Politecnico di Torino

Politecnico di Torino

MSc in Electrical Engineering

Geometrical Parameterization Validation and Design Space Mapping Improvement for Electrically-Excited Synchronous Machines

Candidate: Alessandro Stagnaro

Supervisors: Simone Ferrari
Federica Graffeo
Edoardo Lagorio

Academic Year 2024/25

Contents

1. Introduction

In the following chapter, a brief presentation of the topics discussed in this thesis will be made. In the first place, the context in which this project is conducted will be presented. Following, the main subjects of this work and its objectives will be discussed.

1.1 Overview

In the last few decades, interest towards topics such as sustainability, de-carbonization and environmentally friendly solutions has been growing more and more. These words, previously unknown, are now part of our ordinary lives and influence both our everyday behaviors and the decisions made by governments all around the world.

The great economic and industrial development that has taken place over the last few centuries has undoubtedly brought benefits in terms of improving people's quality of life and general wealth. However, the main players in all this have ignored several fundamental aspects, including the environmental impact of such rapid developments. In fact, in an attempt to meet an ever-increasing demand, the industrial system has made indiscriminate use of fossil fuels, the consequences of which are plain for all to see.

At the present day, climate change is is no longer a possible scenario but reality. In fact, temperatures in 2025 have averagely increased by 1.55 °C compared to pre-industrial levels (1850-1900) [9], sea levels are rising due to pole's ice is melting and atmospheric phenomena are more and more devastating year by year. These events are all related to global warming, a phenomenon caused by the release of greenhouse gases into the atmosphere (GHG). These pollutants have not only negative impacts on climate but also on the population. In fact, the number of people dying from GHG related health issues is increasing on a year by year basis.

Taking into consideration all these events, worldwide organizations and countries' governments have decided to take decisive action in trying to mitigate and possibly solve the environmental question. One of the first steps took place at the Kyoto Conference of 1997, in which 38 industrialized countries committed in trying to reduce their GHG emissions of 5.2%, in the 2008-2012 period, with respect to 1990 levels. [35] Further and more restrictive measures were decided in 2015 Paris Climate Conference, whose main output was the imposition of a global limit in terms of temperature increase of just 1.5°C worldwide. In doing so, each country was able to present its own program to succeed in such task, which included also important financial investments in environmentally friendly technologies. In addition, a trans-

parent global control system was established so that it became possible to verify the effectiveness of the measures taken and also a mandatory update of the single climate goals was imposed every 5 years. [52]

Nevertheless, countries were left free to apply even more restrictive countermeasures at national level. This was the case of the European Union and its European Green Deal. This set of measures is composed by several targets on different time horizons, for example it aims at cutting EU's GHG emissions by 55% by 2030 compared to 1990 levels and to reach net zero emissions by 2050.

Even if these sets of environmental laws were proclaimed, globally the effects are still to observe. In fact, GHG emissions in 2023 reached a record high 53.4 $GtCO_2$, increasing by 0.8% with respect to 2022 [16]. This trend can be observed also for the years prior as Fig. 1.1 shows. However, GHG emissions decreased in the regions that applied more restrictive environmental policies, the global performance is still not enough to satisfy the previously mentioned goals.

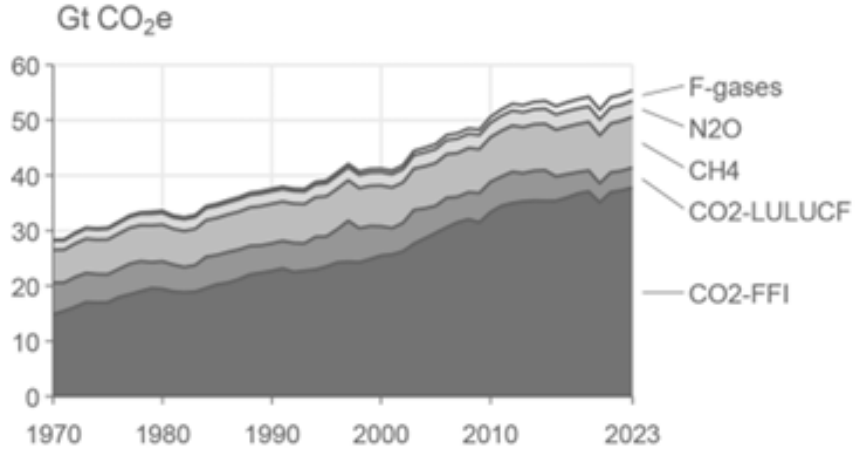


Figure 1.1: GHG Global Emission Levels as of 2023.

1.2 Electric Motors for Traction Applications

Electric motors are one of the spine bones of the current worlwide industry and society. In fact, in 2011, electric machines accounted for nearly 45% the global electricity usage, accordingly to [22]. More recent estimations expect this figure to be in the 45% to 53% range and it is expected to grow even more over the following years. This data is very representative of the importance that electric machines have throughout the world due to their wide range of possible applications, from energy production and industrial machinery to household appliances and transport systems.

1.2.1 The Current Transport System

The focus of this work will be, in particular, on the transport sector. This sector was not immune to the shift in attention towards less environmentally impactful solutions, mentioned in the previous section. In fact, in order to fulfill both global

and local climate goals, for example those convened in the Paris agreements worldwide [52] or in European Green Deal [15] for the European Union (EU), severe actions need to be performed across all areas. Due to the fact that the transport sector consumes approximately 30% of the global energy usage [40], it is consequently one of the largest contributors to global greenhouse gases (GHG) emissions. Considering the enormous amount of energy and emissions that are involved in this field, the benefits that a reduction of both would have towards the environment are evident.

Transports account for approximately 15% of global greenhouse gases emissions in 2019 [12] and, according to more recent estimates, this figure does not appear to have changed significantly during the last years. In fact, as it can be seen from Fig. 1.2, transport related emissions grew on a year basis, with the exception of 2020 due to the COVID-19 pandemic, after which the previous trend continued. [27] According to IEA [24], more than 90% of the energy used for transports worldwide comes from oil derivatives such as gasoline, diesel, and kerosene. This number is due to the fact that the current transport system is still based on internal combustion engines (ICE), which operate on fossil fuels. In general, ICEs are characterized by very low efficiencies, in the 15% to 25% range, meaning that most of the energy extracted from the fuel is not actually used for useful purposes but lost. In addition, the combustion output gases produced by this type of motors are toxic and contribute to worsen both global and local pollution.

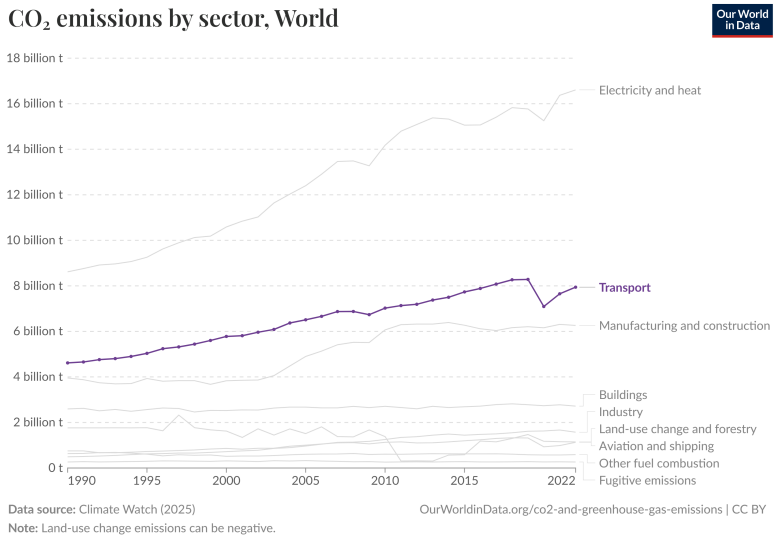


Figure 1.2: Current Transport Sector GHG Emissions

Moreover, the transport field is one of the fastest growing sectors in today's economy as the demands for private vehicles, long range flights and shipments continue to increase, resulting ultimately in higher and higher energy demand. This trend leads to transport-related emissions to be still rising on a year-by-year basis, differently from other sectors whose carbon footprints are reducing. On the other hand, implementing concrete and effective countermeasures to reduce the environmental impact of the entire transport sector is very complex. In fact, certain applications such as aviation, maritime shipping or heavy-duty trucks are challenging to decarbonize as

they require longer ranges and higher energy densities compared to standard vehicles. These requirements constitute a real challenge that will need to be overcome in order to succeed in reducing the entire field GHG emissions. Another element that needs to be taken into account is the fact that the renovation of the transport field is proceeding at a slow rhythm. In fact, many of the vehicles currently in use are outdated and so characterized by lower efficiencies, which ultimately leads to introducing more pollutants into the atmosphere, compared to models of newer generation. This trend is worrying as it can be observed both in high-income countries and low-income ones and it can be associated to an ever-increasing loss of customer's purchasing power caused by inflation growing year by year. Consequently, the challenge is not only to de-carbonize the transport sector, reducing more and more its usage of fossil fuels, by introducing and spreading new "green" solutions but also to make sure that these new environmentally friendly solutions are affordable to customers in order to be actually effective.

1.2.2 The Future of Transports

From the very first moment in which the public opinion has been introduced to the concepts of sustainability and de-carbonization, huge steps have been made and enormous amounts of money invested in order to find suitable alternatives able to resolve the just mentioned issues of the transport sector. If for the previously mentioned heavy-duty sector these problems are not immediately resolvable, the same could not be said for the commercial vehicles. In fact, **electric vehicles (EV)** have been considered, for the last decade, as a key instrument in order to reduce the private transportation's carbon footprint. Differently from standard ICE vehicles, EVs do not produce any output gases during use, consequently reducing pollutants levels both locally and also globally, and so helping in the achievement of the emission's cut goal.

Electric vehicles are powered by electric machines, exploiting the energy stored in a battery pack thanks to power electronic controls, which have an higher efficiency, in the 80% to 95% range, compared to internal combustion engines. Considering the enormous amount of energy consumed by private road transports each year, approximately 80% of the total sector which is equivalent to 25% worldwide [25], the benefits given by any efficiency improvement are notable in terms of both saved energy and not produced emissions.

Electrification of Transports: Hybrid and Battery Electric Vehicles The electrification of road cars was not a new concept made up in the last decades. In fact, the first so-called hybrid vehicle was invented by Ferdinand Porsche in 1900 for the Paris Universal Exhibition. It was a "series" hybrid vehicle, characterized by one gasoline internal combustion engine and two electric motors. The ICE did not operate as the traction motor but as generator, producing the energy required by the electric machines to move the vehicle. Even if this example never reached mass production due to its high production costs and inadequacy of the time's infrastructure, it shows how the benefits of using electric motors on traction vehicles were already well known long before topics such as de-carbonization and electrification became subjects of public domain.

The first modern hybrid vehicle, which entered mass production, was introduced

by Toyota in 1997. Since then, many more electrified vehicles were presented and the resources spent in research for new solutions lead to several new technologies to be implemented.

For what concerns hybrid vehicles, there are four different typologies currently on sale in the market, which differentiate from each other depending on the electrification level and how the battery's energy is exploited.

The first type is represented by **Mild-Hybrid Electric Vehicles (MHEV)** in which the ICE is the traction motor and is supported by a small electric motor and battery, not capable of moving the car alone, during acceleration and coasting phases.

Next, there are **Full Hybrid Vehicles (HEV)** that are characterized by bigger electric motors and batteries, even if the internal combustion engine remains the main traction source, able to move the automobile by themselves for a few kilometers.

After this, it is turn of **Plug-In Hybrid Vehicles (PHEV)**, in which electric motors and batteries are even bigger than for HEVs, but still not the main traction solution, as they are able to move the car for several dozens of kilometers by just exploiting the energy stored in the battery pack. One crucial difference, except for the electric motor and battery pack sizes, between the mentioned technologies is that the batteries in PHEVs can be recharged externally meanwhile for HEVs and MHEVs it can only be recharged using energy produced by the ICE.

Last but not least are the **Extented Range Electric Vehicles (E-REV)**, which exploit a technology very similar to the one developed by Porsche in 1900. In fact, in this case the internal combustion engine present in the vehicle is used to produce the energy stored inside the battery pack which is used by the electric motors for traction purposes. In addition, the battery could not be recharged externally but only using the ICE consequently in order to recharge the pack, it is necessary to refill the fuel tank.

Another possible solution, still not diffused due to infrastructure limitations, is represented by **Fuel Cell Electric Vehicles (FCEV)**. These vehicles exploit a chemical reaction between hydrogen and oxygen to produce electrical energy which is then used by the electric motor for traction purposes. Even if this technology is promising as it bypasses some limitations of standard EVs, its diffusion is slowed down by hydrogen (fuel) procurement, limitations in the dispatch framework and very high costs.

Even if hybrid vehicles represent a step in the right direction towards a reduction of the transport sector emissions, this is still not enough to fulfill the environmental goals imposed by the global institutions and the single countries. The presence of an internal combustion engine, independently on the hybrid vehicle's technology, has the consequence of toxic gases being produced and injected into the atmosphere. In a world in which one of the main targets for the future is reaching carbon neutrality, it is crucial to cut all the un-necessary emissions and private transports related ones fall into this category. For this reason, electric vehicles, with their zero emissions during use, are considered the future of the private transports sector.

Battery Electric vehicles (BEV or EV) have a long history as the first one was invented in the first half of the 19th century, well before the invention of the

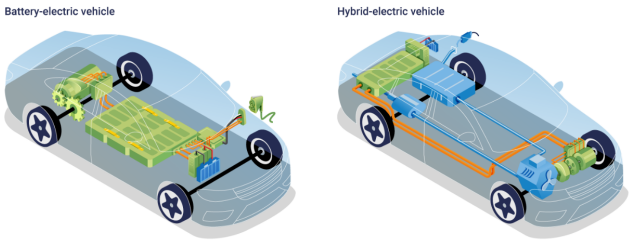


Figure 1.3: Visual Schematic of Hybrid and Battery Electric Vehicles

first gasoline-powered car. At the beginning, they were more electric carriages than real automobiles and they were very popular inside the cities due to their quietness. However, with the advent of the ICE automobiles, they gradually disappeared till the last decades. In fact, the first modern electric vehicle was presented by General Motors in 1996. The term modern refers to the fact that it was not an adaptation of an ICE vehicle to an electric version and that it was the first EV to enter mass production, even if the amount of units assembled was not high. Since then, attention on electric vehicles grew year by year as a solution for sustainable mobility. Consequently, the amount of investments followed this trend, leading to both technological advancement in the field and to automotive manufacturers designing and commercializing their own versions of electric vehicles. The main differences between the previously described vehicles, in terms of components, can be observed in Fig. 1.3.

A brief summary of electrified vehicles' technologies could be observed in Table 1.1.

Technology	**Description**	**Energy Source**	**Motor**
BEV (Battery Electric Vehicle)	Vehicle powered exclusively by battery	Rechargeable battery (Li-ion)	Electric motor
HEV (Hybrid Electric Vehicle)	Internal combustion engine + electric motor, battery charged by engine or regenerative braking	Gasoline/diesel + battery	Internal combustion engine + electric motor
PHEV (Plug-in Hybrid Electric Vehicle)	Similar to HEV but with larger battery rechargeable from external plug	Gasoline/diesel + rechargeable battery	Internal combustion engine + electric motor
FCEV (Fuel Cell Electric Vehicle)	Electric vehicle powered by hydrogen fuel cells	Hydrogen (tank)	Electric motor
MHEV (Mild Hybrid Electric Vehicle)	Internal combustion engine assisted by a small electric motor, battery not rechargeable from external source	Gasoline/diesel + small battery	Internal combustion engine + electric motor

Table 1.1: Summary of Electric Vehicle Technologies

Current EV Situation At the present day, electric vehicles represent 4% of the automobiles around the world. This number has been growing steadily over the last decades, as it can be seen from Fig. 1.4, and is forecast to increase at even higher pace in the following years as, in 2024, more than 20% of new vehicles sold globally were fully electric, with China leading the way at approximately 50%. [26]

This data, even if promising, shows how the spread of EVs is still not wide enough to reach the global climate goals. In fact, a lot of skepticism surrounds these vehicles due to their limitations, in comparison with traditional ones, like waiting additional time for the battery to recharge or the limited ranges, and their higher costs that

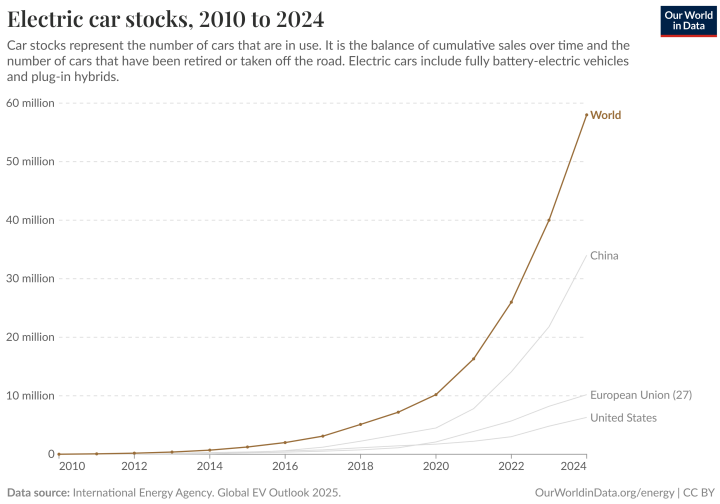


Figure 1.4: Number of Electric Vehicles on the Road Over the Years

prevent their wide-spread. It is not a case that where EVs have more competitive prices, even lower than ICE vehicles, the number of units currently on the road is higher, for example China. For these reasons, governments will have to elaborate possible solutions to incentivize their populations into buying EVs, like for example economic subsidies or advertising campaigns highlighting the advantages of owning one. Only if these strategies are successful, it will be possible to reduce the private transports' carbon footprint, with evident benefits for both earth's ecosystem and population.

Several typologies of electrical machines are capable of fulfilling the requirements imposed by private-traction industry and consequently can be used on electric vehicles. This is due to the fact that electrical machines have an high level of interoperability. However, only a few technologies of electric motors are used in the current automotive market.

In general, permanent magnet motors are the most used in the automotive industry due to their high efficiency and power density levels. Specifically, Internal Permanent Magnet Synchronous Machines (IPMSM) are the most diffused, being installed in approximately 65% of the electric and hybrid vehicles currently on the market. Another very common technology is represented by Induction Machines (IM), which are cheaper and more robust compared to IPMSMs but heavier and less efficient. This category of electric motors covers around 20% of the market share. The remaining percentages are shared between motors of newer generation, like Synchronous Reluctance Machines (SynRM) and Permanent Magnet Assisted Synchronous Reluctance Machines (PMaSynRM), for about 10% and the remaining (approximately 3%) is mainly covered by Electrically Excited Synchronous Machines (EESM) [10]. Fig. 1.5 illustrates graphically the just mentioned percentages for a more immediate comprehension.

During the last years, a shift in paradigm has taken place as the sustainability evaluation of vehicles started considering their entire life cycles, including: their production, the one of their composing materials and apparatuses, operational span

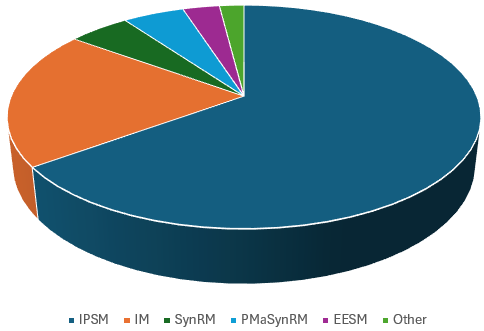


Figure 1.5: Electrical Machines Topologies Shares in the Automotive World

and end-of-life periods. This implies that manufacturers, during the design phase, have to take into account not only the eventual emissions produced by the vehicle during its operational life, but also the ones generated during its production, the ones of its materials and also how the entire system could be recycled or re-used at the end of the life cycle. These new requirements involved all types of vehicles from traditional ICE to electrified one.

In fact, if it is true that electric vehicles do not generate any pollutant gases during their use, the same can not be said for their entire life cycles. Researches testify that to produce an electric vehicle, an higher amount of emissions is created compared to the assembly of an internal combustion engine based car, approximately 50-60% more [21]. However, EVs are more sustainable than traditional vehicles as the overall amount of emissions produced over their entire lives is still lower, as it can be seen from Fig. 1.6. [23] The gap could even become more evident if the electrical energy used by EVs, to perform their traction purposes, arrives from renewable sources.

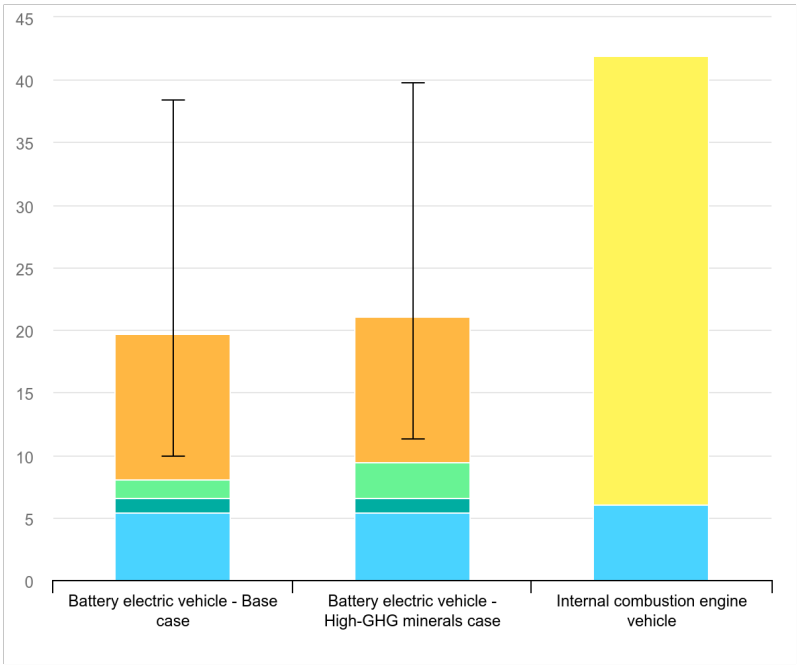


Figure 1.6: Comparative Life Cycle GHG Emissions of an EV and ICE Vehicle

The higher amount of pollutants generated during the EV production phase is mainly due to generation of their unique components. In fact, electric vehicles are made up of a series of components not commonly used before in the automotive industry. The main elements of an EV are: electric motor; battery pack; power electronics (DC-DC converter, inverter and on-board charger) and their respective control, management and cooling systems.

Researches say that the production phase of an electric vehicle, including all components (battery, motors, electronics, chassis, etc.), accounts for approximately 60% of the entire life cycle's emissions. [6] The remaining percentages are taken respectively by the electrical energy consumed by the vehicle, whose environmental impacts depends on the energy mix of the single country, possible changeable components (tires, brakes etc.) and the recyclability of the system.

The battery pack of an EV is for sure the component with the biggest carbon footprint of the entire vehicle. It accounts for about 50% of the vehicle production emissions (equivalent to 30% of the entire life cycle) . Most of these emissions are related to the procurement of the raw materials, like cobalt, nickel or lithium, composing the battery pack as their extraction requires high quantities of energy and takes place in countries with energy mixes highly reliant on fossil fuels.

The manufacturing of the vehicle glider (or chassis) has also an high environmental impact as it accounts for approximately 25% of the entire vehicle production emissions. Once again, most of this share is accountable to the procurement of the raw materials, such as aluminum or composites. In fact, these materials are lightweight but in order to be produced and processed require huge amounts of energy to be consumed.

Also power electronics and electric motors have a certain environmental impact in the overall production emissions of an EV. In fact, they account respectively for about 8% and 10% of GHG emissions. As for the previous components these shares are highly influenced by material extraction, with particular attention for the permanent magnets possibly present in motors, and also the whole manufacturing processes. [21]

The impacts of the single components of an EV in summarized in Table 1.2, where each impact is divided between the production process of the vehicle and the overall life cycle. It is important to underline that the numbers refereed to the energy consumption for traction purposes is an average value as it is highly dependent on how that energy is procured consequently on the owner's provision or the country's energy mix.

Component	**Production Emissions (% of total production)**	**Life Cycle Emissions (% of total LCA)**
Battery pack	50%	30%
Electric motor	10%	5%
Vehicle chassis	25%	15%
Electronics	8%	4%
Energy consumption	0%	45%

Table 1.2: Estimated percentage emissions of electric vehicle components during production and full life cycle.

1.3 Electrically-Excited Synchronous Machines

Electrically-Excited Synchronous Machines are currently one of the most discussed topics in the automotive world. In fact, even if they currently cover a small market share, it is expected that their implementation on vehicles will grow over the following years, as more and more EESM based powertrains are being developed (Fig. 1.7). This trend was originated by the industrial limitations that the current electric motors mounted on EVs impose.

Figure 1.7: Example of EESM based EV Power Train

1.3.1 A Renewed Interest

As mentioned in the previous section, Permanent Magnet Synchronous Machines and their derivatives are currently the most diffused in the automotive world. However, these motors, even if they offer very good performance and efficiency levels, have some limitations. First of all, the presence of rare-earth permanent magnets in most designs creates issues both on a material procurement and environmental levels. In fact, these materials are not commonly available around the world but only in a few countries. This implies that in case of geopolitical tensions, as in the last few years, the supply of these could be reduced or even stopped as a tool of bribery. In addition to that, the extraction of rare-earth materials generates considerable GHG emissions as it requires huge amounts of energy to be performed. Considering also the economical part, the presence of rare-earth magnets in electrical machines implies additional costs as these materials are expensive, which then leads to higher prices of whole vehicle.

For these reasons, some manufacturers have been trying to either remove rare-earth magnets from their electrical machines, reduce their amount or substitute them with different kinds of magnets. Some European constructors have opted instead for investing resources on research and development of other topologies of electrical machines. In this last case, it can be found the Electrically-Excited Synchronous Machine (EESM), which is the main focus of this thesis.

EESM machines surpass the previously mentioned geopolitical, environmental and economical limitations. In fact, these motors substitute permanent magnets with direct current electrical windings, which are more sustainable, less expensive and made of materials easier to procure. Due to the fact that EESM machines are

a well consolidated technology in other applications, they guarantee excellent reliability levels while offering also high efficiencies and power densities, not achievable with other topologies like an Induction Machine.

1.3.2 Structure and Operating Principle

EESM motors, just like any other electrical machine, present a stator and a rotor separated by an airgap. The stator is composed by several sheets of iron stacked one on each other and processed in order to create the slots in which a three-phase electrical current winding, where a three-phase set of alternated (AC) currents flow, is inserted. The configuration of the stator windings and how the single wires are positioned inside each slot, it strictly depends on the single design of the machine and on its application. For what concerns the rotor, it is also made of stacked sheets of iron and it also presents an electrical winding. But, this winding is single phase and it is crossed by a direct current (DC).

The DC rotor winding is also called excitation winding. In fact, this electrical winding performs the same task as permanent magnets in PMSM machines by generating a static magnetic field (or excitation field), originated by the externally provided DC current flowing into the single phase rotor winding. Meanwhile, as the three-phase stator winding is externally supplied with a balanced three-phase power supply, it generates a magnetic field rotating at synchronous speed, which depends on the stator supply frequency and on the number of pole pairs of the machine.

The operating principle of the EESM machine is based on on the fundamental laws of the magnetic field and electromagnetic induction. In fact, the electromechanical conversion is possible from the interaction of stator's magnetic field and rotor's one. Having these two magnetic fields a different rotational speed, as the stator's one rotates at synchronous speed meanwhile the rotor's is static, the interaction between the two generates an electromagnetic torque. This torque forces the rotor, if not immobilized, to start rotating at an increasing speed until it reaches the synchronous one. At this speed, the rotor (and so its magnetic field) has the same rotating speed as the stator's magnetic field.

Consequently, it can easily be deduced that by controlling the DC current flowing in the rotor winding, it can be changed the amplitude of the rotor's magnetic field and consequently the output torque, which is one the main benefits of the EESM machine with respect to PMSM ones.

In addition, the rotor of Electrically-Excited Synchronous Machine can have two different shapes: cylindrical (or not-salient poles) or salient poles. Not-salient poles EESM machines present a cylindrical rotor and so they can be considered as isotropic machines as the airgap thickness remains constant, neglecting the effects caused by the rotor slots, all round the diameter. Instead, salient poles EESM machines present a variable airgap thickness that causes the anisotropy of the system. The consequence of this is having variable inductances along the airgap which leads to the machine torque having two contributions, one caused by the magnetic coupling and the other due to the rotor salient poles. The two different rotor topologies of EESM machines can be observed in Fig. 1.8.

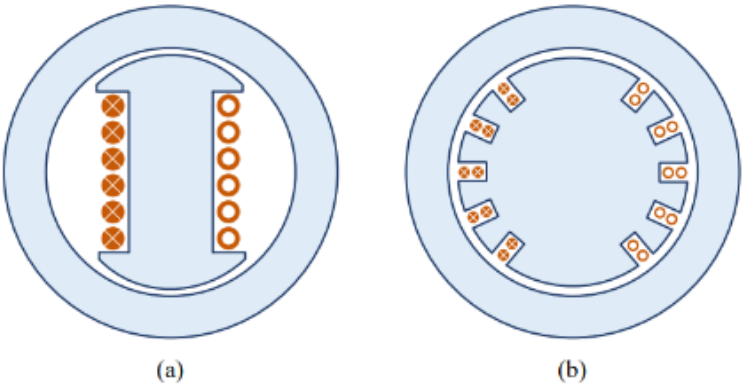


Figure 1.8: EESM Machine Rotor Typologies: a) Salient Poles; b) Cylindrical

The two rotor configurations are interchangeable, in the sense that both can be used for the same applications. Even if in general, cylindrical rotor EESM machines have higher rotational speeds than salient-pole ones. In contrast, within the automotive field, salient-pole EESM machines attracted more interest than cylindrical ones due to the possibility of exploiting two torque contributions instead of one. Considering the fact that torque is one of the main parameters for a traction motor, these type of motors have the potential of providing higher torques due to the sum of two contributions: the reluctance torque and the magnetic coupling one.

1.3.3 Excitation Systems

The rotor excitation system is one of the most crucial components in an EESM machine as a malfunction in this system compromises the correct operation of the entire motor. At the present day, there are several technologies able to perform such task. Traditionally, EESM machines had excitation systems composed by sliding contacts between stationary graphite brushes and rotating slip rings. If it true that this solution is very cheap and offers a fast dynamic response, on the other hand it limited the diffusion of the EESM machine as it caused concerns about the wear of the brushes, which require additional maintenance, and possible safety issues due to eventual sparks in the motor.

During the years, a lot of research and development has been performed on EESM machines in order to find a feasible alternative solution to brushes and slip rings for rotor excitation. Some of the possible options will be briefly explained in the following paragraphs:

Brushless Excitation As it can be deducted from the name, brushless rotor excitation requires no contact between static and moving parts. In fact, in this solution a small synchronous generator is installed on the EESM's rotor, this machine generates a system of three-phase AC currents which is rectified by a diode rectifier installed on the main machine's rotor, which then receives the obtained DC current. This solution partially surpasses the traditional excitation issues as it requires very little maintenance and has high reliability levels. However, it is more complex and offers a worst dynamic control performance.

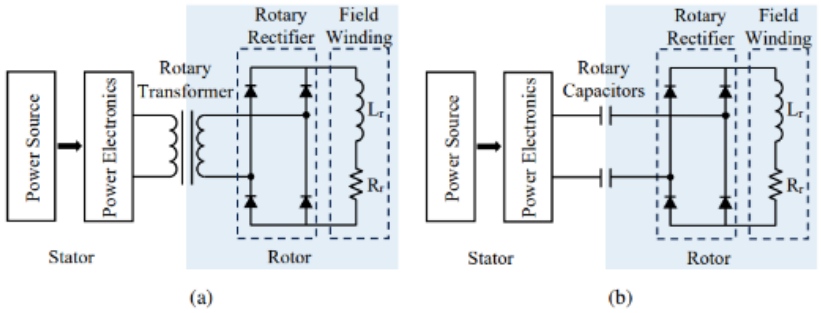


Figure 1.9: Wireless Excitation Circuits: a) Inductive; b) Capacitive

Wireless Excitation This kind of solution has become a feasible solution for EESM excitation systems only during the last years thanks to the researches performed in wireless technologies and power electronics. Currently there are only two possible wireless excitation systems: rotary transformer and rotary capacitors.

Rotary transformers perform an inductive power transfer (IPT) between two AC windings, a primary and a secondary, just like a normal transformer. However the primary winding is installed on the stator of EESM machine meanwhile the secondary one is installed on the rotor and rotates with it. Consequently, when power has been transferred to the secondary winding, the AC current is transformed into DC using a rectifier installed on the rotor. This kind of solution presents some limitations related to its high operational frequencies and the presence of both leakage inductances and parasitic capacitances which alter its correct operation.

Rotary capacitors perform a capacitive power transfer (CPT) between two conductive materials, one positioned on the stator and one on the rotor, by creating an alternated electric field due to the rotor's movement. Once again, the transferred AC current is transformed in a DC one by a rectifier installed on the machine's rotor. This solution offers good performance and high reliability, even at high rotational speeds. It requires very good electrical insulation and protections and it might present some structural issues in presence of mechanical vibrations.

A circuital scheme of the solutions can be observed in Fig. 1.9. [18]

1.3.4 Steady State Model

In order to deliver a discussion as exhaustive as possible, in the following section a *dq*-model of the EESM machine at steady state will be described. In addition, as most of the SyR-e software simulations are performed in static conditions, having further information on the machine under analysis is crucial for a reader's better and more complete understanding.

In general, electrical machines can be represented in a simplified way through models, constituted by specific sets of electrical and magnetic equations. In the EESM case, it can be described using electrical and magnetic equations of the single-phase rotor and the three-phase stator.

As it is know, the variables of these equations can be associated to vectors and consequently the equations can be rewritten in a vector form. These sets of equations representing an electrical machine can be referred to various reference systems, for

example the stator's stationary *1-2-3* frame or the rotor's rotating one.

However, the electrical motors' models can be furtherly simplified by applying a set of transformations. Specifically, the *1-2-3* to α-β and α-β to *d-q* ones. [38]

This thesis will focus on EESM machines having salient pole rotors. As already mentioned, this rotor structure causes the machine to be anisotropic consequently, both the inductances and reluctances are variable along the airgap as its thickness varies depending on the rotor's position.

The application of the α-β and α-β to *d-q* reference transformation brings with it a series of benefits. First, it leads to having isofrequential stator and rotor variables which in the EESM case means having DC variables at steady state. Then these transformations cause the inductance matrix of the machine to be independent of the rotor position. Lastly, it introduces the induced EMF (electro-magnetic force) term into the stator's electrical equation.

The reference system transformations here considered use the controllers convention, as a consequence the amplitude of the variables are invariant from one transformation to another. The electric and magnetic equations that represent an electrical machine can be written in a general matrix form as represented in 1.1.

$$\begin{cases} \boldsymbol{V} = \boldsymbol{RI} + \boldsymbol{\Omega\Lambda} \\ \boldsymbol{\Lambda} = \boldsymbol{LI} + \boldsymbol{MI} \end{cases} \tag{1.1}$$

Each element of these equations, both the vectors and matrices, can be decomposed to highlight the single *d*-axis and *q*-axis components. For example, the structure beneath the voltage $\mathbf{V}$, current $\mathbf{I}$ and flux linkages $\boldsymbol{\Lambda}$ vectors is observable in 1.2 :

$$\mathbf{V} = \begin{bmatrix} V_d \\ V_q \\ V_r \end{bmatrix} \quad \mathbf{I} = \begin{bmatrix} I_d \\ I_q \\ I_r \end{bmatrix} \quad \boldsymbol{\Lambda} = \begin{bmatrix} \Lambda_d \\ \Lambda_q \\ \Lambda_r \end{bmatrix} \tag{1.2}$$

The other elements of 1.1 are related to the characteristics of the electrical machine under analysis, which for this thesis is an EESM one. $\mathbf{R}$ is the resistances matrix, it is a diagonal 3x3 matrix composed by stator resistance R_s and rotor resistance R_r. $\boldsymbol{\Omega}$ is a 3x3 matrix representing the electrical speed $\boldsymbol{\omega} = 2\pi f$. $\mathbf{L}$ is another diagonal 3x3 matrix, containing the *d*-axis inductance L_d, the *q*-axis one L_q and the rotor's L_r. $\mathbf{M}$ is also a 3x3 matrix containing the mutual inductance effect between stator and motor expressed by M_{sr}. It is important to underline that in these considerations the inductances' cross-coupling effect was not considered. In addition, it must highlighted that the previous inductances L_d and L_q are the sum of two contributions: leakage L_δ and magnetizing L_m inductances. 1.5 The structure of the previously mentioned matrices can be observed in 1.3 and 1.4.

$$\mathbf{R} = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_r \end{bmatrix} \quad \boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega & 0 \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1.3}$$

$$\mathbf{L} = \begin{bmatrix} L_d & 0 & 0 \\ 0 & L_q & 0 \\ 0 & 0 & L_r \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} 0 & 0 & M_{sr} \\ 0 & 0 & 0 \\ M_{sr} & 0 & 0 \end{bmatrix} \tag{1.4}$$

$$L_d = L_{\delta d} + L_{md} \qquad L_q = L_{\delta q} + L_{mq} \tag{1.5}$$

Consequently, unfolding the electrical and magnetic equations contained in 1.1, the following equation system 1.6 is obtained. Note that electrical equations are on the right hand side meanwhile magnetic one on the left.

$$\begin{cases} V_d = R_s * I_d - \omega * \Lambda_q \\ V_q = R_s * I_q + \omega * \Lambda_d \\ V_r = R_r * I_r \end{cases} \qquad \begin{cases} \Lambda_d = L_d * I_d + M_{sr} * I_r \\ \Lambda_q = L_q * I_q \\ \Lambda_r = L_r * I_r + M_{sr} * I_d \end{cases} \tag{1.6}$$

By substituting the magnetic equations into stator's electrical ones, 1.7 equation system is obtained, which represents the electrical machine steady state model equations.

$$\begin{cases} V_d = R_s * I_d - \omega * L_q * I_q \\ V_q = R_s * I_q + \omega * L_d * I_d + \omega * M_{sr} * I_d \end{cases} \tag{1.7}$$

It is now possible to rewrite this system of equation in phasorial form as in Eq. 1.8, which can be observed also in graphical form in Fig. 1.10.

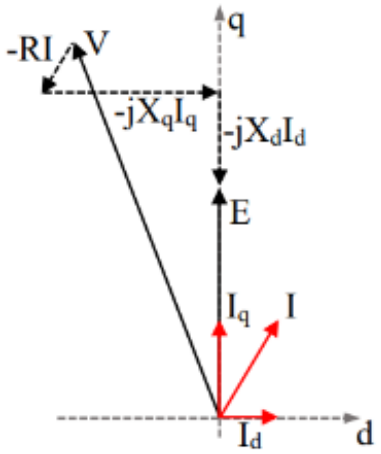


Figure 1.10: EESM Machine Generic Phasor Diagram

$$\underline{V} = R_s * \underline{I} - j * X_q * \underline{I_q} + j * X_d * \underline{I_d} + \underline{E} \tag{1.8}$$

The single elements of 1.8 can be unpacked as in 1.9 and 1.10. Note that $\underline{E}$ is the no load induced voltage.

$$\underline{V} = V_d + j * V_q \qquad \underline{I} = I_d + j * I_q \qquad \underline{I_d} = I_d + j * 0 \qquad \underline{I_q} = 0 + j * I_q \tag{1.9}$$

$$\underline{E} = j * \omega * M_{sr} * I_r \qquad X_d = \omega * L_d \qquad X_q = \omega * L_q \tag{1.10}$$

EESM Output Torque One of the main design parameters of an electrical machine is its output torque. In order to obtain the EESM machine output torque expression, it is necessary to perform an energy balance before. To perform such task, it is necessary to consider the dynamic model of the motor. However, for simplicity's sake and to reduce the length of this section, the steps necessary to obtain the output torque equation will be neglected and only the final expression reported.

Removing the stator and rotor Joule losses and magnetizing power terms, the ones remaining constitute the output mechanical power. At this point, dividing for the mechanical speed $\omega_r = \omega/pp$, where pp indicates the number of pole pairs of the machine, the EESM machine output torque equation is obtained 1.11.

$$T = \frac{3}{2} * pp * (\Lambda_d * I_q - \Lambda_q * I_d) \tag{1.11}$$

Substituting the d-axis and q-axis magnetic equations into the torque one, the obtained result is 1.12.

$$T = \frac{3}{2} * pp * (M_{sr} * I_r * I_q + (L_d - L_q) * I_d * I_q) \tag{1.12}$$

Analyzing the torque equation, the following considerations can be deducted. First, the EESM output torque is composed by two terms, one related to magnetic coupling called "electromagnetic" torque and the other related to the anistropy of the rotor. In case the rotor current would be zero, the torque equation has only the anisotropy contribution and behaves like a relutance synchronous machine, in which the MTPA angle is obtained for a current phasor angle of 45° that maximizes both I_d and I_q. As the rotor current I_r increases in amplitude, the MTPA angle tends to rotate towards negative d-axis current values. In case of a machine operating in the linear iron region, the value of L_d is higher than the one of L_q, leading to a positive difference of the two, consequently in order to sum both torque contributions both I_d and I_q need to be positive, meaning that the MTPA angle is positioned between 45° and the q-axis. Instead, if the rotor current increases even more, causing the machine iron to work in the hard-saturation region, this leads L_d and L_q to have similar values. Consequently, the position of the MTPA coincides with q-axis. [18]

1.3.5 Benefits and Drawbacks

In the following section, a brief overview of the general benefits and drawbacks associated with the usage of EESM machines will be performed.

As already mentioned, EESM machines present a DC rotor winding, which is responsible for generating the excitation field, allowing the removal of permanent magnets from the motors. The benefits connected to this feature in terms of sustainability and independency from geopolitical tensions have already been addressed, so it is now time to talk about the technical ones.

First, being the excitation generated via a DC current flowing in the rotor winding, the EESM machine allows for full controllability of the excitation field. In fact, by reducing or increasing the rotor DC current, it is possible to modify the excitation field's intensity. This leads to the capability of the machine to deliver high torques even at low rotation speeds, without absorbing high stator currents. The ability of controlling the excitation current at any time allows EESM machines not to have any unwanted generation operations, usually present in PMSM ones, as the source of excitation can be rapidly removed by opening the rotor's terminals.

Being the excitation field strictly correlated to the EMF induced at stator, by varying the rotor current amplitude, automatically the amplitude of the induced EMF varies. This leads to the possibility of implicitly controlling the machine's power factor and consequently regulating the reactive power exchanged. Having an high power factor is beneficial as it allows to use windings composed by conductors of smaller sections.

Lastly, EESM machines are characterized by very wide speed ranges in which they are able to operate at constant power, this is due to their crucial ability of controlling the excitation rotor current.

However, EESM machines present also some critical points. For example, the necessity to supply the rotor winding with a DC current introduces additional power electronics devices, already mentioned in a previous section, increasing the system's complexity and costs. In the past, one the EESM machine biggest limitations was related to the necessity of bringing current to rotating parts, in fact, the usage of brushes and rotating rings created both security and reliability concerns, which could be solved nowadays using one of the systems described in Section 1.3.3.

Moreover, the presence of the rotor DC current, flowing in its winding, generates addition Joule losses and consequently heat that needs to be dissipated. Lastly,as for Induction Motors, EESM machines suffer limited cooling capabilities due to the difficulty in reaching the rotor's conductor to cool them.

1.4 SyR-e Software

The analysis performed in this thesis has the software SyR-e as one of crucial subjects. SyR-e (Fig. 1.11) is an electrical machine design software developed by a collaboration between Politecnico di Torino and Politecnico di Bari. Originally created to design synchronous reluctance machines, it is now able to perform such task on more topologies by means of finite element analysis (FEA) and multi-objective optimization. [14]

Figure 1.11: SyR-e Software Logo

In order to successfully perform its analysis, SyR-e has been developed in the Matlab ecosystem and relies on the free FEA software FEMM to perform precise electro-magnetic simulations. The SyR-e work flow is represented in Fig. 1.12. It can be seen that once the user has defined the electrical machines desired parameters, a parameterized drawing of the machine is created as .fem file, which is then analysed by FEMM. The results obtained return to Matlab where a performance analysis is executed.

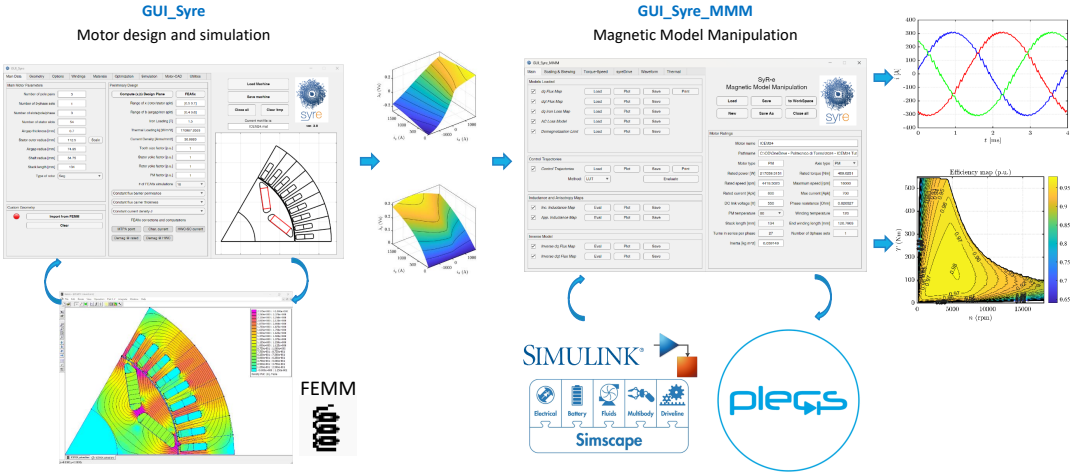


Figure 1.12: SyR-e Software Operational Workflow

In addition, SyR-e is compatible with the Matlab Parallel Computing Toolbox, which allows to perform multiple simulations in parallel. This feature allows to considerably cut the overall simulation time as each simulation can be run on a different computer's core.

To facilitate the user experience while using SyR-e, a Graphical User Interface (GUI) has been developed, where it is possible to apply geometrical adjustments or even modify thermal, magnetic, mechanical or electric parameters of the machine

under analysis. However, it is important to underline that SyR-e can be used also in "manual" mode, without exploiting this feature. Further information about GUI structure will be given in the following subsection.

1.4.1 Graphical User Interface

In order to have a user's more immediate comprehension of the SyR-e software, the GUI feature has been developed. To design an electric machine using SyR-e, the GUI needs to be launched by using the command "GUI_Syre" in Matlab command window.

Once the Graphical User Interface has been launched, its main tab will appear. As soon as the main GUI tab is loaded, a default motor automatically opens; however, the user can decide the work on a pre-existing motor file by clicking on the "Load Machine" button in the right hand side corner and select the desired option. The main GUI window can be observed in Fig. 1.13, where it can be seen that it is divided into 8 secondary tabs, each with a dedicated purpose.

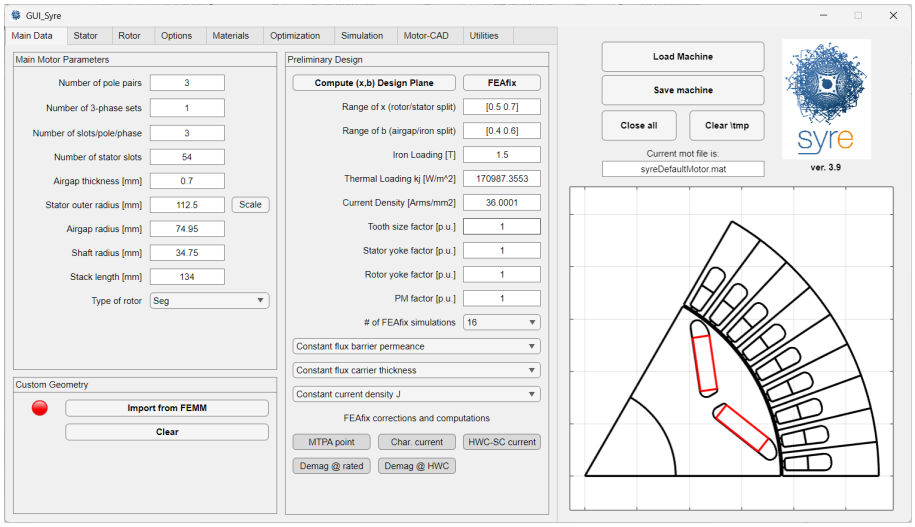


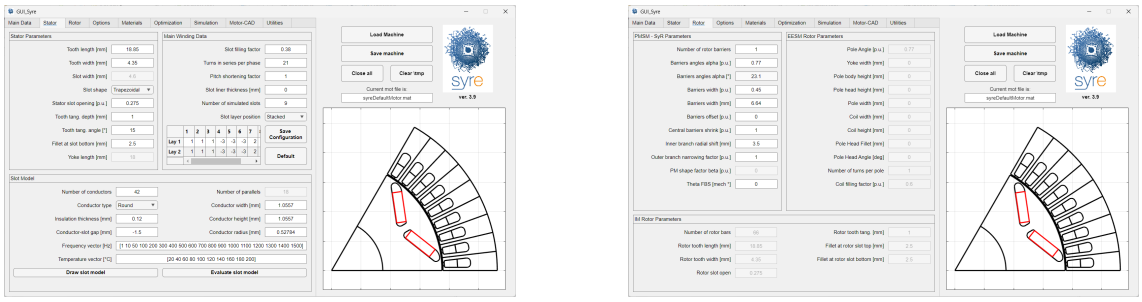
Figure 1.13: SyR-e's GUI Main Tab

In the following paragraphs, an explanation of the single GUI tabs will be given for the reader's sake of clarity.

Main Data In this section, it is possible for the user to set the main machine's parameters for its appropriate design. First of all, it is possible to select the type of rotor and consequently the technology of the machine. It is also possible to decide: number of pole pairs, number of three-phase sets, number of slots per pole per phase, number of stator slots and other main geometrical parameters. While doing so, the user can also observe the consequences of the desired changes as a drawing of one pole of the machine appears on the right side. In addition, the user can also assign some preliminary design parameters like flux and current densities. In short, this section allows to define volumes and material's exploitation of the machine to design.

Stator and Rotor The second and third tabs of SyR-e's GUI, observable in Fig. 1.14 are dedicated to the definition of every aspect of the machine's stator and rotor. For the stator section (Fig. 1.14a), it is possible to assign more geometrical parameters (tooth lengths, tooth widths slot shapes, etc.) but also to define the stator windings and slots. For what concerns the stator winding definition, it is possible to decide: slot filling factor, turns in series per phase, pitch shortening factor and other important winding data. In addition, in order to model the stator's slots, it is possible to select the number, type, width and radius of the conductor inside each slot and also the temperature and frequencies to conduct the thermal and frequency analysis.

The rotor's section (Fig. 1.14b) is similar to the one of the stator but it presents some peculiarities. In fact, in this case, the tab is divided in three sections, each dedicated to a different type of electrical machine. Depending on the rotor type chosen in the first tab, the software allows the user to modify only the parameters correspondent to the technology selected. The variables that can be changed in this section are strictly related to the rotor typology. For example, in EESM case, parameters such as rotor coil filling factor, pole width or pole head fillet can be modified. Meanwhile, for the induction machine the user can select the number the number of rotor bars, the tooth lengths and other parameters.



(a) SyR-e's Stator GUI Tab

(b) SyR-e's Rotor GUI Tab

Figure 1.14: SyR-e's Stator and Rotor GUI Dedicated Tabs.

Options This section, observable in Fig. 1.15, is dedicated to the user's selection of non-electrical parameters of the machine to design. It is divided into three smaller windows, each dedicated to one aspect. The three areas of interest are dedicated to the definition of: thermal parameters, structural parameters and ribs design.

In the window regarding thermal parameters, the user is able to select the main thermal parameters of the machine. The software is able to automatically calculate the machine's nominal current and winding temperature from imposing the housing and copper target temperature and by selecting one of three variables: thermal loading factor k_j [W/m^2], rated losses [W] or field current density [Arms/mm^2]. Selecting as input one of these quantities, the others are calculated as a consequence by SyR-e. In addition, the phase resistance is computed from the geometrical parameters selected in the previous tabs. [14]

SyR-e implements also a simple structural model to be used in the machine's design structural evaluations. The main parameters of this section are the "Over-

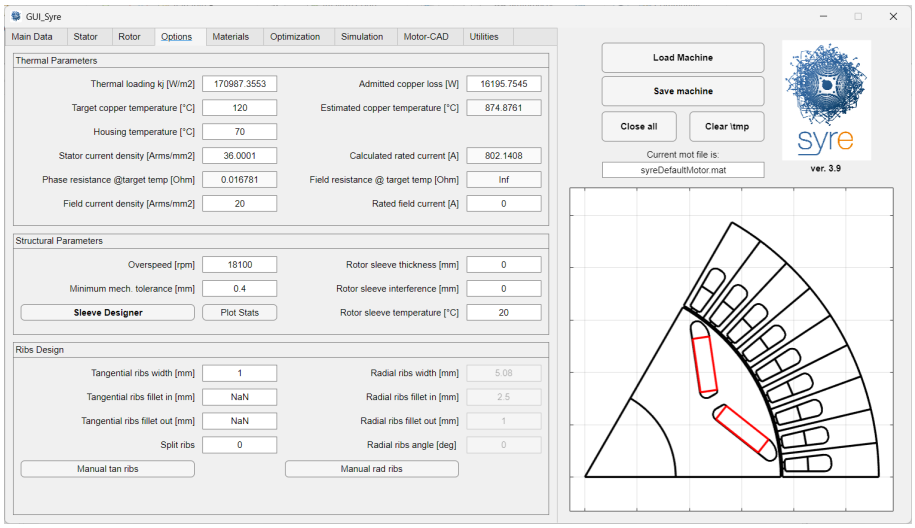


Figure 1.15: SyR-e's GUI Options Tab

speed" [rpm], or maximum rotation speed, and the "Minimum Mechanical Tollerance" [mm]. The first value is used to size radial ribs and it considers only centrifugal forces. While, the second one is used for the sizing of the tangent ribs. In addition, it is possible include also the rotor sleeve, by imposing its thickness, using its dedicated tool. [14]

The last window of this tab is dedicated to the ribs design. This tool is very useful as it allows to design size and dimensions of the flux barriers by defining the thickness of tangent and radial ribs while neglecting overspeed parameters. It is also possible to decide if the ribs need to be split or not and change their slope angle.

Materials GUI's fifth tab (Fig. 1.16) is the one dedicated to materials. In this section, the user assigns to each part of the electrical machine (stator core, stator slot, flux barrier, rotor core, shaft, etc.) its material and defines their weights. Consequently, using the data uploaded in the materials' library, SyR-e automatically computes both the total motor mass and the rotor inertia. This tab is divided into three smaller ones: material data, material library and permanent magnet.

In the material library, the user finds the material database uploaded in SyR-e, it is completely customizable and the user can remove or add materials at its discretion, using the apposite buttons. In addition, for each material present in SyR-e's database, it is possible to observe its characteristic B-H magnetic curve.

The permanent magnet portion is instead dedicated to the design of the permanent magnets mounted on certain topologies with the aim of calculating the characteristic current of the magnet, using the software FEA iterative procedure. [42]

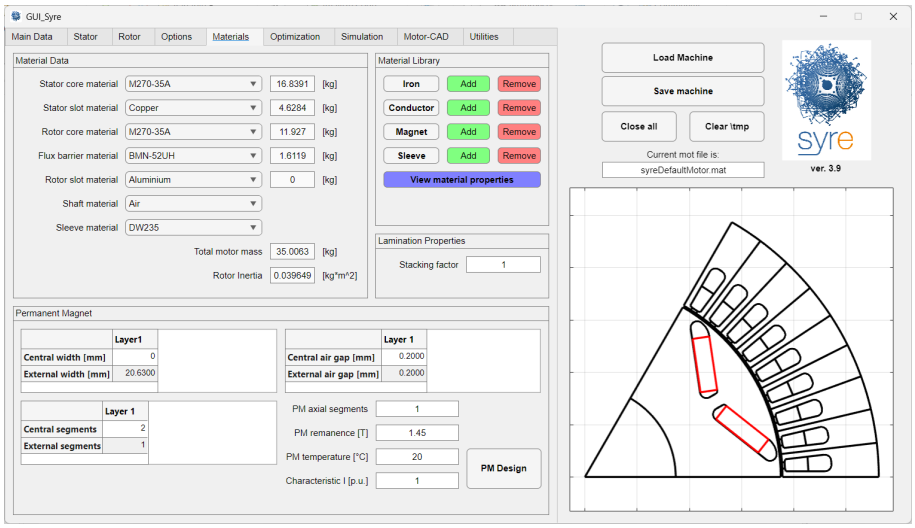


Figure 1.16: SyR-e's GUI Materials Tab

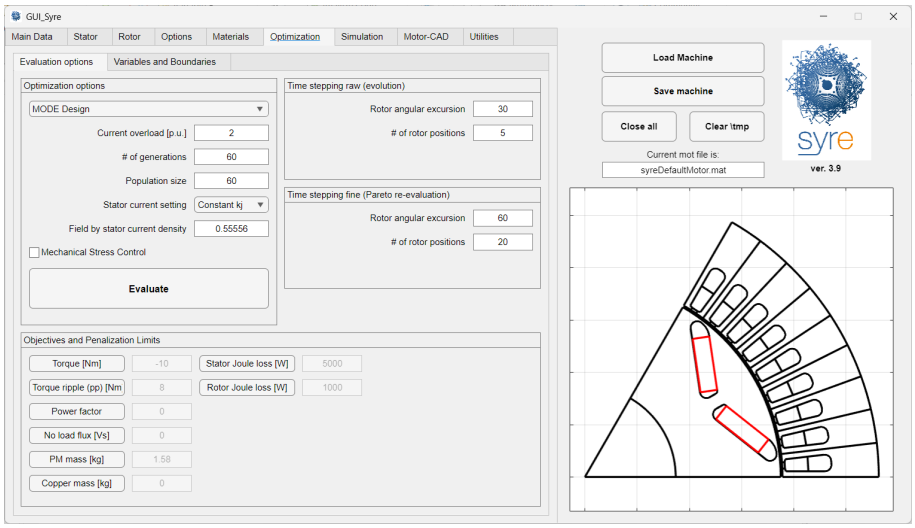
Optimization This tab is one the most important in SyR-e as it allows to perform the optimization of the machine under design. It presents two different pages, which can be seen in Fig. 1.17, in the first the user selects the optimization options while in the second the variables and boundaries.

The first page (Fig. 1.17a) of this section is divided in 4 windows: optimization options; time stepping raw (evolution) and fine (Pareto re-evaluation); objectives and penalization limits. In the first one, the user chooses which kind of optimization to perform and its main parameters (current overload, number of generations, population size and stator current setting). It is also possible to perform a mechanical stress control by enabling its option. In the second and third ones, it is possible to decide the time step, in terms of rotor angular excursion and number of positions, at which the analysis is performed. In the last one instead, the user decides the objectives of the optimization between the ones available (torque, torque ripple, power factor, copper mass, etc).

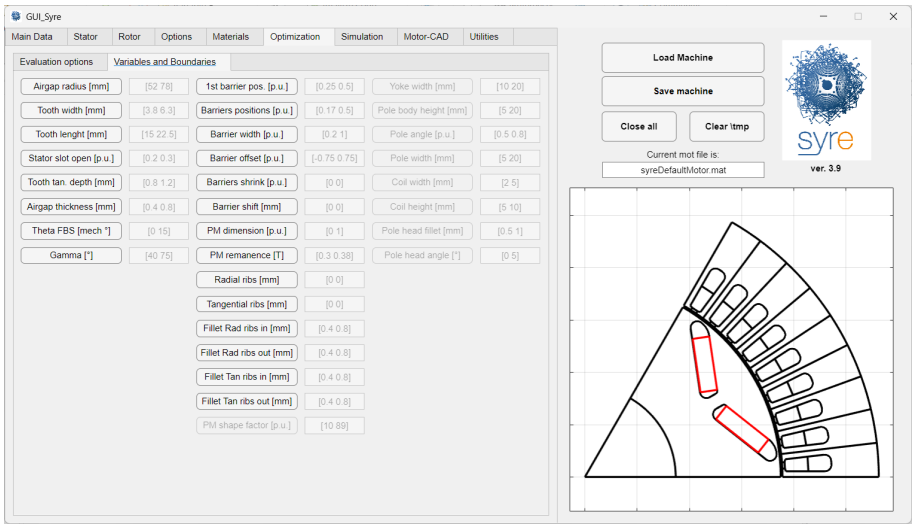
The machine's design optimization is performed using the Multi-Objective Differential Evolution (MODE) family of algorithms, in specific NSGA-II. The setup of the analysis is decided by the user in this GUI tab. As a consequence of the optimization procedure, parameters of stator and rotor can be modified in order to reach the desired objectives. SyR-e offers multiple types of analysis: MODE Design, MODE Refine, Surrogate Model Dataset LHS and Surrogate Model Dataset Sobol. The first two options are MODE type optimization, applied, respectively, to a machine designed from scratch and to an already existing one. Meanwhile, the last two alternatives are types of sensitivity analysis.

The outputs of the optimization algorithm are two files, a Matlab one and a FEMM one, containing all the parameters of the optimized electrical machine.

The second page (Fig. 1.17b) of the optimization tab contains the decision variables and their respective boundaries of the optimization problem. Depending on the machine type chosen in the first tab, different variables can be selected and at user's discretion also their limits can be modified.



(a) SyR-e's Optimization Options GUI Tab



(b) SyR-e's Optimization Variables and Boundaries GUI Tab

Figure 1.17: SyR-e's Optimization Dedicated Tabs.

Simulation In this section of the GUI, it is possible to perform various types of FEA simulations on the electric machine designed in the previous steps. This tab is divided into three smaller windows: simulation setup, custom current and mesh control.

In the first one, it is possible to define the parameters with whom the user desires to perform the simulation. SyR-e offers a wide selection of parameters, for example: current phase angle, phase current amplitude, current load, number of rotor positions, rotor speed, etc. and evaluation type. The evaluation type is the most important one as it indicates the type of simulation the user intends to perform on the electrical machine. There are various types of simulations that can be executed like: single point current computation, torque and flux maps calculation, structural analysis, demagnetization curve analysis, etc.

In the custom current window, by enabling its correspondent flag, it is possible to create from scratch a current waveform, load a pre-existing one and perform the

simulation using this custom current. In addition, the user can also calculate AC losses of the machine.

For what concerns the mesh control zone, here the user can decide the parameters of the mesh what will be used in the FEA simulation. The parameters to select are: mesh size, airgap factor, PM factor and structural PDE.

Once the simulation parameters have been decided, the user can start the computing by clicking on one of the buttons present in the low right hand corner. Each button conducts to softwares and specific types of FEA, like: FEMM; MAGNET; ANSYS and JMAG. It is possible to observe the structure of the Simulation GUI window in Fig. 1.18.

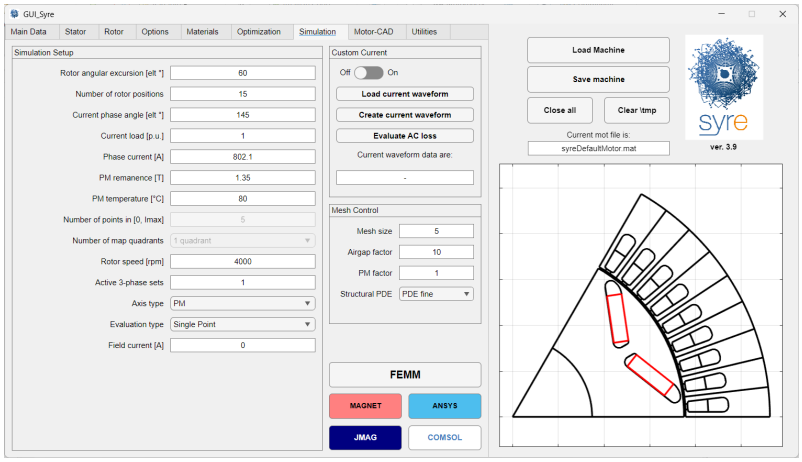


Figure 1.18: SyR-e's GUI Simulation Tab

Motor-CAD This GUI tab is dedicated to the designed electric machine's export to Ansys Motor-CAD. In this section it is possible to export the machine towards other FEA softwares and setup thermal and electromagnetic simulations. It is even possible to decide further parameters for thermal evaluations, including: type of evaluation, transient period, initial temperature, cooling technology, cooling fluid, flow rate, thermal limits, etc.

An overview of this tab can be observed in Fig. 1.19.

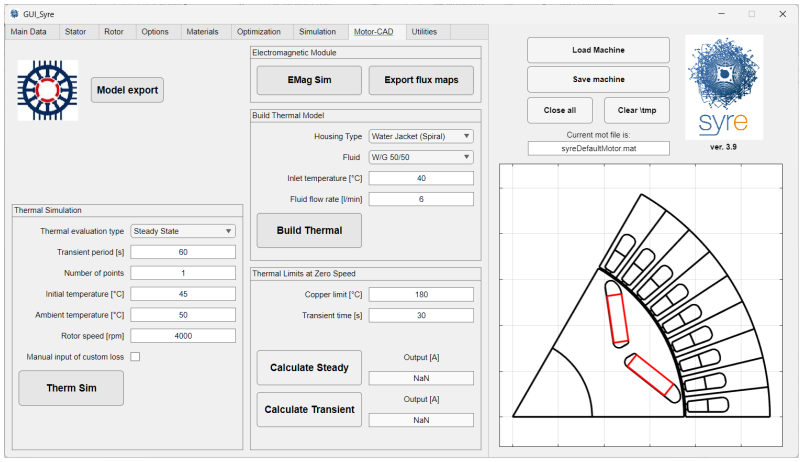


Figure 1.19: SyR-e's GUI Motor-CAD Tab

Utilities The last tab of SyR-e's GUI (Fig. 1.20) has been developed to be helpful for this software's users. In fact, here it is possible to find further SyR-e documentation, parallel computing enabling and disabling buttons, shortcuts to export the electrical machine simulated models and it is also possible to launch the Magnet Model Manipulation section (MMM), where it is possible to calculate flux maps, currents, MTPA, MTPV and evaluate torque and power curves. For further information about the MMM unit, the reader can consult [14].

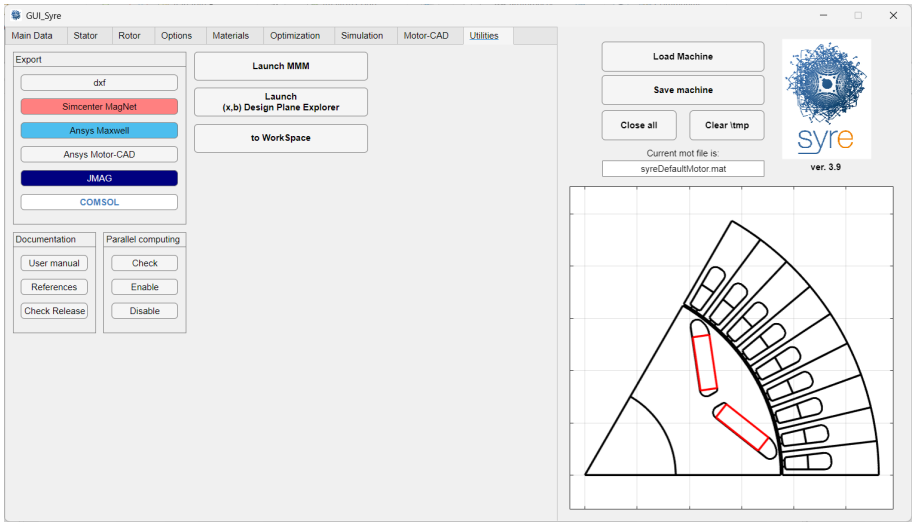


Figure 1.20: SyR-e's GUI Utilities Tab

1.5 Thesis Objectives

This thesis aims to verify the implementation of an additional electrical machine topology in SyR-e, the Electrically Excited Synchronous Machine (EESM). The preliminary analysis, carried out in the previous sections, has highlighted how the EESM motor constitutes a valid alternative option to the traction electrical machine used in the current automotive industry. In fact, the absence of rare-earth permanent magnets in this configuration allows the manufacturers to be more immune to the possible tightening of environmental policies and eventual geopolitical tensions.

At the same time, the description of the SyR-e software has demonstrated how valuable and reliable it is as a tool for designing electrical motors. This is mainly to its wide range of modifiable variables and possible analyses, and its compatibility with even more accurate external software. The immediacy and simplicity with which users can interact with SyR-e makes it a benchmark program for a rapid and cost effective design of electrical machines, although the final considerations of such task always rest to the designer.

In addition, another goal of this work is the development of an automatic MTPA angle calculation tool in SyR-e. This new feature aims to be compatible with all electrical motor topologies and possible analysis that can be performed in SyR-e.

At the present day, SyR-e already has an MTPA angle calculation functionality. However, it is performed only if the designer selects the angle "gamma" as a decision variable in the "Variables and Boundaries" section of the optimization

tool. Implementing a tool that automatically calculates the MTPA angle allows to increment the efficiency of the electrical machine design workflow and the quality of the output data. In fact, even if the designer considers that information not crucial in a first design phase, it might be necessary in a further step and having it already available allows not to launch another optimization procedure, saving precious time.

In the following lines, a brief description of this thesis' chapter structure and the topics covered will be performed.

Chapter 2 will address electrical machine optimization procedures. In particular, the current most used algorithms, specifically the one used in SyR-e, and a new methodology which could become the dominant one in the future.

Chapter 3 will analyze the steps which lead to the implementation of the EESM machine in SyR-e, including how both its geometry and specific parameters were added. This section will also talk about the validation procedures used to verify the correctness of the EESM implementation and an analysis of the results will be performed.

Chapter 4 will address the steps which lead to the automatic MTPA angle research tool addition in SyR-e. In particular, its design steps, its operational procedure and then its validation using a confrontation of two Sobol mapping analysis examples.

Last but not least, chapter 5 will perform a recap of all the arguments treated, draw the conclusions and offer some considerations about further future developments of both SyR-e software and electrical machine optimization techniques.

2. Electrical Machine Design Optimization

This section will offer an overview of the main electrical machine optimization techniques, with particular focus on the one used in SyR-e, for the reader's interest. First, general notions about EM design will be provided, then the single algorithms analyzed. For latter, a discussion about the newest methodologies in the field will be performed.

The design process of an electrical machine is an iterative procedure consisting of a sizing phase and an optimization one. If the sizing provides the machine's sizes, volumes and main parameters, its final geometry and secondary parameters are determined in the optimization phase, which is directly influenced by individual specifications such as power density, output torque, efficiency, etc. [44]

The main focus of this thesis will be on optimization techniques, consequently the sizing stage will be neglected and it will be hypothesized that those electrical machine information are already provided.

Electrical machine design optimization is an interdisciplinary subject as it involves electromagnetic, thermal, economic and structural aspects, reasons why it has been for decades a major area of research [39]. It aims at identifying the best electrical machine configuration able to satisfy multiple requirements simultaneously. As for every optimization problem, also in this case it is necessary to define first the objectives, using the objective functions *f(x)*, of the optimization problem, such as output torque, torque ripple, efficiency, power density, etc. It is also necessary to select the decision variables x of the problem and their variation ranges, or constraints. The latter will be the machine's variables whose values can be modified in order to obtain the optimal EM configuration, after having resolved the problem's set of objective functions *f(x)*.

Being an electrical machine characterized by thousands of variables and tens of equations, belonging to different fields, it is evident that the resolution of an optimization problem is not an easy-to-solve problem. In the past, designers only relied on manual calculation tools, reason why the design of an electrical machine was based on the experience of the designer in applying empirical solutions to new problems or adapting already functioning previous designs. However, with the advent of computers, the design optimization of electrical machines has simplified and sped up slightly, even though it still remains a very complex task in which the ability of the designer is crucial.

In the following section, the currently most used optimization algorithms will be

explained in further detail.

2.1 FEA-based Design Optimization

Finite Element Analysis (FEA) is the basis of the most currently used optimization techniques in the electrical machine design field. It is a crucial instrument due to its ability to model complex physical phenomena with great accuracy, allowing designers not to build expensive machines' prototypes for the first stages of design. In fact, FEA divides the domain under analysis in smaller elements, whose size depends on the mesh density level desired (Fig. 2.1). Consequently, after having defined the proper boundary conditions and the solving equations, the main problem is separated into smaller ones, which are easier and faster to solve. The solutions of the smaller problems (of the single element) are then unified constituting the global solution.

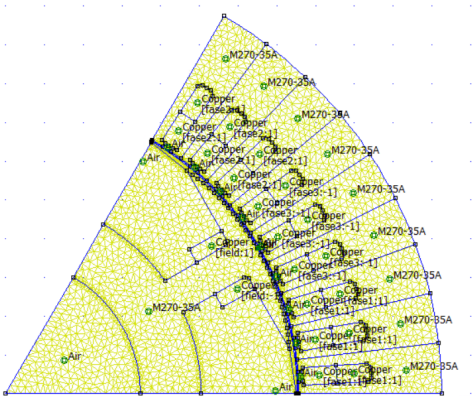


Figure 2.1: Example of Mesh Generation for an Electrical Machine

These characteristics allow FEA to have a wide range of applications as it can be used to solve electromagnetic, structural and thermal problems. Finite Element Analysis' interoperability makes it the perfect instrument of analysis for electrical machine optimization.

The first optimization algorithms adopting FEA were able to solve only single-objective optimization problems, due to the limited computational power that they had available. However, thanks to technological advancement, the computational power of computers increased and with that their capability of solving more complex problems in smaller amounts of time. This paved the way for multi-objective optimization problems to be solvable without recurring to more ingenious solutions.

Electrical machine related optimizations may be single or multi-objective optimization problems, depending on the design's specifications. For this reason, an overview of the main optimization algorithm families, based on FEA, will be performed, with distinction between single and multi-objective ones.

2.1.1 Deterministic Methods

Deterministic optimization algorithms offer consistency in results as they are able to provide the same result under the same initial conditions and parameters, due to the fact that they do not introduce random components in the analysis. In fact, they offer a systematic approach, without relying on chance.

Deterministic algorithms are mainly single-objective, which is why they are able to provide results in reduced amounts of time, compared to other optimization algorithms, due to their smaller computational weights. Even if, more recent evolutions of these algorithms made them able to perform multi-objective optimization, in this case the benefits of this type of analysis partially faint, especially in multi-physics environments with an high number of variables (like for electrical machine design) leading to other types having superior performances. [31]

Even if, the field of electrical machine design usually requires the solution of multi-objective optimization problems, this type of optimization algorithms can still be used for rapid and cost effective single-objective evaluations as they allow to reduce uncertainty in results and to make direct comparisons between different variants of the same design, highlighting the differences.

There are various families of deterministic algorithms, each one differing from another on the type of information used and on the way the solution space is explored. In the following lines, a brief description of each typology will be provided, for the completeness of the discussion:

- **Gradient-based Algorithms**

  As it can been deducted, gradient-based deterministic algorithms are based on derivatives. This type of analysis starts by defining the objective function *f(x)*, a set of problem's resolving equations, and the decision variable's starting values x_0. Then, the gradient of the objective function in x_0 is calculated $\nabla f(x_0)$ and a step variation length α is decided, whose value is either coming from a predefined rule or as a result of another optimization problem aiming at finding its best value. $\nabla f(x_0)$ value provides the direction to follow in order to minimize the objective function as, if the gradient is positive, the successive points under analysis need to be in the opposite direction. Consequently the gradient is then calculated for another point distant α from the first and this process is repeated till a minimum point is found [55], [41], [43].

  Despite being very effective, this family of algorithms have a limitation as the objective function needs to be continuous, differentiable and they might converge on local minimum points.

- **Gradient-free Algorithms**

  These optimization algorithms overcome the main limitations of the previous ones as they do not require the derivative of the objective function to be performed. Consequently, this type of analysis can be executed even if the objective function *f(x)* is not continuous or differentiable. Also in this case, it is necessary to select a decision variable's starting value x_0, in which the objective function value $f(x_0)$ is calculated, and a step variation length. Consequently, the value of the objective function is calculated also for other points chosen

following a certain logic, which depends on the specific gradient-free algorithm selected. Following, the values of the objective function obtained in those points are compared and the one having the highest/minimum value of the lot is selected as new starting point. Consequently a more detailed analysis near this point is performed by using the variation step and if no better point is found, the step is furtherly reduced. After a certain number of attempts, if no better point is found, the point having the highest/minimum value of objective function is labeled as the optimal one. [20], [54]

- **Linear Programming Algorithms**

  Linear Programming optimization algorithms are based on linearity. In fact, in this kind of technique both the objective functions and the constraints are linear. Its standard form can be written in the following way:

$$\begin{cases} min/max \quad (c * \mathbf{x}) \\ A * \mathbf{x} \leq b \end{cases} \tag{2.1}$$

  In this equation: **c** is the vector of the objective function coefficients; **x** is the decision variable's vector; **A** is the matrix of the constraints and **b** is the vector of the constraints' constants.

  These optimization algorithms are characterized by very fast output calculations even in case of systems with an high number of variables under analysis. Differently from the previous two families of algorithms that might have stopped at local minimum/maximum points, LP is able to find global minimum/maximum points of the objective function if the problem is linear.

  Linear Programming workflow operates as follows. First, it observes the constraints of the problem and defines the feasible working space. Then, the algorithm finds the vertex points the operational space and calculates the values of the objective function in those points. A confrontation of the obtained values in these extreme points and in another initial point is performed and, dependently on the results, the algorithm moves the evaluation point towards the extreme having the highest value of objective function. After, several more evaluations around the best vertex are performed, until the point with best performance is found, coinciding with the optimal point [8], [2].

The main characteristics of the algorithms presented before can be observed in Table 2.1 .

Characteristic	**Linear Programming (LP)**	**Gradient-based**	**Non-gradient-based**
Use of derivatives	No	Yes	No
Guarantee of global optimum	Yes, if linear	No, local	No
Speed	Very high	High	Slower (many evaluations)
Robustness to discontinuities	No	No	Yes
Main limitation	Only linear problems	Continuous and differentiable function	Slow

Table 2.1: Comparison of deterministic optimization methods

In the following paragraphs, it will take place the description of the most common deterministic optimization algorithms in the field of electrical machine design.

Method of Moving Asymptotes (MMA) The Method of Moving Asymptotes (MMA) is a deterministic optimization algorithm developed by Krister Svanberg in the 1980s. It has been developed for resolving non-linear constrained problems, reason why it became very common for structural and electromagnetic analysis, as it is able to provide always the same solution if fed with the same inputs, like every deterministic algorithm. As it can be deducted from its name, this algorithm exploits asymptotes (or limits) by constantly adapting them, in an iterative way, depending on the situation, in order to avoid solution instabilities and find the optimal one.

In the following lines, the workflow behind this algorithm will be explained in detail.

First, the problem needs to be defined and its general form is presented in 2.2.

$$\boldsymbol{min_{x \in R^n} f(x)} \quad s.t \quad g_i(x) \leq b, \quad x_j^{min} \leq x_j \leq x_j^{max} \quad i = 1, ..., m \quad j = 1, ..., n \tag{2.2}$$

In this equation each variable has the meaning as follows:

- $\boldsymbol{f(x)}$ is the objective function, the goal of the optimization problem,
- $\mathbf{x}$ are the decision variables, whose value can be modified in order to satisfy the objectives,
- $\boldsymbol{g_i(x)}$ is constraint function, which imposes the limitations on values of the decision variables that need to be respected for the solution to be valid,
- $\mathbf{b}$ constitutes the values of the constraints
- $\boldsymbol{x_j^{min}}$ and $\boldsymbol{x_j^{max}}$ are technological constraints on the values of the decision variables

Once the problem has been defined, the algorithm approximates the objective function and the constraint ones with surrogate functions (convex and rational), based on partial derivatives. This process is repeated for every iteration of the algorithm, till the optimal solution is found. Consequently, the objective and constraint functions can be written as in 2.3 and 2.4, exemplified for k-iteration.

$$f(x) \simeq f^{(k)}(x) = r^{(k)} + \sum_{j=1}^{n} \left(\frac{p_j^{(k)}}{U_j^{(k)} - x_j} + \frac{q_j^{(k)}}{x_j - L_j^{(k)}}\right) \tag{2.3}$$

$$g_i(x) \simeq g_i^{(k)}(x) = r_i^{(k)} + \sum_{j=1}^{n} \left(\frac{r_{ij}^{(k)}}{U_j^{(k)} - x_j} + \frac{s_{ij}^{(k)}}{x_j - L_j^{(k)}}\right) \tag{2.4}$$

The main elements present in these surrogate equations are:

- $\boldsymbol{f^{(k)}(x)}$ and $\boldsymbol{g_i^{(k)}(x)}$ are the surrogate functions approximating respectively the objective function $\mathbf{f(x)}$ and the constraint functions $\boldsymbol{g_i(x)}$,
- $\boldsymbol{p_j^{(k)}}$, $\boldsymbol{q_j^{(k)}}$ and $\boldsymbol{r_{ij}^{(k)}}$, $\boldsymbol{q_j^{(k)}}$ are coefficients obtained from the partial derivatives of respectively the objective functions $\mathbf{f(x)}$ and the constraint functions $\boldsymbol{g_i(x)}$,

- $\boldsymbol{U_j^{(k)}}$ and $\boldsymbol{L_j^{(k)}}$ are the lower and higher asymptotes of the decision variable $\boldsymbol{x_j}$ considered
- $\boldsymbol{x_j}$ is the decision variable under analysis,
- $\boldsymbol{r^{(k)}}$ and $\boldsymbol{r_i^{(k)}}$ are constant terms chosen to make the values of the original functions and the surrogate one coincide in the current valued point.

The asymptotes are one of the most crucial features of this optimization algorithm. In fact, they have multiple roles. First, they act limiting the values that the decision variables can assume. Then, they make sure that the solution obtained is stable and not oscillating. The values of the lower and upper asymptotes are at first selected from the technological limits of the decision variable $\boldsymbol{x_j}$ imposing a certain margin δ. Later, the values of the asymptotes are updated for each iteration of the algorithm using equations 2.5 and 2.6.

$$L_j^{(k+1)} = x_j^k - \gamma * (x_j^k - L_j^{(k)}) \tag{2.5}$$

$$U_j^{(k+1)} = x_j^k - \gamma * (U_j^{(k)} - x_j^k) \tag{2.6}$$

Independently on the iteration number, asymptotes have to always respect the following rule 2.7:

$$x_j^{min} \leq L_j^{(k)} \leq x_j^{(k)} \leq U_j^{(k)} \leq x_j^{max} \tag{2.7}$$

Consequently, once the surrogate functions and asymptotes have been defined, it is possible to re-write the original optimization problem as a series of simpler problems, one for each decision variable. This leads to the conclusion that this algorithm, instead of solving one big and complex optimization problem, allows to solve a series of smaller single-variable ones. So, for each decision variable this procedure is repeated in an iterative way until, the single optimal values are found. The process just explained goes on until, the overall convergence condition $||x^{(k+1)} - x^{(k)} \leq \epsilon||$ is respected, finding the optimal solution of the whole problem.

Even if this algorithm offers good performance in terms of consistency, computational time and it can be adapted to perform multi-objective optimizations (but in its standard form, it is able to perform only single-objective optimizations), it presents some critical points. First, there is no guarantee that the convergence point found is the global optimum point, as it could only be a local one. Second, it is very sensitive on the initial asymptotes selection, a wrong choice can lead to an higher number of iterations in order to reach the convergence condition. Third, its computational weight increases exponentially with the number of variables and constraints (as for electrical machine designs). Last but not least, it requires the functions to be linear as their derivatives need to be estimated. [46] [47]

For what concerns the usage of MMA for electrical machines, it can be exploited for multiple purposes. In fact, it can be used for proper optimization or to perform a sensitivity analysis, in combination with FEA. MMA is able to perform both geometrical optimizations and topology ones; however, the limitations described earlier, especially the slowness with an high number of variables prevent it from being a dominant technology of the field. Final considerations are always left to

the design team and to the level of detail required but, usually other algorithms are preferred over MMA for electrical machine design. Nevertheless, MMA can be useful in a latter design stages to perform a sensitivity analysis. In fact, its ability to provide always the same output if fed with the same inputs allows to perform slight variations on the geometry or the magnetic configuration of the machine and observe the differences with respect to a previous variant. [1], [56]

Fig. 2.2 shows a summary scheme of the MMA optimization algorithm workflow.

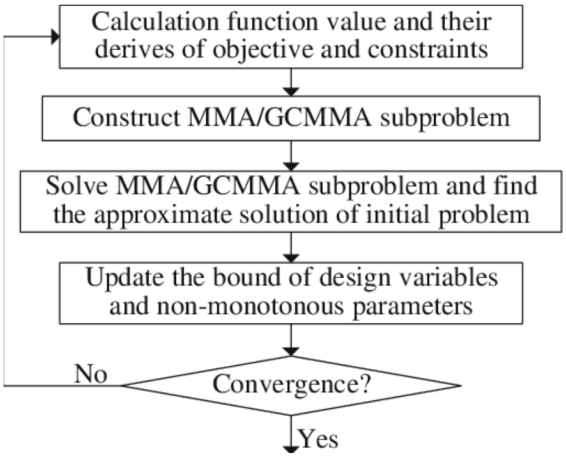


Figure 2.2: MMA Workflow Schematic Diagram

Sequential Quadratic Programming (SQP) Sequential Quadratic Programming is a deterministic optimization algorithm invented in 1970s at Stanford University which became popular in the engineering world during the 1980s.

This algorithm is used for the solution of single-objective constrained optimization problems. It is particularly common due to its accuracy in results, its rapid times of convergence and its compatibility with complex systems. [36] In the sequent lines, a brief description of the SQP algorithm's *modus operandi* will be provided.

The first step consist of defining the optimization problem, which is represented in its standard form by 2.8.

$$\boldsymbol{min_{x \in R^n} f(x)} \quad s.t \quad g(x) \leq 0, \quad h(x) = 0 \tag{2.8}$$

where: $\boldsymbol{f(x)}$ represents the objective function of the problem; $\boldsymbol{g(x)}$ is the inequality constraint vector function; $\boldsymbol{h(x)}$ is the equality constraint vector function and $\boldsymbol{x}$ represents the decision variables' vector.

Once the general problem has been defined, the differences of SQP, in comparison with MMA, come out. In fact, one of the most crucial elements of this algorithm is the definition of the Lagrangian function as it combines both the objective function and inequality and equality constraints in one element, guaranteeing that these limitations are respected. It is defined by the following equation 2.9:

$$L(x, \lambda, \mu) = f(x) + \lambda^T * g(x) + \mu^T * h(x) \tag{2.9}$$

where: λ and μ are respectively the Lagrange multipliers for inequality and equality constraints. In addition, the Lagrangian function is used to verify the

optimality, or convergence, condition. In fact, if the point (x^*, λ^*, μ^*) is the optimal one, in which the objective function is minimized, the following equation 2.10 must be verified:

$$\nabla L(x^*, \lambda^*, \mu^*) = 0 \tag{2.10}$$

After defining the Lagrangian function, SQP algorithm proceeds to approximate both the objective function and the constraint ones, process which is repeated for every iteration. The objective function is approximated using Taylor form in a second grade polynomial. Meanwhile, the constraint functions are approximated as linear. The simplified functions can be observed in 2.11, 2.12 and 2.13, referred to a generic k-iteration and x_k point:

$$f(x_k + d) \simeq f(x_k) + \nabla f(x_k)^T * d + \frac{1}{2} d^T * B_k * d \tag{2.11}$$

$$g(x_k + d) \simeq g(x_d) + J_g(x_d) * d \leq 0 \tag{2.12}$$

$$h(x_k + d) \simeq h(x_d) + J_h(x_d) * d = 0 \tag{2.13}$$

where the main elements of these equations are: $\boldsymbol{B_k}$, which is the Hessian matrix (2nd degree gradient) of the Lagrangian function in point x_k; $\boldsymbol{J_g(x_d)}$ and $\boldsymbol{J_h(x_d)}$ are in order the partial derivatives of *g(x)* and *h(x)* in point x_k; $\boldsymbol{\nabla f(x_k)}$ is the gradient of the objective function in point x_k and $\boldsymbol{d}$ is the direction in which the decision variable vector $\mathbf{x}$ is updated to reach optimality.

At this point, instead of solving the initial optimization problem, this algorithm builds a smaller quadratic problem for each iteration. By solving each quadratic problem, the direction $\boldsymbol{d_k}$ is obtained, which is used to update the point under analysis $x_{k+1} = x_k + \alpha_k * d_k$ (where α_k is the step length, chosen depending on the type of analysis). Consequently, all the elements (Hessian matrix, Lagrange multipliers, etc.) are updated to the new point under consideration and a new quadratic problem is ready to be resolved. This process continues until the global convergence condition presented in 2.10 is verified. [5]

One of the characteristics of this optimization algorithm consists in the fact that it is able to perform its iterative workflow simultaneously for each decision variable, differently from MMA. In addition, it guarantees high levels of accuracy and it is able to cope with both equality and inequality constraints thanks to the usage of the Lagrangian function and the Hessian matrix, which enables it to be used also for non-linear problems.

However, SQP presents also several limitations. First, it is able to perform only single-objective optimizations, limiting its application field. Second, the optimal solution that it finds might be only a local one and not the global optimum, as this algorithm is not able to distinct between a local and global minimum points. Third, it is also very sensitive to the starting point of evaluations, if it starts far from it then it might diverge or take an high number of iterations to reach the minimum point. Last but least, its computational weight is not negligible in case of an high number of variables under analysis.

The features od SQP optimization algorithm allow this analysis to be used in the electrical machines design field, in combination with FEA softwares. In fact, it is able to solve problems of various nature, both geometrical, structural or electromagnetic, if the objective function is well modeled, including also multiple constraints. In addition, its ability to operate with non-linear objectives and non-linear constraints make it a valid alternative solution for single-objective optimizations. This last characteristic is the one limiting the most the application of this algorithm in the EM field as in most cases during EM's design the optimization problems that need to be resolved are multi-objective.

Fig. 2.3 shows a summary of SQP optimization algorithm workflow.

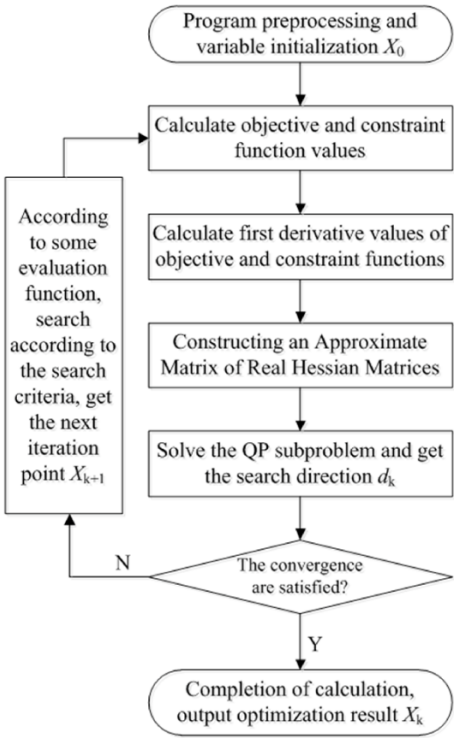


Figure 2.3: SQP Workflow Schematic Diagram

2.1.2 Metaheuristic Algorithms

Metaheuristic Algorithms (MA) are a typology of optimization analysis able to deal with non-linear multi-objective constrained problems. They are able to work in complex environments, where previously discussed deterministic algorithms struggled, without requesting not feasible computational power. Even if, also metaheuristic algorithms do not guarantee to find the global optimal point of the problem, they are built in such a way that they explore the space efficiently in order to find a solution. Once the space has been analyzed, they operate further analysis on a deeper level in those areas that were considered as promising in a first place, minimizing the chances of converging a local optimal point. Another peculiar characteristic of MAs is that multiple algorithms can be combined in hybrid solutions, due to their high level of interoperability. For these reasons, they are the most used solution for electrical machine design optimization as, these algorithms best fit the requirements

of the field, in which they have to deal with complex, constrained and non-linear systems. [49]

Metaheuristic Algorithms can be divided into three different families, whose main features will be explained in the following lines:

- **Evolutionary Algorithms (EA)**

  Evolutionary Metaheuristic Algorithms are based on the concepts of natural evolution. In fact, starting from a population of candidate solutions, which might be also randomly generated, its quality is verified using the objective function (or more than one). The solutions having the best performance are kept, as they have the highest probability of being the optimal solution or that it is in a near space, while the remaining ones are discarded. At this point, the kept solutions are either combined in "child" solutions or mutated by applying random variations, depending on the specific algorithm exploited, and another selection round takes place. The process continues until the convergence condition is verified. If it's true that EAs can be used for multi-objective constrained problems applied to complex systems, they have also limitations, like their not insignificant computational weight, connected to the high number of objective functions evaluations required (and FEA simulations) before converging. Example of evolutionary algorithm are: Genetic Algorithm (GA) and Differential Evolution (DE). [19] [17]

- **Swarm Intelligence-based Algorithms**
  Swarm Intelligence-based Algorithms are metaheuristic algorithms inspired on the human beings social interactions. In most cases, they begin by randomly generating a population of candidate solutions. These solutions are valued using the objective function of the optimization problem. In the next phase, each agent (or possible solution) is free to move, interact and modify itself on the basis of the information given in the previous evaluations, in general they move towards the solutions with the best performance. This iterative process continues until a solution satisfies the algorithm arrest conditions and is labeled as the optimal one. This family of algorithms is very easy to implement and they can be used in different field applications. However, their computational weight is not negligible as they require an high number of iterations and evaluations to converge and they are very sensible on the selection of the starting population. Typical examples of Swarm Intelligence-based Algorithms are: Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). [28]

- **Physics-based Algorithms**
  Physics-based Algorithms are metaheuristic optimization algorithms inspired by physics phenomena, not by biological ones as for the Swarm Intelligence type. The basic idea behind these algorithms is that they simulate a physics system looking for an equilibrium state corresponding to the optimal solution of the problem, which the system tends to reach autonomously. The general workflow starts by generating an initial solution, which can be random or chosen with a certain logic depending on the specific algorithm. This first solution is then modified on the basis of the physics process on which algorithm

is inspired. At this point, this population of candidate solutions is evaluated using the objective function, the best ones are selected and the remaining discarded. The accepted ones are once again modified and further evaluations are performed. The process is iteratively repeated until the limit number of iterations is reached or a solution is accredited as stable. This family of algorithms is very easy to implement, does not require huge populations to converge and is able to avoid local minimum points. However, it is not suitable for multi-objective optimizations and it is slow at finding the optimal solution. Simulated Annealing (SA) and Harmony Search (HS) are two examples of the most common algorithms of this family. [58]

- **Multi-objective Algorithms**
  Multi-objective Algorithms are metaheuristic optimization algorithms developed to solve multi-objective optimization problems. To perform such task, these algorithms produce as outputs Pareto fronts of optimal solutions, through which it is possible to select the optimal output depending on the importance of the objectives or the dominant one overall. Also for these algorithms, their workflow starts by generating an initial population of candidate solutions, decided either randomly or using certain criteria. These first possible solutions are evaluated using the objective functions and then ranked using a Pareto front. At this point, further possible solutions are generated, by either combining the previous ones or mutating them, in such a way that they differ from the first round's ones and then also these new solutions are evaluated, building another Pareto front. This process is repeated iteratively until the arrest condition is reached, either in the number of iteration or the stability of the solutions. At this point, all the Pareto fronts obtained are combined and the result is the non-dominated Pareto front which offers a set of optimal solutions for the objectives. These algorithms are able to work on very complex systems with an high number of variables and offer at the designers the chance to pick the best compromise between the objectives that need to be optimized. However, this comes at a cost as the computational weight of these algorithms is very heavy and they require a significant number of iterations to converge. The most common multi-objective algorithms is NSGA-II. [11] [29]

In the following paragraph a focus on the most popular metaheuristic optimization algorithms used in the electrical machine design field, including their operational workflow will be provided.

Genetic Algorithm (GA) Genetic Algorithm (GA) is a metaheuristic algorithm invented by John Holland in the 1970s and belonging to the Evolutionary Algorithms family, which means it is based on natural selection and genetics. It is able to perform single-objective constrained optimizations on complex and non-linear systems. Its basic concepts are identical to the ones already presented in EA section. [19]

However, a brief description of its workflow will be provided in the following lines, to add further detail to this discussion. A schematic summarizing the main phases of this optimization algorithm can be observed in Fig. 2.4.

Given a certain optimization problem, the one here considered is a minimization of the objective function $f_{obj}(x)$, but GA is able to solve also maximization problems.

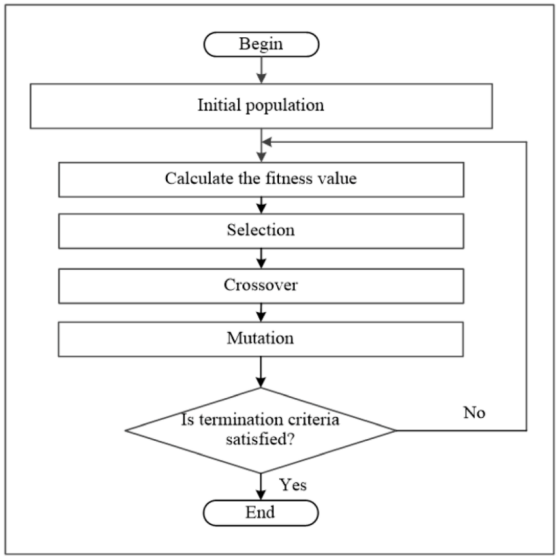


Figure 2.4: Genetic Algorithm Workflow Schematic Diagram

The standard form of the problem can be observed in 2.14:

$$min_{x \in R^n} \boldsymbol{f_{obj}(x)} \quad s.t \quad g(x) \leq 0, \quad h(x) = 0 \quad with \quad \boldsymbol{x} = [x_1, x_2, ..., x_n] \tag{2.14}$$

where: $\boldsymbol{f_{obj}(x)}$ represents the objective function of the problem; $\boldsymbol{g(x)}$ the inequality constraints; $\boldsymbol{h(x)}$ the equality constraints and $\boldsymbol{x}$ the decision variables.

Once the optimization problem has been defined, the Genetic Algorithm builds the fitness function $f_{fitness}(x)$, a function which is used internally to evaluate the quality of a possible solution, in terms of probability of it being discarded or not. This feature is crucial as optimizing the objective function is equal to optimizing the fitness one. Depending on the type of optimization problem, minimization or maximization, the fitness function assumes different forms. In fact, if the optimization problem aims at maximizing the objective then the fitness function is equal to the objective one $f_{fitness}(x) = f_{obj}(x)$. Meanwhile, in case the problem wants to minimize the objective, the fitness function assumes, in the most simple case, the form observable in 2.15.

$$f_{fitness}(x) = \frac{1}{1 + f_{obj}(x)} \tag{2.15}$$

At this point, a random initial population of n candidate solutions $\boldsymbol{P^{(0)}} = [x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}]$ is generated. Each individual, a vector of decision variables, of the initial population is evaluated first by the objective function $f_{obj}(x_i^{(0)})$ and then by the fitness function $f_{fitness}(x_i^{(0)})$. Consequently, the performances of the individuals are observed and the best ones are selected, according to the criteria chosen by the specific declination of GA selected. In general, the individuals are chosen or discarded on the base of a performance ranking or a probability one. The remaining individuals are combined, generating other "child" candidate solutions. An example of how these "child" individuals $\boldsymbol{c_i}$ are generated by the combination of two illustrative parents x^A and x^B can be observed in 2.16.

$$c_1 = [x_1^A, x_2^A, ..., x_k^A, x_{k+1}^B, ..., x_n^B] \tag{2.16}$$

After the "child" individuals have been created, to some of them a random variation is applied, in order to introduce diversity: $x_i* = x_i + \Delta x_i$, where Δx_i is a randomly generated number. At this point, after the selection, combination and mutation processes have been completed, a new population, made of the "child" individuals and the modified ones, is obtained and it is once again evaluated using both objective and fitness functions. The process is repeated iteratively until one of the arrest conditions is met, either the maximum number of generations is reached or the optimal point is found.

In addition to the main features already mentioned before, this algorithm reduces the probability of converging on a local minimum, even if it is still possible, thanks to the diversification of the population explained before. As it is able to perform single-objective optimizations on complex systems, it requires a significant amount of calculations to be performed, especially if the fitness function is non-linear.

GA application in the electrical machine design field, in combination with Finite Element Analysis tools, is quite common. In fact, the design of an electrical machine is a constrained, multi variable and complex optimization problem, which falls into the field of application of GA. In addition, this optimization algorithm is capable of resolving problems of different nature, like: geometrical (best stator or rotor topology), thermal (maximum allowed winding temperature) or even electromagnetic (maximum torque or efficiency). Consequently, GAs are currently used for geometry, topology, thermal and magnetic optimizations of every topology of electrical machines. [45]

It is important to underline, however, that most electrical machine designs nowadays require multi-objective optimization problems to be resolved, reason why evolutions, which will be analyzed later, of Genetic Algorithms have been made.

NSGA-II Non-dominated Sorting Genetic Algorithm II is a metaheuristic optimization algorithm belonging to the Multi-objective Algorithms family and introduce by Deb et al. in 2022. [29]

NSGA-II is the optimization algorithm used in the SyR-e software mentioned in the previous chapter, reason why a description of its general features and workflow will be performed in the following lines. The justifications for the choice of NSGA-II as SyR-e's optimization algorithm stay within its ability to solve multi-objective constrained non-linear problems of various nature, from electromagnetic to thermal ones. It provides as output a non-dominated Pareto front of candidate solutions, by classifying the results' fronts obtained at each step. The solutions supplied are guaranteed not to be too close with each other, thanks to its diversity mechanisms. The basics of NSGA-II have already been introduced in the paragraph dedicated to multi-objective algorithms and partially in the GA's one. However, it is interesting to describe its operation workflow in a more complete manner. A schematic diagram summarizing the main steps of the NSGA-II optimization algorithm can be observed in Fig. 2.5.

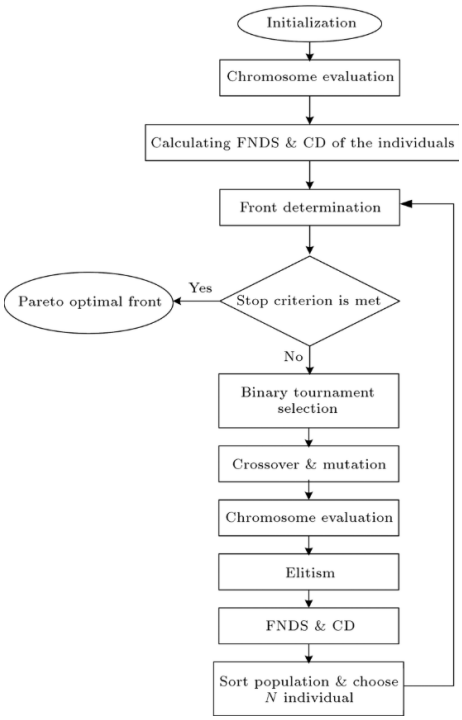


Figure 2.5: NSGA-II Algorithm Workflow Schematic Diagram

Once again, the operational process starts with the generic definition of an optimization problem, which is a multi-objective one this time. Usually NSGA-II performs the minimization of the objective functions but it is also able to maximize them in case it is requested. In fact, it is useful to underline that minimizing a negative function equals maximizing a positive one. The general equation of a typical NSGA-II problem can be observed in 2.17:

$$min\boldsymbol{F(x)} = [f_1(x), f_2(x), ..., f_k(x)] \quad s.t \quad g(x) \leq 0, \quad h(x) = 0 \quad with \quad \boldsymbol{x} = [x_1, x_2, ..., x_n] \tag{2.17}$$

where: $\boldsymbol{F(x)}$ represents the objective vector, containing the various objective functions that need to optimized ; $\boldsymbol{g(x)}$ the inequality constraints; $\boldsymbol{h(x)}$ the equality constraints and $\boldsymbol{x}$ the decision variables.

Once the problem has been defined, a first population of candidate solutions $\boldsymbol{P^{(0)}} = [\boldsymbol{x_1^{(0)}}, \boldsymbol{x_2^{(0)}}, ..., \boldsymbol{x_n^{(0)}}]$ is randomly generated. There are various ways in which $\boldsymbol{X^{(0)}}$ can be created, either randomly or using sampling methods, like Sobol or Latin HyperCube (both implemented in SyR-e). Each single member of the population is a vector containing the values of the decision variables that might solve the problem. At this point, the candidate solutions are evaluated by substituting their variables' values into the objective functions. Consequently, the Pareto dominance is calculated and the non-dominated solutions are inserted into a first Pareto front. It is important to remember that, for example, a solution $\boldsymbol{x_j}$ dominates $\boldsymbol{x_i}$ if the conditions shown in 2.18 are respected.

$$\forall j \in [1, 2, ..., J], \quad f(x_i) \leq f(x_j) \quad and \quad \exists j : f(x_i) < f(x_j) \tag{2.18}$$

Once the first Pareto front has been defined, those non-dominated solution are

removed from the initial population. Another Pareto dominance evaluation is then performed on the remaining solutions of the population, the non-dominated ones are identified and a second Pareto front is built, which is dominated compared to the first one. The process is repeated until each member of the population is inserted in a Pareto front. At this point, NSGA-II algorithm performs the so-called *crowd distancing*, which measures how much a solution is isolated, in comparison with others of the Pareto front to which it belongs; the more distant the solutions are from each other, the more diverse their front is. The distance d_i of a generic solution x_i from the others can be calculated using 2.19.

$$d_i = \sum_{m=1}^{M} \frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}} \tag{2.19}$$

Once the crowd distance has been calculated, it is time of the individuals' selection. This selection is performed using two criteria: the rank of the Pareto front (the least the better) to which they belong and their crowd distance value (the higher the better). So, two candidate solutions (from whatever Pareto front) are confronted on the basis of these two criteria and the one with the best characteristics is selected as winner and the other discarded. This routine continues until every individual of the population is analyzed, in such way the remaining solutions are both non-dominated and diversified. After this step, the "winner" solutions are combined into "child" solutions, which are spatially close to their "parents". In addition, certain "child" solutions are mutated randomly by adding a small random disturbance, in such way the space is better explored reducing the risk of premature convergence. After both the generation and the mutation processes have taken place, a "child" population has been generated. The next step consists of combining the "parent" and the "child" populations, generating a bigger one that that has the double of candidate solutions of the original ones. For this reason, it is necessary to reduce the number of candidate solutions present in this new population and in order to do so, a Pareto domination evaluation is performed. This evaluation organizes the solutions in Pareto fronts of non-dominated solutions and a new population having the size of the starting one is built by adding the Pareto fronts just constructed one at a time. If one of the fronts does not fit entirely in this new population, the solutions which enter are selected on the two criteria used before. Following, new evaluations on this newer population are conducted and the process is repeated iteratively until the convergence or the arrest condition is met. The final result given by NSGA-II algorithm is a Pareto front formed by optimal solutions, each one representing a good compromise between the different objectives. [7]

In addition to main features already mentioned in the first part, NSGA-II algorithm guarantees good diversity of the solutions, thanks to the mechanisms mentioned before, and it is very easy to implement. It is also able to analyze the space in a complete way and this leads to it being able to find the global optimal solution, not converging on local points. However, all these benefits lead to some critical points, as NSGA-II has an high computational weight, especially if it operates in combination with FEA software, due to the high number of objectives functions evaluations which leads to a slow convergence time.

As testified by its use in SyR-e, NSGA-II algorithm is one of the best and most

common solutions for the optimization of every type of electrical machines. This is due to its ability to work on multi-objective non-linear problems taking place in complex systems with an high number of variables. It is able to solve problems of various nature, like geometrical, material, thermal, topology and electromagnetic optimizations, which is an important feature when designing an electrical machine. This allows a wide spectrum of solutions to be considered, with the possibility of including additional features like cost, emissions or machine-control optimizations to the problem without compromising its execution. [44] [34] [57]

In [13] a confrontation between three different multi-objective optimization algorithms, applied on the electrical machine design, has been performed. The families of the algorithms under analysis are: Differential Evolution (DE), in which NSGA-II falls; Genetic Algorithms (GA) and Simulated Annealing (SA). Following all the preliminary considerations about the problem's context and the methodology used for this comparison, the results show how DE is the superior analysis between the three, due to the fact that it has the best performances in terms of convergence time and repeatability of the results.

2.1.3 Hybrid Algorithms

Hybrid optimization algorithms are a type of analysis characterized by the combination of two, or more, optimization techniques in order to exploit the benefits while overcoming the restrictions. This category of algorithms has been created to overcome the limitations of the ones previously described. In fact, deterministic algorithms are valid options when precision and speed are required but their drawback is that they might converge on a local optimal point instead of a global one. On the other hand, metaheuristic algorithms do not generally have this issue, as they perform a good problem space analysis, but they have significant computational weights, which lead to long-waited and less precise optimization results.

Generally, hybrid algorithms try to combine both the good problem space exploration of the metaheuristic ones and also the convergence speed and precision, characteristics of the deterministic algorithms. For these reasons, hybrid approaches constitute one of the best solutions for the analysis of multi-objective, non-linear, constrained optimization problems on complex and multi-dimensional systems, delivering both accurate and fast results. However, these hybrid solutions present several limitations. In fact, these algorithms are more complex to implement, as an higher number of operational parameters (population, mutation, crossover rate, etc.) needs to be set before starting the evaluations. In addition, the combination of the optimization algorithms needs to be developed with proper attention, as if it is done approximately, it might lead to overall worst performances of the optimization than the ones had before, as some examples where the computational weight has increased show. [50] [4]

There are several typologies of hybrid optimization algorithm which will be briefly explored in the following lines:

- **Sequential Hybrids**: these optimization techniques are composed by two different algorithms. The first one is a metaheuristic algorithm (ex: GA or

DE) which is responsible for exploring the problem's space and finding the most promising area. Then, the second algorithm, which is a deterministic one (ex: SQP or Newton-Rapson), analyzes that promising space identifying the optimal solution. The operation of the two algorithms is in series as they operate one at a time and consequently it is of easy implementation.

- **Strictly Integrated Hybrids**: these algorithms are characterized by the full implementation of a deterministic algorithm inside a metaheuristic one. They are also called "memetic" algorithms as the operation of two is not in series but simultaneous. In fact, the metaheuristic techniques performs a problem's space analysis and, periodically or randomly, some solutions are better analyzed and improved by the deterministic algorithm. These solutions proceed then to substitute the original ones. These algorithms offer high levels of efficiency and a meaningful balance between the characteristics of its original ones but the implementation of this type of analysis is very demanding.

- **Multi-Metaheuristic and Multi-Deterministic Hybrids**: these types of optimization algorithms are characterized by the combination of two (or more) algorithms belonging to the same family, for example two metaheuristic or two deterministic ones. The two algorithms can operate in a cooperative way, in which they operate simultaneously (in parallel) and they share information with each other, or in a competitive way, in which they operate one at the time and only the one with best performances continues the resolution. Depending on the families of the chosen algorithms, their benefits and drawbacks can vary according to what has been introduced before.

- **Surrogate-based Hybrids**: these optimization techniques have not insignificant differences with respect to the ones showed before. In fact, surrogate-based hybrid algorithms are characterized by a surrogate model (built via neural networks for example) of the problem, whose aim is to emulate the behavior of the system under analysis but in a simplified and less demanding way. Consequently the global metaheuristic algorithm operates on the surrogate model of the system and not on the real one, having considerable advantages on the computational point of view as this evaluation is faster than performing FEA simulations. Nevertheless, some solutions are tested using the real model of the system in order to verify the correlation between the two. These methods offer a lower computational weight, compared to other hybrid algorithms, which ultimately leads to lower convergence times. However, the surrogate model constitutes a critical part of the technology and for this reason it needs to be constantly updated for it to be accurate. Further detail will be provided into the next section.

- **Co-Evolutionary Hybrids**: these algorithms have some unique features due to the fact that the original optimization problem is divided into smaller ones. Each smaller problem is evaluated by a different optimization algorithm, even with algorithms of other technologies, and have diverse populations. In addition, the algorithms of the smaller problems can operate in a cooperative or in a competitive way. The partial candidate solutions of the smaller problems are then combined to judge the overall quality of the global problem's solution

using the objective functions. Once these evaluations are performed, the feedback is communicated back to the single algorithms of the smaller problems, leading to successive evolutions and newer solutions. This unique feature of decomposing a bigger problem into smaller ones allows these algorithms to be able to manage complex systems with an high number of variables, offering good flexibility performances. However, this strategy has also some limitations as the decomposition strategy needs to properly studied to be effective and also the overall computational weight of the optimization problem increases.

Family	**Strategy**	**Strengths**	**Weaknesses**	**Examples**
Sequential Hybrid	Global search → Local search	Simple, combines exploration and exploitation	Depends on global phase, risk of inefficiency	GA + SQP
Memetic	Local search integrated into evolutionary cycle	High solution quality, fast convergence	Computational cost, risk of premature convergence	GA + Gradient-based
Multi-Metaheuristic and Multi-Deterministic	Combination of multiple metaheuristics (or deterministic)	Good global (or local) exploration, resilient (or sensible) to local minima	Complex (or simple) to design, many parameters	GA + DE, MMA + SQP
Surrogate-Based	Metaheuristic + surrogate model	Reduces computational cost, handles FEM models	Accuracy limited by surrogate, requires updates	GA + Kriging
Co-Evolutionary	Subproblems in cooperative populations	Handles complex problems, parallelizable	High complexity, risk of interdependence	GA for geometry + PSO for materials

Table 2.2: Hybrid Optimization Algorithms Families.

The main characteristics of the Hybrid Algorithms typologies, just presented, are reported in Tab. 2.2 for a more immediate comprehension.

For what concerns the usage of Hybrid Algorithms in the electrical machine design field, these algorithms are one of the most used technology thanks to their ability of being effective in the optimizations of multi-objective non-linear problems of complex systems with a high number of variables. Hybrid algorithms are capable of solving both geometrical, material, thermal and electromagnetic optimizations which make them a very good tool for electrical machine design. In addition, they allow less FEA simulation to be performed with evident reductions of analysis' computational times. The combined usage of different technologies of algorithms allows also to obtain a better compromise between conflicting objectives of various nature, like maximum torque and minimum cost for example. Performing a literature overview, it has been possible to observe how these techniques can be effectively used on almost every electrical machine topology, like induction motors or permanent magnet synchronous machine etc., which enhance even more their validity as optimization tool. [3] [59]

2.2 AI-based Methodologies

During the last years, a real Artificial Intelligence (AI) revolution has been taking place. The term AI refers to the ability of a computer to perform tasks that would normally have been executed by human beings, for example: learning, reasoning, recognizing images, understanding language, or making decisions. The implementation of Artificial Intelligence allows for the automatization of tasks, which are performed in faster and more precise way than if done by humans, and offers support in decision making moments, thanks to its ability of managing and analyzing big quantities of data. Consequently, the reasons behind the large scale adoption of this tool are evident. In fact, it is revolutionizing most sectors of the present society and the engineering field has not been immune to these changes.

In the electrical machine design field, Artificial Intelligence is gradually gaining popularity as AI-based methods offer, in the resolution of complex multi-dimensional problems, performances and efficiencies that traditional algorithms are not able to provide. In fact, electrical machine designs require the resolution of multi-objective non-linear constrained optimization problems, in which commonly the objectives are in conflict with each other, where the classical deterministic algorithms suffer a lack of computational efficiency to perform such tasks with the desired performances.

For AI-based methodologies, a wide range of solutions is intended. In fact, the term AI refers to the capability of an optimization tool to perform the required tasks "intelligently", emulating the concept of human intelligence in a totally autonomous way. For these reasons, there are several methodologies labeled as AI-based even if they still require human intervention to be performed. In the following lines, an overview of the AI-based optimization techniques will be performed.

AI-based optimization techniques can be divided into several families:

- **Metaheuristic Algorithms**: Evolutionary Algorithms, Swarm Intelligence-based Algorithms and Physics-based Algorithms
- **Hybrid Algorithms**
- **Predictive Models** and **Machine Learning**

Both Metaheuristic Algorithms and Hybrid Algorithms have already been discussed in previous sections. Therefore, the reader is invited to consult those parts for further information. Nevertheless, some notions on the reasons behind these families of algorithms to be labeled as AI-based will be provided.

Metaheuristic algorithms can be considered as AI-based, due to the fact that these optimization techniques are founded on natural processes. For example, Genetic Algorithms, one of the biggest exponents of the Evolutionary Algorithms family, use elements like crossover and mutation that resemble natural selection and biology processes, which are intelligent as no human action is required. The same can also be said for Swarm Intelligence-based Algorithms, in which the candidate solutions interact with each other exchanging information in a way similar to what happens during human socialization processes.

For what concerns Hybrid Algorithms, being them originated from the combination of two algorithms, either of the same family or from different ones, they can also

be considered as intelligent as they inherit the best characteristics of their parents.

Predictive Models and Machine Learning : Optimization techniques based on Predictive Models and Machine Learning algorithms are what currently is intended for real AI-based methodologies.

First of all, it is important to underline that a predictive model and machine learning are separate entities. For Machine Learning (ML), it is intended a set of algorithms that enable computers to learn patterns from data without being programmed. This technique is used to build models, which might perform various actions, from a predictive function to a data-generating one. Instead, a predictive model is a mathematical pattern aiming at predicting the behavior of a system by estimating its variables' values. It might be created by a ML algorithm or using traditional methods, for example linear regression.

However, for the sake of this discussion, only predictive models generated using ML algorithms will be considered.

AI-based methodologies using Machine Learning algorithms perform optimization problems by relying not only on mathematical equations, but also by integrating models that they are able to learn from data. In fact, they aim to predict results and reduce computational times. In order for these optimization techniques to be operational, first a set of actions needs to be performed, in the sense that a "training" procedure of the machine needs to take place, for it to be able to solve the problems.

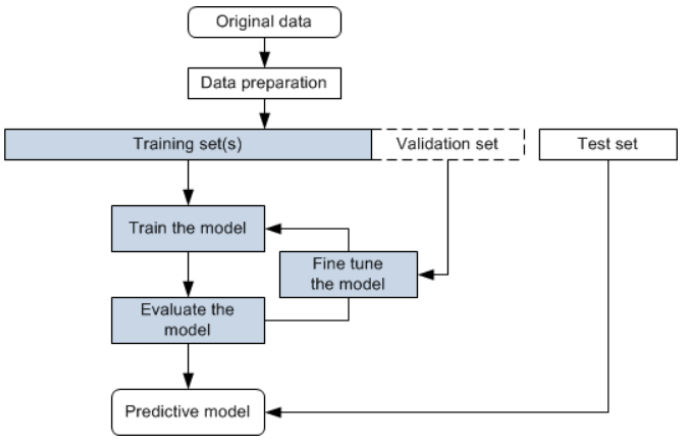


Figure 2.6: Machine Learning Training and Predictive Model Creation Schematic

The workflow behind the "training" procedure and the generation of a predictive model will briefly be explained in the following lines. However, Fig. 2.6 offers a schematic summary of the ML training procedure and predictive model creation for a more immediate understanding.

The first step of "training" consists in feeding a set of consolidated data, coming from a simulation or a previous analysis, to the machine. At this point, using a ML algorithm (for example: neural networks or Gaussian processes) the proper training takes place in the sense that the machine learns the relationship between

the inputs (decision variables) and the outputs (objectives). The output of the "training" is a predictive model with whom the machine is able to predict the behavior of the system under analysis, without having to perform long and expensive simulations. At this point, the predictive model can be inserted into the already consolidated optimization algorithms, whose working principles have already been described, helping in the evaluation of the candidate solutions proposed.

Substantially, at current day AI-based methodologies are more a kind of Hybrid Algorithms that implement Machine Learning techniques more than ad-hoc solutions. In fact, to the author's knowledge, there is not a "full AI" optimization technique, in which artificial intelligence is responsible for every step of the process from start to finish, but only hybrids of AI-based simulations and already in-use optimization algorithms. An example of Machine Learning integration into a consolidated optimization algorithm can be observed in Fig. 2.7, in which ML is combined with GA.

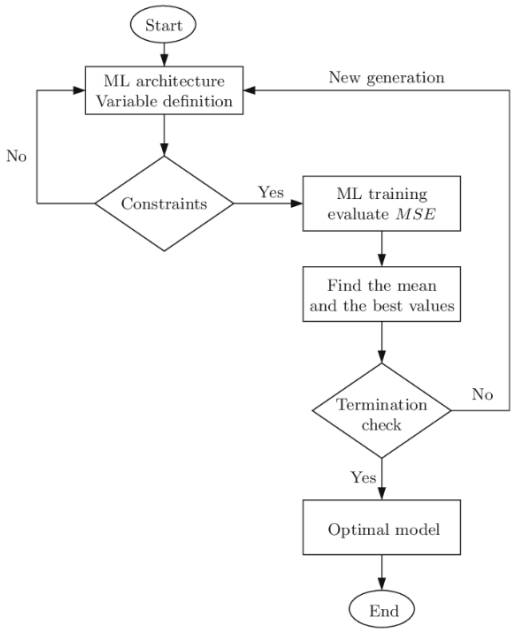


Figure 2.7: Example of Machine Learning Integration into GA

To sum up, the main features of AI-based techniques are: good solution space exploration, as they do not limit the analysis to a few points and so the chance of converge on a local minimum is reduced; introduction of randomness into the analysis , which leads to finding better solutions; feasibility of parallel computing, the task can be divided between various computer's cores and it adapts well to a combination with Finite Element Analysis tools; and significant adaptability, in terms of capability of solving problems of different nature. However, the quality of the training data is one the main limitations of AI-based techniques. In fact, particular attention must be spent on the data given for training the adaptive model, as "bad" data could mean wrong outputs and consequently an optimization problem delivering un-optimal configurations. [51] [30] [32]

Focusing on the electrical motor design scenario, once the "training" has take place and has been done using reliable data, there is no further need to use FEA simulation, except for producing the data used for "training". In fact, the machine,

once trained, is able to predict the behavior of an electric motor by just receiving its input parameters, in effect operating as a "black box". At this point, the predictive model of the electrical machine can be integrated in the algorithms currently in use for EM design (MODE, GA, exc.), with evident benefits in terms of computational time as the AI model is able to respond in milliseconds instead of minutes, like FEA tools.

Following a bibliography research, it has been noted that AI-based techniques are already in use, offering promising results. In [48] AI-driven methodologies are evaluated for electrical machines design, predictive maintenance and control optimization. The results obtained show how the implementation of AI offers significant advancements, both in fault detection and performance optimization of the electrical machines. A similar type of analysis has been performed in [33], in which AI-based hybrid solutions are used to optimize an axial flux PM machine, while performing a study on the minimum number of simulations that can be used to train the machine model, without incurring in unacceptable errors. The results show that these methodologies offer fast optimization's resolution times and result almost identical to the ones obtained with standard simulation tools. [53] quantifies the effective time savings obtained by using machine learning algorithms in electrical machine optimization, the results show how the time required for selecting the algorithm decreased by 10 times compared to before, while effective optimization time lowered by an even bigger factor. Last but not least, [37] shows how even more accurate results can be obtained using proper tools like Scientific Machine Learning (sML) and Physics-informed Machine Learning algorithms. In fact, these tools, if combined with high-fidelity FEA simulation results for training, allow to accelerate computationally expensive tasks in the development of electric machines by hundreds of times.

3. Validation of the EESM Parametrization in SyR-e

This chapter will perform an overview of the steps which leads to the Electrically Excited Synchronous Machine to be implemented in the SyR-e software. The first part will explain how it was modeled, both on a geometrical point of view and on a parameter one. Instead, the second part will talk about the validation processes, analyzing also the results obtained. Fig. 3.1 summarizes the chapter's operational plan.

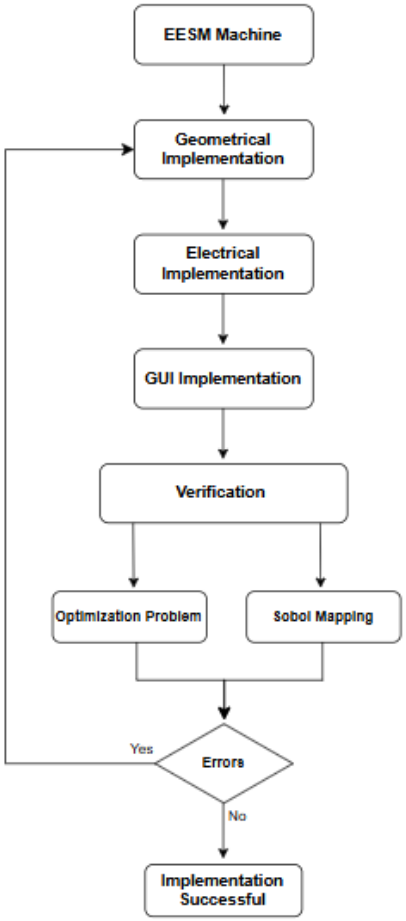


Figure 3.1: EESM Machine Implentation in SyR-e Workflow

3.1 EESM in SyR-e

Being the Electrically Excited Synchronous Machine (EESM) not present between the various electrical machine's topologies offered in SyR-e, it has been necessary to add it. In order to succeed in such task, it was required to perform several steps. In fact, it has been necessary both to create some *ad-hoc* Matlab functions, representing the EESM machine's unique features, and to modify some pre-existing ones, so that they would be compatible with this new instrument of analysis.

In the following sections, the workflow which lead to EESM machine's implementation in SyR-e, both of its geometrical and electrical parameters, will be explained in detail, accompanied by the lines of Matlab code which made it possible.

3.1.1 Geometrical Implementation

As already mentioned in during Chapter 1, EESM machines are characterized by a stator and a rotor, like any other electrical motor.

The EESM's stator is analogous to ones used for Permanent Magnet Synchronous Machines (PMSM) or Induction Motors (IM). In fact, they are characterized by a set (or more) of three-phase electrical windings inserted in the stator's slots following a certain logic, defined in a preliminary design phase. As these topologies are already inserted in SyR-e, to implement the EESM machine's stator it was only necessary to transpose the consolidated functions to this new type of analysis.

The EESM's rotor is instead characterized by a DC current winding, which is a unique feature of this machine. For this reason, it was not possible to adapt the work done previously for implementing other electrical machines' topologies in SyR-e and an *ad-hoc* process was required. As already mentioned in the EESM section, this work will focus only on the implementation of an EESM machine with a salient pole rotor, the cylindrical type is left for a future development.

The geometrical implementation of the EESM machine in SyR-e begins by defining its main geometrical rotor parameters. This information arrives directly from the Graphical User Interface (GUI), which has already been introduced.

The first file under analysis is **data0.m**, whose main function is to translate and divide the machine's dataset, inserted by the user, into smaller classes, depending on the field of the data: *geo* (geometrical), *mat* (material), *per* (performance), bounds (limits of the optimization decision variables). In order to make the discussion leaner and easier to interpret, only **data0.m** code lines regarding the addition of the main geometric parameters of the EESM machine's rotor will be shown below. Furthermore, how these were added to the GUI will be shown at a later stage.

```
186 % EESM
187 geo.lyr          = dataIn.YokeWidth;
188 geo.hpb          = dataIn.PoleBodyHeight;
189 geo.hph          = dataIn.PoleHeadHeight;
190 geo.wp           = dataIn.PoleWidth;
191 geo.wb           = dataIn.CoilWidth;
192 geo.hb           = dataIn.CoilHeight;
```

```
193 geo.thHead_deg   = dataIn.PoleRotHeadAngle;
194 geo.r_fillet     = dataIn.PoleRotHeadFillet;
```

As it can be deducted from these lines, this function receives the values corresponding to several parameters and it associates them with the corresponding field (in this case geometrical) and with a new Matlab variable. The several EESM's rotor geometrical parameters defined in this code will be briefly explained in the following lines:

- ***lyr*** = Yoke Width, quantifies how much each yoke of the rotor's pole is wide;
- ***hpb*** = Pole Body Height, identifies the length of the pole's part interconnecting the yoke and its head, but not including this last one;
- ***hph*** = Pole Head Height, it is "hpb" corresponding quantity for the pole's head, it quantifies the pole head's vertical length;
- ***wp*** = Pole Width, identifies the width of the element interconnecting the yoke with the pole's head;
- ***wb*** = Coil Width, quantifies the outer diameter of the DC winding wounded around each salient pole;
- ***hb*** = Coil Height, shows the vertical length of the DC winding wounded around each salient pole;
- $\boldsymbol{thHead_{deg}}$ = Rotor's Pole Head Angle, identifies the amplitude of the angle between the pole's body and head
- $\boldsymbol{r_{fillet}}$ = Rotor's Pole Head Fillet, quantifies the radius of the circumference's arc connecting the pole's body and head (if this value is zero then the two are connected by a right angle).

Once the main rotor's geometrical parameters have been defined, it is time to draw the EESM's rotor. In order to do so, an *ad-hoc* Matlab script was designed, whose function is identifying the rotor's main nodes (or points) so that it can be drawn in a later stage. The designated Matlab script is **nodes_rotor_EESM.m**, whose lines can be observed in the following:

```
16 function [geo,mat,temp] = nodes_rotor_EESM(geo,mat)
17
18 ri          = geo.Ar;            % internal rotor radius  (mm)
19 p           = geo.p;               % pole pairs          (-)
20 beta_pu     = geo.dalpha_pu;       % pole arc extension  (pu)
21 hry         = geo.lyr;             % rotor yoke height   (mm)
22 hp1         = geo.hpb;           % rotor pole body lengt (mm)
23 hp2         = geo.hph;           % rotor pole head length (mm)
24 wp          = geo.wp;              % rotor pole body width (mm)
25 wb          = geo.wb;              % coil width   (mm)
26 hb          = geo.hb;              % coil height   (mm)
27 MinTol      = geo.pont0;           % minimun tolerance (mm)
28 g0          = geo.g;               % air gap (mm)
```

```
re_lim      = geo.r;                % air gap radius (mm)
thHead_deg  = geo.thHead_deg;       % Head slope  (deg)
r_fillet    = geo.r_fillet;         % Pole head fillet (mm)

narcs = 15; % # of arcs for approximation of the pole head

% parameters' check
if MinTol     <  0.01     MinTol      = 0.01;    end
if p          <= 0        p           = 1;       end
if hry        <= MinTol   hry         = MinTol;  end
if hp1        <= MinTol   hp1         = MinTol;  end
if hp2        <= 2        hp2         = 2;       end
if wp         <= MinTol   wp          = MinTol;  end
if wb         <= MinTol   wb          = MinTol;  end
if hb         <= MinTol   hb          = MinTol;  end
if thHead_deg <= 0        thHead_deg  = 0;       end
if r_fillet   <= 0        r_fillet    = 0;       end
```

```
% parameters' calculation
theta = pi/p/2;       % bisector   (rad)
beta = pi/p*beta_pu;     % pole arc extension   (rad)
sat = 1/(1+p)+0.5; % variable pole body width saturation (pu)
ry = ri + hry;    % rotor's yoke external radius   (mm)
rb = ry + hp1;    % rotor's body's tooth external radius  (mm)
re = rb + hp2;    % rotors external radius  (mm)
```

The function `nodes_rotor_EESM.m` receives as inputs *geo* (geometrical), *mat* (material) variables and provides at output *geo*, *mat* and and *temp* (temporary) ones. The first step consists in associating the respective variables to all the data regarding the geometrical configuration of EESM machine's rotor (internat rotor, pole pairs, rotor yoke height, etc.). Also, the number of arcs *narcs* used to approximate the pole head is defined.

Once the main geometrical parameters have been recovered, their values are controlled. A minimum tolerance value is set and each variable's value is confronted with it. If, they are smaller than this limit, their value is imposed equal to it.

After these preliminary verifications are concluded, this function calculates the remaining geometrical rotor parameters' values, such as: rotor external radius, rotor yoke external radius, etc.

At this point, the real rotor "drawing" takes place as it is constructed point by point using several steps. It is important to underline that this process manages to obtain only the points necessary to draw one pole of the rotor, not the entire pole pair (analogous to SyR-e's GUI).

The process builds the rotor structure starting from the inside towards the outside. In fact, first the rotor's pole yoke is built then its head. A first point is defined by rotating the internal radius r_i by the angle θ, starting from a zero angle. The same is repeated at the rotor's external yoke radius r_y. Consequently, other points are found either by intersection or by rotation of the previous ones. In addition, also geometrical compatibility verifications are performed. This process can be observed

in the lines of Matlab code reported below:

```
% xp1 = ri;
% yp1 = 0;

xp2 = ri*cos(theta);
yp2 = ri*sin(theta);

xp3 = ry*cos(theta);
yp3 = ry*sin(theta);

if yp3 > wp/2
    [xp4, yp4] = intersezione_retta_circonferenza(0,0,ry,0,wp
       /2);
else
    xp4 = xp3;
    yp4 = yp3;
    wp=2*yp3;
end

xp5 = rb;
yp5 = yp4;

xp6 =rb;
yp6 = xp6*tan(beta/2);

xM = re;
yM = 0;
tau = beta/2/narcs;
for ii=1:narcs
    gamma = beta/2 - (ii-1)*tau;
    [xp7(ii),yp7(ii)] = rot_point(xM,yM,gamma);
    xp7(ii) = xp7(ii) - g0*(1/cos(gamma*p)-1)*cos(gamma);
    yp7(ii) = yp7(ii) - g0*(1/cos(gamma*p)-1)*sin(gamma);
end
```

The remaining code lines are neglected to shorten this discussion. Nevertheless, they follow a path similar to the ones mentioned before. In fact, other rotor points are identified by exploiting the geometrical information available, using tools as intersections, projections and rotations. In addition, also the geometrical footprint of the DC winding is performed, in a similar way to the one used up to this point.

At the end of this function, the geometrical parameters defined at the beginning are re-calculated, exploiting the points defined during this process, and their values are updated according to more recent evolutions.

At this point, the focus translates on the function: **build_matrix_EESM.m**. This function receives as inputs temporary and geometrical variables and provides as outputs a matrix *rotore*. This matrix contains all the crucial information about the EESM machine's rotor, both on geometrical and magnetic levels, and it can be used for further analysis, for example in FEA software or magnetic analysis.

In the following lines, the code lines and their respective comments will be pro-

vided. However, to shorten the discussion only the code's most important parts will be reported.

```
function rotore = build_matrix_EESM(temp,geo)
%%
for ii = 2:1:7
    x_var = sprintf('xp%d',ii);
    y_var = sprintf('yp%d',ii);
    eval(sprintf('%s = temp.(x_var);', x_var));
    eval(sprintf('%s = temp.(y_var);', y_var));
end
```

```
narcs = temp.narcs;
theta = temp.theta;
if geo.r_fillet ~= 0
    xc_fillet = temp.xc_fillet;
    yc_fillet = temp.yc_fillet;
end
rotore = [];
Mag = [];
% Cu <- Mag || Mag <- Cu

materialCodes;
indexEle = 1;
```

In these first lines of code, the function extracts, from *temp* (temporary folder), the points x_p, necessary to define the rotor profiles' geometry, and the points x_c (whose lines are not reported), which indicate the boundaries of inner parts of the rotor where conductors or magnets are positioned. In the following lines, some geometrical parameters are defined, like the number of arcs *narcs* necessary to define the profile. In presence of rotor's fillets, their coordinates are also extracted from the temporary folder. At this point, the matrices *rotore* and *Mag*, defining the rotor's iron and conductor geometries are initialized and then the single materials' codes are extracted.

```
rotore = [rotore
    0     0     xp3      yp3      xp4   yp4   -1 codMatFeRot
       indexEle
    xp4   yp4   xp5(1)   yp5(1)   NaN   NaN    0 codMatFeRot
       indexEle
    ];
```

```
for ii = 1:1:narcs-1
    if ii == narcs-1
        [xo(ii), yo(ii)] = calc_center_given_3pts(xp7(ii),
           yp7(ii), xp7(ii+1), yp7(ii+1), xM, yM);
    else
        [xo(ii), yo(ii)] = calc_center_given_3pts(xp7(ii),
           yp7(ii), xp7(ii+1), yp7(ii+1), xp7(ii+2), yp7(ii
           +2));
    end
    rotore = [rotore
```

```
78          xo(ii) yo(ii) xp7(ii) yp7(ii) xp7(ii+1) yp7(ii+1) -1
               codMatFeRot indexEle
79          ];
80  end
```

The function then proceeds to define the rotor's first sections by adding lines to the *rotore* matrix, including: the coordinates of the single points, the materials codes and the type of geometric element (0 for lines and 1 for arcs). In case of presence of fillets, additional information is added. Consequently, each arc of the rotor is created using an iterative process, which calculates its center using a geometrical function **calc_center_given_3pts.m**. This process is repeated iteratively for each arc and then every single element is added to *rotore* matrix.

```
89  Mag = [Mag
90      xc1 yc1 xc2(1) yc2(1) NaN NaN 0 codMatCuRot indexEle
91      ];
```

```
102 Mag = [Mag
103     xc3 yc3 xc4 yc4 NaN NaN 0 codMatCuRot indexEle
104     xc4 yc4 xc1 yc1 NaN NaN 0 codMatCuRot indexEle
105     ];
106
107 rotore = [rotore;Mag];
```

The final part of **build_matrix_EESM.m** proceeds to build the *Mag* matrix, for the conductors of the EESM rotor, in a way analogous to the one used to build *rotore*. Also in this case, the coordinates of the geometrical elements and their materials are included.

At the end of these processes, the created matrices are unified creating one unique matrix called *rotore*, which contains a complete mathematical description of the rotor's geometry in terms of lines/arcs and materials.

All the information obtained by running the functions **nodes_rotor_EESM.m** and **build_matrix_EESM.m** is then used in the function **drawPole.m**. This feature exploits the processes ran previously to build a complete model of the rotor in matrix form, both on a geometrical point of view and on a material one, which can be used by the FEMM software for analysis. In fact, it defines the materials used in each part of the rotor, and applies rotation and symmetry tools, delivering complete information ready to be used in the FEA simulations.

Consequently, the inputs of this function are *geo* (geometrical) and *mat* (material) data and *fem* (FEA) parameters, like mesh density etc. Meanwhile ,the outputs are: *geo*, containing information on the rotor's geometry and the position of each material; *mat* and *temp* (temporary), which stores information that might be used in successive steps of analysis.

In the following lines the most important parts of the function and the ones regarding the EESM machine will be reported.

```
15  function [geo,temp,mat] = drawPole(geo,mat,fem)
16  %
```

```
% This function draw a single standard pole, in zero position (as drawn in
% SyR-e), with labels matrix (BLKLABELSrot),
```

```
% 1) Find the design points (nodes_rotor_xxx), build rotor matrix
...
case 'EESM'
        [geo,mat,temp] = nodes_rotor_EESM(geo,mat);
        rotor = build_matrix_EESM(temp,geo);
end
```

The first step of this function consists in a confrontation between the rotor type, selected by the user, and the various types that SyR-e offers. Once, the type of rotor under analysis is identified, in this case the EESM's one, the function proceeds to build the rotor's matrix finding the design points and materials, exploiting the functions **nodes_rotor_EESM.m** and **build_matrix_EESM.m** previously described.

```
%2) mirror the half pole
rotNeg=rotor;
rotNeg(:,[2 4 6 7]) = -rotor(:,[2 4 6 7]);
rotNeg(:,9) = rotNeg(:,9)+max(rotor(:,9));
rotor=[rotor;rotNeg];

%3) find the centers of all blocks (PMs and air)
BarCenter = defineBlockCenters(temp,fem,geo);

% Rotate rotor in zero position (rotor matrix and labels)

rotor = rotateMatrix(rotor,90/geo.p*pi/180);
rotor = checkPlotMatrix(rotor,1e-9);

% Rotate block labels selection points
[xtemp,ytemp] = rot_point(BarCenter(:,1),BarCenter(:,2),90/geo.p*pi/180);
BarCenter     = [xtemp,ytemp,BarCenter(:,3:end)];
clear xtemp ytemp;

% Magnetization direction rotation
xtemp=cos(atan2(BarCenter(:,7),BarCenter(:,6))+(pi/2/geo.p-eps));
ytemp=sin(atan2(BarCenter(:,7),BarCenter(:,6))+(pi/2/geo.p-eps));
BarCenter(:,6)=xtemp;
BarCenter(:,7)=ytemp;
clear xtemp ytemp;

%%% OUTPUT DATA %%%

%%% Block centers %%%
geo.BLKLABELS.rotore.xy = BarCenter;
geo.rotor = rotor;
```

Once the rotor's geometrical information has been retrieved, the next steps consists of mirroring the part just built so that the full salient poles of the rotor are represented. After that, the central points of each element are identified, so that the respective material of each member of the rotor can be assigned properly. The rotor and all the identified points are then rotated in zero position of the analysis. Consequently, the magnetization directions are also rotated, which is essential for having accurate simulations of the magnetic fluxes crossing the salient poles of the rotor.

Finally, the outputs of the function are defined, which are: *geo.rotor*, a geometric element that contains the rotor's matrix defining each geometrical element and the single materials; *geo.BLKLABELS.rotore.xy*, a matrix containing the coordinates of the central points of each element of the rotor for material assignment; *temp* and *mat*, which have been already discussed.

The last part of the EESM's rotor geometric implementation in SyR-e is composed by the **ROTmatr.m** function. It is crucial as it is responsible for the machine's rotor drawing in FEMM (Finite Element Method Magnetics), the FEA software used by SyR-e to perform simulations and analysis. Also for this function, its inputs are *geo* (geometrical) and *mat* (material) data and *fem* (FEA) parameters. Meanwhile, its outputs are: *rotor*, a matrix describing the rotor's geometry using lines and arcs for it to be drawn in FEMM; *BLKLABELSrot*, another matrix containing the boundary conditions and the coordinates of each element; *geo* and *mat*, which contain information about the rotor's final geometry and materials.

Being this function very long and complex, only its initialization and the parts regarding the EESM machine will be reported in the following lines.

```
15 function [rotor,BLKLABELSrot,geo,mat] = ROTmatr(geo,fem,mat)
16
17 % Rotor construction.
18 % rotor:                          one row per FEMM line or arc
19 % BLKLABELSrot.xy:           center points of FEMM blocks
20 % BLKLABELSrot.boundary:    one row per FEMM bounday
      condition
21 % BLKLABELSrot.BarName:     names of flux barrier blocks
22
23 p        = geo.p;
24 ps       = geo.ps;
25 th_FBS   = geo.th_FBS;
26 r        = geo.r;
27 % Ar       = geo.Ar;
28 lm       = geo.hc_pu*geo.g;
29 RotType  = geo.RotType;
30 matFBS   = mat;
31 hs       = geo.hs;
32 g0       = geo.g;
```

```
38 % draw a single, straight pole or a rotor slot
39 if strcmp(geo.RotType,'IM')
40     [geo,~,mat] = drawBar(geo,mat,fem);
```

```
41 else
42     [geo,temp,mat] = drawPole(geo,mat,fem);
43 end
44
45 Ar = geo.Ar;
46
47 % initialize the matrix for geometry and labels for the total
       rotor (ps poles)
48 rotor0 = geo.rotor;
49 xy0    = geo.BLKLABELS.rotore.xy;
50
51 [nRow_mat,nCol_mat] = size(rotor0);
52 [nRow_xy,nCol_xy]   = size(xy0);
53
54 rotor     = zeros(ps*nRow_mat,nCol_mat);
55 BarCenter = zeros(ps*nRow_xy,nCol_xy);
```

These first lines of code initialize the function. In fact, geometrical information is retrieved and assigned to the variables that will be later used. In addition, depending on the machine's rotor typology, a first pole or slot is drawn. Lastly, the matrices for the rotor geometry and the boundary conditions are generally created.

```
74  else
75         % replicate the pole (ps-1) times (geometry and
               labels)
76         for ii=1:ps
77             % geometry
78             rotorTmp = rotateMatrix(rotor0,(ii-1)*pi/p);
79             indexEle = rotorTmp(:,9);
80             rotorTmp(:,9) = indexEle+max(indexEle)*(ii-1);
81             rotor(1+nRow_mat*(ii-1):nRow_mat*ii,:) = rotorTmp
                   ;
82
83             % labels
84             if strcmp(RotType,'EESM')
85                 [xtemp,ytemp] = rot_point(xy0(:,1),xy0(:,2),(
                       ii-1)*pi/p);
86                 magdir = atan2(xy0(:,7),xy0(:,6))+((ii-1)*pi/
                       p-eps)+(cos((ii-2)*pi)+1)/2*pi;
87                 if ~mod(ii,2)
88                     BarCenter(1+nRow_xy*(ii-1):nRow_xy*ii,:)
                           = [xtemp, ytemp, xy0(:,3:5), cos(
                           magdir), sin(magdir), -xy0(:,8)];
89                 else
90                     BarCenter(1+nRow_xy*(ii-1):nRow_xy*ii,:)
                           = [xtemp, ytemp, xy0(:,3:5), cos(
                           magdir), sin(magdir), xy0(:,8)];
91                 end
92                 if ii<ps
93                     BarCenter(4*ii-1,:) = NaN*ones(1,nCol_xy)
                           ;
```

```
94                  else
95                      if ps == 2*p
96                          BarCenter(end-1,:) = NaN*ones(1,
                               nCol_xy);
97                      end
98                  end
99              else
100                 [xtemp,ytemp] = rot_point(xy0(:,1),xy0(:,2),(
                       ii-1)*pi/p);
101                 magdir = atan2(xy0(:,7),xy0(:,6))+((ii-1)*pi/
                       p-eps)+(cos((ii-2)*pi)+1)/2*pi;
102                 BarCenter(1+nRow_xy*(ii-1):nRow_xy*ii,:) = [
                       xtemp, ytemp, xy0(:,3:5), cos(magdir), sin
                       (magdir), xy0(:,8)];
103             end
104         end
```

In these lines, the first step is to replicate the pole drawn previously *ps-1* times, in terms of both geometry and coordinates for the boundary conditions. In the EESM machine specific part, an additional calculation is performed. In fact, the magnetic direction, in terms of orientation of the rotor's poles, is estimated. These processes are performed using geometrical tools like rotations and arc-tangents, aiming at building the rotor's complete structure.

In the following lines, analogous steps are performed also for the other rotor typologies implemented in SyR-e. After these processes, additional rotor sections are built, like the the connecting sections between the rotor's iron and the airgap or the shaft. These steps are performed, once again, recurring to the geometrical tools already mentioned. In this procedure, a distinction between a partial machine analysis (of just one pole) and a full machine one is performed. To shorten the discussion, only the lines regarding the full machine analysis will be reported.

```
255 % full machine
256     xre2 = re;
257     yre2 = 0;
258     xre3 = -re;
259     yre3 = 0;
260
261     xra2 = Ar;
262     yra2 = 0;
263     xra3 = -Ar;
264     yra3 = 0;
265     if strcmp(RotType,'EESM')
266         rotor = [rotor
267             0 0 xra2 yra2 xra3 yra3 1 codMatShaft indexEle+1
268             0 0 xra3 yra3 xra2 yra2 1 codMatShaft indexEle+1
269             ];
270         for ii = 1:1:2*p
271             [xA,yA] = rot_point(temp.xp7(1),temp.yp7(1),pi/p
                    /2+ii*pi/p);
272             [xB,yB] = rot_point(temp.xp7(1),temp.yp7(1),pi/p
```

```
                /2+ii*pi/p+(1-geo.dalpha_pu)*pi/p);
            rotor = [rotor
                xA yA xB yB NaN NaN 0 codMatAirRot indexEle
                    +1+ii
                ];
        end
    else
```

```
% add label for rotor iron
    elseif strcmp(RotType,'EESM')
    [xtemp,ytemp] = rot_point(mean([Ar Ar+geo.lyr]),0,pi/2/p)
        ;
    BarCenter = [BarCenter; xtemp ytemp codMatFeRot,fem.res
        ,1,NaN,NaN,NaN];
```

```
    elseif strcmp(geo.RotType,'EESM')
    [xRotBound1,yRotBound1] = rot_point(mean([Ar,Ar+geo.lyr])
        ,0,-90/p*pi/180);
    [xRotBound2,yRotBound2] = rot_point(mean([Ar,Ar+geo.lyr])
        ,0,(ps-1/2)*180/p*pi/180);
    [xRotAirBound1,yRotAirBound1] = rot_point(mean([Ar+geo.
        lyr,r-hs]),0,-90/p*pi/180);
    [xRotAirBound2,yRotAirBound2] = rot_point(mean([Ar+geo.
        lyr,r-hs]),0,(ps-1/2)*180/p*pi/180);
    [xSleeveBound1,ySleeveBound1] = rot_point(mean([r-hs,r])
        ,0,-90/p*pi/180);
    [xSleeveBound2,ySleeveBound2] = rot_point(mean([r-hs,r])
        ,0,(ps-1/2)*180/p*pi/180);
```

In these last two code blocks reported, first a specific EESM machine rotor's iron label is added, identifying it as material block. Then, also the other elements of the rotor are labeled but, the code lines are neglected. Following, the EESM specific boundary conditions, including the borders between airgap, iron and shaft, are defined. It is important to underline that these represent only the EESM machine proprietary boundary conditions but, also limits common to all machines' typologies exists. However, these common boundary conditions are not reported here.

```
    %%% OUTPUT DATA %%%
%%%%%%%%%%%%%%%%%%%%

geo.rotor = rotor;

%%% Block centers %%%
BLKLABELSrot.xy      =   BarCenter;
BLKLABELSrot.BarName =    BarName';

% Boundaries %%%
BLKLABELSrot.boundary = [
    xShaftBound1   yShaftBound1   codBound_periodic;
    xShaftBound2   yShaftBound2   codBound_periodic;
    xRotBound1     yRotBound1     codBound_periodic;
```

```
402     xRotBound2    yRotBound2    codBound_periodic;
403     xRotAirBound1 yRotAirBound1 codBound_periodic;
404     xRotAirBound2 yRotAirBound2 codBound_periodic;
405     xSleeveBound1 ySleeveBound1 codBound_periodic;
406     xSleeveBound2 ySleeveBound2 codBound_periodic;
407     ];
408
409 BLKLABELSrot.boundary = BLKLABELSrot.boundary(~isnan(
    BLKLABELSrot.boundary(:,1)),:);    %remove NaN rows
410
411 % Rotate boundary selection points
412 [xtemp,ytemp]=rot_point(BLKLABELSrot.boundary(:,1),
    BLKLABELSrot.boundary(:,2),90/p*pi/180);
413 BLKLABELSrot.boundary=[xtemp,ytemp,BLKLABELSrot.boundary(:,3:
    end)];
414 %
```

The last lines of the **ROTmatr.m** function are dedicated to define its outputs. In fact, the *rotor* matrix, defining the machine's rotor geometry, and the *BLKLABELSrot.boundary* one, containing all the information about the single block of the rotor, materials and boundary conditions, are created.

3.1.2 Electrical Parameters Implementation

As it has been underlined in previous sections, the Electrically Excited Synchronous Machine (EESM) has some unique features, not present in the other machines' topologies implemented in SyR-e. For this reason, a series of new electrical parameters had to be added for this type of machine to be correctly implemented.

The presence of the rotor DC winding has consequences in terms of additional electrical parameters that need to be defined for an effective implementation of the EESM machine. In fact, this winding is crossed by a certain "field" current i_f, which is responsible for generating the excitation field, that needs to be modeled in the machine's SyR-e transposition. Consequently, both a field current i_f and a rotor current density J_f need to be added as electrical variables. In addition, the DC winding is characterized by a certain resistance value, called field resistance R_f.

In order to insert these parameters in SyR-e, once again, the **data0.m** function has been used. Following the user selection of the amplitude of these entities in the GUI section, these parameters, which are then part of a dataset, are translated by this function and assigned to the respective field of interest, either *geo* (geometrical), *mat* (material), *per* (performance), *bounds* (boundary conditions) or *objs* (objectives).

The lines of code regarding the implementation of these EESM-original parameters will be reported below.

```
15     function [bounds, objs, geo, per, mat] = data0(dataIn)
```

```
47 per.kj    = dataIn.ThermalLoadKj;
```

```
per.i0   = dataIn.RatedCurrent;
per.Rs   = dataIn.Rs;
per.Lend = dataIn.Lend;
per.J    = dataIn.CurrentDensity;
per.Jf   = dataIn.RotorCurrentDensity;
per.if   = dataIn.FieldCurrent;                % rotor current
    for EESM
per.Rf   = dataIn.Rf;                         % field circuit
   resistance
per.if0  = dataIn.RatedFieldCurrent;
per.JfPU = dataIn.FieldStatorCurrentDensityRatio;
```

```
geo.win.kcuf = dataIn.RotorConductorFillingFactor;
geo.win.Nf   = dataIn.FieldTurns;   % turns in series per
   pole
```

In these lines of code, it can be observed how the various EESM specific parameters, present in the dataset, have been divided between the geometrical field and the performance one. The electrical parameters added, inserted within the performance data of the machine, are: rotor current density J_f; field current i_f; field resistance R_f; rated field current i_{f0}, which is the nominal value of reference for the field current, and the ratio between the field and stator current densities J_{fPU}. Instead, for what concerns the additional geometrical entities, these are related to the DC winding. In fact, the two implemented parameters are respectively: rotor conductor filling factor k_{cu}^f, which quantifies how much the conductors' area is actually filled, and number of turns of the field winding N_f.

At this point, the proper implementation of the EESM machine is SyR-e is almost complete. However, some more steps need to be executed, adapting pre-existing functions.

The first function under analysis is **FEMMfitness.m**. This function is one of the most crucial ones of the SyR-e software. In fact, it is responsible for running the FEA simulations, in FEMM, by using the data coming either from a pre-existing machine or from the optimization setup decided by the user in the GUI. The inputs of this function are several sets of data, in specific: *RQ*, which is the set of parameters whose value needs to be optimized; *geo*, geometrical information about the machine under analysis; *per*, contains the performance variables of the machine; *mat*, expresses the materials of the elements present; $eval_t ype$, which defines the type of analysis that needs to be performed (either an optimization of a new machine or of an existing one, etc.), and *filenameIn*, which is a vector containing the name of the input file of this function. **FEMMfitness.m**'s outputs are instead: *cost*, a matrix containing the values of the objectives after the optimization has been performed; *out*, a matrix containing the values of the performance variables after the analysis has been performed; *pathname*, which contains the path necessary to retrieve the output information of the process; *geo* and *mat*, whose meaning has already been explained.

As **FEMMfitness.m** Matlab code is very long and complex, only a portion of it will be reported. This part is the most relevant one for the EESM machine implementation, as it introduces two *ad-hoc* functions that have been developed appositely to add this machine to SyR-e.

```
10      function [cost,geo,mat,out,pathname] = FEMMfitness(RQ,geo
            ,per,mat,eval_type,filenameIn)
11      ...
12      if ~isempty(RQ)
13
14      % MODE optimization (RQ geometry)
15      RQ=RQ';
16      geo.pathname=pwd();
17
18      if strcmp(eval_type,'MO_OA')
19  %           RQ % debug .. when syre crashes it is useful to
        have visibility of last RQ
20      end
21
22      [geo,gamma,mat] = interpretRQ(RQ,geo,mat);
23      per.gamma=gamma;
24
25      [geo,mat] = draw_motor_in_FEMM(geo,mat,pathname,filename)
            ;
26
27      [~,geo] = calc_endTurnLength(geo);
28      [~,geo] = calc_endTurnFieldLength(geo);
29
30      %       flag_OptCurrConst = 1;
31      switch per.flag_OptCurrConst
32          case 0 % constant thermal loading
33              per.loss = NaN;
34              per.J    = NaN;
35          case 1 % constant current density
36              per.kj   = NaN;
37              per.Loss = NaN;
38          case 2 % constant current
39              per.kj   = NaN;
40              per.Loss = NaN;
41              per.J    = NaN;
42      end
43      per = calc_i0(geo,per,mat);
44      % warning('Define the ratio between stator and rotor
            current density')
45      per.Jf = per.J*per.JfPU;
46      per = calc_if(geo,per,mat);
47      per.if = per.if0;
48
49      %       if any(strcmp(geo.OBJnames,'Fdq0'))
50      %           per0 = per;
51      %           per0.overload = 0;
52      %           per0.gamma = 0;
53      %           per0.nsim_singt = 1;
54      %       end
55
56  else
```

```
57      % post proc or FEMM simulation (existing geometry)
58      copyfile(filenameIn,[pathname filename]); % copy .fem in
            the temporary folder
59  end
```

In these lines, the function proceeds first to check the type of analysis that needs to be performed, if it is an optimization or another kind. Then, it withdraws geometrical and material data, in combination with the analysis' objectives and decision variables. After this step, it proceeds to draw the electrical machine in the FEMM software. At this point, this function estimates the windings' end turns length. If this procedure for the stator AC windings is already consolidated, the same can not be said for the DC rotor winding. In fact, it has been necessary to develop a proper function, called **calc_endTurnFieldLenght.m**, in order to estimate the length of the DC winding end turns. The code of this inner function can be observed below.

```
15      function [lendf,geo] = calc_endTurnFieldLength(geo)
16  % computation of the field coil end-winding length
17
18  wp = geo.wp;
19  wb = geo.wb;
20
21  lendf = pi*(wp/2+wb/2);
22
23  geo.lendf = lendf;
```

This very simple function is able to calculate the field winding end turn length, in average value, by exploiting the rotor pole body width w_p and the coil width w_b. The DC winding end turn length is estimated by using the formula of the circumference perimeter, in which the diameter is composed by the sum of half w_p and w_b.

After this simple estimation, **FEMMfitness.m** proceeds first to define if the optimization is performed imposing one between the thermal loading, current density or the current as constant. Then, this function estimates the values of various electrical parameters. The first one to be estimated is the nominal stator current i_0, which is obtained using a proper function already implemented for other topologies. Then, the field current density J_f is calculated, by exploiting the product between the stator current density J and the ration between the two J_{fPU}. At this point, it is time to estimate the field currenti_f, flowing into the rotor DC winding. To estimate this entity, an *ad-hoc* function, named **calc_if.m**, has been developed and its code lines can be observed in the sequent lines.

```
16  function [per] = calc_if(geo,per,mat)
17
18  Jf  = per.Jf;
19  if0 = per.if0;
20
21  l       = geo.l/1e3;            % stack length [m]
22  lendf   = geo.lendf/1e3;        % end-winding length [m]
23  Nf      = geo.win.Nf;           % number of turns in series per
        pole
24  if isfield(geo,'Acoilf')
```

```
25      Acoilf = geo.Acoilf/1e6;       % slot area [m^2]
26  else
27      Acoilf = NaN;
28  end
29
30  kcuf    = geo.win.kcuf;          % slot filling factor
31  p       = geo.p;                % pole pairs number
32
33  tempCu = per.tempcu;          % target copper temperature [C]
34
35  if exist('mat','var')
36      ro0 = 1/mat.BarCond.sigma;
37      alphaCond = mat.BarCond.alpha;
38      rocu = ro0*(1+alphaCond*(tempCu-20));
39  else
40      rocu = (1.7241e-08)*(234.5+tempCu)/(234.5+20);
41      warning('Copper rotor winding computation');
42  end
43
44  flag = 1;
45  if ~isnan(Jf)
46      if0 = Jf*(Acoilf*1e6*kcuf)/Nf;
47  elseif ~isnan(if0)
48      Jf = if0/(Acoilf*1e6*kcuf)*Nf;
49      flag = 0;
50  else
51      warning('Wrong field currrent input!')
52  end
53
54  Rf = rocu*(1+lendf)/(Acoilf*kcuf)*Nf^2*(2*p);
55
56  per.Rf = Rf;
57  per.if0 = if0;
58  per.Jf = Jf;
```

This inner function is able to estimate both the field current i_f and the field winding resistance R_f. It is able to do so by exploiting both the geometrical and performance data available and by consequently applying the right formulas depending on the single case scenario. The procedure used can be read in the lines code, which are of immediate and simple interpretation.

At this point, **FEMMfitness.m** proceeds then to perform various analysis belonging to different fields, depending on the type of examination required and on the objectives selected in the setup phase. Both mechanical and electromagnetic simulations, of the machine under analysis, can be performed. For this specific discussion, only the electromagnetic simulations will be considered. These simulations are performed, after having drawn the machine in FEMM, by running the function **simulate_xdeg.m**. This function is able to emulate the electromagnetic behavior of an electrical machine and consequently, it offers as output all the variables' values of main interest, which will constitute the *out* and *cost* vectors, like: torque, currents (on *d*-*q* axis), fluxes, torque ripple, etc. The importance of this function is then evi-

dent for the overall success of the optimization process. It is important to underline that **simulate_xdeg.m** is completely compatible with the EESM machine, even if some adaptation, here neglected to shorten the discussion, were necessary.

3.1.3 GUI Implementation

As it has been previously discussed, the Graphical User Interface is a very convenient SyR-e tool. In fact, it allows for an immediate and simplified comprehension of each electrical machine parameter, to choose rapidly the analysis to perform and to select additional fields of study, depending on the single case scenario. In a few words, the GUI simplifies the designer's work, causing significant savings of time compared to SyR-e usage without this tool.

Every electrical machine topology supported by SyR-e is implemented in the GUI and for the EESM machine, no exception was made. In order to do so, several pre-existing functions have been modified to efficiently support the EESM machine.

The first function that was adapted for implementing the EESM machine is **GUI_APP_SetParameters.m**. It is responsible for managing the GUI's interface, which allows to insert the electrical machine's parameters. This function is the one behind the benefits given by the usage of the GUI. In fact, exploiting its features, the user is able to modify the values of the various parameters by just clicking and writing in the desired box, instead of using SyR-e's proper code to apply modifications.

In the lines below, only some of the lines regarding the implementation of the EESM machine are reported, due to the fact that this function is very complex and extensive.

```
15 function GUI_APP_SetParameters(app)
16
17 dataSet = app.dataSet;
18
19 set(app.currentMotFileName,'Value',dataSet.currentfilename);
20
21 % Main data panel
22 set(app.PolePairsEdit,'Enable','on','Value',num2str(dataSet.
       NumOfPolePairs));
23 set(app.NumOfSlotsEdit,'Enable','on','Value',num2str(dataSet.
       NumOfSlots));
24 set(app.NumberofstatorslotsEditField,'Enable','on','Value',
       int2str(dataSet.NumOfStatorSlots));
25 set(app.Numberof3phasesetsEditField,'Enable','on','Value',
       int2str(dataSet.Num3PhaseCircuit));
26 set(app.GapThiEdit,'Enable','on','Value',num2str(dataSet.
       AirGapThickness));
27 set(app.StatorOuterRadEdit,'Enable','on','Value',num2str(
       dataSet.StatorOuterRadius));
28 set(app.AirGapRadiusEdit,'Enable','on','Value',num2str(
       dataSet.AirGapRadius));
29 set(app.ShaftRadEdit,'Enable','on','Value',num2str(dataSet.
       ShaftRadius));
```

```
30 set(app.StackLenghtEdit,'Enable','on','Value',num2str(dataSet
       .StackLength));
31 set(app.TypeOfRotorList,'Enable','on','Value',dataSet.
       TypeOfRotor);
32 ...
33 elseif (strcmp(dataSet.TypeOfRotor,'IM')||strcmp(dataSet.
       TypeOfRotor,'EESM'))
34     set(app.NumberOfLayersEdit,'Enable','off','Value',num2str
           (dataSet.NumOfLayers));
35     set(app.AlphapuEdit,'Enable','off','Value',mat2str(
           dataSet.ALPHApu));
36     set(app.AlphadegreeEdit,'Enable','off','Editable','off','
           Value',mat2str(dataSet.ALPHAdeg));
37     set(app.hcpuEdit,'Enable','off','Value',mat2str(dataSet.
           HCpu));
38     set(app.hcmmEdit,'Enable','off','Editable','on','Value',
           mat2str(dataSet.HCmm));
39     set(app.DxEdit,'Enable','off','Value',mat2str(dataSet.
           DepthOfBarrier));
40     set(app.BetaEdit,'Enable','off','Value',mat2str(dataSet.
           betaPMshape));
41     set(app.CentralBarriersShrinkEdit,'Enable','off','Value',
           mat2str(dataSet.CentralShrink));
42     set(app.NarrowFactorEdit,'Enable','off','Value',mat2str(
           dataSet.NarrowFactor));
43     set(app.RadShiftInnerEdit,'Enable','off','Value',mat2str(
           dataSet.RadShiftInner));
44     set(app.ThetaFBSEdit,'Enable','off');
45     set(app.TanRibEdit,'Enable','off','Value',mat2str(dataSet
           .TanRibEdit));
46     set(app.RadRibEdit,'Enable','off','Value',mat2str(dataSet
           .RadRibEdit));
47     set(app.SplitRibsEditField,'Enable','off','Value','0');
48     set(app.RadRibCheck,'Enable','off','Value',dataSet.
           RadRibCheck);
49     ...
50 % EESM motor panel
51 if strcmp(dataSet.TypeOfRotor,'EESM')
52     % set(app.PoleAnglepuEditField,'Enable','on','Editable','
           on','Value',num2str(dataSet.PoleAnglepu));
53     set(app.PoleAnglepuEditField,'Enable','on','Editable','on
           ','Value',num2str(dataSet.ALPHApu));
54     set(app.YokewidthmmEditField,'Enable','on','Editable','on
           ','Value',num2str(dataSet.YokeWidth));
55     set(app.PolebodyheightmmEditField,'Enable','on','Editable
           ','on','Value',num2str(dataSet.PoleBodyHeight));
56     set(app.PoleheadheightmmEditField,'Enable','on','Editable
           ','off','Value',num2str(dataSet.PoleHeadHeight));
57     set(app.PolewidthmmEditField,'Enable','on','Editable','on
           ','Value',num2str(dataSet.PoleWidth));
58     set(app.CoilwidthmmEditField,'Enable','on','Editable','on
```

```
        ','Value',num2str(dataSet.CoilWidth));
59      set(app.CoilheightmmEditField,'Enable','on','Editable','
        on','Value',num2str(dataSet.CoilHeight));
60      set(app.PoleHeadFilletmmEditField,'Enable','on','Editable
        ','on','Value',num2str(dataSet.PoleRotHeadFillet));
61      set(app.PoleHeadAngledegEditField,'Enable','on','Editable
        ','on','Value',num2str(dataSet.PoleRotHeadAngle));
62      set(app.NumberofturnsperpoleEditField,'Enable','on','
        Editable','on','Value',num2str(dataSet.FieldTurns));
63      set(app.CoilfillingfactorpuEditField,'Enable','on','
        Editable','on','Value',num2str(dataSet.
        RotorConductorFillingFactor));
64  ...
65  if strcmp(dataSet.TypeOfRotor,'EESM')
66
67      set(app.YokeWidthBouCheck,'Enable','on');
68      set(app.PoleBodyHeightBouCheck,'Enable','on');
69      set(app.PoleAnglepuBouCheck,'Enable','on');
70      set(app.PoleWidthBouCheck,'Enable','on');
71      set(app.CoilWidthBouCheck,'Enable','on');
72      set(app.CoilHeightBouCheck,'Enable','on');
73      set(app.PoleRotHeadFilletBouCheck,'Enable','on');
74      set(app.PoleRotHeadAngleBouCheck,'Enable','on');
75
76      if dataSet.YokeWidthBouCheck
77          set(app.YokeWidthBou,'Enable','on');
78      else
79          set(app.YokeWidthBou,'Enable','off');
80      end
81  ...
```

Observing the upward code lines, it can be seen how this function operates. In fact, it enables the possibility for the user of modifying the single parameters of the electrical machine under analysis. First, it is the turn of the main ones, like: number of pole pairs, number of slots, number of three-phase sets, etc. Then, depending on the topology selected, the same is repeated for the specific variables of the single topologies. This is the case also for the EESM machine. In fact, when the function confronts the type of rotor and detects that an EESM is under analysis, it automatically enables the possibility of editing the parameters of this machine and blocks the ones of the other topologies. In addition, also the verification that the values inserted respect the boundaries, obtained from the **back_compatibility.m** file, is performed and if the value inserted is higher or lower then its respective limits, the function changes it autonomously to the value of the closest limit. This process is repeated for each EESM variable, even if in the code lines only an example is reported.

The other function that was modified is: **GUI_APP_DrawMachine.m**. This function is responsible for drawing one pole of the electrical machine under analysis in the GUI. In fact, the user is able to observe in real time the consequences of the changes of the single parameters by just looking at the sketch of the machine in the low-right hand side corner of the GUI. This feature is very useful as, it makes of

easier comprehension the meaning of the machine's single parameters.

In the following lines, a brief overview of the function will be exhibited, with a focus on the EESM machine characteristic parts.

```
function app = GUI_APP_DrawMachine(app)

% flag_plot = 'Y';
h = app.AxisGeometry;
dataSet = app.dataSet;
...
[~, ~, geo,per,mat] = data0(dataSet);
[geo,gamma,mat] = interpretRQ(geo.RQ,geo,mat);
...
% nodes
[rotor,~,geo] = ROTmatr(geo,fem,mat);
[geo,stator,~] = STATmatr(geo,fem);

GUI_Plot_Machine(h,rotor);
GUI_Plot_Machine(h,stator);
...
% Rated current computation (thermal model)
[~,geo] = calc_endTurnFieldLength(geo);
per = calc_if(geo,per,mat);
dataSet.RotorCurrentDensity = per.Jf;
dataSet.RatedFieldCurrent = per.if0;
dataSet.Rf = per.Rf;
dataSet.RotorCurrentDensity = per.Jf;
...
% Mass and Inertia computation
geo.pShape = dataSet.pShape;
geo.mCu = calcMassCu(geo,mat);
geo.mPM = calcMassPM(geo,mat);
geo.mAl = calcMassAl(geo,mat);
[geo.mFeS,geo.mFeR] = calcMassFe(geo,mat);
if strcmp(geo.RotType,'EESM')
    geo.J = NaN;
    warning('Rotor inertia not yet computed')
    ...
```

These lines of code start by defining the function. Then, the function proceeds to retrieve the data, which is required to draw the electrical machine under analysis and perform some immediate estimations. Consequently, geometrical, material, performance, objective and decision variables data is extracted and saved. This data is then used to draw both the stator and the rotor, by defining first their nodes and elements, using the inner functions **ROTmatr.m** and **STATmatr.m**. After the two main geometrical elements of the machine have been drawn, the function proceeds to perform some immediate estimations, like calculating rated stator current (whose steps are here neglected). For the EESM specific case, the following additional calculations are also performed: DC winding end turns l_{endF}, field current i_f, rotor current density J_f, field winding resistance R_f and rated field current i_{f0}. The knowledge of these parameters is crucial as they will be used not only in elec-

tromagnetic simulations but also in thermal ones. In addition, this function is also able to estimate mechanical parameters like the mass and inertia of the machine. After the other lines of code, here neglected for space reasons, are run, this function completes its task of drawing a pole of the electrical machine under study in SyR-e's GUI, having estimated both electrical, thermal and mechanical parameters in the meanwhile.

3.2 EESM Implementation Validation

In order to verify the correct operation of the EESM implementation in SyR-e, a series of test analysis has been performed. These procedures are necessary to detect the presence of possible errors, either in the geometrical or in the electrical parts, committed in the functions' development or adaptation. For these reasons, two different simulation runs are executed: a multi-objective optimization with 60 generations of 60 individuals each and a design space sampling through Sobol set mapping (with more than 9000 cases). Summarizing, an optimization first and then a sensitivity analysis will be performed, in such a way that these modifications are tested in different environments and dealing with tasks of various nature.

The validation procedures that will be carried out and described in the following lines will be executed using the **ICEM24** motor as basis. This electrical machine is present in SyR-e's library of machines' examples. However, being the ICEM24 not an EESM, it has been modified by changing its rotor type to an EESM. It is known that this leads to having a not properly designed machine but, this ensures that the results obtained are even more satisfactory, as the analysis have been carried out on an un-optimal sample. Consequently, the result obtained for a proper-EESM machine will be even better.

3.2.1 Validation with MODE

As already mentioned, the first type of validation procedure is an optimization one. This analysis is performed exploiting the Multi-objective Differential Evolution algorithm, which has been discussed in the previous chapter. The optimization problem setup has been carried out selecting only two objectives: an output torque value and maximum torque ripple one. It is important to underline that, even if the target torque value is negative, the algorithm aims at maximizing its value. In fact, optimizing a positive variable is equivalent to minimizing a negative one. For what concerns the decision variables, seven have been selected. It is important to underline that all the decision variables connected to the EESM machine rotor were only selected, as this procedure aims to verify if the implementation process was executed properly. Being the stator identical to the well-consolidated **ICEM24.m** machine's one, it is supposed that it will not generate any issues in the optimization process. The full optimization setup, with the target values and boundaries of each variable, can be observed in Fig. 3.2 and Fig. 3.3.

Before launching the first full optimization procedure, a manual code verification has been carried out by running the functions manually in Matlab's Debug Mode. After these manual checks have delivered positive results, the real optimization process was launched. The proper optimization test kept going for several hours,

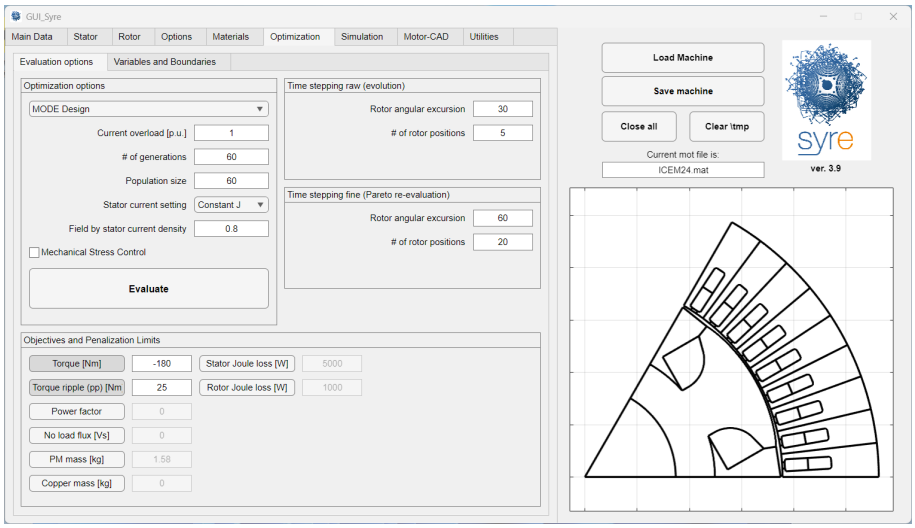


Figure 3.2: MODE EESM Validation Objectives

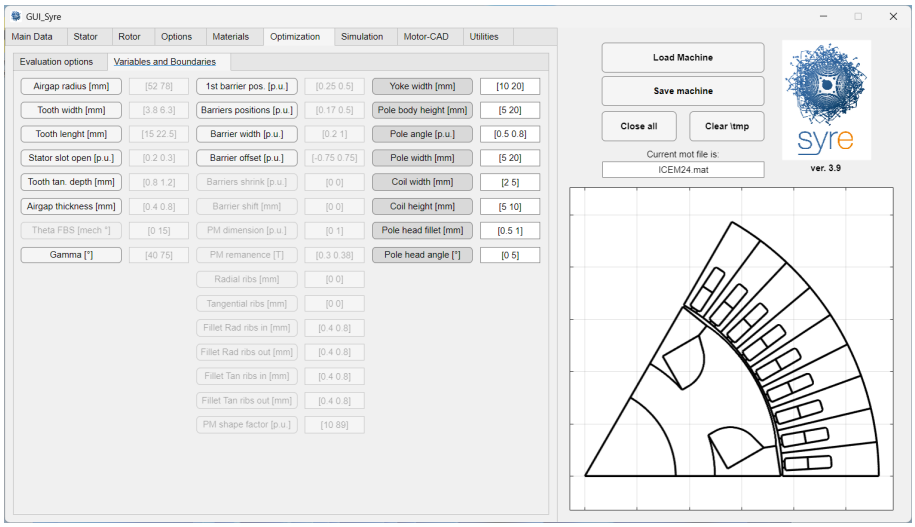


Figure 3.3: MODE EESM Validation Decision Variables

approximately five, until it stopped as it concluded its evaluations on the populations generated. Consequently, it can be concluded that the EESM implementation was successful in terms of the optimization analysis, even if the results will be analyzed in a following section.

3.2.2 Validation with Sobol Set Mapping

The second analysis test conducted is composed by a sensitivity analysis using the Sobol space mapping method. Sobol is quasi-random sampling algorithm used to explore efficiently the design parameters' space. It is capable of generating a set of uniformly distributed points in the multidimensional variables' space. It is usually preferred over standard random sampling algorithms as it guarantees a more uniform space analysis, less variation in the simulation results, obtained using also less samples. The use of Sobol allows to perform efficient sensitivity analysis and consequently, to identify the relationship between the design parameters and the machine's performances, without recurring to computationally significant additional

FEA simulations.

If the machine under test was the same of the first analysis (still an EESM adaptation of the ICEM24 motor example present in SyR-e), some features were different. In fact, this verification was composed of two identical smaller ones. For each simulation, the population was set equal to 4096 individuals and the analysis performed imposing a constant current density, with a ratio of 0.8 between the rotor and stator current densities.

For this case study, more objectives were selected: output torque, torque ripple, copper mass, stator Joule losses and rotor Joule losses. The chosen decision variables were all the ones available for the EESM machine rotor. No decision variables regarding the machine's stator were selected as it is not subject of analysis, coming from a well-consolidated technology. In such way, the simulation is the most complex one, from the rotor's point of view, and positive results indicate that the implementation of the EESM machine does not create any issues and works properly.

It is important to underline that the real objective of this analysis regards only the validation of the EESM machine in SyR-e, not the results obtained.

In addition to the preliminary setup, it is crucial to highlight that the mechanical and thermal analysis performed automatically by SyR-e have been turned off for this simulation. This choice was made to avoid errors from taking place as the EESM machine has not been properly implemented in those field. Consequently, the upcoming analysis will focus only on the geometrical and electro-magnetical aspects.

The overall simulation setup can be observed in Fig. 3.4 and in Fig. 3.5.

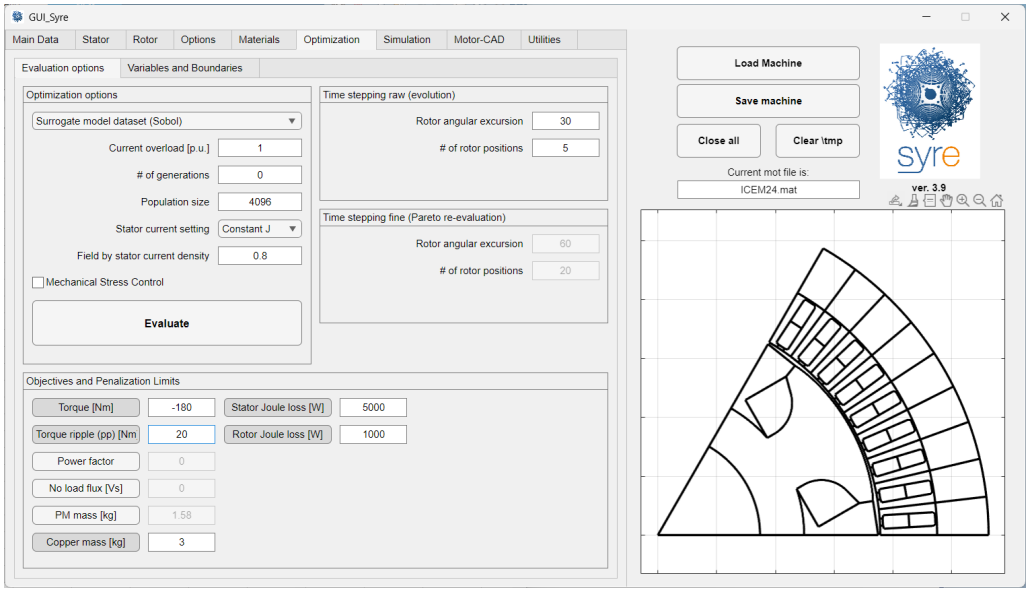


Figure 3.4: Sobol Sampling EESM Objectives

Also in this case, before launching the real analysis, some manual testing was performed exploiting Matlab's Debug Mode. Being the results promising, in the sense that no errors occurred, the real problems were launched. Both simulations concluded without delivering errors, even if their computational time was longer than the one of the first analysis, concluding after approximately 10 hours each.

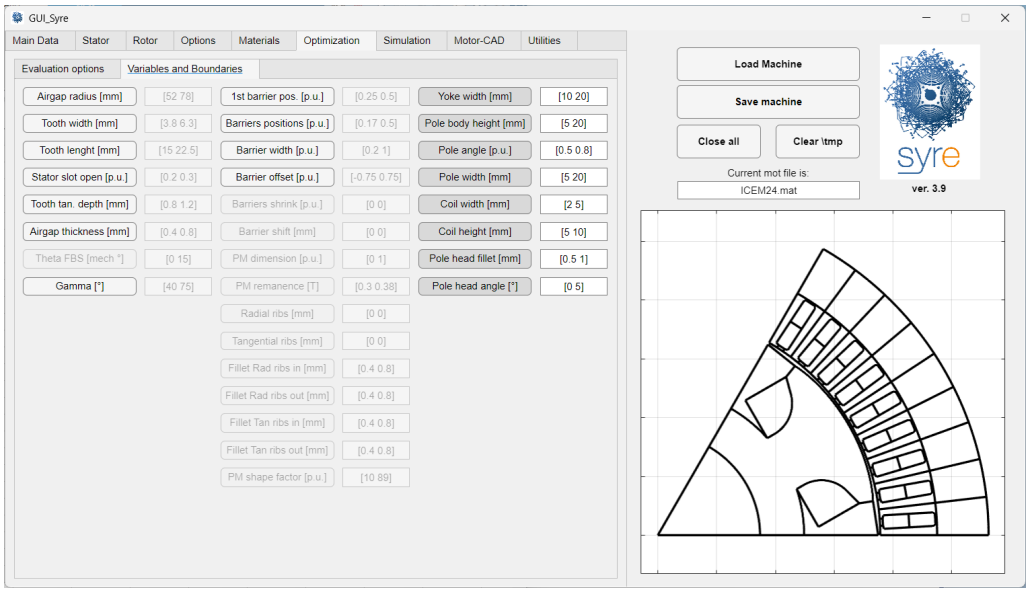


Figure 3.5: Sobol Sampling EESM Decision Variables

Nevertheless, both simulations delivered their respective results which will be discussed later. Consequently, it can be concluded that the EESM optimization was successful for this type of review also.

3.3 Results and Comments

The following section aims at presenting the results obtained from the validation analysis performed on the newly implemented EESM machine in SyR-e. In addition to the pure results, explanatory comments and figures will be reported for a more immediate interpretation.

3.3.1 MODE Optimization Results

After all preliminary verifications and setup steps were completed, as depicted in the previous sections, the proper optimization validation procedure has been performed. The computational weight of this type of analysis was not insignificant, reason why it has been necessary to use a virtual machine, made available by Politecnico di Torino. Exploiting this machine's larger computational capabilities, it was possible to obtain the optimization results in approximately five hours, partly thanks to SyR-e's ability to exploit Matlab's Parallel Computing toolbox.

After all the steps of the analysis were completed, SyR-e automatically provided the results in terms of both pure data, in Matlab files, and graphical figures for a more immediate understanding. The software has been able to generate bar charts regarding each one of the objectives and the decision variables, highlighting the distribution of their values between the various machines generated in the population. In addition, also a Pareto front was delivered to underline the best compromise between the desired objectives and their goal values.

In the following pages, the just mentioned figures will be reported, starting from

the decision variables and then going to the objectives. The Pareto front will be neglected as the only goal of this simulation is the pure verification of the correctness of the EESM implementation procedure.

Fig. 3.6 shows the distributions of the decision variables' obtained from this first simulation. In particular, Fig. 3.6a and Fig. 3.6b represent respectively the values taken by the field winding coil height *hb* and by rotor's pole body height *hpb*. Observing the left hand side figure, it can be seen that in most cases the variable reaches the higher limit value. In addition, the most of remaining ones have a value of 5mm, coinciding with the constraint lower value. This leads to the conclusion that the constraint limit values are too strict for this value to reach its optimal value.

A similar pattern can be observed also in the right hand side figure. However, in this case only a few subjects reach the higher limit value and most of them are within the limits, far from the lower one. In fact, the lowest value observed is approximately 16.4mm, which is significantly closer to the upper constraint than to the lower one. Consequently, it could be interesting to observe the results in case the limits of the constraints are enlarged.

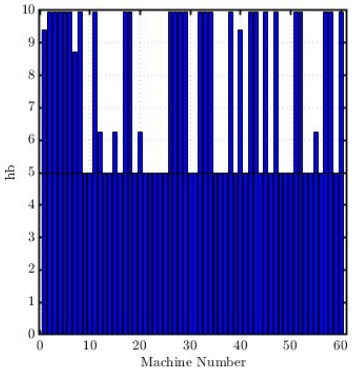
Fig. 3.6c and Fig. 3.6d represent respectively the values taken by the rotor's yoke width *lyr* and its pole head fillet r_{fillet}. By rapidly observing these figures, it can be observed that the values taken by the yoke widths in the candidate machines are, once again, very close to their upper limit value of the constraints. This trend may signal that the constraints used for this analysis are too strict and may be enlarged in a future simulation to obtain the optimal value of the variable.

A different trend can be seen for the pole head fillet. In fact, it never reaches the higher limit value but, instead, the most common value coincides with the lower constraint limit. Consequently, it is possible that the optimal value of this variable stays below the lower limit value.

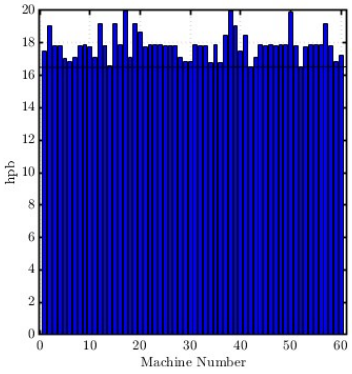
Fig. 3.6e and Fig. 3.6f represent respectively the values taken by the rotor's pole head angle *thHead-deg* and the field winding coil width *wb*.

From a rapid observation of the graphs, it can be seen that the most common pole head angle is equal to zero, with a few exceptions having values higher than one. Meanwhile, the coil width has a very different distribution. The variety of values is more diversified this time. In fact, for most candidates their value is very high and close to the upper limit of the constraint. However, for some cases the value is lower and lowest one is 2mm, which coincides with the lower limit of the constraint. Nevertheless, most individuals present an amplitude higher half of the range, which lead to the conclusion that in a future analysis its limits could be enlarged.

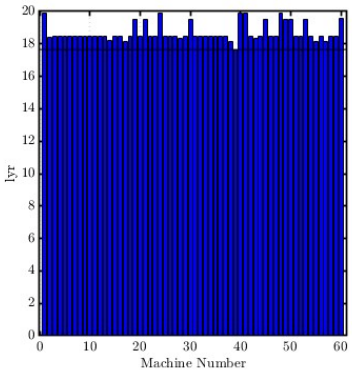
Fig. 3.6g shows the distribution of the pole width *wp* values of the candidate machines obtained from the optimization problem. As it can be seen from the bar chart, the values are not evenly distributed within the constraint limits. In fact, there are several candidates having values coinciding with upper limit and an even bigger part characterized by a smaller one, a few coinciding with lower limit value of this constraint. The discrepancy observed in the previous figures is observable also in this one and they can all be re conducted to unorthodox adaptation of a PMSM born machine to being an EESM one.



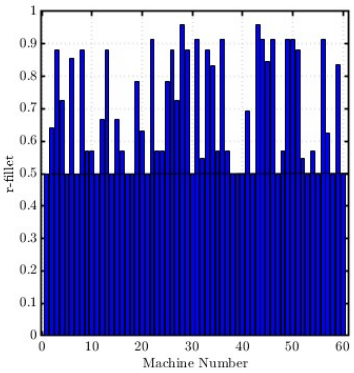
(a) Rotor Coil Height



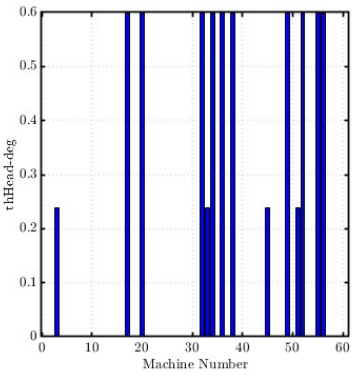
(b) Rotor Pole Body Height



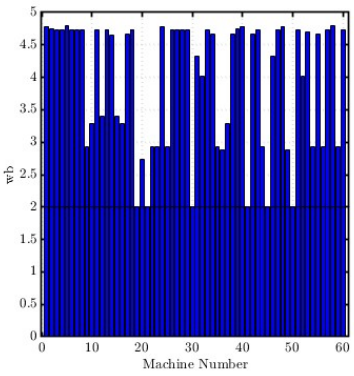
(c) Rotor Yoke Width



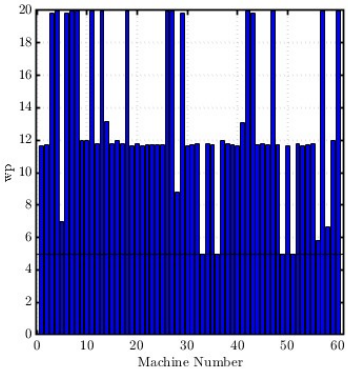
(d) Rotor Head Fillet



(e) Rotor Pole Head Angle



(f) Rotor Coil Width



(g) Rotor Pole Width

Figure 3.6: MODE Validation Decision Variables Distributions.

Nevertheless, to provide an higher level of completeness of this discussion, also the other results regarding the objectives will be reported in the following pages.

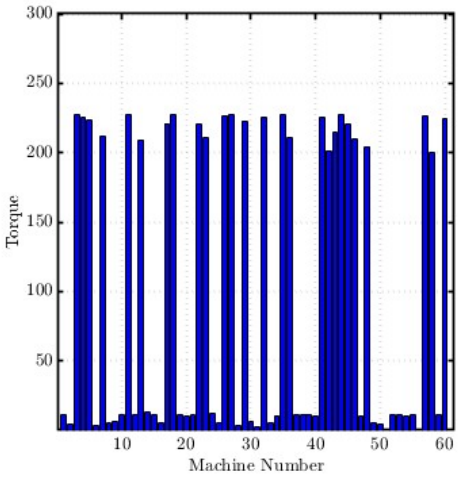


Figure 3.7: Candidate Machines' Output Torque Distribution

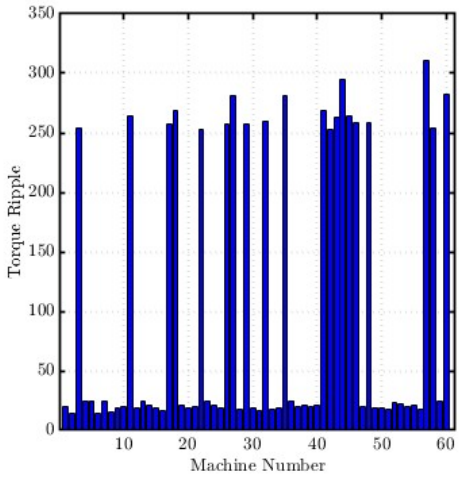


Figure 3.8: Candidate Machines' Torque Ripple Distribution

Fig. 3.7 and Fig. 3.8 show the optimization outputs regarding the two objectives of torque and torque ripple. As it can be observed from the left hand side figure, only a limited number of machines respect the analysis initial objective target value of an output torque equal to 180 Nm. In fact, most of the machines provide a torque value significantly smaller than the one initially wanted.

An even more worrying trend can be observed for the torque ripple objective. In fact, in the simulation setup a target torque ripple value of 25 Nm was selected, which constitutes more than 10% of the target torque. From the right hand figure, it can be seen that most of the proposed machines do not respect the initial target value. In fact, it can be observed that the machines delivering the requested torque produce a very high torque ripple. The only individuals that have torque ripple values in the same range of the objective provide an output torque significantly lower than the one requested initially.

The previous considerations highlight how the results obtained from this first optimization test for an adapted EESM version of the **ICEM24.mat** machine are not acceptable, in terms of pure machine designing. The poor quality of the output results is strictly connected to the "raw" adaptation of the electrical machine geometry, born to be used with permanent magnets, to a new type of rotor. In addition, the optimization procedure waists iterations in trying to calculate the MPTA angle γ, even when not selected as variable, which leads to the worst overall design space exploration, ultimately resulting in the discrepancies observed between the values of the same objectives. Last but not least, also the fact that the stator of the machine was not subject of the optimization process, but only its rotor, made the results even worse.

However, the main goal of this type of analysis was not to deliver a final design of an EESM machine. In fact, it was to verify if the EESM implementation process,

previously described, was successful or not. Considering the fact that the simulation proceeded without stopping for unexpected errors and delivered the presented results (even if they were not satisfactory), it can be concluded that the main objective of this analysis was completed.

3.3.2 Sobol Set Mapping Results

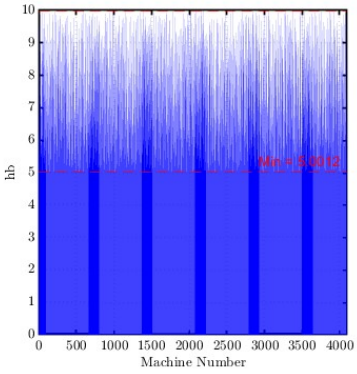
After the simulations setup were prepared, as explained previously, the two Sobol mapping simulations were launched. These analysis were carried out one at the time. Even if each analyzed a population of 4096 individuals, the subjects under study were different between the two. In fact, the individuals studied in the second one are directly adjacent to the ones of the first. In this way, it has been possible to sample a wider space, delivering a more accurate dataset, and test even more the EESM machine implementation. However, it is important to highlight that, being the MTPA angle γ not selected as decision variable, the overall sampling performed by Sobol was not of highest efficiency, for reasons that will be explained in a later section.

Once both simulations were completed, SyR-e provided the results in terms of pure datasets. At this point, a post-processing of the delivered data was necessary for it to be observable in a figure form. This process started by retrieving, for each simulation, the OUT matrix containing all the output information. Of all the data contained in this matrix, two smaller ones were of greatest interest, the matrices *Xpop* and *Jpop*. *Xpop* contains the values assumed by decision variables for each electrical machine (individual of the population) analyzed, each column representing a variable. *Jpop* has a similar structure but this time each column represents the values of the objectives. This operation was necessary to retrieve the data on which the graphs contained in Fig. 3.9 and 3.10 are based.

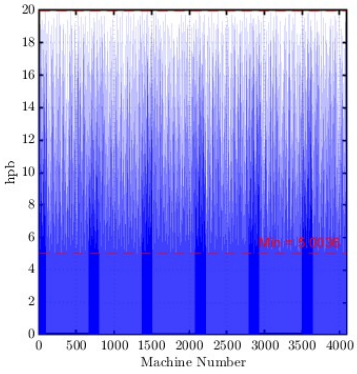
It is crucial to highlight that only the figures regarding the decision variables will be reported in the following pages. This choice was due to the fact that, as already mentioned, the real objective of this analysis is not really to find data on which to perform further analysis, but to verify if the implementation of the EESM machine was done in the right way, without delivering unwanted errors. For this reason, only the decision variables will be reported, to underline the wideness of the variables' space analyzed.

Fig. 3.9a and Fig. 3.10a show the coil heights' *hb* values assumed by the field winding in the various cases analyzed. It can be seen that both the minimum and maximum values taken by this variable are different between the two runs, highlighting how these are adjacent. In addition, it is also observable the variety of the values assumed by this variable, within the limits defined in the setup, underlining the density of the sampling performed in this space mapping analysis.

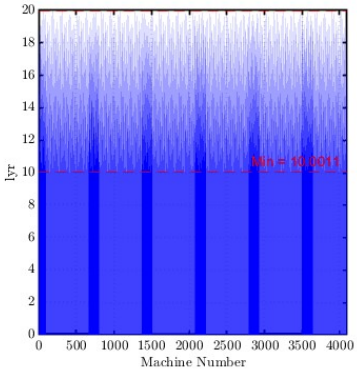
Fig. 3.9b and Fig. 3.10b represent the heights *hpb* taken by the rotor's pole body along the populations studied. Also in this case, it can be noted that the values assumed by this variable have an high variety and their minimum and maximum amplitudes are close to the limits imposed by the predefined constraints but different from each other.



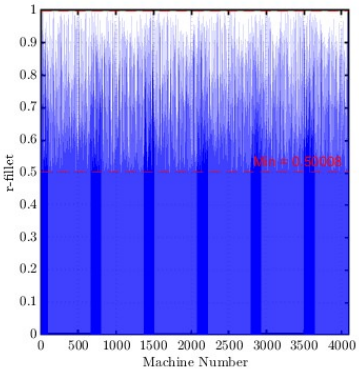
(a) Rotor Coil Height



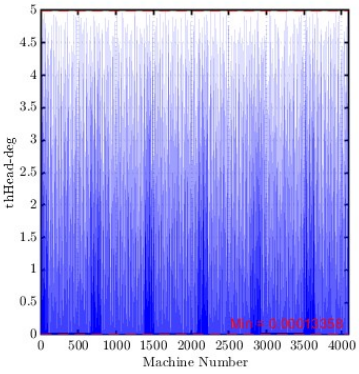
(b) Rotor Pole Body Height



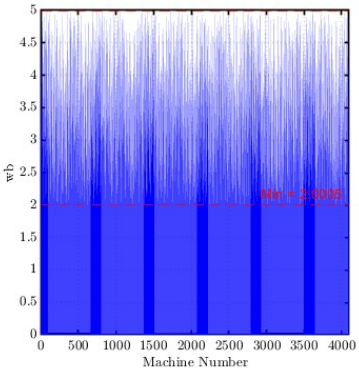
(c) Rotor Yoke Width



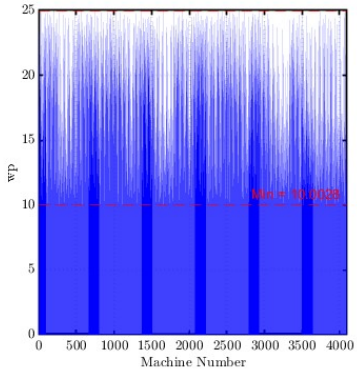
(d) Rotor Head Fillet



(e) Rotor Pole Head Angle

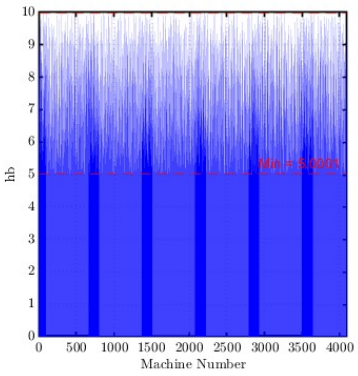


(f) Rotor Coil Width

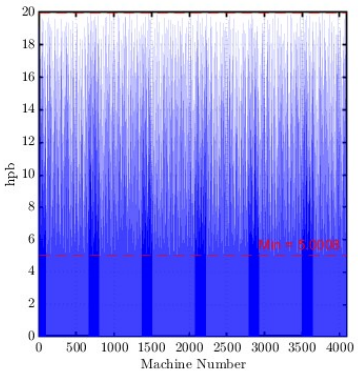


(g) Rotor Pole Width

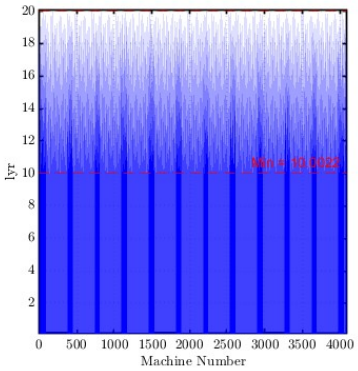
Figure 3.9: Sobol First Run of Validation Decision Variables Distributions.



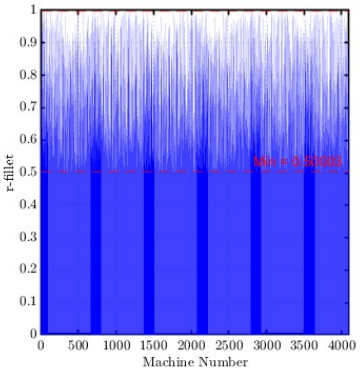
(a) Rotor Coil Height



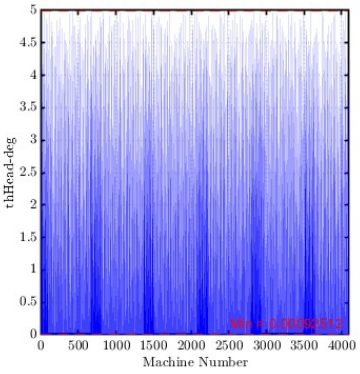
(b) Rotor Pole Body Height



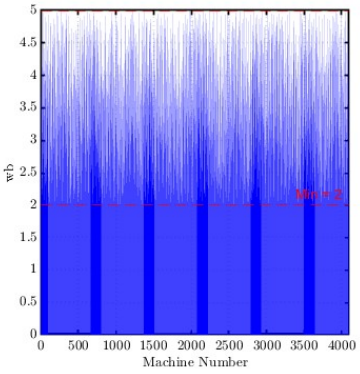
(c) Rotor Yoke Width



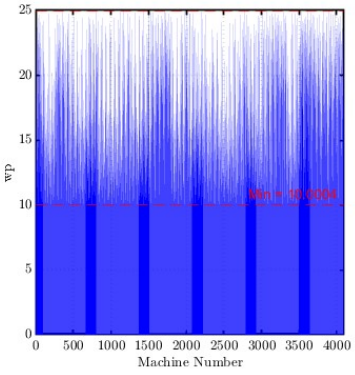
(d) Rotor Head Fillet



(e) Rotor Pole Head Angle



(f) Rotor Coil Width



(g) Rotor Pole Width

Figure 3.10: Sobol Second Run of Validation Decision Variables Distributions.

Fig. 3.9c and Fig. 3.10c show the amplitudes of the rotor's yoke widths *lyr* studied in the two analysis. Fig. 3.9d and Fig. 3.10d make observable the values taken by the rotor's head fillet *r-fillet* during the simulation runs performed. Fig. 3.9e and Fig. 3.10e show the distribution of the values assumed by rotor's pole angle *thHead-deg*. Fig. 3.9f and Fig. 3.10f represent the values assumed by the field winding's coil width *wb*. Last but not least, Fig. 3.9g and Fig. 3.10g highlight the amplitudes taken by the rotor's pole widths *wp*.

As already mentioned, observing the different patterns of the figures, various common aspects can be deducted. The first one regards the maximum and minimum values of each decision variable represented. In fact, it is evident how both values, for each run performed, are close to the limits of the constraints selected in the simulations' setup, in line with the Sobol objective of performing an efficient design space exploration. In addition to that, by a rapid observation of the two figures proposed for each decision variable analyzed, it can be noted how each one assumes a various spectrum of values within those predefined limits. This highlights even more how the variables' design space sampling performed was very wide, being the number of individuals studied so high. This choice was made on purpose to put under stress the implementation of the EESM machine in SyR-e. In fact, being both the number of individuals and the one of selected decision variables large, the simulations performed coincided with worst-case scenario for this new feature. However, as explained in the previous lines, it was still possible to obtain the presented results without errors. Consequently, it can be concluded that also for this case study, the implementation of the EESM machine in SyR-e was successful.

4. Improvement of the Design Space Sampling Procedure

This chapter is dedicated to explanation of the steps which lead to the development of a new SyR-e tool. This new feature allows to determine the MTPA of an electrical machine independently on the fact that this angle γ is a decision variable or not, in the analysis setup, and on the topology of the machine under analysis. In the latter section of this chapter, the validation of the proposed procedure is assessed and a confrontation of results between the previous and the current methodologies is performed.

4.1 Addition of MTPA search during MODE and Design Space Mapping

As it is know, the Maximum Torque Per Ampere (MTPA) angle is one of the most important information to know about an electrical machine. In fact, it allows to identify the current angle corresponding to the maximum output torque for that current amplitude. Consequently, the knowledge of the MTPA angle γ is crucial for developing an efficient and reliable electrical machine control strategy. The value of this variable is not constant, but changes depending on the electrical machine under analysis, due to the topology and other motor's parameters (like the d-axis and q-axis inductances), and its working conditions.

The implementation of an automatic MTPA calculation tool is beneficial for every type of analysis. However, Design Space Mapping simulations receive the biggest boost in terms of performance if compared with the one received by an optimization problem. In fact, generally speaking, the MTPA angle γ must always be a decision variable of any type of analysis, even when not selected. Proof of this is that during optimization problems, the algorithm wastes iterations trying to find the MTPA angle, instead of performing a better exploration of the geometrical design space, resulting in a worst overall performance. Nevertheless, for an optimization analysis, this fact does not constitute a real problem as when the convergence of the problem is reached both the MTPA angle and the optimal solutions are delivered as outputs. However, the same can not be said for Space Mapping analysis, like Sobol mapping. In fact, for these types of studies, the presence of an additional electrical variable in the analysis' space constitutes a non-negligible issue as consequently, the geometrical design space is not sampled in a satisfactory way, leading to a lower overall quality of the delivered dataset. Consequently, an automatic MTPA angle calcu-

lation removes the limitations associated with the presence of this additional (and unexpected) electrical variable. This paves the way to a more efficient and dense geometrical space sampling, especially for smaller populations, which ultimately leads to the delivery of a more accurate dataset, as each machine (individual) is already evaluated along its MTPA angle.

In the following lines, the workflow procedure utilized to develop the MTPA automatic research tool will be explained. This new feature has been inserted within the **FEMMfitness.m** file and it is based on iterative processes.

The first step consists in defining a set of variables useful for the iterative cycles' initialization.

```
% Variables necessary for MTPA calculation
flagMTPA = 0;

maxIter = 40;
gammaStep = 2;
direction = 0;

if isfield(per,'if0')
    per.if = per.if0;
else
    per.if = 0;
end
if ~isempty(RQ)
    RQ(end) = 90;
end

if any(strcmp ('gamma', geo.RQnames))
    flagMTPA = 0;
else
    flagMTPA = 1;
end
```

As it can be seen from the Matlab code lines reported, the first step is the definition of a flag variable *flagMTPA*, which is initialized equal to zero. Following, the maximum number of iterations, the amplitude of the variations applied to γ and their directions are also added. Consequently, a differentiation between the EESM machine and the other topologies is performed by assigning the field current value depending on the study case. A starting value of the MTPA angle γ is also defined and imposed equal to 90 degrees. In the final lines of the preliminary initialization, an automatic *flagMTPA* value assignment is performed. In fact, depending on the fact that $\boldsymbol{\gamma}$ is selected as decision variable or not in the optimization setup, the value of *flagMTPA* is imposed equal to zero or one.

In the code lines reported below, the proper iterative process leading to the calculation of the MTPA angle will be reported and also explained in detail.

```
    if ~isempty(RQ)      % MODE optimization (RQ geometry)

    % Cost functions
```

```
308     cost = zeros(1,length(geo.OBJnames));
309     temp1 = 1;
310     % Torque
311     if strcmp(geo.OBJnames{temp1},'Torque')
312         if ~flagMTPA
313             cost(temp1) = -out.T;
314         else
315             % aggiungere ricerca MTPA
316             gamma0 = RQ(end);
317             jj = 1;
318             done = 0;
319             TVect    = nan(1,maxIter);
320             gVect    = nan(1,maxIter);
321             dTppVect = nan(1,maxIter);
322             idqVect  = nan(1,maxIter);
323             fdqVect  = nan(1,maxIter);
324
325             perTmp = per;
326             TmpSOL_old = [];
327             TmpSOL_new = [];
328
329
330              while ~done
331                     if jj==1
332                         gammaSim = gamma0;
333                     elseif jj==2
334                         gammaSim = gamma0+gammaStep;
335                     elseif jj==3
336                         gammaSim = gamma0-gammaStep;
337                     else
338                         gammaSim = gammaSim+direction*
                            gammaStep;
339                     end
340
341                      RQ(end) = gammaSim;
342                      TmpSOL = simulate_xdeg(geo,perTmp,mat,
                            eval_type,pathname,filename);
343                      TVect(jj)    = mean(TmpSOL.T);
344                      gVect(jj)    = gammaSim;
345                      dTppVect(jj) = max(TmpSOL.T) - min(
                            TmpSOL.T);
346                      idqVect(jj)  = mean(TmpSOL.id)+j*mean(
                            TmpSOL.iq);
347                      fdqVect(jj)  = mean(TmpSOL.fd)+j*mean(
                            TmpSOL.fq);
348
349                     % To save the last 2 results of the
                          iteractive process
350                      TmpSOL_old = TmpSOL_new;
351                      TmpSOL_new = TmpSOL;
352
```

```
353
354                     if jj==3
355                             [~,index] = max(TVect,[],'omitnan
                                ');
356                             if index==1
357                                done=1;
358                             elseif index==2
359                                    direction=+1;
360                             else
361                                    direction=-1;
362                             end
363                      elseif jj>3
364                          if TVect(jj)<TVect(jj-1)
365                             done=1;
366                          end
367                     end
368
369                     if jj==maxIter
370                          done=1;
371                     end
372
373                     jj = jj+1;
374                     disp(['Simulation␣' int2str(jj-1) '␣done
                        '])
375             end
376
377             [~,index] = max(TVect,[],'omitnan');
378
379
380
381             % Output data
382              OUT.geo    = geo;
383              OUT.per    = perTmp;
384              OUT.mat    = mat;
385              OUT.T      = TVect(index);
386              OUT.dTpp   = dTppVect(index);
387              OUT.gamma  = gVect(index);
388              OUT.idq    = idqVect(index);
389              OUT.fdq    = fdqVect(index);
390              OUT.RQ     = RQ;
391              %OUT.nFEMM = nFEMM;
392              OUT.Pjs    = 3/2*per.Rs*abs(OUT.idq)^2;
393
394              % Identify Rf value depending on rotor geometry
395              if strcmp(geo.RotType, 'EESM')
396                 Rf=per.Rf;
397                 OUT.Pjf    = Rf*out.if;
398              else
399                  Rf=0;
400                  OUT.Pjf = nan;
401              end
```

```
402
403                 if mean(TmpSOL_new.T) > mean(TmpSOL_old.T)
404                     OUT.SOL = TmpSOL_new;
405                 else
406                     OUT.SOL = TmpSOL_old;
407                 end
408
409                 out.SOL = OUT.SOL;
410                 out.id      = real(OUT.idq);
                                                          % [A]
411                 out.iq      = imag(OUT.idq);
                                                          % [A]
412                 out.fd      = real(OUT.fdq);
                                                          % [Vs]
413                 out.fq      = imag(OUT.fdq);
                                                          % [Vs]
414                 out.T       = OUT.T;
                                                                % [
                    Nm]
415                 out.dTpp    = OUT.dTpp;
                                                              % [Nm]
416                 out.gamma = OUT.gamma;
                                                              % [
                    degrees] MTPA angle
417
418                 cost(temp1) = -mean(OUT.T);
419                 temp1 = temp1+1;
420
421         end
422
423     end
```

First of all, these lines of code are run independently on the problem under analysis, it could be either an optimization or a Sobol sampling or another type included in SyR-e. Then a *cost* vector, having the length of the problem's number of objectives, is initialized and with it also a counting variable *temp1*.

Following, the real differentiation between the cases explained before takes place. In fact, depending on the value of *flagMTPA*, the MTPA automatic calculation is either skipped, if γ is already a decision variable of the problem and consequently the output torque value is directly calculated, or not. For explaining this procedure, a value of *flagMTPA* equal to 1 will be hypothesized.

As soon as the automatic MTPA research code is run, a set of new variables is defined. These new parameters are local ones and are only used in these lines of code for either: assigning temporary values to real machine entities or defining variables useful for the iterative process later run. For example, *gamma0* is the initial value of the MTPA angle from which the calculations will start. Meanwhile, *done* is a flag used to highlight if the iterative process has ended or not. *perTmp* represents temporary values of machine performance variables.

At this point, the real iterative process begins. In fact, as long as the *done* flag is not equal to 1, the process continues to take place updating the counting variable *jj* value at each iteration, until either the MTPA angle is found or the maximum number of iterations is reached. Depending on *jj*'s entity, the value of the temporary variable *gammaSim*, representing the MTPA angle, is modified. For the first iteration, it is imposed equal to *gamma0* but in following ones, it is modified first to identify a search direction and then to look for the final MTPA angle value.

After the value of *gammaSim* has been modified according to the previous rules, the consequent value of the variables of the electrical machine under analysis are calculated using the **simulate_xdeg.m** function and saved as a temporary solution in *TmpSOL*. The main variables of interest (torque, torque ripple, currents, fluxes and γ) are then saved singularly in their respective array vectors, which will contain the values obtained from each iteration of this process. In addition, the last two solution vectors *TmpSOL* are saved in a singular way for future analysis.

As soon as the number of iterations performed is at least three, a confrontation between the calculated torque values is performed. Depending on the iteration number corresponding to the biggest one, the process either stops or identifies the direction in which to perform further analysis. In case the arrest condition is not immediately met, additional iterations are performed. The torque values obtained in the last two calculations are then confronted, ending the process if the penultimate value is bigger than the last one calculated.

Once the stop condition of the iterative process has been met, the output data processing begins. First, the electrical machine variables corresponding to the maximum torque value are saved into the *OUT* struct vector. Then, taking advantage of the fact that the last two calculations performed have been saved, the *SOL* vector corresponding to it is saved in *OUT*. Consequently, the just mentioned *OUT.SOL* vector is saved in **FEMMfitness.m** output vector *out* and the main variables of interest are then saved singularly. The process ends with the delivery of the calculated output torque (which is an objective of the optimization) and its respective angle, corresponding to the MTPA angle γ.

4.2 Procedure Validation and Assessment

As introduced in the previous section, the automatic MTPA research process is able to work on problems of various nature, from an optimization to a space sampling one. However, being Sobol sampling the one obtaining the most benefits from its introduction, for the reasons already explained, it has been decided to validate the development of this new feature by launching two Sobol simulations and confronting their results to highlight the differences. The two simulations launched are identical, and similar to the Sobol mapping one described in Chapter 3. They again regard an EESM adaptation of the **ICEM24.mat** motor present in SyR-e's library. The only difference between the two analysis presented below consists in the selection or not of the MTPA angle γ as a decision variable in the preliminary setup. For both cases, the analysis has been carried out considering a population of 512 individuals, imposing a constant current density type of analysis and a ratio between the field

and stator current densities of 0.8. The selected objective are: torque, torque ripple, copper mass, stator and rotor Joule losses. The selected decision variables are, also in this case, connected only to the rotor, for the reasons explained in the previous chapter, and every single one available has been enabled.

In addition, also for these study cases, it has been decided to turn off both the mechanical and thermal SyR-e's analysis tool, this is due to the fact the EESM machine has not been properly implemented in those fields yet. Consequently, to avoid unwanted simulations errors due to external causes, those two objectives have been avoided.

The common simulations' setup can be observed in Fig. 4.1 and Fig.4.2.

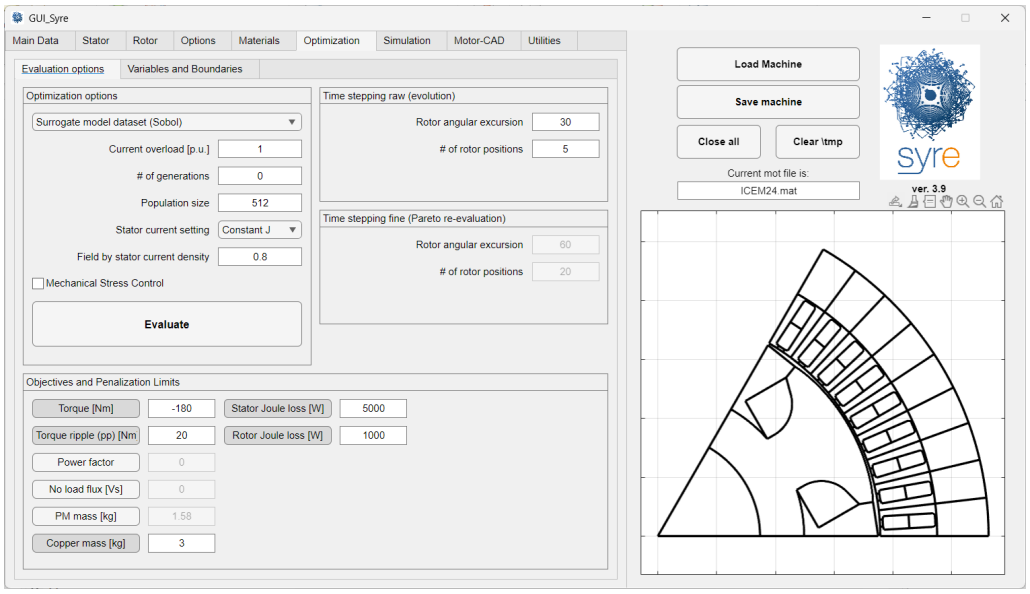


Figure 4.1: Sobol Sampling MTPA Procedure Validation Objectives

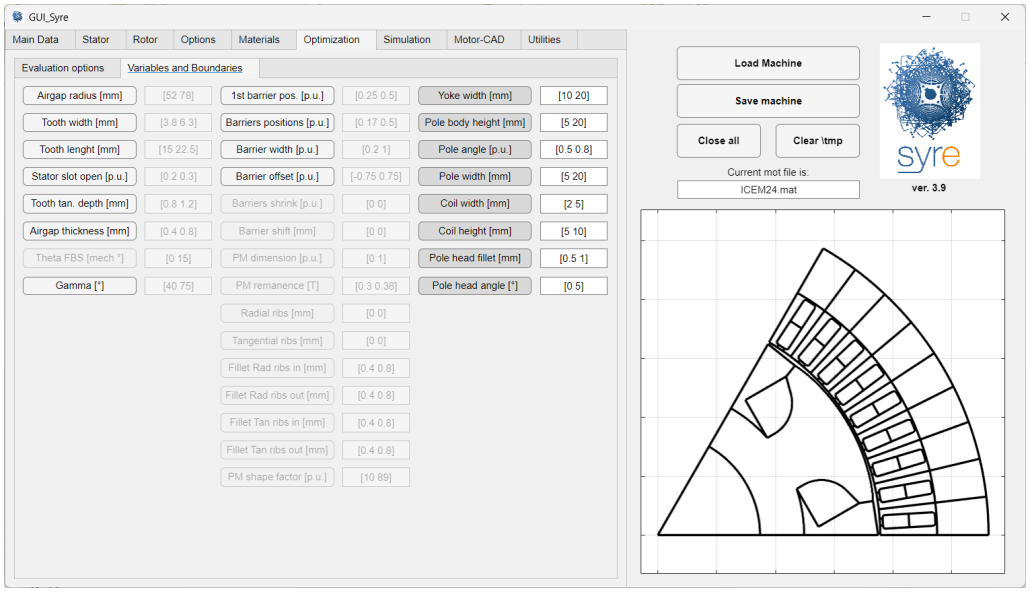


Figure 4.2: Sobol Sampling MTPA Procedure Validation Decision Variables

It is important to highlight that before launching these simulations several manual runs using Matlab's Debug Mode have been performed. These experiments have been useful for identifying possible critical issues and to validate the effectiveness of the developed code. During these test, no errors were found using the final version of the MTPA automatic research and the results obtained were aligned with the expectations.

Differently from the EESM implementation validation Sobol analysis, in this case, the figures representing the distributions of the single decision variables are substituted by different graphical representations. This is due to the fact that the main objective of this analysis is to prove both the correct operation of the automatic MTPA tool and its effectiveness in improving the overall performance of the space mapping performed during the analysis. The choice of these figures was done to highlight the differences between the two studied cases in terms of space sampling density. In the following lines, the figures representing the results obtained from the two simulations will be reported.

Fig. 4.3 illustrates the normalized amplitudes of the selected decision variables obtained from the two simulations performed. In particular, Fig. 4.3a is the output of the Sobol analysis performed selecting the MTPA angle γ as decision variable. Meanwhile, Fig. 4.3b is the output of the analysis performed exploiting the new automatic MTPA research tool.

This type of graphical tool helps to highlight the main purpose of the automatic MTPA research tool. In fact, in this way, Sobol mapping is able to perform a more dense sampling of the geometrical space as no iterations are wasted to look for the addition electrical variable constituted by the MTPA angle γ.

From a confrontation between Fig. 4.3a and Fig. 4.3b, it is possible to note how the second figure has an higher density of lines. If this characteristic is not directly observable in the central area of the figures, which is quite chaotic, it becomes more evident in the peripheral one. In fact, observing the areas close to the x-axis and to the unit amplitude line, it is possible to notice how Fig. 4.3b has more lines with respect to Fig. 4.3a.

In order to highlight the benefits generated by the introduction of the automatic MTPA research tool, an *ad-hoc* Matlab script, calculating the distances between the 512 points analyzed in the 7-dimensional variables' space, has been developed. Even if the choice of calculating the distances between the points obtained from the Sobol mapping analysis might seem unconventional, it has been executed after several considerations on the which could be the most efficient way to effectively show the higher sampling density performed in the simulation using the automatic MTPA tool. The direct calculation and consequent observation of the distances' distributions seemed as the best pick for a more immediate understanding of the matter.

The developed *ad-hoc* Matlab script will be reported in the following lines.

```
1 P = OUT.Xpop(:,1:7);
2
3 [N, d] = size(P);
```

```
fprintf('Points Number: %d, Dimensions: %d\n', N, d);

% Distances Calculation
if exist('pdist', 'file') == 2
    fprintf('Using pdist...\n');
    D = squareform(pdist(P, 'euclidean'));
...
dist_values = squareform(D);
dmin  = min(dist_values);
dmax  = max(dist_values);
dmean = mean(dist_values);
...
figure;
histogram(dist_values, 50);
title('Distances Distribution (OUT.Xpop)');
xlabel('Distance');
ylabel('Frequency');
```

This very simple Matlab code is able to estimate the distances between the 512 points subjects of the Sobol analysis in an efficient and cost effective way. In fact, it retrieves the first seven columns of the *OUT.Xpop* matrix, containing the values of the decision variables, and assigns them to a new variable *P*. Then, exploiting the *pdist* function, present in the Matlab Statistics and Machine Learning Toolbox, it is able to calculate rapidly the distances between each point, which corresponds to a line of the matrix *OUT.Xpop*. At this point a matrix *D*, containing the amplitudes of the distances between each point and the others, is created. Consequently, a single line form of the matrix *D*, called *dist_value* is formed. This is particularly useful to perform a statistical analysis by retrieving the minimum, maximum and mean values of the calculated distances. Last but not least, the distribution of the values assumed by the points' distances is summarized in an histogram graph. It is important to note that the Matlab code previously described has been developed just to perform this analysis. For this reason andhaving it has no other application in terms of electrical machine design, it has not been added to the SyR-e library.

As soon as the output data obtained from the two Sobol simulations performed, with and without the automatic MTPA research, were available, they have been inserted in the just described Matlab script. Consequently, it has been run two times, one for each analysis, and its outputs can be observed in Fig. 4.4 and in Tab. 4.1. Specifically, Fig. 4.4 shows both the distributions of the points' distances for the Sobol analysis performed, with and without the MTPA tool. It is particularly useful as it allows to execute an immediate comparison between the results obtained from the two simulations. In addition, Tab. 4.1 illustrates the results from a statistical point of view, highlighting the minimum, maximum and average values of the distances obtained from the two analysis.

Confronting the histogram graphs present in Fig. 4.4, it is possible to note how the two distributions are similar in shape. Nevertheless, they present some interesting differences. In fact, it can be observed that the second distribution, related to simulation performed with automatic MTPA research, presents an higher frequency of lower values of distances than the ones shown in the first. Consequently,

the second histogram shows, generally, a slightly lower frequency of higher distances compared to the first. This trend is aligned with expectations. In fact, as already mentioned, the distances for the analysis with the automatic MTPA were predicted to be lower than the ones of first one, in which the angle γ is selected as variable. The reason behind this forecast is connected to the higher space sampling density offered, thanks to the removal of the search for the MTPA angle γ, which allows exploring the same design space in a more efficient way, without wasting iterations to find this additional electrical variable.

However, it is important to underline that there are a few cases, observable in Fig. 4.4, in which this trend is not completely respected. Consequently, in these anomalous points, the MTPA-related distribution presents higher values of distances than the ones correspondent to the analysis performed selecting γ as variable. This discrepancy is connected to the size of the populations analyzed in these simulations. In fact, in order to limit the computational times required for the Sobol mapping simulations, it has been decided to perform these study cases on a population of only 512 individuals. Incrementing the size of the population under analysis should resolve the just mentioned anomalies. In fact, in that case, the Sobol mapping algorithm would have an higher number of individuals to sample the design space and consequently, the differences, between the usage of the MTPA tool or not, should be enlarged even more. Nevertheless, this choice, in posterity, has been appropriate as the time required to perform the MTPA-related simulation was almost four times longer than that occupied by the one without this tool.

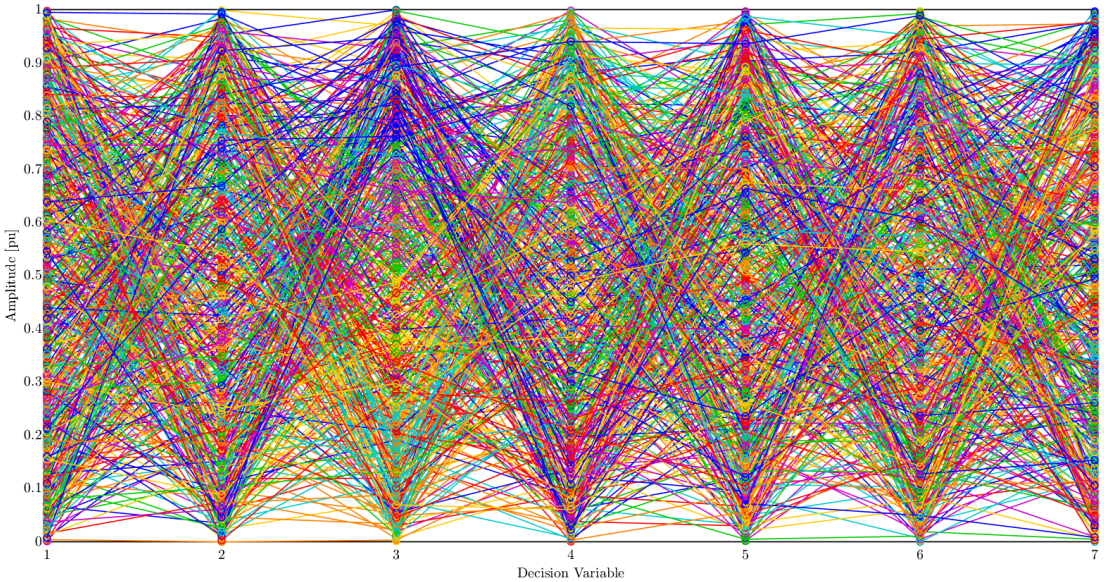
	Simulation 1 (No MTPA)	**Simulation 2 (Yes MTPA)**
Min. Value	1.3628	1.3002
Max. Value	22.2090	21.5976
Average Value	9.5429	9.5420

Table 4.1: Sobol MTPA Tool Simulations Statistics

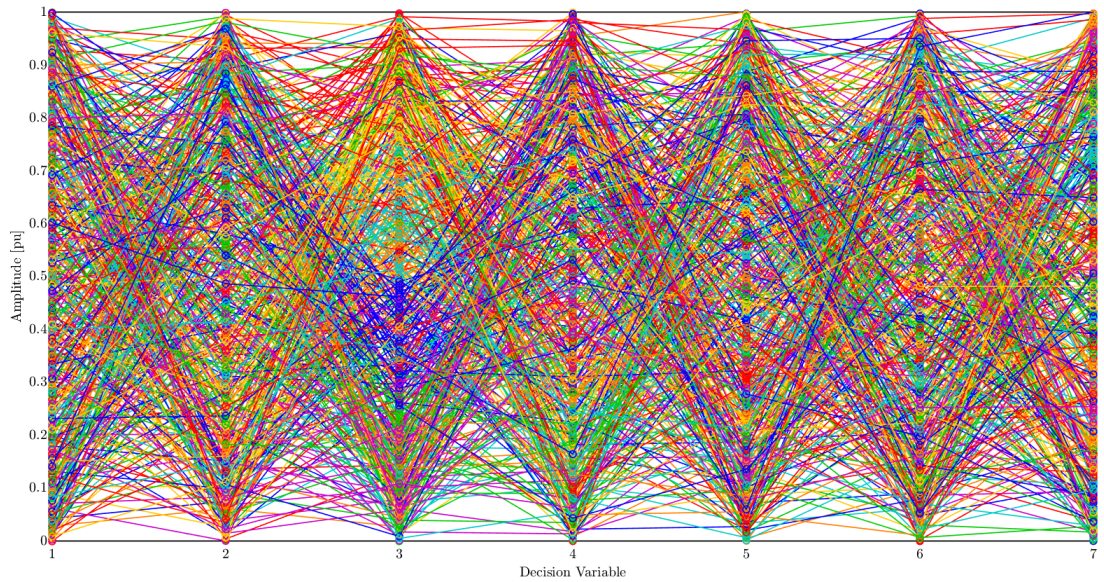
Anyway, the predicted trend is confirmed also by observing the statistical values reported in Tab. 4.1. In fact, this table summarizes, between all the calculated distances of the two simulations, the minimum, maximum and average amplitudes found. Confronting the single values obtained from the simulations, it is clear that the second one (obtained using the automatic MTPA research) has better performances than the first. In fact, all three values analyzed were lower. This confirms that the points obtained from the second simulation are closer between each other, consequently they are more densely distributed in the design space analyzed, even if both the space and the number of elements under study are the same.

These differences would be even more evident if the same study cases were repeated using a larger population, which was not done due to both time and computational power limitations.

Summarizing, it can be affirmed that the implementation of the automatic MTPA tool was successful as no errors took place during the simulation runs. In addition, the expected benefits, in terms of sampling density, carried by the development of this feature have found an experimental correlation with the data obtained from the two Sobol analysis.



(a) Sobol Simulation Normalized Decision Variables' Amplitudes Without Automatic MTPA Research



(b) Sobol Simulation Normalized Decision Variables' Amplitudes With Automatic MTPA Research

Figure 4.3: Sobol Mapping Decision Variables' Normalized Amplitudes.

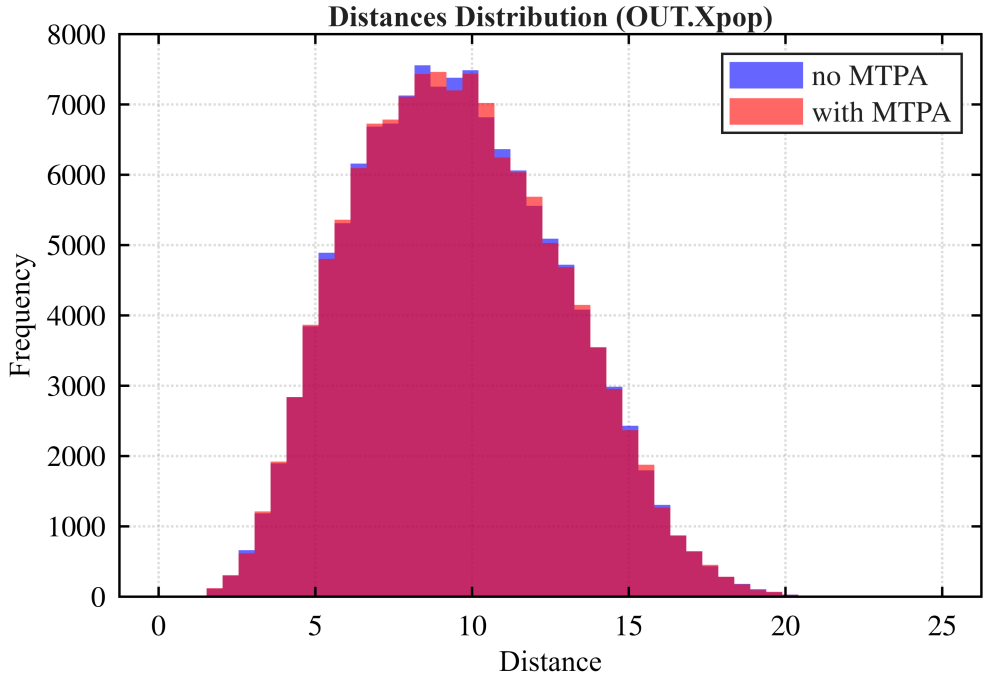


Figure 4.4: Comparison of points distances during Sobol mapping with and without MTPA search

5. Conclusions and Future Works

In conclusion, this thesis has touched on several arguments, starting from an overview on the current transport system, including its limitations and future goals, then developed by introducing the real subjects of the analysis, the EESM machine and SyR-e software. Consequently, a more detailed discussion about electrical machine design optimization techniques has been developed.

Once the theoretical bases were depicted, the discussion's focus shifted towards the description of the more practical section performed.

The first topic regarded the steps behind the implementation of the EESM machine in the SyR-e software. This procedure has been divided into three smaller sections for an easier and more organized comprehension. The first one concerns the geometrical aspects behind the introduction of this new electrical machine topology in SyR-e, in specific the addition of new parameters and the steps behind its drawing for FEA purposes. The second regards the implementation of electrical parameters unique to the EESM machine, which has been possible thanks to the development of *ad hoc* Matlab functions and the adaptation of several pre-existing ones. Lastly, also the GUI implementation of this new electrical machine topology has been discussed, due to the fact that this is a crucial tool for SyR-e users as it allows for a more immediate and intuitive analysis setup, saving a considerable amount of time.

At this point, proper verifications of the correctness of the previously described steps were necessary to validate the procedure. In order to do so, it has been decided to perform two different simulations. The first one consisted in launching an optimization problem on an EESM adaptation of an electrical machine already present in the SyR-e library. Instead, the second regards two runs of the Sobol mapping algorithm on the same machine used in the first.

Both analysis concluded successfully without interruptions due to errors. Consequently, it can be concluded that both geometrical and electrical implementations of the EESM machine were successful. In fact, even if the quality of the output data generated from the simulations was not satisfying, on an electrical machine design point of view, the main objective of these analysis was to verify if the designed workflow worked properly, not to design a new machine.

Once the EESM related verifications were successfully completed, the focus of this discussion shifted towards the development of a new tool for the SyR-e software, the automatic MTPA angle research. This additional feature allows to autonomously determine the MTPA angle γ of the machine under analysis, even when it is not selected as decision variable in the simulation setup, and it is compatible with every topology and type of analysis present in SyR-e.

First, the steps leading to the development of this additional feature and its Matlab code were reported and described in detail. Then, this procedure has been verified by launching two separate Sobol mapping simulations, one selecting γ as variable of interest and the second exploiting this new SyR-e tool. This choice was made to have a direct confrontation between the data delivered by the two simulation runs, in such way to highlight the main differences and observe the benefits introduced by the automatic MTPA tool. Both analysis were performed on the adapted EESM machine used for the previous study cases. However, the number of individuals studied was smaller this time due to limitations in the computational power available.

The two simulations ended successfully, without delivering unexpected errors. Consequently, it can be concluded that the implementation of this new tool was performed correctly.

At this point, the simulations' output data was processed in such a way that the main differences between the two were more immediate to observe. In fact, at first a figure containing the normalized amplitudes of the analysis' common decision variables was generated. From this element, it was possible to see how the simulation using the MTPA tool delivered a more dense design space sampling, using the same number of elements, than the first one. However, due to the fact that this feature was not clearly visible enough, it has been decided to generate a second figure showing the distributions of the distances between the points delivered by the two studies. This has been possible by appositely developing a Matlab code performing such task. A confrontation between the figures shows how, even if the distributions are quite similar in shape, the one related to the simulation using the new MTPA tool generally has lower values of distances than the one that is not, with the exception of a few anomalous points. The same pattern can be observed also when the overall values of minimum, maximum and average distances of the two analysis are confronted. In fact, the second one has lower amplitudes in all three categories.

Consequently, it can be concluded that the development of the automatic MTPA research tool was successful. In fact, it introduces benefits for every kind of analysis performed in SyR-e. In the optimization problems case, it allows to not waste iterations in the research of the MTPA angle γ, leading to a more efficient exploitation of resources towards the fulfillment of the analysis objectives. However, the type of analysis benefiting the most from this addition is the Sobol space mapping one. In fact, the introduction of this features allows this space sampling algorithm to perform a better design space exploration, at parity of space limits, compared to the one executed previously as no resources are wasted in looking for the MTPA angle. Consequently, the space sampling performed is more dense leading to a more accurate dataset delivered as output.

Nevertheless, these improvements in terms of output data quality comes at a cost as the computational time of the simulation increases by approximately four times compared to the time achieved without using this new tool.

Concluding this discussion, it can be stated that the main objectives of this thesis were fulfilled as both the outputs obtained from the EESM verification simulations and the development of the automatic MTPA research tool gave results in line

with the expectations. However, further analysis on both subjects might still be interesting to perform. In fact, if it is true that the EESM machine has been successfully implemented on a geometrical and electrical points of view, SyR-e is able to execute analysis also on thermal and mechanical aspects of the electrical machines, which are still missing from the current EESM model. In fact, the just mentioned limitations forced the removal of the thermal and mechanical analysis from being performed in the Sobol mapping simulations, both in the case of the EESM verification and for the MTPA tool assessment. Consequently, it would be intriguing to repeat the simulations when these additional aspects of analysis are added. In addition, in the MTPA feature case, it would be interesting to perform additional simulations increasing the size of the populations analyzed also. In such way, the benefits introduced in terms of quality of the output dataset and in sampling density executed should become even more evident. However, a compromise between the number of individuals under study and the computational time required to perform the analysis should be found for it not to be excessively long. Nevertheless, these additional aspects are left as hints for future discussions.

Bibliography

[1] Niels Aage, Erik Andreassen, and Boyan S. Lazarov. Topology optimization using the method of moving asymptotes. 2015.

[2] Mokhtar S. Bazaraa and John J. Jarvis. Linear programming and network flows. 1977.

[3] N. Bianchi and S. Bolognani. Brushless dc motor design: an optimisation procedure based on genetic algorithms. 1997.

[4] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. 2003.

[5] P.T. Boggs and J.W. Tolle. Sequential quadratic programming. 1995.

[6] Visual Capitalist. Life-cycle emissions of electric, hybrid, and ice cars. 2024.

[7] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. Evolutionary algorithms for solving multi-objective problems. 2007.

[8] George B. Dantzig. Linear programming and extensions. 1963.

[9] World Energy Data. Climate and energy summary. 2025.

[10] Raksha Sharma DataIntelo. Ev traction motor core market. 2023.

[11] K. Deb. Multi-objective optimization using evolutionary algorithms. 2001.

[12] J.C. Minx F.L. Toth A. Abdel-Aziz M.J. Figueroa Meza K. Hubacek I.G.C. Jonckheere Yong-Gun Kim G.F. Nemet S. Pachauri X.C. Tan T. Wiedmann Dhakal, S. 2022: Emissions trends and drivers. 2022.

[13] G. Pellegrino F. Cupertino and C. Gerada. Design of synchronous reluctance motors with multi-objective optimization algorithms. 2014.

[14] S. Ferrari. Syr-e user manual. 2023.

[15] Directorate-General for Research and Innovation (European Commission). European green deal. 2021.

[16] P. M. Forster, C. Smith, T. Walsh, W. F. Lamb, R. Lamboll, C. Cassou, M. Hauser, Z. Hausfather, J.-Y. Lee, M. D. Palmer, K. von Schuckmann, A. B. A. Slangen, S. Szopa, B. Trewin, J. Yun, N. P. Gillett, S. Jenkins, H. D. Matthews, K. Raghavan, A. Ribes, J. Rogelj, D. Rosen, X. Zhang, M. Allen, L. Aleluia Reis, R. M. Andrew, R. A. Betts, A. Borger, J. A. Broersma,

S. N. Burgess, L. Cheng, P. Friedlingstein, C. M. Domingues, M. Gambarini, T. Gasser, J. Gütschow, M. Ishii, C. Kadow, J. Kennedy, R. E. Killick, P. B. Krummel, A. Liné, D. P. Monselesan, C. Morice, J. Mühle, V. Naik, G. P. Peters, A. Pirani, J. Pongratz, J. C. Minx, M. Rigby, R. Rohde, A. Savita, S. I. Seneviratne, P. Thorne, C. Wells, L. M. Western, G. R. van der Werf, S. E. Wijffels, V. Masson-Delmotte, and P. Zhai. Indicators of global climate change 2024: annual update of key indicators of the state of the climate system and human influence. 2025.

[17] D.E. Goldberg. Genetic algorithms in search, optimization, and machine learning. 1989.

[18] F. Graffeo. Modeling and design of wfsm for transportation applications. 2023.

[19] J.H. Holland. Adaptation in natural and artificial systems. 1975.

[20] Anders Holmström. Derivative-free algorithms in engineering optimization. 2000.

[21] Georg Bieker ICCT. A global comparison of the life-cycle greenhouse gas emissions of combustion engine and electric passenger cars. 2021.

[22] IEA. Walking the torque. 2011.

[23] IEA. Comparative life-cycle greenhouse gas emissions of a mid-size bev and ice vehicle. 2021.

[24] IEA. Transport – energy system. 2023.

[25] IEA. World energy outlook. 2023.

[26] IEA. Global ev outlook for 2025. 2024.

[27] Our World in Data. Climate watch (2025) – with major processing by our world in data - greenhouse gas emissions by sector. 2025.

[28] J.Kennedy and R.Eberhart. Particle swarm optimization. 1995.

[29] S.Agarwal K. Deb, A.Pratap and T.Meyarivan. A fast and elitist multi-objective genetic algorithm: Nsga-ii. 2002.

[30] Wissam Lahoud and Elias et Al. Khalil. Predict-and-optimize techniques for data-driven optimization problems: A review. 2025.

[31] Remigijus Paulavičius Linas Stripinis. An extensive numerical benchmark study of deterministic vs. stochastic derivative-free global optimization algorithms. 2022.

[32] Xiaodong Liu, Huaizhou Qi, Suisui Jia, Yongjing Guo, and Yang Liu. Recent advances in optimization methods for machine learning: A systematic review. 2025.

[33] P. Asef M. Vatani, D. R. Stewart and D. M. Ionel. Optimal design of coreless axial flux pm machines using a hybrid machine learning and de method.

[34] S.Bolognani N. Bianchi and M. Dai Pre. Design considerations for fractional-slot permanent magnet machines. 2004.

[35] United Nations Framework Convention on Climate Change. Kyoto protocol to the united nations framework convention on climate change. 1997.

[36] W. Murray P. E. Gill and M. H. Wright. Practical optimization. 1982.

[37] P. Gottipatti M. Solveson P. Han, J. Liang. Design and analysis of em assisted by scientific ml.

[38] R.H. Park. Two-reaction theory of synchronous machines generalized methoo of analysis part-i. 1929.

[39] Zhitong Ran, Zi Qiang Zhu, Zhiqian Chen, Matthew Younkins, and Philippe Farah. Novel system level driving cycle oriented design co-optimization of electrical machines for electrical vehicles. 2024.

[40] REN21. Renewables global status report 2024. 2024.

[41] Sebastian Ruder. An overview of gradient descent optimization algorithms. 2016.

[42] G. Pellegrino S. Ferrari. Torque ripple optimization of pm-assisted synchronous reluctance machines via assymetric rotor poles. 2019.

[43] R. K. Salgotra, R. Rani, and A. Abraham. Gradient-based optimizer (gbo): A review, theory, variants, and applications. 2022.

[44] M. Schoening. Automated electrical machine design with differential evolution techniques. 2011.

[45] Longjie Sun. Application of improved multi-population genetic algorithm in structural optimization of automotive electrical equipment. 2024.

[46] Krister Svanberg. The method of moving asymptotes—a new method for structural optimization. 1987.

[47] Krister Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. 2002.

[48] B. B. Monchusi T. H. Mokwana. Empowering electrical machine performance: Convergence of ai and motor design. 2024.

[49] E.G. Talbi. Metaheuristics: From design to implementation. 2009.

[50] El-Ghazali Talbi. A taxonomy of hybrid metaheuristics. 2002.

[51] Xuecheng et Al. Tian. Data transformation in the predict-then-optimize framework: Enhancing decision making under uncertainty. 2023.

[52] UNFCCC. Paris climate change conference - november 2015. 2018.

[53] V. Yesina M. Sukhonos V. Pliushin, M. Pan. Using azure ml cloud technology for em optimization.

[54] Wikipedia. Derivative-free optimization.

[55] Wikipedia. Newton's method in optimization.

[56] Y. Zhou, W. Sun, and B. Li. Topology optimization of electric machines based on finite element analysis and the method of moving asymptotes. 2018.

[57] Z. Q. Zhu and D. Howe. Electrical machines and drives for electric, hybrid and fuel cell vehicles. 2007.

[58] J.H. Kim Z.W. Geem and G.V. Loganathan. A new heuristic optimization algorithm: Harmony search. 2001.

[59] Mehmet Çunkaş and Ramazan Akkaya. Design optimization of induction motor by genetic algorithm and comparison with existing motor. 2006.