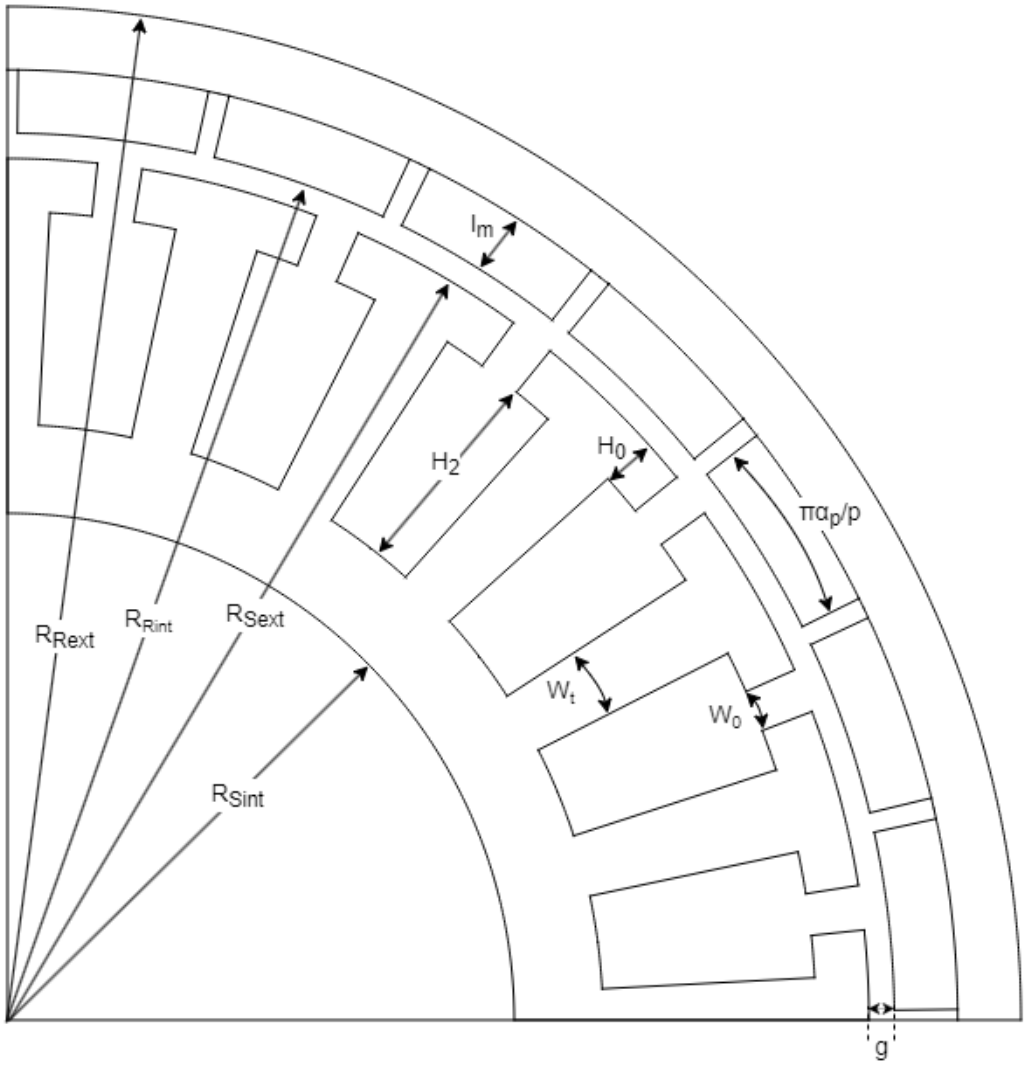
Multi-Objective Optimisation Framework for Electrical Machines based on Open Source Platforms

Ansh Anil Rajdev



TU Delft

Multi-Objective Optimisation Framework for Electrical Machines based on Open Source Platforms

by

Ansh Anil Rajdev

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 25, 2022 at 14:00.

Student number: 5233909
Project duration: October 19, 2021 – August 25, 2022
Thesis committee: Dr. P. Bauer, TU Delft
Dr. J. Dong, TU Delft
Dr. D. Ragni, TU Delft

TU Delft

Acknowledgements

It has been a wonderful two years studying at TU Delft. Working on this thesis has been a fun challenge which I could not have completed without the help and support I got from the people here at TU Delft.

I would like to extend my deepest gratitude to my supervisor Dr Jianning Dong who has been a constant support and has guided me through the process. He has always been available to answer my doubts and questions. I would also like to thank Dr Pavol Bauer and Dr Daniele Ragni for being on my thesis committee and providing me with such an interesting problem to work on.

I would like to thank my family, my Mom, Dad and Sister who have constantly provided me with support and encouragement throughout this thesis. I would also like to thank my friends who often were echo chambers for me to process my ideas and were patient enough to hear me ramble about python errors.

Ansh Anil Rajdev

Abstract

The design of electrical machines is complicated due to the number of variables involved and due to competing objectives like efficiency, weight and cost. Another important aspect is the involvement of different physical phenomena such as torque production, electromagnetic fields and thermal heat flow, Thus designs need to satisfy multiple constraints and fulfill competing objectives making the design process tedious. Multi-Objective Optimisation algorithms provide a set of designs that are Pareto optimal i.e. any improvement in one objective comes at the cost of performance in another objective. This gives the designer a set of designs with different trade-offs between objectives and they can choose the design that satisfies the objectives and constraints the best.

There are numerous commercial software available that provide these functionalities. However the methods used to model machines within these packages are not available or cannot be changed. They are also usually expensive. This makes the exploration of new limits and new topologies difficult. Using open source packages allows us to modify these methods according to our application and gives greater control over the process.

This thesis uses PYLEECAN python library that is based on FEMM software to perform the analysis of the machine. A six time-step magnetostatic analysis method to calculate the average torque and iron losses in the stator and rotor core is presented along with methods to calculate the copper losses and windage losses. A steady state Lumped Parameter Thermal Network (LPTN) is developed to calculate the temperatures of various parts of the machine. The LPTN is capable of estimating the temperatures under natural convection and forced air cooling conditions.

Finally a MOO framework was developed using the models developed and the PYMOO python library. This thesis uses NSGA-II to perform the MOO. The MOO framework was used to optimise the design of a machine for a drone application and explore the specific power density limit of the machine. The power density limit was found to $5 - 7$ kW/kg based on different slot pole combinations, winding temperature limits and core material used. Further, insights into how different machine parameters affect the specific power density are presented.

Contents

1 Introduction

1.1. Background

The design of electrical machines is complicated due to the number of variables involved and due to competing objectives including efficiency, weight and cost. Another important aspect is the involvement of different physical phenomena such as torque production, electromagnetic fields and thermal heat flow, which makes it important to have multiple models to predict these behaviours. Currently, there are various commercial packages available that provide these functionalities. However, these packages have expensive licenses making dedicating these resources to exploring newer boundaries and designs difficult. Furthermore, since these software packages are proprietary, the methods used to perform the calculations for various models are either unknown or cannot be changed. Having an open-source software package with a free license can help us overcome these problems and allow us to have greater control over the models.

Electrical machines are the movers of our world. With applications ranging from small drone motors to large wind turbine generators, electrical machines have increasing penetration in the industries of the world. This has been possible mainly due to the development in power electronics in recent years allowing us to design machines that can operate at higher frequencies and torque. One of these applications is in the aerospace domain. The development of electric aircraft as well as drones has led to tighter requirements for specific torque density as well as higher efficiencies to increase the flight time from the same battery size [1]–[4].

To fulfil this demand traditional methods of machine design are not enough [5], [6]. Current machine design methods rely on the experience of the designer to reduce the design space to manageable levels and use insights developed with experience to decide the flow of the optimisation. However, this makes the entry of new individuals as well as the development of new topologies difficult. A better method of analysing the design space to reduce choices is required. This makes use of Multi-Objective Optimisation (MOO) an attractive option. MOO techniques can be used to explore the design space systematically and derive insight into what parameter choices lead to better machines.

There have been many projects that use MOO to achieve various objectives for different machine topologies [7]–[9]. [10] provides an overview of various developments where MOO has been used for electrical machine designs. These developments use analytical or numerical calculations to model the machines. Numerical analysis especially for the magnetic field distribution is preferred over analytical equations due to the non-linear nature of magnetic field distribution as well as lower accuracy due to simplifying assumptions made in analytical equations [11]. However numerical methods have high time complexity and could lead to large evaluation times for each generation of MOO which is undesirable.

Completing one MOO of electrical machines using numerical methods to solve for electromagnetic fields can take 1-2 days [8] or even up to a week [10] using high end computers. Another method used is the computationally efficient modelling method where the two techniques are combined by utilizing inherent symmetries in the machine and using space-time transformations to extract total information from a small part of the machine that is simulated [12]. This method, however, is problem specific and cannot be extended to the evaluation of different designs for example for different slot pole combinations and is hence not useful for developing a generic framework. However, it could lead to large improvements in time complexity for problems where it can be used. Thus numerical methods are used in this thesis specifically 2D Finite Element Analysis for its good compromise between time complexity and accuracy of results.

There are many MOO algorithms available that can be used based on the optimisation that has to be run. Population-based MOO algorithms have been gaining popularity for the design of electrical machines as they do not need to calculate gradients and can handle a large variety of objective functions [10]. These algorithms instead of providing a single result provide a set of results that are non-dominated and Pareto optimal. Pareto optimal solutions are solutions for which any improvement in one objective comes at the cost of loss of performance in another objective.

The flowchart shown in figure 1.1 gives a general idea of how population-based MOO algorithms work. A new generation of solutions is created by using the surviving population of the previous generation. For the first iteration, a random parent population is chosen. We then evaluate these solutions on the objective criteria and check the constraint conditions. This is followed by implementing survival criteria that choose better performing solutions while maintaining diversity in the population to avoid local convergence. The surviving population is then used to create a new generation and this process is continued till the exit condition of the algorithm is reached.

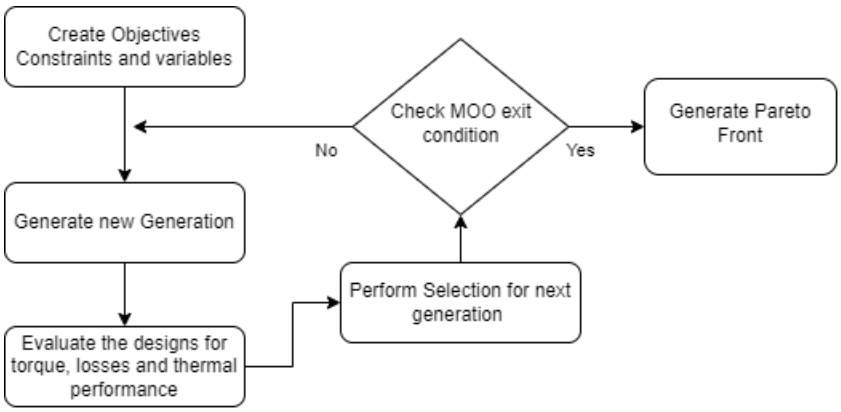


Figure 1.1: Flow chart of MOO Routine

Since the algorithm choice is dependent on the application it is being used for, no particular algorithm is chosen. Non-Dominated Sorting Genetic Algorithm (NSGA-II) is used to demonstrate how the framework works but this can be easily changed based on requirements or preferences.

1.2. Research Focus

This thesis focuses on the development of a MOO framework for electrical machines based on multi-physical models. This framework developed is then used to gain insights towards design trade-offs of electrical machines for drones. The models developed are applicable for Surface Mounted Permanent Magnet Synchronous Machines to limit the scope of the thesis. However, efforts have been made to write the code in a modular way such that snippets can be reused to develop models for other topologies. Furthermore, the development of the power electronics drives as well as the mechanical design and structural analysis is beyond the scope of this thesis.

1.3. Thesis Objective

This thesis originates from a desire to develop a MOO framework that can be used to explore the limits of performance of existing topologies as well as new machine topologies. This framework is based on open-source software packages to make it accessible for all. The framework needs to accurately analyse hundreds of machine designs in a short time. To achieve this goal three objectives have to be fulfilled:

- Develop multi-physical models of electrical machines with low time complexity;
- Design a multi-objective optimisation framework based on the multi-physical models;
- Explore the specific power density(kW/kg) for an aerospace application using the developed framework.

1.4. Research Approach and Thesis Layout

The thesis is divided into six chapters:

- **Chapter 1** covers the Introduction which provides the motivation and objectives of the thesis.
- **Chapter 2** deals with the development of the multi-physical models and the calculations performed to deliver the torque, losses and thermal distribution of the machine.
- **Chapter 3** verifies the multi-physical models using previous studies and also checks the viability of the thermal model.
- **Chapter 4** deals with the MOO and its implementation.
- **Chapter 5** presents the results obtained from the MOO and presents insights into the design of drone machines.
- **Chapter 6** provides a conclusion of the work performed and recommends future developments that can be performed.

2

Multi-Physical Modelling of an Electrical Machine

This chapter introduces the methods used to evaluate machine designs by considering multi-physical performance indicators. We begin by performing electromagnetic analysis of the magnetic field distribution in the machine using 2D Finite Element Analysis. This is then used to calculate the losses in the machine which can be used to calculate the efficiency of the machine as well as the thermal performance. Then we perform thermal calculations to complete our evaluation.

In the coming sections, the analysis of the machine is performed in the synchronous reference frame using park transformation [13]. Using the park transformation is advantageous as it simplifies the analysis by converting sinusoidal variables in the stationary reference frame to DC variables in the synchronous reference frame which is rotating at synchronous speed. The amplitude invariant form of park transformation is used in this analysis.

2.1. Magnetic Model

Estimating the magnetic field distribution inside an electrical machine is a complex task due to the non-linear BH curve for the core iron as well as a large number of design variables. There are analytical methods that use simplifications and iterative calculations to calculate the magnetic field using magnetic circuit theory [14]. However, this method does not yield any information about the MMF harmonics and leakage flux. It is usually used at the beginning of the design process to get an estimate of the machine's dimensions.

With increasingly powerful computers being available numerical methods have become preferable compared to analytical calculations for detailed machine analysis. The geometry of the machine is defined and broken into smaller cells (often triangular) called a mesh as shown in the example of figure 2.1. We then solve Poisson's equation numerically to get the magnetic field distribution [15]. However numerical analysis methods are time-consuming and therefore a trade-off between performance and time complexity has to be considered especially when performing optimisation. A 3D Finite Element Analysis (3D-FEA) provides accurate information about the machine and a complete model of the machine can be created. 2D-FEA ignores the end effects on the magnetic field as well as the end windings cannot be modelled in 2D but the time complexity is significantly lower. This makes 2D-FEA much more attractive for MOO. This thesis uses 2D-FEA to perform electromagnetic analysis.

There are numerous finite element solvers available commercially as well as open source software which can fulfil our requirements. The next section discusses the choice of solver in detail.

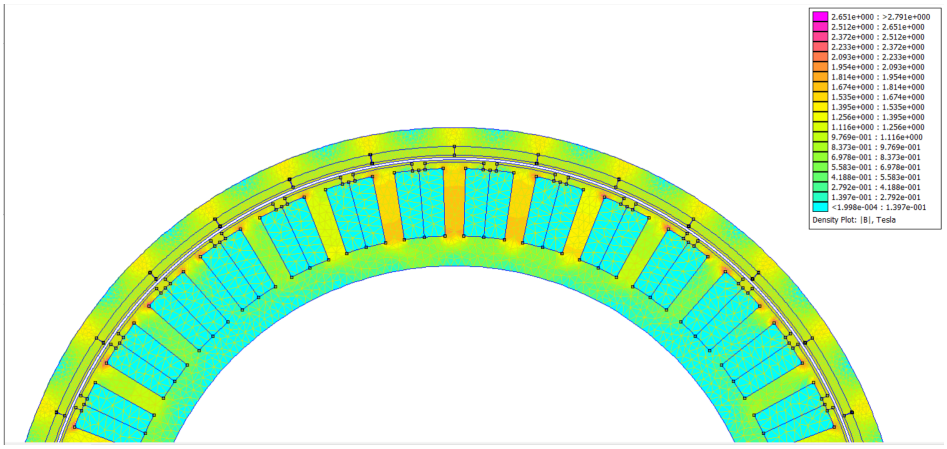


Figure 2.1: 2D FEA model of an external rotor PMSM showing the geometry and meshing created using PYLEECAN

2.1.1. Choice of Solver

Since one of the aims of this thesis is to develop an open-source optimising platform for researchers it is important that the software packages used are free and open-source. Commercial packages are often robust and require little to no coding knowledge and thus are easy to use. But commercial packages are often restrictive in their use as the calculation methods for various features are either not available or cannot be changed [16]. By using an open source package we have the source code of the software available which provides us with greater control over the execution and opens up the possibility of performing parallel computations which may end up saving computation time [16]. Moreover, for research purposes, it is often required to implement new models or methods, which is impossible to do in closed commercial packages.

FEMM is one of the most commonly used open source FE solvers available [17]. FEMM also supports use through MATLAB which can be used to perform MOO using existing libraries available in MATLAB and utilize MATLAB for post-processing and visualisation [16], [18]. There are some open source software packages available that provide support for creating the geometries of specific machine types [17], [19]. However, this still leaves the complex task of generating the geometries and assigning materials to the different parts of the machine to the user which might be time-consuming when comparing across machine types.

Smeklib is a MATLAB toolbox that provides 2D Finite Element Analysis ability [20]. However, the problem of generating machine geometry is not solved and is only available on a subscription basis. PYLEECAN is a python library which also uses FEMM as its FE solver but easily generates the machine geometry from the design variables [21]. The library is currently under development and provides online support as well. There are some other libraries which can be used but their support is found to be lacking [22], [23]. This makes it the ideal choice for this project as we can directly focus on the optimisation without having to worry about geometry generation. The library further has post-processing scripts which are open source making it easy for the user to customise them to their particular use case. Table 2.1 provides a further comparison of PYLEECAN with some other options available.

Table 2.1: Comparison of PYLEECAN with some other open source software

Criteria	PYLEECAN	xfemm	SMEKlib
Geometry Generation from machine parameters	Yes	No	No
Object-oriented programming for pick and plug application	Yes	No	No

Although PYLEECAN supports the evaluation of different kinds of machines such as Induction Machine, Synchronous Reluctance Machines, Interior Permanent Magnet Machines and so on and even provides a way to define new topologies [21], the developments in this thesis have been limited to Surface Mounted Permanent Magnet Synchronous Machines (SPMSM) since it is the most promising topology

used in drones [2], [4], [24], [25]. However, these multi-physical models can be easily extended to other topologies either directly or with small modifications.

The code used for generating the machine design is presented in appendix A. This is used in further sections to generate the models that are being discussed and processed for various calculations.

2.1.2. Flux Linkage Calculation

The flux linkage of any contour is defined as the total magnetic flux passing the contour.

$$\Psi = \int_S \vec{B} \cdot \vec{dS} \tag{2.1}$$

For the context of coils, the flux linkage is defined as

$$\Psi = T_c \int_S \vec{B} \cdot \vec{dS} \tag{2.2}$$

Where Ψ is the flux linkage of the winding, T_c is the number of turns in series per coil, $\vec{B}$ is the magnetic flux density and S is the total surface area enclosed by the windings. PYLEECAN directly provides a function to calculate this for each static FEA performed.

The flux linkage of a winding depends on the permeance seen by the magnetic field and the turns distribution of the winding. Ideally, we assume that the turns are placed in the air gap directly and the winding turns are sinusoidally distributed. This gives us a sinusoidal flux linkage. However in reality the turns are placed in slots cut in the stator instead of the air gap to keep the air gap reluctance low [26]. This means that the air gap permeance is a function of rotor position. Also, the turns cannot be realistically placed sinusoidally are placed in bunches in the slots. This leads to the flux linkage having higher harmonics as well. These higher harmonics do not contribute to torque production as their torque output averages to zero over one electrical cycle. They increase the losses and vibrations in the machine as well as the torque ripple. In [27], the authors discuss that the most significant harmonics present in the flux linkage are of the 6^{th} order and its multiples in the synchronous reference frame.

Figure 2.2 shows the d axis flux linkage in the synchronous reference frame for an Outer Rotor Surface Mounted Permanent Magnet Synchronous Motor described in section 3.1 over one electrical cycle. It can be seen that the flux linkage has 6^{th} order harmonics and its multiples. Since our goal is to analyse hundreds if not thousands of designs, the addition of even one more time step could lead to an increase of many minutes of evaluation time in the MOO routine. Thus we must perform sufficient analysis to get accurate enough results and not more than that.

If we sample the flux linkage such that the harmonics present are antiperiodic they will be eliminated when we average the flux linkage across the samples. For example let us consider that the d-axis flux linkage has zeroth harmonic Ψ_{d0} and 6^{th} and 12^{th} harmonics have peaks Ψ_{d6} and Ψ_{d12} and phase angle ϕ_{d6} and ϕ_{d12} respectively. The d-axis flux linkage as a function of rotor position (in electrical radians) θ^e is then given by

$$\Psi_d(\theta^e) = \Psi_{d0} + \Psi_{d6} e^{j(6\theta^e + \phi_{d6})} + \Psi_{d12} e^{j(12\theta^e + \phi_{d12})} \tag{2.3}$$

If we sample at rotor positions (in electrical radians) $\theta^e = [0, \frac{\pi}{12}, \frac{\pi}{6}, \frac{\pi}{4}]$

$$\Psi_d(0) = \Psi_{d0} + \Psi_{d6} e^{j\phi_{d6}} + \Psi_{d12} e^{j\phi_{d12}} \tag{2.4}$$

$$\Psi_d(\frac{\pi}{12}) = \Psi_{d0} + \Psi_{d6} e^{j(\frac{\pi}{2} + \phi_{d6})} + \Psi_{d12} e^{j(\pi + \phi_{d12})} \tag{2.5}$$

$$\Psi_d(\frac{\pi}{6}) = \Psi_{d0} + \Psi_{d6} e^{j(\pi + \phi_{d6})} + \Psi_{d12} e^{j(2\pi + \phi_{d12})} \tag{2.6}$$

$$\Psi_d(\frac{\pi}{4}) = \Psi_{d0} + \Psi_{d6} e^{j(\frac{3\pi}{2} + \phi_{d6})} + \Psi_{d12} e^{j(3\pi + \phi_{d12})} \tag{2.7}$$

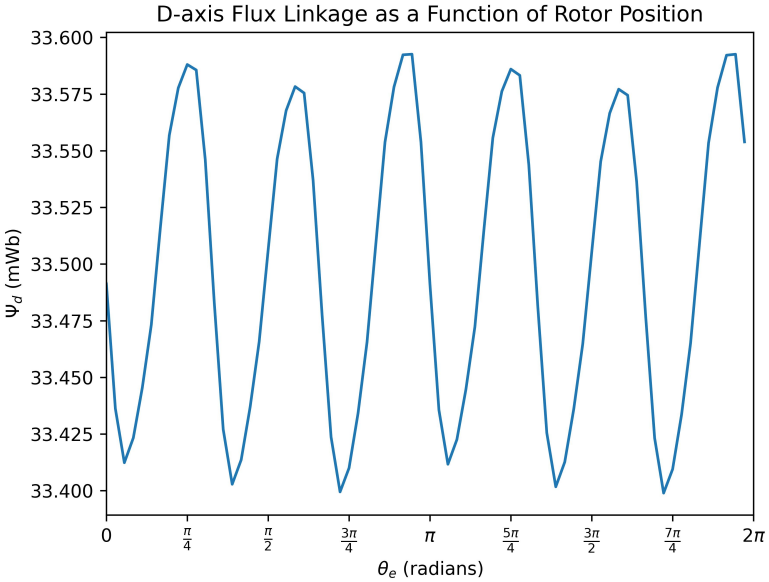


Figure 2.2: D-axis flux linkage in the synchronous reference frame for an outer rotor SPMSM

Adding the four equations we get

$$\Psi_d(0)+\Psi_d(\frac{\pi}{12})+\Psi_d(\frac{\pi}{6})+\Psi_d(\frac{\pi}{4}) = 4\Psi_{d0}+\Psi_{d6}e^{j\phi_{d6}}(1+j-1-j)+\Psi_{d12}e^{j\phi_{d12}}(1+-1+1-1) \quad (2.8)$$

$$\Psi_{d0} = [\Psi_d(0)+\Psi_d(\frac{\pi}{12})+\Psi_d(\frac{\pi}{6})+\Psi_d(\frac{\pi}{4})]/4 \quad (2.9)$$

Similarly, we can calculate for q-axis flux linkage

$$\Psi_{q0} = [\Psi_q(0)+\Psi_q(\frac{\pi}{12})+\Psi_q(\frac{\pi}{6})+\Psi_q(\frac{\pi}{4})]/4 \quad (2.10)$$

We can see that the 6^{th} and 12^{th} harmonics are eliminated from the average for the chosen rotor positions. An example of this sampling is shown in figure 2.3.

In electrical machines, only the zeroth flux linkage components in the synchronous reference frame lead to useful torque, so it is important to extract them from all harmonics. This derivation can also be extended to torque calculations as shown in the next section.

2.1.3. Torque Calculation

The torque production in a PMSM is due to the interaction of the magnetic field produced by the magnets and the copper coils. The relation between torque and input current is given by

$$T_{em} = \frac{3}{2}p\vec{\Psi}\times\vec{I} \quad (2.11)$$

where T_{em} is the electromagnetic torque, p is the number of pole pairs,$\vec{\Psi}$ is the flux linkage space vector and $\vec{I}$ is the current space vector. For SPMSM with maximum torque per ampere control, this equation simplifies to

$$T_{em} = \frac{3}{2}p\Psi_M I_q \quad (2.12)$$

Where Ψ_M is the flux linkage due to permanent magnets and I_q is the q axis current both in the synchronous reference frame.

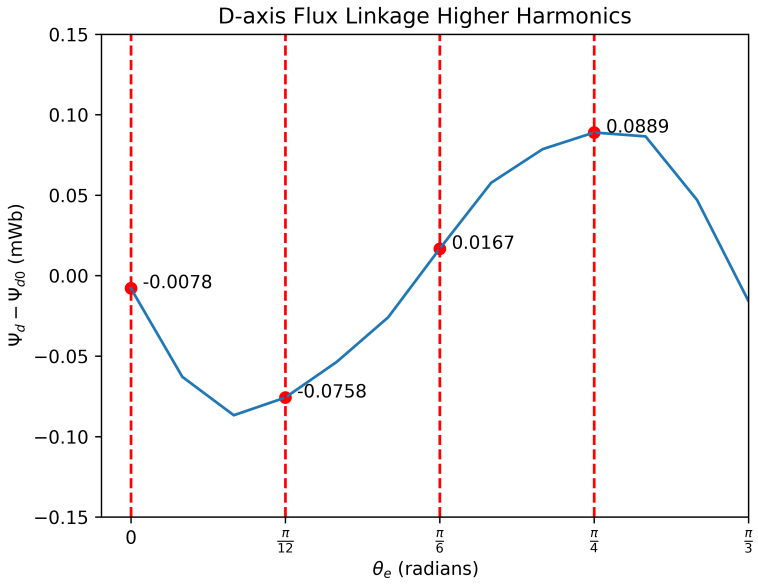


Figure 2.3: D-axis flux linkage higher harmonics with specified sampling points

Another method for calculating the torque is to use Maxwell's stress tensor [28]. Since we know the airgap flux density in the radial and tangential direction from the 2D-FEA this is an easy way to calculate the torque.

$$T_{em} = \frac{l}{\mu g} \int_S r B_r B_{tan} dS \tag{2.13}$$

Where l is the effective stack length, g is air gap thickness, r is the average bore radius, B_r and B_{tan} are radial and tangential magnetic field density and S is the area composed of the air gap in the plane of the machine.

If we consider that the input current is purely sinusoidal we can then see that the harmonics of torque will follow the harmonics of the flux linkage. The assumption of input current being sinusoidal is not always true since power electronics will introduce time harmonics, however, in the design process, we only consider the average torque hence higher harmonics are neglected. Thus we can calculate the average torque delivered as

$$T_{em,av} = [T_{em}(0) + T_{em}(\frac{\pi}{12}) + T_{em}(\frac{\pi}{6}) + T_{em}(\frac{\pi}{4})]/4 \tag{2.14}$$

Where $T_{em}(\theta^e)$ is the torque calculated from 2D-FEA simulation results using Maxwell's stress tensor shown in equation (2.13).

2.1.4. Back EMF Calculation

The back EMF in an electrical machine is present due to the varying magnetic field and is defined as

$$E = \frac{d\Psi_p}{dt} \tag{2.15}$$

Where Ψ_p is the flux linkage of one phase. We have already calculated the zeroth harmonic of the flux linkage, since the flux linkage is sinusoidal the equation can be simplified as

$$\Psi_p = \sqrt{\Psi_{d0}^2 + \Psi_{q0}^2}$$

$$E = \omega_0 \Psi_p \tag{2.16}$$

$$\omega_0 = 2\pi f_{elec} = \pi p \frac{n}{30}$$

where f_{elec} is the electrical frequency, p is the number of pole pairs and n is rotor speed in RPM.

2.2. Loss Calculations

Loss calculations is one of the most important evaluation performed when designing electrical machines, not only because they help us estimate the efficiency of the machine but also because they help us determine where the heat is being generated so that a powerful enough cooling system can be added. Major sources of losses in electrical machines include iron losses, copper losses, magnet losses, windage losses and bearing losses [28]. Iron losses, copper losses and windage losses calculations are discussed in the coming sections. Magnet losses are ignored as they are usually small in magnitude [29]. If the magnet losses are significant we can reduce them either by segmenting the magnets or reducing the spatial harmonics [30], [31]. Bearing losses are dependent on the diameter of the bearing and the forces applied to it [28]. These values are difficult to calculate but will not change for a constant operating point and are thus ignored when calculating the losses for the machine. However, the actual efficiency will be lower than that used for the MOO. A sufficient margin should be assumed when selecting designs from the Pareto optimal solution set.

2.2.1. Iron Loss Modelling

Two methods for iron loss modelling are explored. The first method uses a static 2D-FEA to estimate the losses and the second method uses the time step simulation with two more rotor positions evaluated.

For the static method, a relation between the stator teeth and stator yoke magnetic field density and air gap magnetic field density is derived. We use magnetic circuit theory for this [14]. Let the rotor be positioned such that the magnetic field is maximum in the stator teeth and yoke. We assume that the relative permeability of the stator iron is high and all the magnetic flux is travelling through the stator iron. Then the magnetic flux passing through the teeth is equal to the total magnetic flux entering the slot pitch for that teeth. For the yoke, there are two paths which the flux can travel and thus the magnetic flux in the yoke is half that entering the slot pitch. Typical flux lines for stator teeth and back iron are shown in figure 2.4.

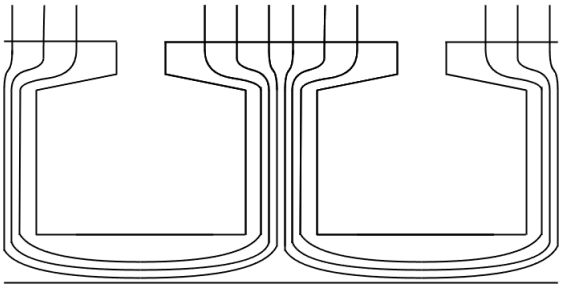


Figure 2.4: Typical Flux lines in stator teeth and back iron

This leads to the following relations.

$$B_{t,pk} = \frac{\tau_s}{w_t} B_{ag,pk} \tag{2.17}$$

$$B_{y,pk} = \frac{1}{2} \frac{\tau_s}{w_y} B_{ag,pk} \tag{2.18}$$

Where $B_{ag,pk}$ is the peak airgap magnetic flux density, $B_{t,pk}$ and $B_{y,pk}$ are the peak magnetic flux density in the teeth and yoke respectively, τ_s is the slot pitch in meters and w_t and w_y are the thickness of the teeth and the yoke respectively.

The magnetic field in the air gap rotates with the rotor. Thus the spatial distribution of the magnetic field density in the air gap will be seen by each teeth and yoke as a function of time. We can thus use the calculated peak magnetic flux density to estimate the losses assuming that it oscillates at the operating frequency because of the air gap relation. Thus we can calculate the iron loss density using the Bertotti Loss model [32] for the yoke and teeth and finally the total loss can be calculated.

$$p_{den} = K_{hys} f |B|^2 + K_{edd} f^2 |B|^2 + K_{exc} f^{1.5} |B|^{1.5} \tag{2.19}$$

$$P_l = p_{den} vol \rho_d \tag{2.20}$$

Where K_{hys},K_{edd} and K_{exc} are the loss hysteresis, eddy current and excess loss coefficients which are dependent on the material properties. vol is the volume of the part whose losses are being determined and ρ_d is the mass density of the material. It is important to note here that the loss coefficient should match the calculation whether it is mass-based or volume-based.

It should be noted that we assume that all the flux from the air travels through the stator teeth and stator yoke and thus the calculation can be inaccurate when a large amount of flux returns from the teeth tip or through the slot (especially in machines with an almost equal number of slots and poles) as shown in figure 2.5.

This method cannot be extended to the rotor iron loss calculations as the air gap flux density rotates with the rotor and is thus stationary if seen from the rotor. The rotor losses are due to higher harmonics and these cannot be extracted using a static 2D-FEA.

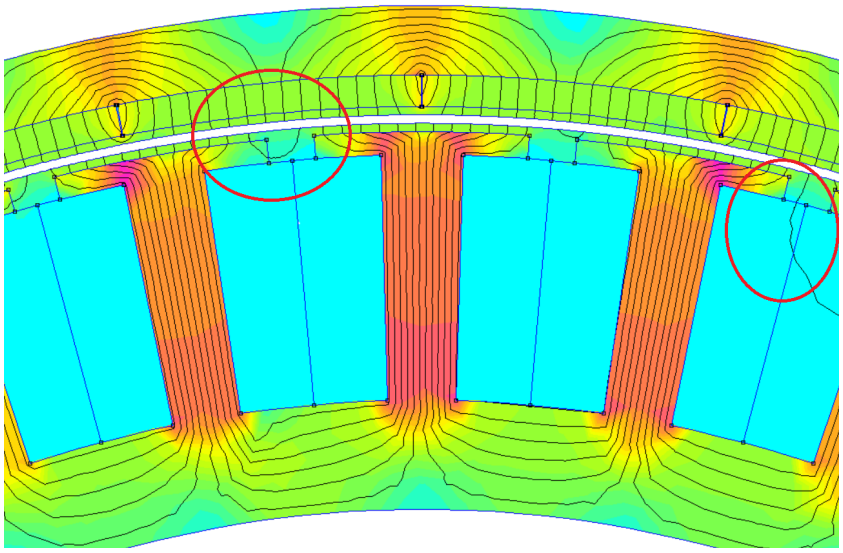


Figure 2.5: Flux returning through stator teeth tip and slot instead of going through the teeth and yoke

The static 2D-FEA method fails to provide any information about the harmonics as well as averages the flux density across the whole stator this is not correct as there are some parts of the stator that get significantly higher flux densities like near the teeth and yoke junction and some parts where the flux density is lower. Thus a time-step FEA which can measure the waveform of the magnetic flux density per cell in the mesh is needed.

Since we already have the time-step FEA solutions for $\theta^e = [0, \frac{\pi}{12}, \frac{\pi}{6}, \frac{\pi}{4}]$ we can extract more information by adding two more time steps for rotor positions $\theta^e = [\frac{\pi}{2}, \frac{3\pi}{4}]$ and extract the peak of 1^{st} and 3^{rd} harmonic of the x and y component of the magnetic field density per cell in the mesh solution generated using Discrete Fourier Transforms. These peaks can then be used to calculate the loss density due to both changes in the x and y directions which can be summed to get the loss density per cell for all the cells within the stator or the rotor [33].

Since this method uses the FEA solutions directly there is no assumption regarding the permeability of the material and the results are more accurate. The reason for only computing the 1^{st} and 3^{rd} harmonic is to keep the computation time at a minimum as extracting more harmonics will require more time steps. To sufficiently measure a harmonic of f frequency we need to sample the wave at least $2f$ frequency according to Nyquist's theorem.

We use the four time-step samples at $\theta^e = [0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}]$ to generate 8 samples by using the anti periodicity condition for the other four samples at $\theta^e = [\pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}]$. Figure 2.6 shows the magnetic field density in the x and y direction for a cell from the 2D FEA of the motor in section 3.1 with the sampling points marked.

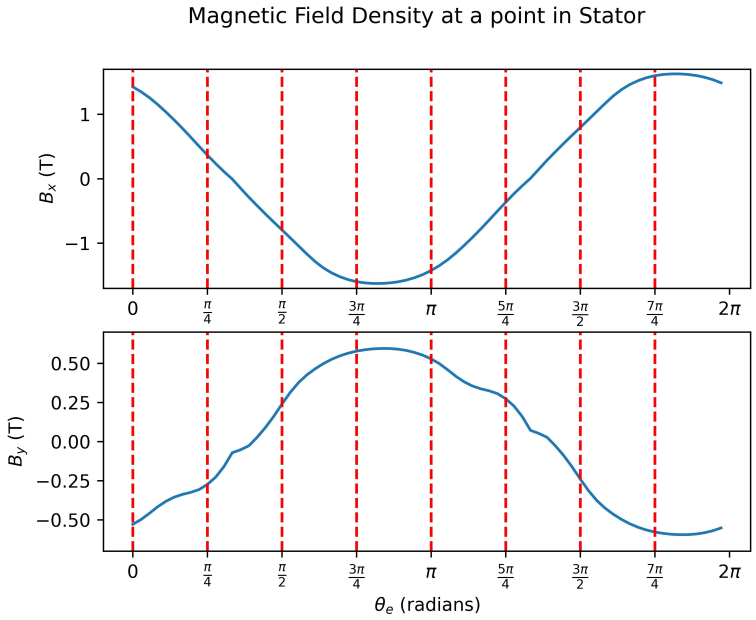


Figure 2.6: Magnetic Field Density at an example cell with sampling points marked

Now by using Discrete Fourier Transform on the i_{th} cell in the mesh

$$B_{x,i,k} = \sum_{n=0}^{7} B_n e^{-j(\frac{\pi}{4}kn)} \tag{2.21}$$

$$B_{y,i,k} = \sum_{n=0}^{7} B_n e^{-j(\frac{\pi}{4}kn)} \tag{2.22}$$

$$\hat{B}_{x,i,k} = 2\frac{|B_{x,i,k}|}{8} \quad \hat{B}_{y,i,k} = 2\frac{|B_{y,i,k}|}{8} \tag{2.23}$$

We multiply by 2 to account for the two-sided spectrum. We can use $\hat{B}_{x,i,k}$ and $\hat{B}_{y,i,k}$ to calculate the loss density using the Bertotti model as shown before. The calculated $\hat{B}_{x,i,k}$ and $\hat{B}_{y,i,k}$ for the example cell are shown in figure 2.7.

The loss density is calculated per cell and we know the weight or volume of each cell to calculate the loss per cell. We can sum over all cells to get the total loss.

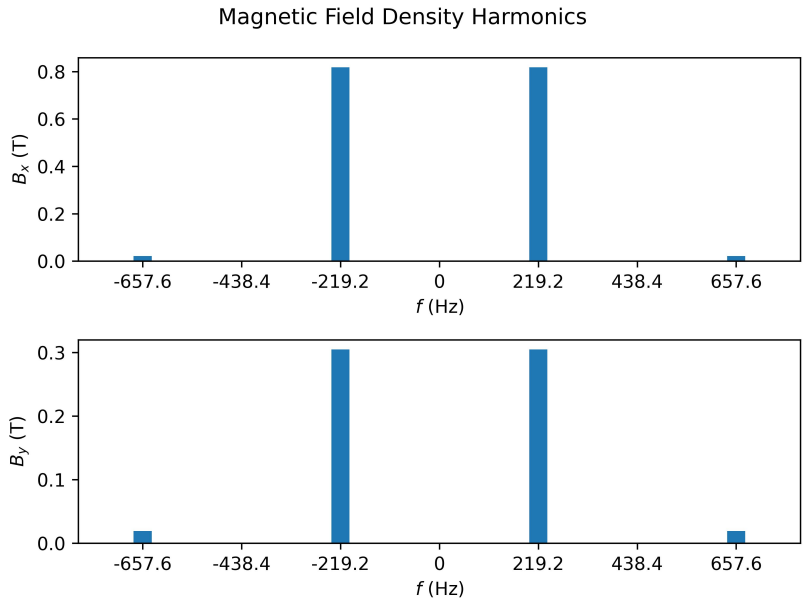


Figure 2.7: Calculated Magnetic Field Harmonics in x and y direction

$$p_{den,i} = \sum_{k=0}^{3} [K_{hys} f_k |\hat{B}_{x,i,k}|^2 + K_{edd} f_k^2 |\hat{B}_{x,i,k}|^2 + K_{exc} f_k^{1.5} |\hat{B}_{x,i,k}|^{1.5} + K_{hys} f_k |\hat{B}_{y,i,k}|^2 + K_{edd} f_k^2 |\hat{B}_{y,i,k}|^2 + K_{exc} f_k^{1.5} |\hat{B}_{y,i,k}|^{1.5}]$$

$$P_l = \sum_{i=0}^{n_{cell}} p_{den,i} vol_i \rho_d \tag{2.24}$$

The code for both static iron loss calculation and six time-step iron loss calculation are shown in appendix B.

2.2.2. Copper Loss Modelling

Copper loss calculation for electrical machines gets complicated at higher frequencies due to the skin effect and proximity effect. Skin effect is a phenomenon where the current density is concentrated at the surface of the conductor and the effective resistance of the conductor increases. This effect occurs due to the presence of opposing eddy currents present in the conductor generated by the changing magnetic field. Proximity effect is similar but occurs in parallel conductors. The current distribution again moves to the outer surface of the conductors and away from the other wires as well.

[28] provides a derivation of the skin effect and proximity effect factor. The resulting equations are given below

$$\alpha = \sqrt{\frac{1}{2} \omega \mu_0 \sigma_c \frac{b_{c0}}{b}} \tag{2.25}$$

$$\xi = \alpha h_{c0} = h_{c0} \sqrt{\frac{1}{2} \omega \mu_0 \sigma_c \frac{b_{c0}}{b}} \tag{2.26}$$

$$k_R = 1 + 0.59 \frac{z_t^2 - 0.2}{9} \xi^4 \tag{2.27}$$

$$P_{Cu} = 3k_R I_{rms}^2 \frac{l}{\sigma_c A_{wire}} \tag{2.28}$$

Where b_{c0} is the wire thickness, h_{c0} is the wire height, ω is the angular frequency of the current,σ_c is the conductivity of copper, b is the slot width, z_t is the number of layers of winding, l is the length of one turn per phase and A_{wire} is the equivalent area of one turn per phase.

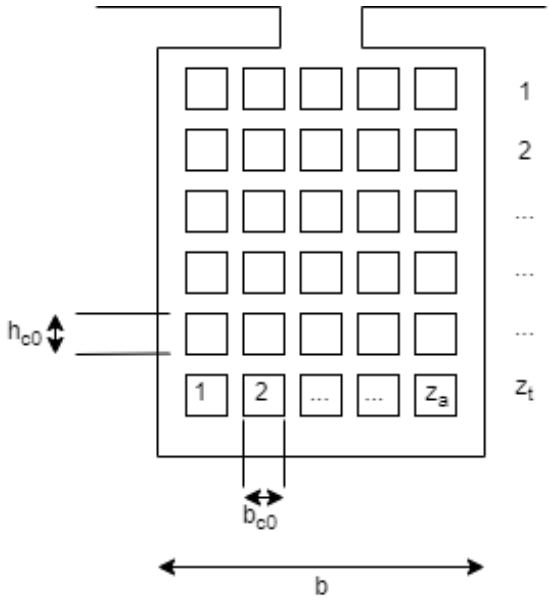


Figure 2.8: Parameters of the winding for skin effect and proximity effect calculation

The code for copper loss calculation are shown in appendix C and appendix D. The copper loss calculation is modified for MOO as the full specification of the winding is not known. This is further explained in chapter 4.

2.2.3. Windage Loss Modelling

Windage losses are losses that occur due to the friction between air and the rotor. Windage losses depend on the speed of the rotor as well as the surface area of the rotor. The equation for calculating windage losses is given by [34] and [35]

$$Re_g = \frac{\rho_f \Omega D_r g}{2\mu} \tag{2.29}$$

Where Re_g is Couette Reynolds number ρ_f is the density of air, Ω is the angular velocity, D_r is the rotor bore diameter, g is the air gap length and μ is the dynamic viscosity of air. We can now calculate the torque coefficient using the Couette Reynolds number.

$$C_M = 10\frac{(2g/D_r)^{0.3}}{Re_g}, \quad Re_g < 64 \tag{2.30}$$

$$C_M = 2\frac{(2g/D_r)^{0.3}}{Re_g^{0.6}}, \quad 64 < Re_g < 500 \tag{2.31}$$

$$C_M = 1.03\frac{(2g/D_r)^{0.3}}{Re_g^{0.5}}, \quad 500 < Re_g < 10^4 \tag{2.32}$$

$$C_M = 0.065\frac{(2g/D_r)^{0.3}}{Re_g^{0.2}}, \quad 10^4 < Re_g \tag{2.33}$$

The windage power loss is given by

$$P_{fr} = \frac{k_1 C_M \rho \pi \omega^3 D_r^4 l}{32} \tag{2.34}$$

Where k_1 is the smoothness factor of the rotor surface.

2.3. Thermal Model

We must consider the flow of heat within the machine to ensure that the machine does not suffer from thermal runaway and the insulation is within its thermal limits. The problem of thermal runaway exists because as the temperature of the windings increases, the copper losses increase which in turn again increases the temperature and this cycle continues. To stop thermal runaway the cooling of the machine should be sufficient to ensure that the temperatures can stabilise.

There are various ways of making thermal models of electrical machines. The most simple method is the Lumped Parameter Thermal Network (LPTN). This method makes an analogy between the flow of heat and the flow of current with various parts in the machine being modelled as resistances. The power losses in various parts are modelled as current sources and the temperatures of these parts are equivalent to voltages of nodes [36]. This method is less accurate as it relies on simplifications in geometry and averaging over the whole body to calculate the temperature as opposed to calculating the temperature at each point in the body. This method also fails to identify hot spots that can occur within the body as the modelling precision is low. A more accurate method would be the use of 2D-FEA and 3D-FEA to solve for the temperature using Poisson's equation. 2D-FEA can provide accurate temperatures for a fine mesh and this feature is available in FEMM as well. However often the main path of flow of heat is in the axial direction, especially for the windings but this information is not available in 2D-FEA. 3D-FEA are time-consuming and are not adopted in this thesis for analysing machines quickly.

Steady State Lumped Parameter Network is used as including the transient calculations will increase the complexity of the model. Also if the system performs well at the steady state we can verify the transient behaviour in the latter part of the design process and make changes to the design if necessary.

Another important part of the thermal model is the interaction of the machine with the cooling fluid. This part of the modelling is usually done by performing experiments and deriving empirical equations which can be used in various cases. [37] and [38] provide many such empirical derivations which have been used in this thesis. The alternative to using empirical equations is to perform Computational Fluid Dynamics (CFD) analysis to estimate the fluid flow which then gives us an idea about the heat exchanged. This however again has a large time complexity as well as requires prior knowledge of fluid dynamics and is not considered suitable for our case.

The following sections will discuss how various thermal resistances have been calculated to create the LPTN.

2.3.1. Thermal Conduction in Solid Parts

The thermal resistance of any part can be calculated in the same way that we calculate electrical resistance:

$$R = \frac{l}{\Lambda_t A} \tag{2.35}$$

where l is the length of the part, A is the area of cross-section and Λ_t is the thermal conductivity of the material. However, most of these parts also generate heat and thus behave as heat sources as well. The obvious thing to do would be to add the total loss of the part and add it in series to the thermal resistance. However, the average temperature calculated using this method is incorrect as this method assumes that the total loss is generated outside the body which is not true. To counter this problem "T-networks" were introduced [39]. This method has shown to produce accurate enough results [34], [35].

The T-network method represents different parts of the machine using three thermal resistances as shown in figure 2.9 where the middle point of the three resistances is the average temperature of the part. R_1 and R_2 are the resistances from the midpoint of the body to the two extremes and R_3 is the

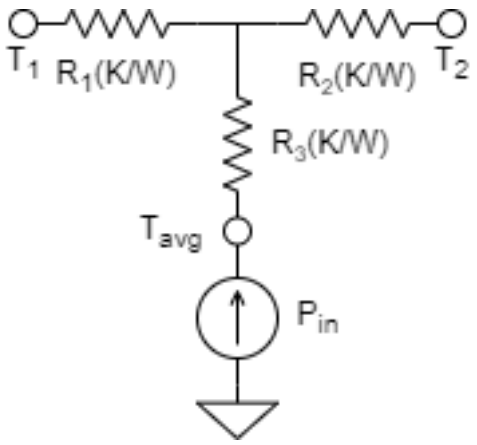


Figure 2.9: A T network representation of a machine part with P_{in} losses being generated within the body

correction resistance to get the actual average temperature of the body. The T-network represents the flow of heat in one direction but for electrical machines, this is not always true as the heat flows in radial as well as axial directions and we assume that the body is isothermal in the tangential direction. If we assume that the two direction flows are independent we can represent the part by connecting two T networks in parallel to the same source [36].

The three resistances of the T network for axial flow are given by [34]

$$R_1 = R_2 = \frac{d}{\Lambda_z A} = \frac{L}{2\Lambda_z A} \tag{2.36}$$

$$R_3 = -\frac{R_1}{3} = -\frac{L}{6\Lambda_z A} \tag{2.37}$$

where Λ_z is the thermal conductivity of the material in the axial direction, L is the length of the body in the axial direction (often half of the stack length) and A is the area of cross-section.

For the radial flow of heat, the shape of the part is of importance. For a hollow cylinder the three thermal resistances are given by [34], [35]. The dimensions of the cylinder as used in the following equation are shown in figure 2.10.

$$R_1 = \frac{1}{4\pi\Lambda_r L}[1 - \frac{2r_2^2 ln(\frac{r_1}{r_2})}{r_1^2 - r_2^2}] \tag{2.38}$$

$$R_2 = \frac{1}{4\pi\Lambda_r L}[\frac{2r_1^2 ln(\frac{r_1}{r_2})}{r_1^2 - r_2^2} - 1] \tag{2.39}$$

$$R_3 = -\frac{1}{8\pi\Lambda_r L(r_1^2 - r_2^2)}[r_1^2 + r_2^2 - \frac{4r_1^2 r_2^2 ln(\frac{r_1}{r_2})}{r_1^2 - r_2^2}] \tag{2.40}$$

For modelling the teeth we approximate it as a cuboid and the equations for heat flow in the radial direction are given as below [35].

$$R_1 = R_2 = \frac{h_s}{\Lambda_r t_t L} \tag{2.41}$$

$$R_3 = -\frac{R_1}{3} = \frac{h_s}{3\Lambda_r t_t L} \tag{2.42}$$

where t_t is the thickness of the teeth and h_s is the slot height. The thermal resistance between the teeth and the windings is covered with the winding calculation.

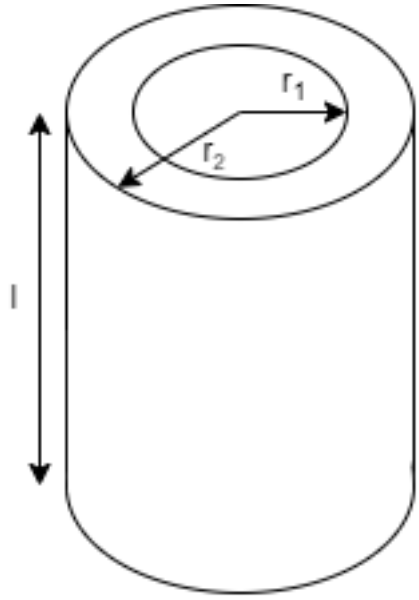


Figure 2.10: Hollow cylinder parameters for T-network thermal resistance calculation

The magnets are modelled as partial hollow cylinders by multiplying the resistances by $\frac{2\pi}{\alpha_p}$. In addition, we divide this by $2p$ to account for the parallel paths.

Another important thing to remember is that the materials used in electrical machines are not isotropic especially when it comes to thermal properties. The iron core has a higher thermal conductivity in the radial direction as it consists of metal sheets whereas axial conductivity is bad due to the presence of insulators. Similarly, windings have higher conductivity in the axial direction as the copper runs axially whereas the radial conductivity is bad due to the presence of insulators. Thus the core iron is usually cooled radially whereas the heat produced in the windings usually flows to the end winding part and is exchanged with the fluid in the end winding space (usually air).

For the calculations performed further in this thesis the axial flow of heat in the iron core is ignored. This is justified by the poor conductivity as well as the smaller area of cross-section in the axial direction. A comparison of the axial and radial thermal properties for the cores of the motor analysed in section 3.1 is presented in table 2.2. These resistances can however be included in future work if the heat flow in the axial direction is found to be significant for example in pancake machines (radius is much larger than axial length).

Table 2.2: Comparison of Axial and Radial Thermal Resistance for an example machine (Nuna Motor)

Machine Part	Axial Thermal Resistance	Radial Thermal Resistance
Stator Yoke	$R_1 = 1.68$ K/W $R_2 = 1.68$ K/W $R_3 = -0.56$ K/W	$R_1 = 0.94$ K/W $R_2 = 0.026$ K/W $R_3 = -0.007$ K/W
Rotor Yoke	$R_1 = 7.58$ K/W $R_2 = 7.58$ K/W $R_3 = -2.53$ K/W	$R_1 = 0.92$ K/W $R_2 = 0.003$ K/W $R_3 = -0.001$ K/W

2.3.2. Thermal Conduction in Windings

Windings are difficult to model because of the presence of two different materials as well as the pattern of the windings. [28] provides the derivation of the winding's thermal model. The resulting equations are presented below.

$$R_{tw} = \frac{1}{LS_s}\left[\frac{b_i}{(2h_s + b_s)\Lambda_i} + \frac{1}{(2h_s + b_s)\Lambda_{co}}\right] \tag{2.43}$$

R_{tw} is thermal resistance between the winding and the stator teeth. S_s is the number of stator slots, L is the stack length and Λ_{co} is the thermal conductivity of copper. h_s is the height of the slot, b_s is the width of the slot and b_i is the thickness of the insulation lining as shown in figure 2.11. Now we calculate the internal resistance and the resistance between the windings and end winding air space.

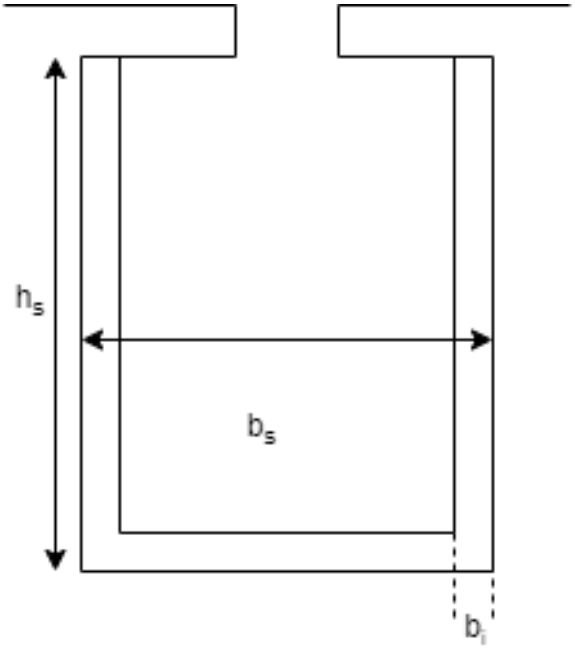


Figure 2.11: Dimensions of a slot used for thermal calculations

$$G_w = \frac{1}{R_{tw}} \tag{2.44}$$

$$R_W = \frac{L}{S_s k_f A_c \Lambda_c} \tag{2.45}$$

where A_s is the area of the slot k_f is the copper fill factor and Λ_c is the thermal conductivity of copper.

$$R_1 = R_2 = \sqrt{\frac{R_W}{G_W}} tanh\left(\sqrt{\frac{R_W G_W}{2}}\right) \tag{2.46}$$

$$R_3 = \frac{1}{G_W}\left(\frac{\sqrt{R_W G_W}}{sinh\left(\sqrt{\frac{R_W G_W}{2}}\right)} - 1\right) \tag{2.47}$$

2.3.3. Thermal Resistance of the Air Gap

There is heat flowing from the stator to the rotor or vice versa through the air present in the air gap. This flow can be significant, especially when modelling naturally cooled machines. The modelling of this thermal resistance is made difficult because the rotor is moving, leading to turbulent flow of air. [40] has developed a model for the flow of gas and thermal transfer coefficient in machine air gaps which is used here.

The modelling of heat flow through air uses some constants to determine the state of flow(turbulent or laminar). We first calculate Taylor's number.

$$Ta = \frac{\rho^2 \omega^2 r_m g^3}{\mu_a^2} \tag{2.48}$$

where ρ is the mass density of the fluid, ω is the angular velocity of the rotor, r_m is the mean air gap radius, g is the air gap length and μ_a is the dynamic viscosity of the fluid. This number is modified to account for Taylor-Couette flow as explained in [34].

$$Ta_m = \frac{Ta}{F_g} \tag{2.49}$$

where F_g is

$$F_g = \frac{\pi^4[\frac{2r_m-2.304g}{2r_m-g}]}{1697[0.0056+0.0571(\frac{2r_m-2.304g}{2r_m-g})^2](1-\frac{g}{2r_m})} \tag{2.50}$$

The Nusselt number can thus be calculated for different conditions of flow based on the modified Taylors number [34].

$$Nu = 2 \qquad TA_m < 1700 \tag{2.51}$$

$$Nu = 0.128TA_m^{0.367} \qquad 1700 < Ta_m < 10^4 \tag{2.52}$$

$$Nu = 0.409TA_m^{0.241} \qquad 10^4 < Ta_m \tag{2.53}$$

The thermal convection coefficient is then calculated as [34]

$$\Lambda_a = \frac{Nuk}{d_h} \tag{2.54}$$

$$d_h = g\sqrt{\frac{8}{3}} \tag{2.55}$$

where k is the thermal conductivity of the fluid and d_h is the hydraulic diameter. The thermal resistance can then be calculated as

$$R = \frac{1}{\Lambda_a A} \tag{2.56}$$

where A is the area of contact between the rotor or stator and the air gap. We must also add the thermal transfer due to radiation in this calculation. The calculation for the radiation conductivity coefficient is covered in the next section.

2.3.4. Thermal Convection: Natural Convection

To calculate the convection coefficients the orientation of the machine becomes important, especially for natural convection. Here we assume that the machine is a horizontal cylinder without fins or any other cooling constructions. Thus we have three surfaces from which convection and radiation take place, the two flat faces and the curved face as shown in figure 2.12.

For the curved face, the convection coefficient is calculated using the following equations as given in [37].

$$Gr = \frac{\beta g\theta\rho^2 L^3}{\mu^2} \tag{2.57}$$

where β is the coefficient of cubical expansion of the fluid, g is the gravitational force of attraction, θ is the temperature difference between the surface and the fluid,ρ is the fluid mass density, L is the characteristic length of the surface (height of the cylinder) and μ is the dynamic viscosity of the fluid.

$$Pr = \frac{c_p\mu}{k} \tag{2.58}$$

where c_p is the specific heat capacity of the fluid and k is the thermal conductivity of the fluid.

$$Ra = GrPr \tag{2.59}$$

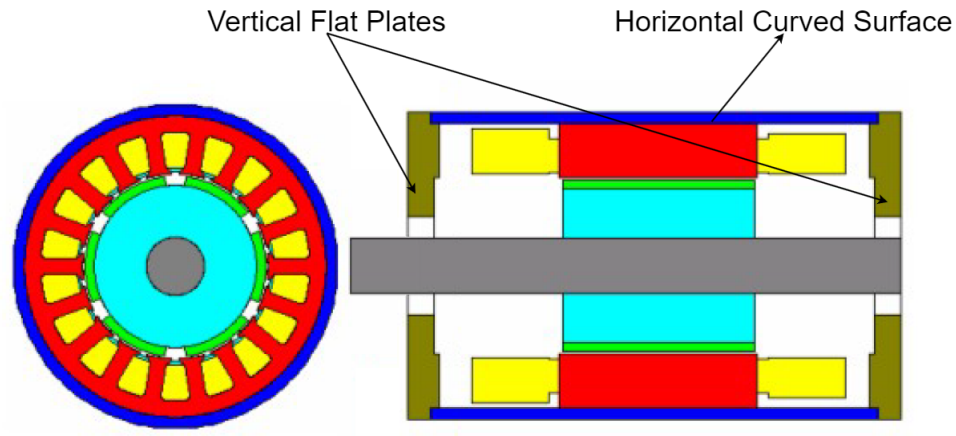


Figure 2.12: Three surfaces through which a natural convection machine is cooled [37]

$$Nu = 0.525Ra^{0.25} \quad 10^4 < Ra < 10^9 \tag{2.60}$$
$$Nu = 0.129Ra^{0.33} \quad 10^9 < Ra < 10^{12} \tag{2.61}$$

$$\Lambda_c = \frac{Nuk}{L} \tag{2.62}$$

For the flat plates, the calculation remains the same with only the Nusselts number(Nu) being adjusted. The characteristic length is the diameter of the cylinder.

$$Nu = 0.59Ra^{0.25} \quad 10^4 < Ra < 10^9 \tag{2.63}$$
$$Nu = 0.129Ra^{0.33} \quad 10^9 < Ra < 10^12 \tag{2.64}$$

When the machine is cooled through natural convection the heat flow by radiation can be significant and should be calculated to get an accurate estimate of the cooling capability. The radiation thermal conductivity is calculated through the following equation as given by [28]

$$\Lambda_r = \epsilon_{thr}\sigma_{SB}\frac{T_s^4 - T_a^4}{T_s - T_a} \tag{2.65}$$

where ϵ_{thr} is the relative emissivity between the surface and air and σ_{SB} is the Stefan-Boltzmann constant. T_s and T_a are the surface and air temperature respectively.

Thus the thermal resistance from any surface to air is given by

$$R = \frac{1}{(\Lambda_c + \Lambda_r)A} \tag{2.66}$$

The lumped parameter thermal network generated for an external rotor machine with natural convection cooling is shown in figure 2.13.

2.3.5. Thermal Convection: Forced Air Cooling

Under the forced cooling regime cool air is blown over the hot surface and it absorbs more heat from the surface when compared to natural convection and thus allowing us to have higher loss density and more compact machines.

In our model, there are three paths where forced cooling takes place, the end windings, the rotor frame and the stator jacket. To calculate the thermal conductivity for rotor frame and stator jacket is straightforward as these surfaces are essentially flat plates with air being blown over at a fixed speed. The stator winding is difficult to model as the surface of the winding is rough as well as toroid shaped

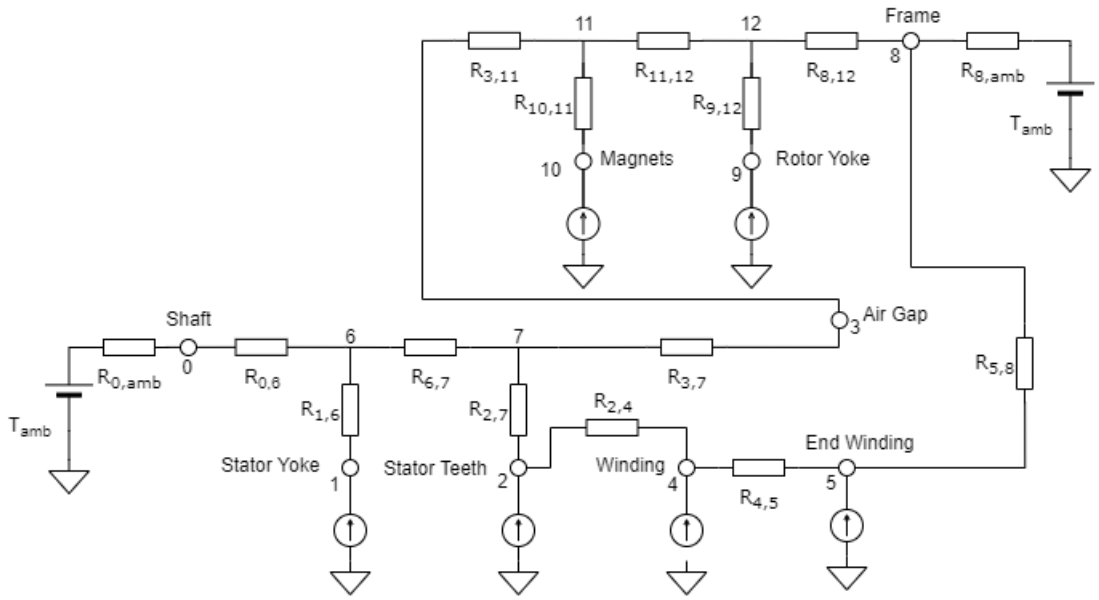


Figure 2.13: Lumped Parameter Thermal Network for an Outer Rotor SPMSM with natural convection cooling

making it difficult to analyse. Nevertheless, [35] used the flat plate calculations on the end winding with satisfactory results and we will make the same approximation here.

The thermal conductivity for a flat plate with air being blown over it is given by the following set of equations [37].

$$Re = \frac{\rho v L}{\mu} \tag{2.67}$$

where Re is called the Reynolds number and ρ is the mass density of the fluid, v is the velocity of the fluid, L is the characteristic length (length in the direction of airflow) and μ is the dynamic viscosity of the fluid.

$$Nu = 0.664 Re^{0.5} Pr^{0.33} \quad Re < 5x10^5 \tag{2.68}$$

$$Nu = (0.037 Re^{0.8} - 871) Pr^{0.33} \quad Re > 5x10^5 \tag{2.69}$$

We can use the Nusselt Number to calculate the heat transfer coefficient as done in the previous sections.

The lumped parameter thermal network generated for an external rotor machine with forced cooling is shown in figure 2.14. Changes, when compared to natural convection, have been highlighted in red. The coefficients from the frame and shaft to the ambient air have changed. Also, the end windings are now directly cooled by ambient air blowing over them.

2.3.6. Solving the model using Python

To solve the lumped parameter thermal network in python we use network theory to solve a resistive network with independent current inputs. Since the copper losses are dependent on the temperature of the winding we have to iterate till a stable solution is found [41]. The linear equations for the thermal model are given by [42]

$$GT = P \tag{2.70}$$

$$G = \begin{bmatrix} \sum_{i=0}^{n-1} \frac{1}{R_{0,i}} + \frac{T_{amb,0}}{R_{amb,0}} & -\frac{1}{R_{0,1}} & \cdots & -\frac{1}{R_{0,n-1}} \\ -\frac{1}{R_{0,1}} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & -\frac{1}{R_{n-2,n-1}} \\ -\frac{1}{R_{0,n-1}} & \cdots & -\frac{1}{R_{n-2,n-1}} & \sum_{i=0}^{n-1} \frac{1}{R_{n-1,i}} + \frac{T_{amb,n-1}}{R_{amb,n-1}} \end{bmatrix} \tag{2.71}$$

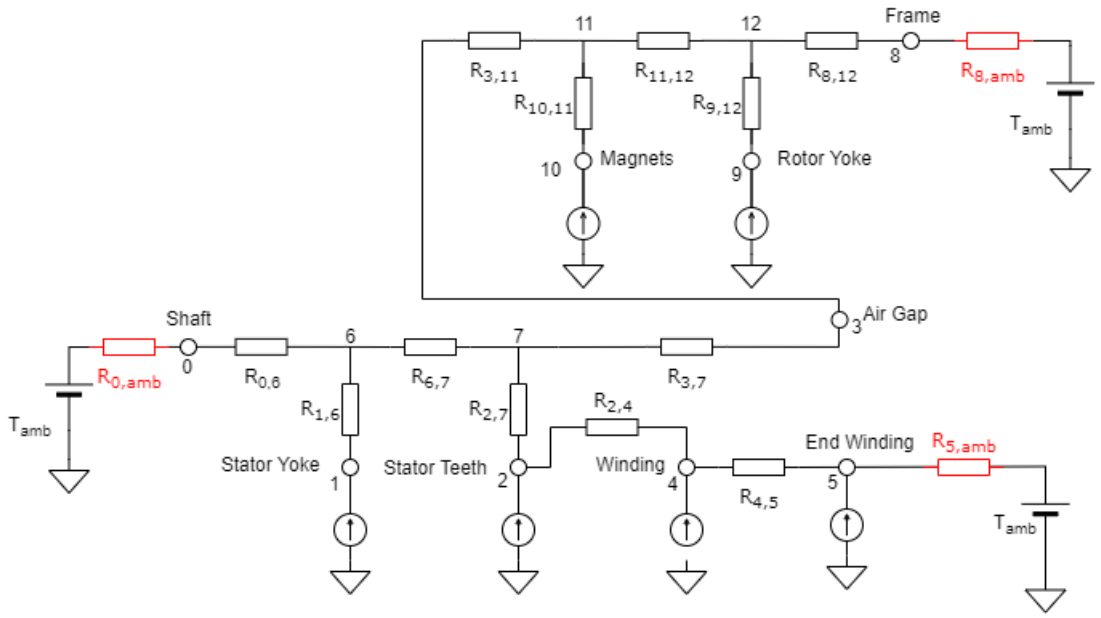


Figure 2.14: Lumped Parameter Thermal Network for an Outer Rotor SPMSM with forced cooling

$$T = \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{n-1} \end{bmatrix} \tag{2.72}$$

$$T = \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{n-1} \end{bmatrix} \tag{2.73}$$

where $R_{i,j}$ is the thermal resistance between nodes i and j, T_i is the absolute temperature of the part represented by node i and P_i is the loss generated in part represented by node i.

For this thesis, a thermal model with 13 nodes is generated. The important nodes and their numbers are presented in table 2.3. The rest of the nodes are the T points for various components.

Table 2.3: Different Parts of the machine and their corresponding node numbers

Part Name	Node Number
Shaft	0
Stator Yoke	1
Stator Teeth	2
Air Gap	3
Winding	4
End Winding	5
Frame	8
Rotor Yoke	9
Magnets	10

Since the increase in temperature increases the copper losses as well, a single computation of the temperature distribution may not be enough to get accurate results. Hence the temperature distribution is calculated iteratively to get better results. The new temperature distribution is used to calculate the

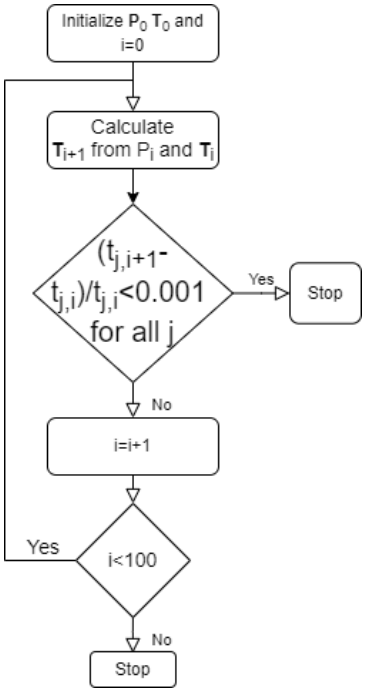


Figure 2.15: Flow chart of the thermal calculation to account for increase in copper losses with temperature

new losses distribution (copper loss) and this is then used to calculate the new temperature distribution. This is continued till the difference between two generations is small (relative tolerance of 1%) or if the exit condition has been reached (100 calculations). If the exit condition is reached then the system is likely in thermal runaway and such a design is rejected by the MOO routine. The flow chart for the temperature calculation is shown in figure 2.15.

The code presented in appendix E and appendix F correspond to an outer rotor permanent magnet machine for natural convection and forced cooling respectively. The code can be easily adapted for an inner rotor machine by changing the connections for the frame and the shaft nodes to the corresponding parts. The change in directions of the thermal resistances in the T network is already accounted for.

3

Validation of Multi-Physical Modelling

Before we start MOO using the multi-physical models created in the last chapter it is important that we first verify if the models are accurate. This verification is performed in two phases, first, we verify the model with a machine whose most parameters are known and performance characteristics are known from previous studies, then we move to a new problem where very few parameters of the machine are known. The next section deals with the verification of the known model.

3.1. Validation of Totally Enclosed Outer Rotor Surface PMSM

This section is based on a previous study in [43] of a motor which is used by the Nuna Solar car team. The machine is an Outer Rotor Surface Mounted Permanent Magnet Synchronous Machine. The machines dimensions are presented in table 3.1 and the structure is shown in figure 3.1.

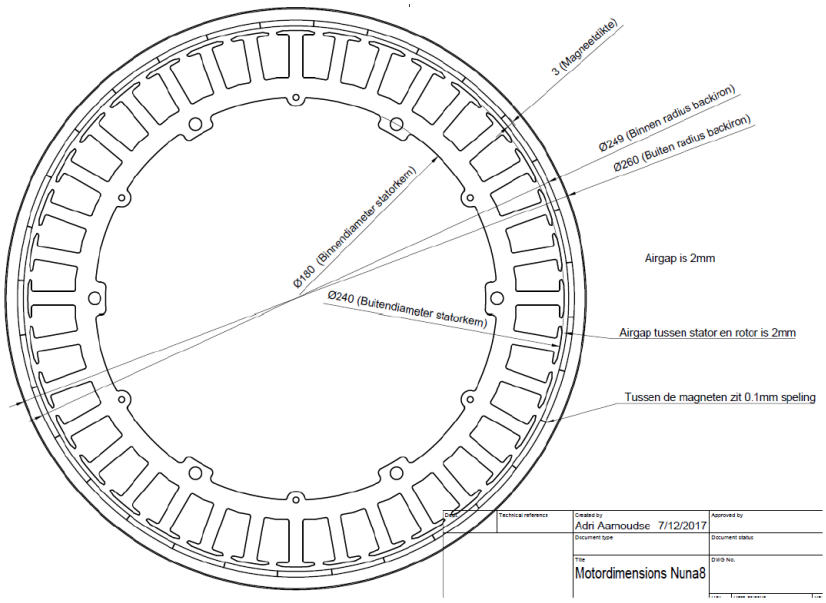


Figure 3.1: Nuna Machine Structure

To perform the verification a particular operation point of $I_{drms} = 0$ $I_{qrms} = 6.8589$ A and rotor speed $N0 = 822$ RPM is taken. The results of the analysis are presented in table 3.2.

Table 3.1: Dimensions of Nuna machine

Dimension	Value
Stack Length (L)	40 mm
Stator Inner Radius (R_{Sint})	90 mm
Stator Outer Radius (R_{Sext})	120 mm
Air Gap length (g)	2 mm
Rotor Inner Radius (R_{Rint})	122 mm
Rotor Outer Radius (R_{Rext})	130 mm
Magnet Height (l_m)	3 mm
Magnet Angle (α_p)	0.1955 rads
Number of Pole Pairs (p)	16
Number of Slots (N_s)	36
Slot Opening Angle (W_0)	0.03175 rads
Slot Height (H_2)	19.5 mm
Tooth Width(W_t)	3.8 mm
Number of turns per slot(N_{tps})	30
Wire thickness including insulation	1.6 mm
Iron Material	M270-35A(assumed)
Magnet Material	N35H (assumed)

Table 3.2: Comparison of Results produced by the multi-physical model with the previous study. [43]

Property	Previous Study Results (AE Group)	Calculated Results
Torque (Static)	-	11.21 Nm
Torque (6 time-step)	9.31 Nm	11.8 Nm
Iron Losses (Static)	-	31.65 W
Iron Losses (6 time-step)	21.6063 W	23.66 W
Copper Losses	4.64 W	6.51 W
Winding Temperature	55^o C (assumed)	101^o C
Windage Losses	2.57 W	2.51 W

The calculated torque matches the experimental torque. There is still a difference in the calculated torque but this is due to the lack of knowledge about the exact material properties used in the machine. Another point of difference is that the shape of the teeth has been approximated to a simple geometry easily constructed through code and we do not know the exact dimensions of the teeth.

The loss calculations are found to be in agreement with previous studies. The copper losses predicted by the model are higher than previous studies as they assume the temperature to be 55^o C whereas the LPTN predicts the temperature to be 101^o C. Since we calculate the temperature from the properties of the machine the calculation has fewer approximations when compared to the previous studies.

3.2. Validation of Forced Air Cooling Outer Rotor Surface PMSM

For the MOO a drone motor is analysed. This motor is an External Rotor Surface Mounted Permanent Magnet Synchronous Machine. The datasheet of the motor [44] provided limited information about the dimensions of the motor and no information about the materials used. Nevertheless, we can verify

the torque production and temperature calculation by making some assumptions regarding the internal dimensions. The dimensions of the machine known from the datasheet are shown in figure 3.2. The assumed dimensions are presented in table 3.3 and the results of the calculations are shown in table 3.4.

We have assumed that the frame of the machine is directly attached to the rotor and is rotating with the rotor. Another assumption is regarding the speed of the air cooling the shaft, it is assumed that the speed is equal to the rotor surface speed. This speed might differ according to the design of the outer frame of the machine with the placement of openings but this assumption is made to simplify the design process.

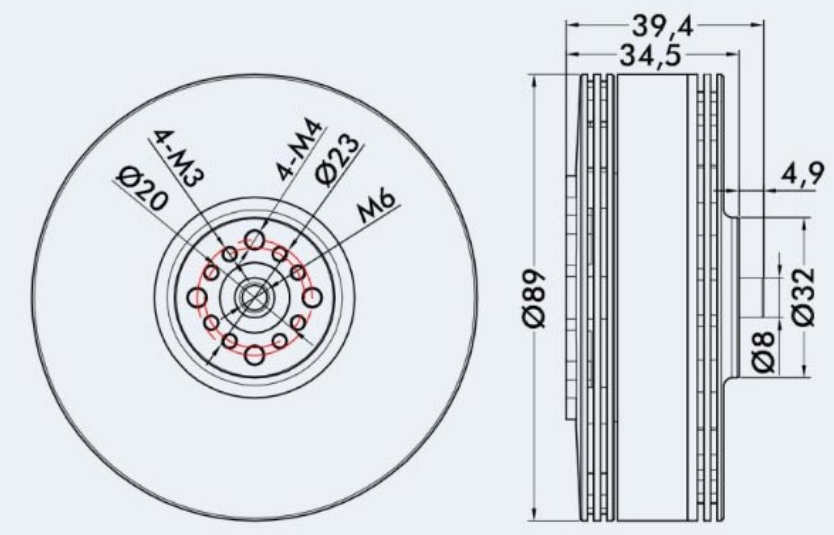


Figure 3.2: 24 Slot 28 Pole outer rotor PMSM Drone machine dimensions

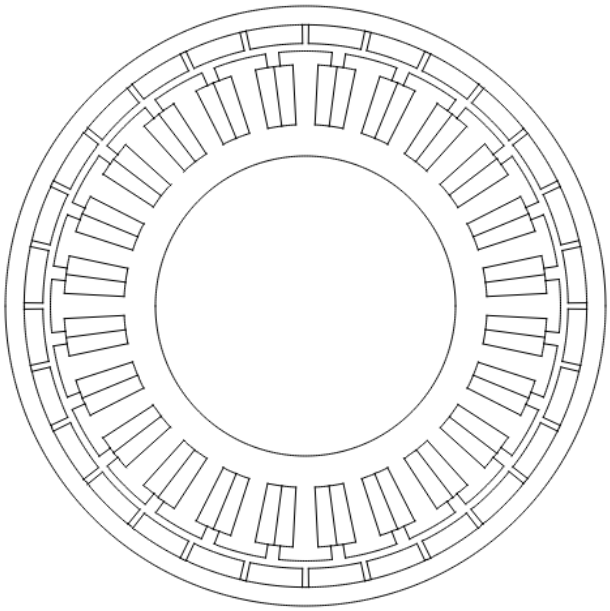


Figure 3.3: Structure of the machine used for 2D FEA

Table 3.3: Dimension of the 24 Slot 28 Pole Outer Rotor SPMSM used for drone application

Dimension	Value
Stack Length (L)	30 mm
Stator Inner Radius (R_{Sint})	20 mm
Stator Outer Radius (R_{Sext})	34 mm
Air Gap length (g)	1 mm
Rotor Inner Radius (R_{Rint})	35 mm
Rotor Outer Radius (R_{Rext})	40 mm
Magnet Height (l_m)	2.5 mm
Magnet Angle (α_p)	0.202 rads
Number of Pole Pairs (p)	14
Number of Slots (N_s)	24
Slot Opening Angle (w_0)	0.0523 rads
Slot Height (H_2)	7.98 mm
Tooth Width(W_t)	2.95 mm
Iron Material	M270-35A(assumed)
Magnet Material	N35H (assumed)

Table 3.4: Results from the Multi-physical model for the drone motor

Property	Datasheet Value	Calculated Results
Torque (Static)	3.45 Nm	3.30 Nm
Torque (6 time-step)	3.45 Nm	3.45 Nm
Iron Losses (Static)	-	31.6 W
Iron Losses (6 time step)	-	26.1 W
Copper Losses	-	116.4 W
Winding Temperature	max 200^o C	182^o C
Surface Temperature	78.5^o C	45.3^o C
Windage Losses	-	2.51 W

The surface temperature is lower than that obtained from the datasheet but an exact match of the properties is difficult due to the lack of information about the machine. However, the results verify that the winding temperature is within limits and that the cooling method used is sufficient to cool the machine.

4

Multi-Objective Optimisation

Now that the multi-physical model has been verified we can start using it to create the optimisation platform. The next section provides the motivation for multi-objective optimisation in the development of electrical machines followed by an explanation of the developed framework. This framework is then applied to a specific case study of optimisation of a SPMSM for drone application.

4.1. Multi-Objective Optimisation

The design of electrical machines often has conflicting requirements which means that improvement in one aspect of the machine will lead to some other aspect becoming worse. For example, in order to reduce the losses we need to use thicker wires or more core iron which increases the weight and the cost of the machine, This combined with the high number of variables and constraints involved in designing makes the task difficult. Thus the development of a tool to reduce design space and depict the trade-offs between different designs is essential. Multi-Objective Optimisation algorithms are these tools which use gradient-based or evolutionary methods to generate design candidates. Since the design of electrical machines involves discrete variables and also has functions whose gradient is difficult to define, the use of gradient-based methods is difficult.

Many evolutionary algorithms have been developed for different use cases. In electrical machine optimisation, three evolutionary algorithms have been used extensively - Non-Dominated Sorting Genetic Algorithm, Particle Swarm Optimisation and Differential Evolution. These algorithms return a set of solutions which are Pareto optimal i.e. for these solutions any improvement in an objective comes at the cost of performance in another objective. Some publications have made comparisons of these algorithms and suggest Differential Evolution [10] or Particle Swarm Optimisation [45] to be the best. [10] argues that the most suitable algorithm is based on the application and all three algorithms return satisfactory results. This thesis uses the Non-Dominated Sorting Generic Algorithm II (NSGA-II) algorithm for performing the optimisation to generate the Pareto Optimal fronts due to its easy setup and beginner friendliness. [10] furthermore argues that defining the problem correctly is more important than the algorithm used to obtain a feasible set of solutions. The next section provides a brief introduction to NSGA-II followed by the definition of the problem.

4.2. Non-Dominated Sorting Genetic Algorithm II

Non-Dominated Sorting Genetic Algorithm II (NSGA-II) is a MOO algorithm that is inspired by the natural selection and evolution of a species through survival of the fittest. The algorithm can be broken down into the following steps: Generating New Generation, Sorting, Selection and Mutation.

A new generation is generated by performing a crossover of the input variables of two parents to generate a child population. The parent and child populations are combined and sorted by generating levels of non-dominated solutions. This process is what gives the algorithm its name. After the solutions are ranked into different levels we can use a selection technique to define the parents of the next generation to ensure the survival of the best performing individuals as well as ensure genetic diversity in the population. In NSGA-II this is done using the crowding distance selection with the solutions that are far apart being given a higher priority. After selection Mutation is performed to increase exploration of the design space. Once the new generation of the parent population is ready we can repeat the process until the exit condition, usually, a fixed number of generations, is reached. [46] provides a succinct explanation of the algorithm and how it is implemented for beginners.

For implementing NSGA-II in python multiple libraries were found. A comparison of these libraries is provided in table 4.1 Out of these PYMOO is chosen because it provides a diverse set of algorithms with constraint handling inbuilt in the package. PYMOO has in-built objects predefined for commonly used selection, crossover and mutation techniques and helps the user get a start in MOO without worrying about in-depth details. Now that the algorithm and implementation are decided the problem needs to

Table 4.1: Comparision of python libraries that can be used to perform Multi-Objective Optimisation

Python Library	MOO algorithms support	Constraint Handling	Ease of Use
PYMOO	Best	For most Algorithms	Best
PYGMO	Best	For some algorithms	Best
jMetalPy	good	No inbuilt support	good

be defined and parameters and objectives are finalised.

4.3. Definition of the MOO problem

The problem that we optimise is the specific torque density of the drone machine subject to thermal, efficiency and torque production constraints. The thermal constraints ensure that the machine can operate within safe temperature conditions such that the lifetime of the parts is not affected. Having a highly inefficient machine will lead to additional weight that the drone has to carry due to requiring a bigger battery size. There is a need to restrict the design space to a manageable level so that the amount of time spent on evaluations can be reduced. Thus some design decisions are taken before beginning the optimisation process.

The most common machine topology used in drone machines is the Surface Mounted Permanent Magnet Synchronous Machines [2], [4], [24], [25]. This is because they have high power density because of the possibility of using Neodymium magnets. Although Interior Permanent Magnet Synchronous Machines(IPMSM) also use Neodymium magnets their torque-producing capability is lower due to the loss of flux in pole bridges. In [47] the authors argue that Neodymium SPMSM have the highest torque producing capability across most commonly used topologies including ferrite PMSM, IPMSM, Synchronous Reluctance Machine(SRM), Wound Field Synchronous Machine (WFSM) and Induction Machine (IM). Outer rotor PMSM are generally used in drone applications since they have a higher bore or air gap diameter compared to inner rotor machines which leads to them having a larger torque capability in the same volume as we know that the torque of a machine is dependent on the air gap diameter [48]. Another important advantage of outer rotor machines is that the frame can be directly attached to the rotor and the machine is cooled by the motion of the rotor in still air. An internal rotor machine would require a cooling mechanism on the frame to ensure sufficient cooling. Thus only the outer rotor SPMSM is considered in this thesis, however, the method developed can analyse inner rotor SPMSM as well as other topologies with some modifications.

There are many choices for windings with various pros and cons to each choice. Double Layer Fractional Slot Concentrated Windings are suitable for drone applications due to their smaller end winding length [49]. End windings do not contribute to torque production but do lead to copper losses and increase the weight of the machine. However, the choice of the number of slots and poles has a big

impact on the winding factor for double layer fractional slot windings and hence care should be taken when choosing these values.

The variables that affect the design of a machine are listed in table 4.3 and table 4.4. These variables have been divided into two categories: discrete and continuous which helps in performing optimisation as we only deal with one category of variables at a time. This list of variables is not exhaustive and the resulting designs will not be sufficient to manufacture a machine. However, the resulting designs provide a good starting point for the whole process. Notable variables excluded are the tooth shape and the winding specifications. Winding specification especially the number of turns and number of strands is dependent on the choice of voltage source as well as mechanical constraints. Choosing a winding specification independent of these values is not possible and they will change for each design candidate. Hence only the copper fill factor is specified and used for calculations. Care is taken that the conductor strand thickness is not larger than the skin depth to minimize an unnecessary increase in copper losses. However, if stranded wires and multiple turns are required this will lead to an increase in copper losses due to the proximity effect and sufficient care should be taken. The connection of the coils of the winding is provided by PYLEECAN directly and they use the star of slots method [50]. The connection of turns as well as the number of turns per coil can be selected once the optimal design is selected from the Pareto front based on the voltage and current constraints from the power source.

This still leaves the task of determining the q-axis current required to produce the torque required. We can measure the magnet's flux linkage by performing a no-load test and measuring the d-axis flux linkage. After this, the required q-axis current can be calculated using equation (2.12). However, since we are using a magnetostatic simulation to determine the magnet's flux linkage, the zeroth harmonic estimated might be incorrect due to the presence of higher harmonics. Thus we might have to change the current to get the required torque. Modified Richardson's iteration is used to calculate the q-axis current. Assuming that the magnet's flux linkage is Ψ_m and T_r is the required torque we can calculate the current using the flow chart shown in figure 4.1.

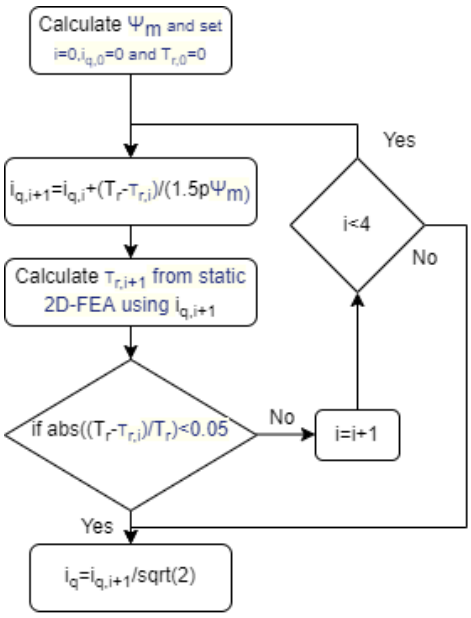


Figure 4.1: Flow Chart for determining q-axis current to get a specific torque

4.4. Bounding of Design Space

Now that we have a set of variables we need to define the values these variables can take. The discrete variables of rotational speed and torque are chosen from the datasheet of the drone machine [44]. In [51], a study of iron core materials is performed especially Cobalt Iron alloys when compared to Silicon Steel. Silicon steel is the most commonly used material in electrical machines. Cobalt Iron alloy sheets

have a higher saturation magnetization $(2.4$ T) when compared to silicon steel $(1.6$ T) for a similar mass density. However, However, higher magnetic field density will lead to higher iron loss density within the core leading to an interesting trade-off which is to be explored. The loss coefficients for the materials are derived from the loss data in appendix H for SiFe and in for appendix I for VaCoFe respectively.

The slot pole combination directly affects the winding factor, which is to be explored as well. [52] provides a detailed analysis for choosing slot pole combinations keeping the winding factor in mind. Here we explore a few slot pole combinations in the neighbourhood of the example machine's slot/pole of 24/28. For having a balanced 3 phase system the number of slots(S_s) should be a multiple of three as well for the value of integer n

$$\frac{S_s}{3n} = HCF(S_s, 2p) \tag{4.1}$$

The winding factor k_w of a winding can be found by multiplying the distribution factor k_d and pitch factor k_p.

$$k_w = k_d k_p \tag{4.2}$$

$$k_d = \frac{sin(\frac{\alpha_u}{2})}{zsin(\frac{\alpha_u}{2z})} \tag{4.3}$$

where $\alpha_u = \frac{P\pi}{S_s}$ and $z = \frac{S_s}{gcd(S_s,3P)}$.

$$k_p = sin(\frac{P\pi}{2S_s}) \tag{4.4}$$

[53] provides an easy reference spreadsheet with these values calculated already for a large number of slot pole combinations. table 4.2 provides the analysis for some slot pole combinations in the vicinity of 24/28.

Table 4.2: Comparison of the winding factor for different pole slot combinations

N_s/P	**24**	**26**	**28**	**30**	**32**
18	0.866	-	-	-	-
21	unbalanced	-	0.866	unbalanced	-
24	unbalanced	0.949	0.933	unbalanced	0.866
27	0.945	-	-	0.945	-
30	unbalanced	0.936	0.951	unbalanced	0.951

Out of these the combinations 24/26, 27/24 and 27/30 are selected. Slot pole combinations with 30 slots are ignored as the area of the slot will be too low compared to 24 slots.

Table 4.3: Discrete Variables and their values

Discrete Variables	**Values**
Stator and Rotor Core Material	SiFe; VaCoFe
Magnet Material	N35H
Slot/Pole combinations	24/26,24/28,27/24,27/30
Rotational Speed	4300 RPM
Torque at Rotational Speed	3.45 Nm
Copper Fill Factor	0.40
Air Gap Length	1 mm

Table 4.4: Continuous Variables and their limits

Continuous Variables	Lower Limit	Upper Limit
Rotor external radius (R_{Rext})	40 mm	66.5 mm
Stator internal radius (R_{Sint})	10 mm	30 mm
Stator outer radius/rotor outer radius($\frac{R_{Sext}}{R_{Rext}}$)	0.5	0.8
stack length (L)	10 mm	50 mm
magnet height rotor height ratio($\frac{l_m}{R_{Rext}-R_{Rint}}$)	0.4	0.9
magnet angle (α_m)	0.6	0.9
Teeth height ratio ($\frac{H_0}{R_{Sext}-R_{Sint}}$)	0.05	0.2
Slot height ratio ($\frac{H_2}{R_{Sext}-R_{Sint}}$)	0.4	0.7
Slot opening ratio ($\frac{W_0}{\tau_s}$)	0.1	0.3
teeth thickness ratio ($\frac{W_t}{\tau_s}$)	0.1	0.3

4.5. Objectives and Constraints

Now that we have defined the variables and their bounds we can work on the parameters used to evaluate the designs. The optimisation is run with two objectives: minimizing losses at the operating point and minimizing the active weight of the machine. Selection of efficiency and specific torque density as objectives is also possible but they will be a maximizing optimisation.

Optimising the design to minimise losses and active weight will yield a Pareto optimal front that can be used to find the lowest weighing machine at an acceptable level of losses. We can then use the data generated to produce efficiency vs specific power density curves as well.

These machines should be able to deliver the performance required without causing lasting damage, Thus constraints related to the average torque and operating temperature of the winding were added. These constraints are listed below.

$$0.95 \geq \frac{T_{em}}{T_{em,ref}} \leq 1.05 \tag{4.5}$$

$$T_{wind} \leq 200 \tag{4.6}$$

Although we need the torque to be equal to 3.45 Nm, equality constraints are usually not used in MOO. This is due to the fact that equality constraints make the convergence of the algorithm difficult as a large fraction of the population is eliminated due to constraints not leaving enough parent population to maintain diversity in the population. Thus equality constraints are converted to inequality constraints and then used to run the algorithm. Once a design is finalised a more localised MOO with tighter constraints can be defined to get a design with the required torque.

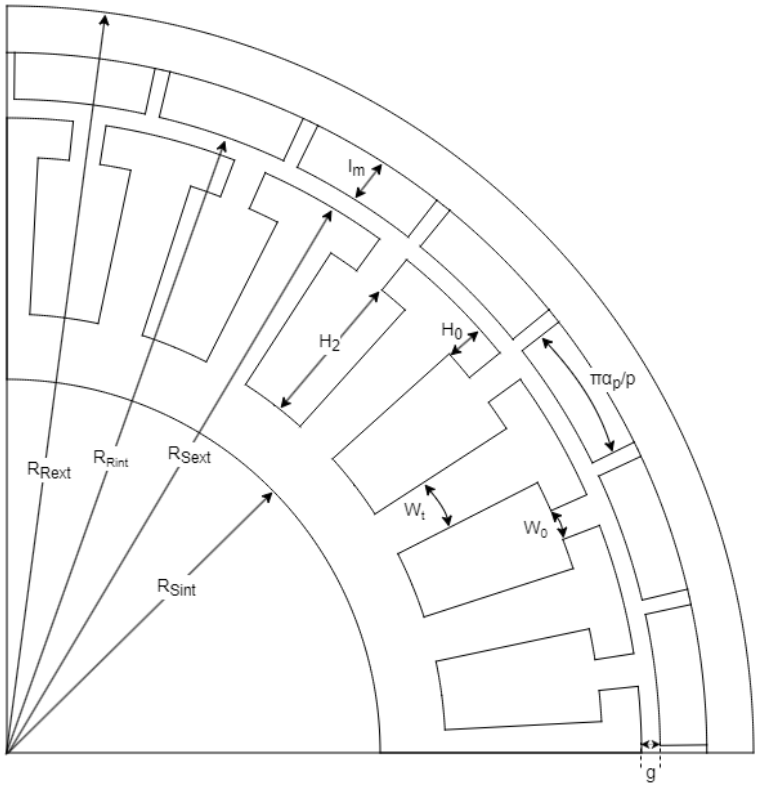


Figure 4.2: Variables used to define the structure of the machine to run MOO

5

Results and Discussion of MOO

This chapter presents the results obtained from the Multi-Objective Optimisation and discusses some key findings obtained from it. The results have been divided into sections according to parameters that were varied to get the results.

5.1. Effect of Slot Pole Combination

To study the effect of slot pole combinations on the performance of the machine, four sets of MOO with discrete variables modified according to the selected slot pole combinations are performed with a population of 100 and for 25 generations. The loss versus active weight curve for these is presented in figure 5.1.

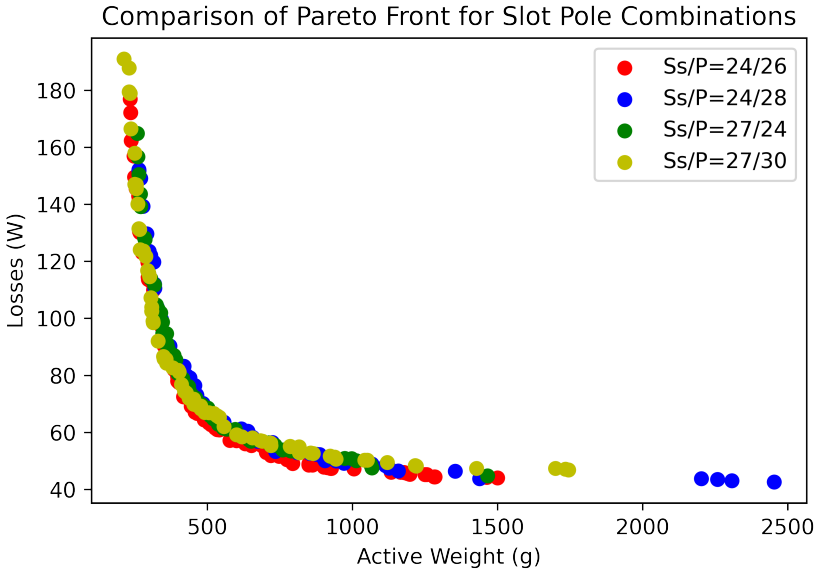


Figure 5.1: Comparison of Pareto Fronts for Slot Pole Combinations

We can see that the Pareto fronts generated have the same shape for all slot pole combinations.

This is expected as the factors affecting the losses and weight are the materials used and the thermal constraints hence the same Pareto fronts are generated. The slight differences in performance can be explained by the difference in winding factors for these slot pole combinations. The efficiency vs specific power density curves for different slot pole combinations is shown in figure 5.2.

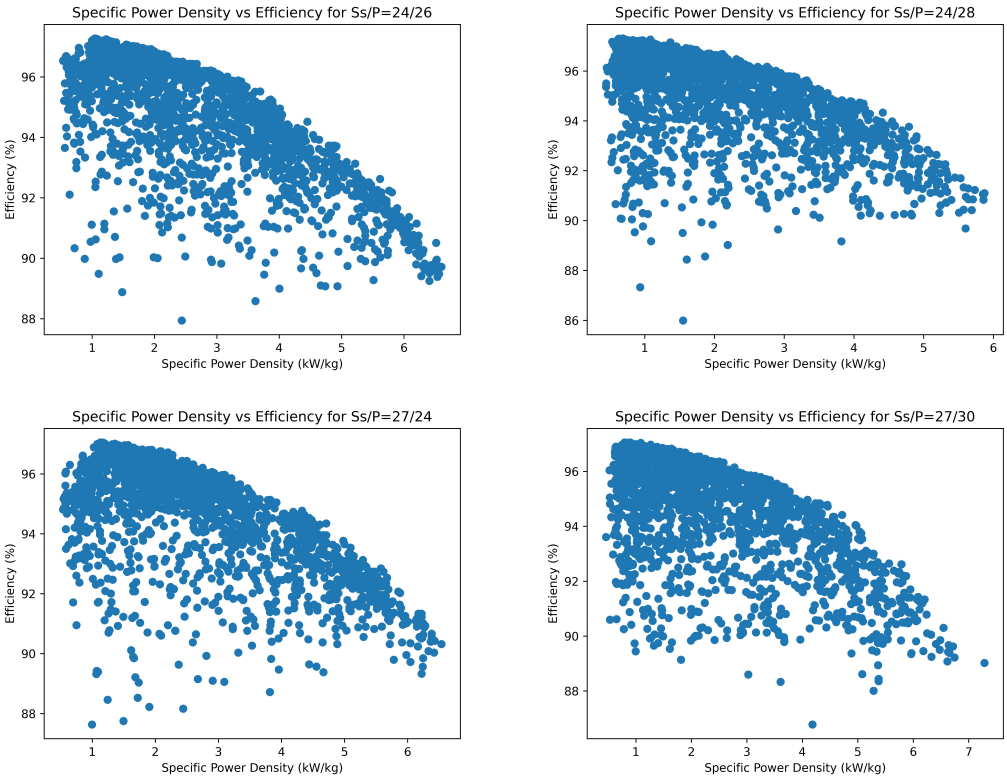


Figure 5.2: Specific Power Density vs Efficiency for Different Slot Pole Combinations

There is a clear trade-off between efficiency and specific power density. This is explained by the increase in copper and iron losses as we try to reduce the weight of the machine. The magnetic field density in the core for the same flux increases if the area of cross-section is reduced and the wire resistance increases if we use thinner wires. Both lead to higher losses. A specific power density of about 6 kW/kg is possible for forced cooling machines based on SiFe core and Neodymium magnets. The slot pole combination of 27 slots and 30 poles has the highest specific power density of about 7 kW/kg. It is important to note here that these are not the only criteria used to select the machine design and further exploration especially of torque ripple and manufacturability needs to be performed before a decision can be made.

5.2. Effect of Temperature constraints

To study the effect of the thermal constraints on the performance of the machine, three sets of MOO were performed by choosing the slot pole combination of 24 slots and 28 poles and changing the winding thermal limit constraint to 130^o C, 150^o C and 200^o C. The population size and number of generations are again chosen to be 100 and 25 respectively. The losses versus active weight curves for these are presented in figure 5.3.

It is clear from the loss vs active weight graph that the losses are limited by the temperature constraint and the Pareto front remains the same. Thus there is a limit on the minimum weight that is possible for a particular temperature constraint and thus a limit on the specific power density. The change in temperature constraint doesn't affect the Pareto front only the maximum possible losses. This can be explained by the fact that the constraint doesn't affect the actual cooling of the machine which depends

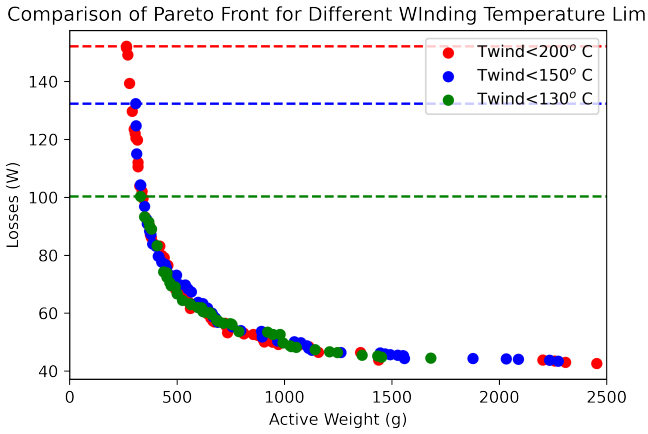


Figure 5.3: Comparison of Pareto Fronts for Different Winding Temperature Limits (Ss/P=24/28)

on the materials used and the cooling method used.

Specific power density limit increases from about 5 kW/kg for winding temperature constraint of 130^o C to about 6 kW/kg for winding temperature constraint of 200^o C. Thus if we have materials capable of handling higher temperatures we can have higher specific power density.

5.3. Effect of machine parameters

In this section, the effects of machine geometry variables are studied. This is done by studying the MOO for the 24/28 slot pole combination with a population size of 100 and 25 generations.

5.3.1. Effect of Stack Length and Air Gap Radius

Essen's rule is a commonly used equation for finding the initial size of the machine. The equation is given by:

$$T_{em} = \frac{\pi^2}{\sqrt{2}} k_{w1} A \hat{B_g} D_{ag}^2 L_{st} \tag{5.1}$$

Where T_{em} is the torque produced, k_{w1} is the winding factor of the winding, A is the RMS value of the linear current density, $\hat{B_g}$ is the peak air gap flux density, D_{ag} is the bore diameter and L_{st} is the effective stack length.

An important conclusion to make from figure 5.4 is that the air gap radius is much larger than the stack length. This is expected as increasing the diameter has a larger effect on the output torque than the stack length.

5.3.2. Rotor External Radius and Stator Inner Radius

The outer radius of the machine is an important factor in drone application due to the limited space available. Since our objective was to minimize weight it is expected that the algorithm tries to push the external rotor radius variable (R_{Rext}) to its lower limit, however, this is not the case as is shown in figure 5.5.

Although the algorithm reduces the radius, the losses also increase when the radius is reduced. Also reducing the outer radius increases the thermal resistance of the frame (lesser cooling area) and thus the machine heats up more and thus the machine designs may become thermally infeasible. We can see that the lower boundary of the solution space ($R_{Rext} = 40$ mm) is not touched by the MOO routine.

Similarly, the Stator inner radius is increased to reduce the weight of the machine to get a higher specific power density as shown in figure 5.5. Also increasing the stator inner radius increases the area over which the cool air flows allowing higher losses and hence higher specific power density. Since the stator

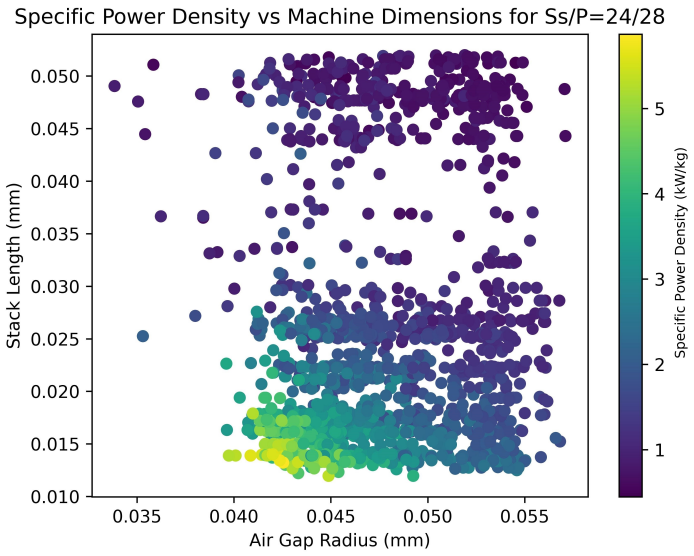


Figure 5.4: Specific Power Density vs Machine Dimensions

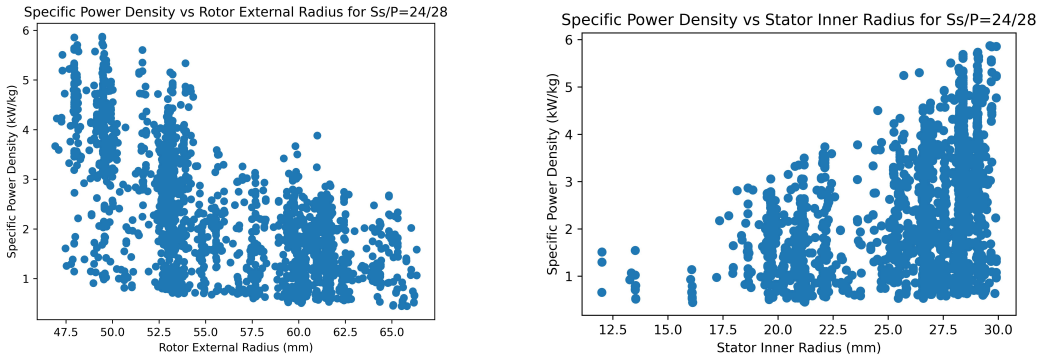


Figure 5.5: Specific Power Density vs Rotor External Radius and Stator Inner Radius for different Slot Pole Combinations

inner radius is approaching the upper limit further investigation can be performed to find the optimum value.

5.3.3. Teeth width and Stator Back Iron Thickness

The magnetic flux in the machine goes through the airgap, stator teeth, stator back iron, rotor back iron and magnets. Having thicker teeth and back iron will mean a reduction in the amount of magnet required to get the same air gap flux density. Thus an interesting compromise of the amount of magnets and amount of core steel is present in the problem. The figure 5.6 shows the specific power density with yoke and teeth thickness. We can see that the algorithm has found a compromise between reducing the thickness to the minimum or increasing it to the maximum.

From figure 5.7 it can be seen that the Specific Power Density is highest at the lower limits of the stator yoke thickness. Thus having more amount of magnet is preferable and the yoke is pushed to be saturated to have the highest specific power density.

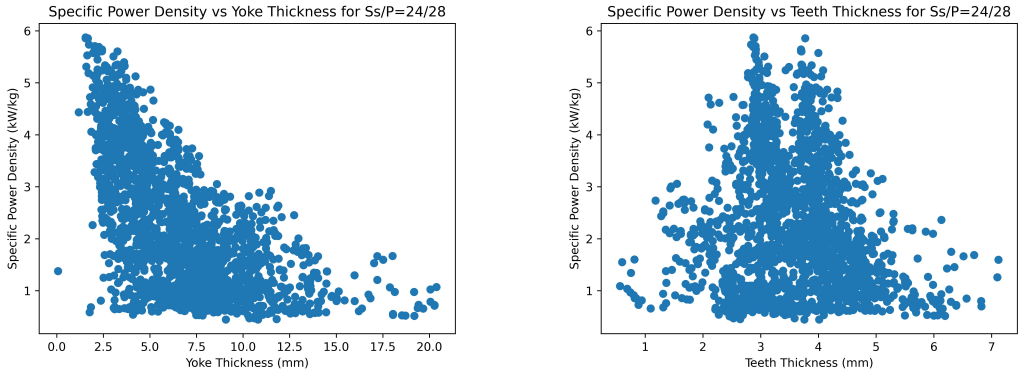


Figure 5.6: Comparison of Specific Power Density with Stator Yoke and Teeth Thickness

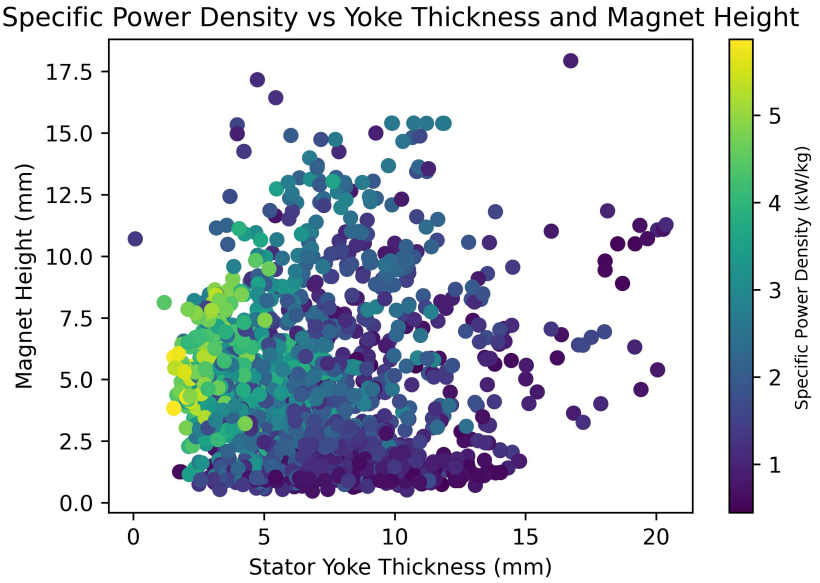


Figure 5.7: Comparison of Specific Power Density with Stator Yoke and Teeth Thickness for Ss/P=24/28

5.3.4. Effect of Magnet Angle Ratio and Slot Opening Angle

The magnet angle and slot opening angle affect the air gap magnetic flux density. Having a higher magnet angle will mean better utilization of rotor space with more magnets being present for the same magnet height and having a smaller slot opening reduces the air gap reluctance, which leads to higher air gap flux density for the same amount of magnets. These two values are restricted by mechanical constraints, a minimum slot opening is needed to wind the windings and the magnets also require space for the adhesive to hold them in place. For the MOO routine, these constraints were ignored. The slot opening angle as a percentage of slot pitch and the magnet angle as a percentage of half pole pitch ($\frac{2\pi}{P}$) vs specific power density is shown in figure 5.8.

Reducing the slot opening angle increases the specific power density as expected but only up to a limit. After about 40 %, reducing the slot opening will increase the flux leaking through the stator teeth tip instead of increasing the flux linkage of the magnets. The same is true for the magnet angle increasing the magnet angle above about 75 % will lead to more flux returning to the adjacent magnets without linking the winding.

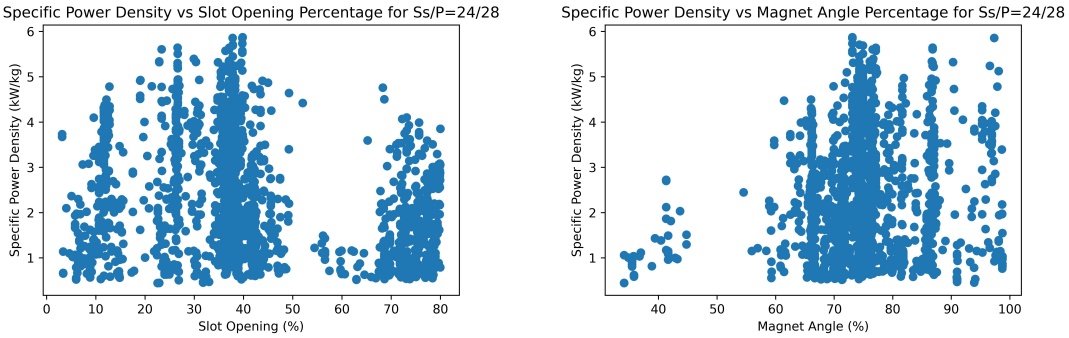


Figure 5.8: Comparison of Specific Power Density with Slot Opening Angle and Magnet Angle

5.4. Effect of Core Material

This section discusses the effect of the core material on the design. This is done by comparing two types of core materials SiFe and CoFe. This helps us gain insight if using an expensive material with a higher saturation point that increases the specific power density limit.

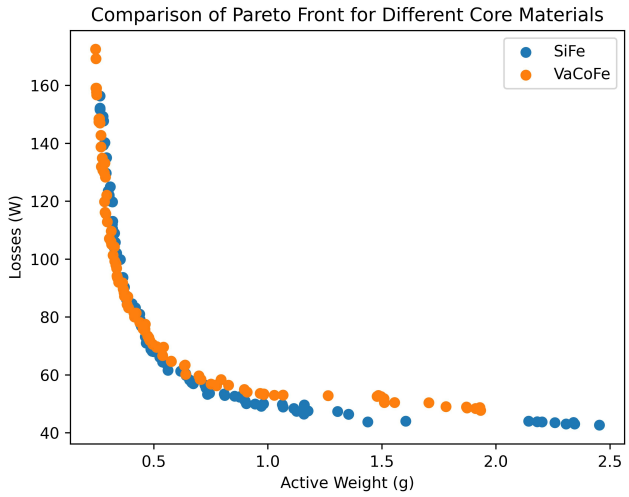


Figure 5.9: Comparison of Pareto Front for Different Core Materials

The Pareto fronts for Losses vs Active Weight for the two core materials for 24 slots and 28 poles are shown in figure 5.9. From the figure, we can see that VaCoFe has lower losses at lower weights. VaCoFe is also able to operate at higher losses due to better thermal conductivity of the material.

The average air gap magnetic field density for the two materials shown in figure 5.10. VaCoFe is able to achieve higher air gap magnetic flux density for the same weight. Thus the VaCoFe core is more saturated compared to SiFe core but still has lower losses at the same weight. This means that the specific power density for VaCoFe Core should be higher.

The specific power density vs efficiency graph for VaCoFe core is shown in figure 5.11. The maximum specific power density of VaCoFe core is 6.325 kW/kg compared to 5.871 kW/kg for SiFe core for the same slot pole combination of 24/28. Thus using VaCoFe core leads to a 7.7 % increase in specific power density.

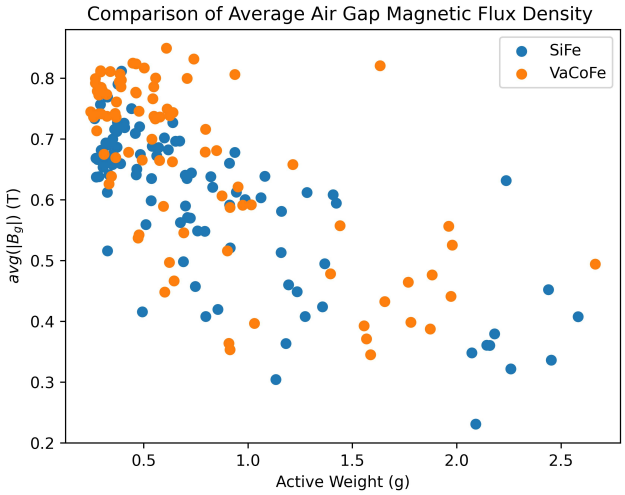


Figure 5.10: Comparison of Average Air Gap Magnetic Flux Density for Different Core Materials

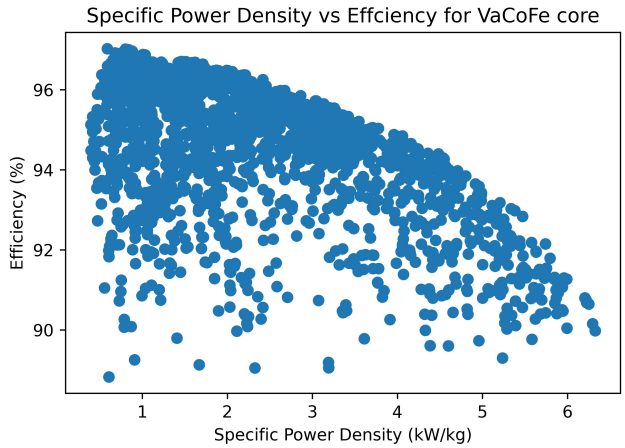


Figure 5.11: Specific Power Density vs Efficiency for VaCoFe core for Ss/P=24/28

6

Conclusion and Future Work

This chapter summarises the developments that take place in this thesis and discusses the findings of the MOO of the drone machine. This is then followed by recommendations for future work and avenues that can be further investigated.

6.1. Conclusions

Designing electrical machines is a complex endeavour which involves calculating the performance of the machine with respect to different physical aspects including magnetic field distribution, torque calculation, loss calculation and heat flow. The traditional methods used to design electrical machines are based on rules of thumb derived from prior experience and do not yield the most optimum design. The presence of conflicting objectives such as increasing efficiency, reducing cost etc. leads to the process being iterated over until a design satisfying all objectives and constraints can be found. This makes Multi-Objective Optimisation an attractive option to help speed up the process.

The aim of this thesis was to develop a Multi-Objective Optimisation (MOO) platform using open source platforms. This task has been completed successfully by using PYLEECAN which is a python library that provides an easy way to create machine geometries for many topologies. PYLEECAN also provides a link to FEMM which is an open-source finite element solver. Since the MOO routine will be analysing hundreds of designs, it is important that we reduce the time spent on analysing one design as this will allow us to explore more designs. Thus a compromise between accuracy and calculation time needs to be found especially for the Finite Element Analysis (FEA) since it has the highest time complexity. 2D-FEA was found to have the best compromise among the analytical and numerical methods explored in this thesis.

A 6 time-step magnetostatic analysis was developed to calculate the average torque performance of the machine and iron losses in the stator and rotor core. This calculation was complemented by calculating the copper losses taking end winding resistance and skin and proximity effect into account. Furthermore, windage losses were calculated based on empirical equations. Magnet losses and bearing losses were ignored due to high time complexity and lack of information respectively. The results were comparable to previous studies that used transient analysis to extract the same information.

For estimating the thermal heat flow in the machine steady state Lumped Parameter Thermal Networks (LPTN) were used. LPTN method draws an analogy between the flow of current in electrical circuits with the flow of heat in the machine. LPTN were preferred over 3D-FEA thermal analysis and Computation Fluid Dynamic (CFD) due to its lower time complexity. 2D-FEA thermal analysis could not be used as the heat flow in electrical machines is in both axial and radial directions. However, LPTN still does not

provide accurate information about hot spots in the machine and sufficient care should be taken when choosing the final design. LPTN for both natural convection cooling and forced air cooling was created. These models were found to have satisfactory performance when compared with previous studies.

There are many MOO algorithms available each with its specific pros and cons. This thesis uses Non-Dominated Sorting Genetic Algorithm-II which has been chosen for its simplicity and ease of use. NSGA-II is inspired by genetic evolution and natural selection observed in nature to get the best set of results called Pareto optimal solutions.

Once the code for the MOO framework is completed this is then used to optimise a machine for drone application. To make the design space manageable, outer rotor permanent magnet synchronous machines are chosen for their high torque capability and efficiency. The windings are chosen to be double-layer fractional slot concentrated windings to reduce the end winding length and get lower weight and lower copper losses. The

The results from the MOO show the following insights into the design of electrical machines for drone applications:

- In general the cooling method and material used has the biggest impact on the specific power density
- The slot pole combination of 27/30 was found to have the highest specific power density of 7 kW/kg. This is due to the winding factor being higher for the 27/30 combination which leads to better utilization of the materials used.
- Having a smaller stack and larger air gap radius is preferred to get a higher specific power density. This can be explained by Essen's rule where the torque developed is a function of the square of air gap radius and of the effective stack length.
- Using VaCoFe core leads to a higher specific power density of about 6.3 kW/kg compared to 5.8 kW/kg for SiFe. VaCoFe is able to have a higher air gap magnetic flux density and thus higher torque and specific power density.
- The steel core has the highest contribution to the machine weight and hence to get the highest specific power density the core is made to be highly saturated. Thus choosing yoke and teeth thickness at their lowest possible value will lead to higher specific power density

Now that we have discussed the results it is important to reflect on the conditions under which these results were derived. The MOO routine evaluated various machine designs but at a particular operating point. Electrical machines especially those used in the transportation industry seldom operate at a constant load. For example, for drones, much higher torque is required when the drone is stationary to generate enough thrust for lift-off. But to maintain the drone at a constant height the motor has to only generate enough thrust (torque) to counter gravity. Thus if we were to complete the design of the drone machine other operating points have to be considered. If we optimise a machine to have the highest efficiency at its highest operating point we would have over-designed it as the machine will not be operating at that point for a long time. Usually machines are designed to have overloading capability where they can handle a higher current for a short while to produce more torque without causing thermal damage. For electric vehicle motors the operating speeds are more varied and a high efficiency is expected over a larger range of speeds. Thus the objectives have to be defined keeping different operating points in mind. The advantage of using MOO is that this can be easily implemented by changing how the objectives are calculated without reconfiguring the complete code.

The results discussed are also only valid at the said operating point. They cannot be generalised and applied to all machines independent of size. The winding configuration can be different leading to a different compromise for copper losses leading to the machine preferring longer stack length. Thus each configuration has to be analysed to get the full picture that is valid for that scenario.

6.2. Recommendations and future work

The results shown in chapter 5 have some assumptions that should be further investigated. These suggestions for future work are listed below.

- This Thesis ignores the rotor loss calculations. However, for some machines, these losses can be significant and some investigation into extracting this information from the same six-step time step analysis can be made by measuring the MMF harmonics.
- The thermal model for forced air cooling assumes that the air cooling the stator has the same speed as surface air. The effect of the stator cooling air can be investigated.
- Two important constraints could not be included in the MOO routine: the mechanical constraints on minimum thickness for various parts and the demagnetisation constraint of the magnets. These can lead to some designs being infeasible and they could be included in the MOO.

Bibliography

[1] D. Golovanov, D. Gerada, G. Sala, *et al.*, “4-MW Class High-Power-Density Generator for Future Hybrid-Electric Aircraft,” English, *IEEE Transactions on Transportation Electrification*, vol. 7, no. 4, pp. 2952–2964, 2021, ISSN: 2332-7782. DOI: 10.1109/TTE.2021.3068928.

[2] U. H. Lee, C.-W. Pan, and E. J. Rouse, “Empirical Characterization of a High-performance Exterior-rotor Type Brushless DC Motor and Drive,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Nov. 2019, pp. 8018–8025. DOI: 10.1109/IROS40897.2019.8967626.

[3] Y. C. Chong, D. Staton, Y. Gai, H. Adam, and M. Popescu, “Review of Advanced Cooling Systems of Modern Electric Machines for EMobility Application,” in *2021 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD)*, Apr. 2021, pp. 149–154. DOI: 10.1109/WEMDCD51469.2021.9425675.

[4] J. Z. Bird, “A Review of Electric Aircraft Drivetrain Motor Technology,” *IEEE Transactions on Magnetics*, vol. 58, no. 2, pp. 1–8, Feb. 2022, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/TMAG.2021.3081719.

[5] S. A. Semidey, Y. Duan, J. R. Mayor, R. G. Harley, and T. G. Habetler, “Optimal Electromagnetic-Thermo-Mechanical Integrated Design Candidate Search and Selection for Surface-Mount Permanent-Magnet Machines Considering Load Profiles,” *IEEE Transactions on Industry Applications*, vol. 47, no. 6, pp. 2460–2468, Nov. 2011, Conference Name: IEEE Transactions on Industry Applications, ISSN: 1939-9367. DOI: 10.1109/TIA.2011.2168589.

[6] Y. Duan and R. G. Harley, “A Novel Method for Multiobjective Design and Optimization of Three Phase Induction Machines,” *IEEE Transactions on Industry Applications*, vol. 47, no. 4, pp. 1707–1715, Jul. 2011, Conference Name: IEEE Transactions on Industry Applications, ISSN: 1939-9367. DOI: 10.1109/TIA.2011.2156372.

[7] Y. Zheng, L. Zhou, J. Wang, Y. Ma, J. Zhao, and Z. Chen, “A Neural Network and NSGA-II Based Multi-objective Optimization Design Method for Permanent Magnet Synchronous Machine,” in *2019 19th International Symposium on Electromagnetic Fields in Mechatronics, Electrical and Electronic Engineering (ISEF)*, Aug. 2019, pp. 1–2. DOI: 10.1109/ISEF45929.2019.9097021.

[8] M. van der Geest, H. Polinder, J. A. Ferreira, and D. Zeilstra, “Optimization and comparison of electrical machines using particle swarm optimization,” in *2012 XXth International Conference on Electrical Machines*, Sep. 2012, pp. 1380–1386. DOI: 10.1109/ICElMach.2012.6350058.

[9] N. Rivière, M. Stokmaier, and J. Goss, “An Innovative Multi-Objective optimization Approach for the Multiphysics Design of Electrical Machines,” in *2020 IEEE Transportation Electrification Conference & Expo (ITEC)*, ISSN: 2377-5483, Jun. 2020, pp. 691–696. DOI: 10.1109/ITEC48692.2020.9161650.

[10] G. Lei, J. Zhu, Y. Guo, C. Liu, and B. Ma, “A Review of Design Optimization Methods for Electrical Machines,” en, *Energies*, vol. 10, no. 12, p. 1962, Dec. 2017, Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1996-1073. DOI: 10.3390/en10121962. [Online]. Available: https://www.mdpi.com/1996-1073/10/12/1962 (visited on 07/14/2022).

[11] Y. Duan and D. M. Ionel, “A Review of Recent Developments in Electrical Machine Design Optimization Methods With a Permanent-Magnet Synchronous Motor Benchmark Study,” *IEEE Transactions on Industry Applications*, vol. 49, no. 3, pp. 1268–1275, May 2013, Conference Name: IEEE Transactions on Industry Applications, ISSN: 1939-9367. DOI: 10.1109/TIA.2013.2252597.

[12] P. Zhang, G. Y. Sizov, M. Li, *et al.*, “Multi-objective tradeoffs in the design optimization of a brushless permanent magnet machine with fractional-slot concentrated windings,” in *2013 IEEE Energy Conversion Congress and Exposition*, ISSN: 2329-3748, Sep. 2013, pp. 2842–2849. DOI: 10.1109/ECCE.2013.6647070.

[13] C. J. O’Rourke, M. M. Qasim, M. R. Overlin, and J. L. Kirtley, “A Geometric Interpretation of Reference Frames and Transformations: Dq0, Clarke, and Park,” *IEEE Transactions on Energy Conversion*, vol. 34, no. 4, pp. 2070–2083, Dec. 2019, Conference Name: IEEE Transactions on Energy Conversion, ISSN: 1558-0059. DOI: 10.1109/TEC.2019.2941175.

[14] D. Hanselman, *Brushless Permanent-magnet Motor Design*, ser. New Horizons in Comparative Politics. McGraw-Hill, 1994, ISBN: 978-0-07-026025-2. [Online]. Available: https://books.google.nl/books?id=PuhSAAAAMAAJ.

[15] Z. Zhu, D. Howe, E. Bolte, and B. Ackermann, “Instantaneous magnetic field distribution in brushless permanent magnet DC motors. I. Open-circuit field,” *IEEE Transactions on Magnetics*, vol. 29, no. 1, pp. 124–135, Jan. 1993, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/20.195557.

[16] M. Zaheer, P. Lindh, L. Aarniovuori, and J. Pyrhönen, “Comparison of Commercial and Open-Source FEM Software: A Case Study,” *IEEE Transactions on Industry Applications*, vol. 56, no. 6, pp. 6411–6419, Nov. 2020, Conference Name: IEEE Transactions on Industry Applications, ISSN: 1939-9367. DOI: 10.1109/TIA.2020.3015827.

[17] T. Lupu, R. A. Marťiş, A. I. Nicu, and C. S. Marțiş, “Open source software based design and optimization tool for electrical machines,” in *2021 9th International Conference on Modern Power Systems (MPS)*, Jun. 2021, pp. 1–5. DOI: 10.1109/MPS52805.2021.9492624.

[18] R. Crozier and M. Mueller, “A new MATLAB and octave interface to a popular magnetics finite element code,” in *2016 XXII International Conference on Electrical Machines (ICEM)*, Sep. 2016, pp. 1251–1256. DOI: 10.1109/ICELMACH.2016.7732685.

[19] K. M. de Andrade, H. E. Santos, W. M. Vilela, T. E. P. de Almeida, and G. T. de Paula, “PeMSyn: A Free Software to Assist the Design and Performance Assessment of Permanent Magnets Synchronous Machines,” in *2019 IEEE 15th Brazilian Power Electronics Conference and 5th IEEE Southern Power Electronics Conference (COBEP/SPEC)*, ISSN: 2643-9778, Dec. 2019, pp. 1–6. DOI: 10.1109/COBEP/SPEC44138.2019.9065650.

[20] A. Lehikoinen, T. Davidsson, A. Arkkio, and A. Belahcen, “A High-Performance Open-Source Finite Element Analysis Library for Magnetics in MATLAB,” in *2018 XIII International Conference on Electrical Machines (ICEM)*, ISSN: 2381-4802, Sep. 2018, pp. 486–492. DOI: 10.1109/ICELMACH.2018.8507235.

[21] P. Bonneel, J. Le Besnerais, R. Pile, and E. Devillers, “Pyleecan: An Open-Source Python Object-Oriented Software for the Multiphysic Design Optimization of Electrical Machines,” in *2018 XIII International Conference on Electrical Machines (ICEM)*, ISSN: 2381-4802, Sep. 2018, pp. 948–954. DOI: 10.1109/ICELMACH.2018.8506884.

[22] D. van Riesen, C. Monzel, C. Kaehler, C. Schlensok, and G. Henneberger, “iMOOSE-an open-source environment for finite-element calculations,” *IEEE Transactions on Magnetics*, vol. 40, no. 2, pp. 1390–1393, Mar. 2004, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/TMAG.2004.825471.

[23] *GetDP: A General Environment for the Treatment of Discrete Problems*. [Online]. Available: http://getdp.info/ (visited on 08/05/2022).

[24] B.-S. Jun, Y.-S. Kook, J.-S. Park, and C.-Y. Won, “A Development of Electronic Speed Control (ESC) for PMSMs Driving used in Drone,” in *2018 IEEE International Power Electronics and Application Conference and Exposition (PEAC)*, Nov. 2018, pp. 1–4. DOI: 10.1109/PEAC.2018.8590355.

[25] V. Carev, J. Roháč, M. Šipoš, and M. Schmirler, “A Multilayer Brushless DC Motor for Heavy Lift Drones,” en, *Energies*, vol. 14, no. 9, p. 2504, Jan. 2021, Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1996-1073. DOI: 10.3390/en14092504. [Online]. Available: https://www.mdpi.com/1996-1073/14/9/2504 (visited on 08/08/2022).

[26] Z. Zhu and D. Howe, “Instantaneous magnetic field distribution in brushless permanent magnet DC motors. III. Effect of stator slotting,” *IEEE Transactions on Magnetics*, vol. 29, no. 1, pp. 143–151, Jan. 1993, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/20.195559.

[27] N. Bianchi, L. Alberti, M. Popescu, and T. J. E. Miller, “MMF Harmonics Effect on the Embedded FE-Analytical Computation of PM Motors,” in *2007 IEEE Industry Applications Annual Meeting*, ISSN: 0197-2618, Sep. 2007, pp. 1544–1551. DOI: 10.1109/07IAS.2007.239.

[28] J. Pyrhonen, T. Jokinen, and V. Hrabovcova, *Design of Rotating Electrical Machines*. Wiley, 2009, ISBN: 978-0-470-74008-8. [Online]. Available: https://books.google.nl/books?id=%5C_y3LSh1XTJYC.

[29] S. Bandyopadhyay, “Design Optimisation of Surface Mounted Permanent Magnet Synchronous Machine for In-wheel Electric Vehicle Applications,” en, 2015. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3Af84440ae-73be-44a3-8061-2c4785312d2b (visited on 07/15/2022).

[30] Y. Wang, J. Ma, C. Liu, G. Lei, Y. Guo, and J. Zhu, “Reduction of Magnet Eddy Current Loss in PMSM by Using Partial Magnet Segment Method,” *IEEE Transactions on Magnetics*, vol. 55, no. 7, pp. 1–5, Jul. 2019, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/TMAG.2019.2895887.

[31] P. Sergeant and A. Van den Bossche, “Segmentation of Magnets to Reduce Losses in Permanent-Magnet Synchronous Machines,” *IEEE Transactions on Magnetics*, vol. 44, no. 11, pp. 4409–4412, Nov. 2008, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/TMAG.2008.2001347.

[32] G. Bertotti, “General properties of power losses in soft ferromagnetic materials,” *IEEE Transactions on Magnetics*, vol. 24, no. 1, pp. 621–630, Jan. 1988, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/20.43994.

[33] *Rotating losses in a surface mount permanent magnet motor*. [Online]. Available: https://www.femm.info/wiki/SPMLoss.

[34] J. Nerg, M. Rilla, and J. Pyrhonen, “Thermal Analysis of Radial-Flux Electrical Machines With a High Power Density,” *IEEE Transactions on Industrial Electronics*, vol. 55, no. 10, pp. 3543–3554, Oct. 2008, Conference Name: IEEE Transactions on Industrial Electronics, ISSN: 1557-9948. DOI: 10.1109/TIE.2008.927403.

[35] J. Saari, *Thermal analysis of high-speed induction machines*, en. Helsinki University of Technology, Jan. 1998, Accepted: 2012-02-10T09:12:34Z ISSN: 0001-6845, ISBN: 978-951-22-5576-4. [Online]. Available: https://aaltodoc.aalto.fi:443/handle/123456789/2155 (visited on 07/15/2022).

[36] T. Lipo, *Introduction to AC Machine Design*, ser. IEEE Press Series on Power and Energy Systems. Wiley, 2017, ISBN: 978-1-119-35216-7. [Online]. Available: https://books.google.nl/books?id=jPY2DwAAQBAJ.

[37] *Resources*. [Online]. Available: https://www.motor-design.com/resources/ (visited on 08/10/2022).

[38] J. R. Simonson, *Engineering heat transfer*. Macmillan, 1992.

[39] X. Liu, Y. Zhao, M. Lu, Z. Chen, and S. Huang, “Multi-Objective Optimization for a Dual-Flux-Modulator Coaxial Magnetic Gear With Double-Layer Permanent Magnet Inner Rotor,” *IEEE Transactions on Magnetics*, vol. 57, no. 7, pp. 1–5, Jul. 2021, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/TMAG.2021.3065464.

[40] M. Kuosa, P. Sallinen, and J. Larjola, “Numerical and experimental modelling of gas flow and heat transfer in the air gap of an electric machine,” *Journal of Thermal Science*, vol. 13, pp. 264–278, Aug. 2004. DOI: 10.1007/s11630-004-0041-4.

[41] Q. Chen, D. Wu, G. Li, W. Cao, Z. Qian, and Q. Wang, “Development of a Fast Thermal Model for Calculating the Temperature of the Interior PMSM,” en, *Energies*, vol. 14, no. 22, p. 7455, Jan. 2021, Number: 22 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1996-1073. DOI: 10.3390/en14227455. [Online]. Available: https://www.mdpi.com/1996-1073/14/22/7455 (visited on 02/09/2022).

[42] M. Skalicky, R. Pechanek, and L. Veg, “Algorithm for Creating of the Equivalent Thermal Circuit for PMSM,” in *2020 19th International Conference on Mechatronics - Mechatronika (ME)*, Dec. 2020, pp. 1–6. DOI: 10.1109/ME49197.2020.9286645.

[43] D. Bindhu Sreevalsan, “Optimisation of Interior Permanent Magnet Generator in Propeller Type Tidal Turbines,” en, 2021. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3Aef5f2813-d5cf-4338-bea3-ecc0597b7d38 (visited on 08/10/2022).

[44] *MN801-S KV120_navigator Type_motors_multirotor_t-MOTOR Store-Official Store for T-motor drone motor,ESC,Propeller*. [Online]. Available: https://store.tmotor.com/goods.php?id=721 (visited on 08/10/2022).

[45] Y. Duan, R. Harley, and T. Habetler, "Comparison of Particle Swarm Optimization and Genetic Algorithm in the design of permanent magnet motors," in *2009 IEEE 6th International Power Electronics and Motion Control Conference*, May 2009, pp. 822–825. DOI: 10.1109/IPEMC.2009.5157497.

[46] *Evolutionary Computation for Single and Multi-Objective Optimization - Course*. [Online]. Available: https://onlinecourses.nptel.ac.in/noc21_me43/preview (visited on 08/10/2022).

[47] T. A. Lipo and W. Liu, "Comparison of AC Motors to an Ideal Machine Part I—Conventional AC Machines," *IEEE Transactions on Industry Applications*, vol. 56, no. 2, pp. 1346–1355, Mar. 2020, Conference Name: IEEE Transactions on Industry Applications, ISSN: 1939-9367. DOI: 10.1109/TIA.2019.2961351.

[48] B. Ge, M. Liu, J. Dong, and W. Liu, "Torque Production Limit of Surface Permanent Magnet Synchronous Machines and Their Electromagnetic Scalability," *IEEE Transactions on Industry Applications*, vol. 57, no. 5, pp. 4353–4362, Sep. 2021, Conference Name: IEEE Transactions on Industry Applications, ISSN: 1939-9367. DOI: 10.1109/TIA.2021.3084552.

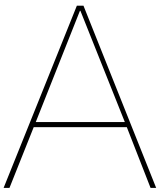
[49] A. M. EL-Refaie, "Fractional-Slot Concentrated-Windings Synchronous Permanent Magnet Machines: Opportunities and Challenges," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 1, pp. 107–121, Jan. 2010, Conference Name: IEEE Transactions on Industrial Electronics, ISSN: 1557-9948. DOI: 10.1109/TIE.2009.2030211.

[50] *Welcome to SWAT-EM's documentation! — SWAT-EM 0.6.3 documentation*. [Online]. Available: https://swat-em.readthedocs.io/en/latest/ (visited on 08/10/2022).

[51] P. P. C. Bhagubai and J. F. P. Fernandes, "Multi-Objective Optimization of Electrical Machine Magnetic Core Using a Vanadium–Cobalt–Iron Alloy," *IEEE Transactions on Magnetics*, vol. 56, no. 2, pp. 1–9, Feb. 2020, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/TMAG.2019.2950880.

[52] J. Hendershot and T. Miller, *Design of Brushless Permanent-magnet Machines*. Motor Design Books, 2010, ISBN: 978-0-9840687-0-8. [Online]. Available: https://books.google.nl/books?id=n833QwAACAAJ.

[53] R. Parsons, *Things in Motion: Selecting the best pole and slot combination for a BLDC (PMSM) motor with concentrated windings*, Jan. 2019. [Online]. Available: https://things-in-motion.blogspot.com/2019/01/selecting-best-pole-and-slot.html (visited on 07/14/2022).



Machine Geometry Generation using PYLEECAN

```
from pyleecan.Functions.load import load
from pyleecan.Classes.MachineSIPMSM import MachineSIPMSM
from pyleecan.Classes.LamSlotWind import LamSlotWind
from pyleecan.Classes.LamSlot import LamSlot
from pyleecan.Classes.SlotW22 import SlotW22
from pyleecan.Classes.Winding import Winding
from pyleecan.Classes.CondType12 import CondType12
from pyleecan.Classes.EndWindingCirc import EndWindingCirc

from pyleecan.Classes.LamSlotMag import LamSlotMag
from pyleecan.Classes.SlotM11 import SlotM11
from pyleecan.Classes.Magnet import Magnet
from numpy import pi,sqrt
def machine_generator(design_variable,discrete_variables):
    RRext=design_variable[0]
    RSint=design_variable[1]
    Rratio=design_variable[2]
    L=design_variable[3]
    lm_per=design_variable[4]
    alpha_m=design_variable[5]
    h0=design_variable[6]#teeth
    h2=design_variable[7]#slot
    w0=design_variable[8]#opening angle
    w2=design_variable[9]#slot width angle
    Steel=discrete_variables['iron']
    Copper=discrete_variables['copper']
    Insulator=discrete_variables['insulator']
    magnet=discrete_variables['magnet']
    Ns=int(discrete_variables['Ns'])
    P=int(discrete_variables['P']/2)
    g=discrete_variables['g']
```

```
kf=discrete_variables['k_f']
end_wind=EndWindingCirc()
stator_hieght=Rratio*(RRext-RSint)
RSext=RSint+stator_hieght
stator = LamSlotWind(Rint=RSint,Rext=RSext,L1=L,Nrvd=0,Kf1=0.93,is_internal=True,
    is_stator=True,mat_type=Steel)
RRint=RSext+g
lm=lm_per*(RRext-RRint)
slot_pitch=2*pi/Ns

stator.slot = SlotW22(Zs=Ns,H0=h0,H2=h2,W0=w0,W2=w2)
stator.winding = Winding(qs=3,Ntcoil=1,Npcp=1
                         ,p=P,type_connection=0,
                         Lewout=0.0001,Nlayer=2,Nslot_shift_wind=0,is_aper_a=True,end_winding
                             =end_wind)
stator.winding.conductor = CondType12(Nwppc=1, Wwire=4e-3,Wins_cond=1.6e-3,cond_mat=
    Copper,
                                      ins_mat=Insulator)
copper_area=stator.slot.comp_surface_active()*kf/2
stator.winding.conductor.Wwire=sqrt(copper_area)
pole_pitch=pi/P
rotor_slot=SlotM11(W0=pole_pitch, H0=0, Hmag=lm, Wmag=alpha_m*pole_pitch, Zs=P*2)
mag=Magnet(mat_type=magnet, type_magnetization=0, Lmag=L)
#define rotor lamination and size
rotor = LamSlotMag(magnet=mag,slot=rotor_slot,L1=L,Nrvd=0,Wrvd=0,Kf1=0.93,is_internal=
    False,Rint=RRint+lm,
                Rext=RRext,is_stator=False,mat_type=Steel)
machine=MachineSIPMSM(name="machine1",rotor=rotor,stator=stator,type_machine=1,shaft=None
    )
return machine
```

B

Iron Loss Calculations(Static and Time-Step)

```
import numpy as np
from numpy import pi
def iron_loss_static(out,coeff):
    #calculate the air gap magnetic field density then take a moving average to remove peaks
    Na_tot=out.simu.input.Na_tot
    comp_rad=out.mag.B.components["radial"].values.reshape(-1,1)
    comp_tan=out.mag.B.components["tangential"].values.reshape(-1,1)
    B_air=np.concatenate([comp_rad,comp_tan])
    B_air_mag=np.linalg.norm(B_air,axis=1)
    N=int(Na_tot/20)
    B_air_mag=np.convolve(B_air_mag, np.ones(N)/N, mode='valid')
    B_air_max=np.amax(B_air_mag) #peak air gap B
    stator=out.simu.machine.stator
    rotor=out.simu.machine.rotor
    N0=out.simu.input.OP.N0
    P=out.simu.machine.rotor.get_pole_pair_number()
    rho=stator.mat_type.struct.rho
    L=stator.L1
    Zs=stator.slot.Zs
    slot_pitch=2*pi*stator.Rint/Zs
    Tyoke=stator.Rext-stator.slot.comp_height()-stator.Rint
    if stator.is_internal:
        R_stator_bottom=stator.Rint+Tyoke
        Weight_yoke=pi*(R_stator_bottom**2-stator.Rint**2)*L*rho
    else:
        R_stator_bottom=stator.Rext-Tyoke
        Weight_yoke=pi*(stator.Rext**2-R_stator_bottom**2)*L*rho
    Rbo=stator.get_Rbo()

    stator_slot_mid=stator.slot.comp_radius_mid_active()
    stator_slot_width=stator_slot_mid*stator.slot.comp_angle_active_eq()
    Tteeth=2*pi/Zs*stator_slot_mid-stator_slot_width #teeth thickness at middle of slot to
```

```
        take average thickness
    Weight_teeth=(abs(pi*(Rbo**2-R_stator_bottom**2))-stator.slot.comp_surface()*stator.slot.
        Zs)*L*rho
    T_mag=(rotor.slot.Wmag)*(rotor.get_Rbo())

    f=N0*P/60
    B_teeth=B_air_max*slot_pitch/Tteeth #flux in teeth is from one stator pitch
    B_yoke=B_air_max*T_mag/2/Tyoke #flux in yoke is accumalated per rotor pole
    loss_per_kg=coeff[0]*f*B_teeth**coeff[1]+ coeff[2] * (f * B_teeth)**coeff[3]+ coeff[4] *
        (f * B_teeth)**coeff[5]
    loss_teeth=Weight_teeth*loss_per_kg
    loss_per_kg=coeff[0]*f*B_yoke**coeff[1]+ coeff[2] * (f * B_yoke)**coeff[3]+ coeff[4] * (f
         * B_yoke)**coeff[5]
    loss_yoke=Weight_yoke*loss_per_kg
    iron_loss_dict=dict(zip(['stator teeth','stator yoke','stator','rotor'],[loss_teeth,
        loss_yoke,loss_teeth+loss_yoke,0]))
    return iron_loss_dict

def iron_loss_transient(out,coeff):
    #Calculate the iron loss by calculating iron loss density per cell then estimating total
        loss
    machine=out.simu.machine
    P=machine.get_pole_pair_number()
    N0=out.simu.input.OP.N0
    rho=machine.stator.mat_type.struct.rho
    if out.simu.mag.is_periodicity_a:#if antiperiodicity is used losses for a part of the
        mesh are calculated and multiplied
            sym, is_antiper_a, _, _ = out.get_machine_periodicity()
            sym *= is_antiper_a + 1
            print(sym)
    else:
            sym = 1

    meshsol_stator=out.mag.meshsolution.get_group(group_names="stator core")#extracts the
        mesh and solution for the stator
    mesh_stator=meshsol_stator.mesh[0]
    sol_stator=meshsol_stator.get_solution(label="B")

    area_teeth,area_yoke=iron_loss_area_calc(mesh_stator,stator=machine.stator)
    loss_density_stator=iron_loss_density(sol_stator,P,N0,coeff)
    stator_teeth_loss=np.sum(area_teeth*loss_density_stator*machine.stator.L1*rho)*sym#
        multiplies loss density with volume and sums over whole mesh
    stator_yoke_loss=np.sum(area_yoke*loss_density_stator*machine.stator.L1*rho)*sym

    meshsol_rotor=out.mag.meshsolution.get_group(group_names="rotor core")#extracts the mesh
        and solution for rotor
    mesh_rotor=meshsol_rotor.mesh[0]
    sol_rotor=meshsol_rotor.get_solution(label="B")

    area_rotor=iron_loss_area_calc(mesh_rotor)
    loss_density_rotor=iron_loss_density(sol_rotor,P,N0,coeff)*sym
    rotor_loss=np.sum(area_rotor*loss_density_rotor*machine.stator.L1*rho)
    iron_loss_dict=dict(zip(['stator teeth','stator yoke','stator','rotor'],[
        stator_teeth_loss,stator_yoke_loss,stator_teeth_loss+stator_yoke_loss,rotor_loss]))
```

```
    return iron_loss_dict
def iron_loss_density(sol,P,N0,coeff):
    #calculates the iron loss density per cell for the input mesh solution
    comp_x=sol.field.components["comp_x"]
    comp_y=sol.field.components["comp_y"]
    B_x1=np.delete(comp_x.values,[1,2],0)# removing field corresponding to angle pi/12 and pi
        /6
    B_x2=B_x1*-1
    B_x=np.concatenate((B_x1,B_x2),axis=0)#antiperiodicity condition to convert 4 samples to
        8
    B_y1=np.delete(comp_y.values,[1,2],0)
    B_y2=B_y1*-1
    B_y=np.concatenate((B_y1,B_y2),axis=0)
    B_yf=2*abs(np.fft.fft(B_y,axis=0))/8 #peak B per cell for different harmonics
    B_xf=2*abs(np.fft.fft(B_x,axis=0))/8
    f=N0*P/60 #first harmonic contribution
    loss_density=coeff[0]*f*B_xf[1]**coeff[1]+ coeff[2] * (f * B_xf[1])**coeff[3]+ coeff[4] *
         (f * B_xf[1])**coeff[5]
    loss_density+=coeff[0]*f*B_yf[1]**coeff[1]+ coeff[2] * (f * B_yf[1])**coeff[3]+ coeff[4]
        * (f * B_yf[1])**coeff[5]
    f=N0*P/60*3  #third harmonic contribution
    loss_density+=coeff[0]*f*B_xf[3]**coeff[1]+ coeff[2] * (f * B_xf[3])**coeff[3]+ coeff[4]
        * (f * B_xf[3])**coeff[5]
    loss_density+=coeff[0]*f*B_yf[3]**coeff[1]+ coeff[2] * (f * B_yf[3])**coeff[3]+ coeff[4]
        * (f * B_yf[3])**coeff[5]
    return loss_density
def iron_loss_area_calc(mesh,stator=None):
    #returns the area per cell for the supplied mesh, if stator then area is split between
        teeth and yoke
    node=mesh.node
    if stator is not None:
        radius=np.linalg.norm(node.coordinate,axis=1).transpose()
        Rint=stator.Rint
        #R_mid=out.simu.machine.stator.slot.comp_radius_mid_active()
        Tyoke=stator.Rext-stator.slot.comp_height()-stator.Rint
        if stator.is_internal:
            node_indice=np.argwhere(radius<(Rint+Tyoke))
        else:
            node_indice=np.argwhere(radius>stator.Rext-Tyoke)
        nodes=node.indice[node_indice]
        cells = np.array([], dtype=int)
        for i in node_indice:
            alpha=np.argwhere(mesh.cell['triangle'].connectivity==i)
            cells=np.append(cells,alpha)
        area=mesh.get_cell_area()
        area_teeth=area.copy()
        area_teeth[cells]=0
        area_yoke=area-area_teeth
        return area_teeth,area_yoke
    else:
        area=mesh.get_cell_area()
        return area
```

C

Copper Loss Calculation within MOO routine

```
from numpy import pi, sinh, cos, sin,cosh,sqrt
from numpy import ndarray
#following code is from PYLEECAN but implemented differently
def comp_skin_effect_resistance(conductor, freq, T_op=20, T_ref=20, b=None, zt=None):
    """Compute the skin effect factor on resistance for the conductors from "Design of
        Rotating Electrical Machines", J. Pyrhonen, second edition
    All parameters are defined p.270 / 271
    Parameters
    ----------
    conductor : Conductor
        an Conductor object
    b: float
        Slot width [m]
    zt: int
        Number of turns in series per coil
    freq: float
        electrical frequency [Hz]
    T_op: float
        Conductor operational temperature [degC]
    T_ref: float
        Conductor reference temperature [degC]
    Returns
    ----------
    kr_skin : float
        skin effect coeff for resistance at given frequency and temperature
    """

    if b is None:
        # Compute slot average width
        slot = conductor.parent.parent.slot
        b = slot.comp_surface_active()/slot.comp_height_active()
```

```
    if zt is None:
        # Get number of turns in series per coil
        zt = conductor.parent.Ntcoil

    # check if wires are round or rectangular
    is_round_wire = True

    # conductor height
    hc0 = conductor.Wwire
    # conductor width
    bc0 = conductor.Wwire
    # Number of circumferential adjacent wires
    za = int(sqrt(conductor.Nwppc))
    # Number of radial adjacent wires
    zp =int(sqrt(conductor.Nwppc))
    # Equivalent height of conductor
    hc = zp * hc0

    # Electrical conductivity accounting for temperature increase
    rho20=conductor.cond_mat.elec.rho
    rho=rho20*(1+conductor.cond_mat.elec.alpha*(T_op-20))
    sigma = 1/rho
    # Magnetic permeability
    mu0 = 4 * pi * 1e-7
    mur=1
    # Electrical pulsation
    w = 2 * pi * freq

    # reduced conductor height Eq(5.24) p.270 + adding relative permeatibility
    ksi = hc * sqrt((1 / 2) * w * mu0 * mur * sigma * za * bc0 / b)

    if not isinstance(ksi, float):
        # Avoid numerical error with 0
        ksi[ksi == 0] = 1e-4

    # average resistance factor
    if is_round_wire:
        # Use round wire approximation Eq(5.28) p.271
        kr_skin = 1 + 0.59 * ((zt ** 2 - 0.2) / 9) * ksi ** 2
    else:
        # resistance factor function phi Eq(5.26) p.271
        phi = ksi * (sinh(2 * ksi) + sin(2 * ksi)) / (cosh(2 * ksi) - cos(2 * ksi))

        # resistance factor function psi Eq(5.27) p.271
        psi = 2 * ksi * (sinh(ksi) - sin(ksi)) / (cosh(ksi) + cos(ksi))

        kr_skin = phi + ((zt ** 2 - 1) / 3) * psi  # Eq(5.28) p.271
        # kr_approx = 1 + (zt ** 2 - 0.2) / 9 * ksi ** 4  # Eq(5.29) p.271

    return kr_skin
def Copper_Loss_MOO(stator,f,I,T,kf=0.4):
    #modified calculation for MOO routine as winding data is incomplete
    N_strands=stator.winding.conductor.Nwppc #number of strands per turn
    copper_area=stator.slot.comp_surface_active()*kf/2 #copper area from area of slot for
```

```
    double layer winding
A=copper_area/N_strands
L1=stator.L1
Ns=stator.slot.Zs
qs=stator.winding.qs
slot=stator.slot
Rext=stator.Rext
R_wavg=(Rext-slot.H0-slot.H2/2)*slot.W2/2 # average radius of end windings from slot
    shape
L=(2*pi*R_wavg+2*L1)*Ns/qs*N_strands
I_strand=I/N_strands
rho20=stator.winding.conductor.cond_mat.elec.rho
rho=rho20*(1+stator.winding.conductor.cond_mat.elec.alpha*(T-20))#Rise in temperature
R=rho*L/A
Copper_loss=3*I_strand**2*R*comp_skin_effect_resistance(conductor=stator.winding.
    conductor,freq=f,T_op=T)
#sometimes output is array following code makes data type correction
if type(Copper_loss)==ndarray:
    return Copper_loss[0]
else:
    return Copper_loss
```

D

Copper Loss Calculation For PMSM

```
import numpy as np
from numpy import pi, sinh, cos, sin,cosh,sqrt
from numpy import ndarray
def comp_skin_effect_resistance(conductor, freq, T_op=20, T_ref=20, b=None, zt=None):
    """Compute the skin effect factor on resistance for the conductors from "Design of
        Rotating Electrical Machines", J. Pyrhonen, second edition
    All parameters are defined p.270 / 271
    Parameters
    ----------
    conductor : Conductor
        an Conductor object
    b: float
        Slot width [m]
    zt: int
        Number of turns in series per coil
    freq: float
        electrical frequency [Hz]
    T_op: float
        Conductor operational temperature [degC]
    T_ref: float
        Conductor reference temperature [degC]
    Returns
    ----------
    kr_skin : float
        skin effect coeff for resistance at given frequency and temperature
    """

    if b is None:
        # Compute slot average width
        slot = conductor.parent.parent.slot
        b = slot.comp_surface_active()/slot.comp_height_active()

    if zt is None:
        # Get number of turns in series per coil
```

```
        zt = conductor.parent.Ntcoil

    # check if wires are round or rectangular
    is_round_wire = True

    # conductor height
    hc0 = conductor.Wwire
    # conductor width
    bc0 = conductor.Wwire
    # Number of circumferential adjacent wires
    za = int(sqrt(conductor.Nwppc))
    # Number of radial adjacent wires
    zp =int(sqrt(conductor.Nwppc))
    # Equivalent height of conductor
    hc = zp * hc0

    # Electrical conductivity accounting for temperature increase
    rho20=conductor.cond_mat.elec.rho
    rho=rho20*(1+conductor.cond_mat.elec.alpha*(T_op-20))
    sigma = 1/rho
    # Magnetic permeability
    mu0 = 4 * pi * 1e-7
    mur=1
    # Electrical pulsation
    w = 2 * pi * freq

    # reduced conductor height Eq(5.24) p.270 + adding relative permeatibility
    ksi = hc * sqrt((1 / 2) * w * mu0 * mur * sigma * za * bc0 / b)

    if not isinstance(ksi, float):
        # Avoid numerical error with 0
        ksi[ksi == 0] = 1e-4

    # average resistance factor
    if is_round_wire:
        # Use round wire approximation Eq(5.28) p.271
        kr_skin = 1 + 0.59 * ((zt ** 2 - 0.2) / 9) * ksi ** 2
    else:
        # resistance factor function phi Eq(5.26) p.271
        phi = ksi * (sinh(2 * ksi) + sin(2 * ksi)) / (cosh(2 * ksi) - cos(2 * ksi))

        # resistance factor function psi Eq(5.27) p.271
        psi = 2 * ksi * (sinh(ksi) - sin(ksi)) / (cosh(ksi) + cos(ksi))

        kr_skin = phi + ((zt ** 2 - 1) / 3) * psi  # Eq(5.28) p.271
        # kr_approx = 1 + (zt ** 2 - 0.2) / 9 * ksi ** 4  # Eq(5.29) p.271

    return kr_skin
def Copper_Loss_PMSM(stator,f,I,T,kf=0.4):
    #standard calculation when winding data is complete
    Copper_loss=stator.comp_resistance_wind(T=T)*I**2*3*comp_skin_effect_resistance(conductor
        =stator.winding.conductor,freq=f,T_op=T)

    if type(Copper_loss)==ndarray:
```

```
        return Copper_loss[0]
    else:
        return Copper_loss
```

E

Thermal Model for Natural Convection

```
"""This file creates a 14 node lumped parameter thermal network for a surface mounted
    permanent magnet machine """
"""The code can be adapted for other machines as well as more nodes"""
from numpy import pi, r_
from numpy import zeros
from numpy import log as ln
from numpy import sqrt
from numpy import tanh
from numpy import sinh
from pyleecan.Classes.SlotW22 import SlotW22
from pyleecan.Classes.SlotM11 import SlotM11

from Copper_Loss_PMSM import Copper_Loss_PMSM
import numpy as np
def yoke_resistance_calculator(r1,r2,L,lam):
    """determines radial thermal resistance for cylindrical yoke for T network LPTN """
    #r1 is outer radius
    #r2 is inner radius
    #L is effective stack length
    #lam is the radial thermal conductivity
    # returns the three resistances for the T network
    R1=(1-(2*r2**2*ln(r1/r2)/(r1**2-r2**2))/(4*pi*lam*L))
    R2=((2*r1**2*ln(r1/r2)/(r1**2-r2**2)-1)/(4*pi*lam*L))
    R3=((4*r1**2*r2**2*ln(r1/r2)/(r1**2-r2**2)-r1**2-r2**2)/(8*pi*lam*L)/(r1**2-r2**2))
    return R1,R2,R3

def annular_resistance_calculator(r1,r2,L,lam,alphap):
    """determines radial thermal resistance for cylindrical yoke for T network LPTN """
    #r1 is outer radius
    #r2 is inner radius
    #L is effective stack length
    #lam is the radial thermal conductivity
    #alphap is the magnet angle times the number of pole pairs
    #thermal resistance of a cylidrical shell is known multiply by 2pi and divide by magnet
```

```
        angle
    #multiply 2p to account for parallel paths
    # returns the three resistances for the T network
    R1=(1-(2*r2**2*ln(r1/r2)/(r1**2-r2**2))/(4*alphap*lam*L))
    R2=((2*r1**2*ln(r1/r2)/(r1**2-r2**2)-1)/(4*alphap*lam*L))
    R3=((4*r1**2*r2**2*ln(r1/r2)/(r1**2-r2**2)-r1**2-r2**2)/(8*alphap*lam*L)/(r1**2-r2**2))
    return R1,R2,R3

def yoke_axial_resistance_calculator(r1,r2,L,lam):
    """determines axial thermal resistance for cylindrical yoke for T network LPTN """
    #r1 is outer radius
    #r2 is inner radius
    #L is effective stack length
    #lam is the radial thermal conductivity
    # returns the three resistances for the T network
    if lam==0:
        return 0,0,0
    R1=L/(2*pi*lam*(r1**2-r2**2))
    R2=L/(2*pi*lam*(r1**2-r2**2))
    R3=L/(6*pi*lam*(r1**2-r2**2))
    return R1,R2,R3

def annular_axial_resistance_calculator(r1,r2,L,lam,alphap):
    """determines axial thermal resistance for cylindrical yoke for T network LPTN """
    #r1 is outer radius
    #r2 is inner radius
    #L is effective stack length
    #lam is the radial thermal conductivity
    #alphap is the magnet angle times the number of pole pairs
    # returns the three resistances for the T network
    if lam==0:
        return 0,0,0
    R1=L/(2*alphap*lam*(r1**2-r2**2))
    R2=L/(2*alphap*lam*(r1**2-r2**2))
    R3=L/(6*alphap*lam*(r1**2-r2**2))
    return R1,R2,R3
def rectangular_resistance_calculator(h,A,lam):
    #h is the height of the block
    #A is the area of cross section perpendicular to heat flow
    #lam is the thermal conductivity in direction of calculation
    #returns three resistance for the T network
    R1=h/(2*A*lam)
    R3=-R1/3
    return R1,R1,R3
def shaft_calculator(R,L,lam,T_surface,T_air):
    lam_contact=1100
    R1=1/(2*pi*R*L*lam_contact)
    R2=L/(2*pi*R**2*lam)
    R3=frame_axial(R*2,pi*R**2,T_surface,T_air=T_air)
    R=R1+(R2+R3)/2


    return R
def frame_axial(D,area,T_surface,T_air):
```

```
    g=9.8
    theta=T_surface-T_air
    beta=3400e-6
    cp=1.005e3
    k=0.025
    rho=1.225
    mu=1.6e-5
    ethr=0.85
    sigma_sb=5.67e-8
    Pr=cp*mu/k
    Gr=beta*g*theta*rho**2*D**3/mu**2
    Ra=Gr*Pr
    if 1e4<=Ra<1e9:
        Nu=0.59*Ra**0.25
    elif 1e9<=Ra<1e12:
        Nu=0.129*Ra**0.33
    alpha_c=Nu*k/D
    T1=T_surface+273
    T2=273+T_air
    alpha_r=ethr*sigma_sb*(T1**4-T2**4)/(T1-T2)
    R=1/(alpha_c+alpha_r)/area
    return R
def frame_ambient(D,area,T_surface,T_air):
    g=9.8
    theta=T_surface-T_air
    beta=3400e-6
    cp=1.005e3
    k=0.025
    rho=1.225
    mu=1.6e-5
    ethr=0.85
    sigma_sb=5.67e-8
    Pr=cp*mu/k
    Gr=beta*g*theta*rho**2*D**3/mu**2
    Ra=Gr*Pr
    if 1e4<=Ra<1e9:
        Nu=0.525*Ra**0.25
    elif 1e9<=Ra<1e12:
        Nu=0.129*Ra**0.33
    alpha_c=Nu*k/D
    T1=T_surface+273
    T2=273+T_air
    alpha_r=ethr*sigma_sb*((T1**4)-(T2**4))/(T1-T2)
    R=1/(alpha_c+alpha_r)/area
    return R
def airgap_calculator(rm,g,omega,area,T_surface,T_air):
    #rm is average of stator and rotor bore radius
    #g is the air gap length
    #omega is the angular velocity of the rotor
    rho=1.225#mass density of air todo implement thermal effect on properties based on
        increase in air temp
    mu=1.81e-5#dynamic viscosity of air
    lam=0.025 #thermal conductivity of air
    ethr=0.85
```

```
    sigma_sb=5.67e-8
    Ta=rho**2*omega**2*rm*g**3/mu**2
    x=(2*rm-2.304*g)/(2*rm-g)
    Fg=(pi**4*x)/(1697*(0.0056+0.0571*x**2)*(1-g/(2*rm)))
    Tam=Ta/Fg
    if Tam<1700:
        Nu=2
    elif Tam<1e4:
        Nu=0.128*Tam**0.367
    elif Tam<1e7:
        Nu=0.409*Tam**0.241
    dh=g*(8/3)**0.5
    alpha_c=Nu*lam/dh
    T1=T_surface+273
    T2=273+T_air
    alpha_r=ethr*sigma_sb*((T1**4)-(T2**4))/(T1-T2)
    R=1/(alpha_c+alpha_r)/area
    return R
def stator_calculator(out,Rmat,T,Tambient):
    stator=out.simu.machine.stator
    machine=out.simu.machine
    r_in=stator.Rint
    r_out=stator.Rext
    slot_height=stator.slot.comp_height_active()
    if stator.is_internal:
        r_yoke_in=r_in+slot_height
    else:
        r_yoke_in=r_out-slot_height
    L=stator.L1*stator.Kf1
    lam_r_iron=stator.mat_type.HT.lambda_x
    [R1,R2,R3]=yoke_resistance_calculator(r_out,r_yoke_in,L,lam_r_iron)
    Rmat[6,7]+=R1 if stator.is_internal else R2
    Rmat[0,6]+=R2 if stator.is_internal else R1
    Rmat[1,6]+=R3
    slot_base_width=stator.slot.W2*(r_yoke_in)
    Ns=stator.slot.Zs
    teeth_area=(2*pi*r_yoke_in/Ns-slot_base_width)*L
    [R1,R2,R3]=rectangular_resistance_calculator(slot_height,teeth_area,lam_r_iron)
    Rmat[6,7]+=R1/Ns
    Rmat[3,7]+=R2/Ns
    Rmat[2,7]+=R3/Ns
    rm=(stator.get_Rbo()+machine.rotor.get_Rbo())/2
    g=abs(stator.get_Rbo()-machine.rotor.get_Rbo())
    stator_teeth_area=(2*pi-Ns*stator.slot.comp_angle_opening())*stator.get_Rbo()*L
    omega=out.simu.input.OP.N0/30*pi
    Rmat[3,7]+=airgap_calculator(rm,g,omega,stator_teeth_area,T_surface=T[5],T_air=20)

    slot_insulation_thickness=0.1e-3
    lam_insulation=0.11
    lam_stator_core_contact=0.2
    area_copper=stator.comp_fill_factor()*stator.slot.comp_surface_active()
    lam_r_copper=stator.winding.conductor.ins_mat.HT.lambda_x
    lam_a_copper=stator.winding.conductor.cond_mat.HT.lambda_z
```

```
    #winding calculation
    R1=(1/(4*pi*lam_r_copper)+slot_insulation_thickness/((2*slot_height+slot_base_width)*
        lam_insulation)+1/((2*slot_height+slot_base_width)*lam_stator_core_contact))
    Rw=slot_height/(Ns*lam_a_copper*area_copper)
    Gw=1/R1
    x=sqrt(Rw*Gw)
    y=sqrt(Rw/Gw)
    R2=y*tanh(x/2)
    R3=R1*(x/sinh(x)-1)
    Rmat[2,4]+=R1
    Rmat[4,5]+=R2
    Rmat[4,5]+=R3
    lam_a_iron=stator.mat_type.HT.lambda_z
    #[R1,R2,R3]=yoke_axial_resistance_calculator(r_out,r_in,L,lam_a_iron)
    #Rmat[1,1]=R3+R1*R2/(R1+R2)
    l_end=stator.winding.comp_length_endwinding()*2
    r_sew1=r_yoke_in+slot_insulation_thickness
    r_sew2=r_sew1+slot_height
    r_sew=(r_sew1+r_sew2)*0.5
    R3=1.5*l_end/(Ns*area_copper*lam_a_copper)
    R2=1.5/(4*pi**2*(r_sew1+r_sew2)*lam_r_copper)
    Rmat[4,5]+=R3
    Rmat[5,8]+=R2/2
    end_winding_area=pi**2*r_sew*(r_sew1+r_sew2)
    R1=airgap_calculator(rm,g,omega,end_winding_area,T_surface=T[5],T_air=20)
    R2=airgap_calculator(rm,g,0,end_winding_area,T_surface=T[5],T_air=20)
    Rmat[5,8]+=R1*R2/(R1+R2)
    if stator.is_internal:
        R=machine.rotor.Rext
        Rmat[5,8]+=frame_axial(R*2,pi*R**2,T_surface=T[8],T_air=20)
    else:
        R=machine.stator.Rext
        Rmat[5,8]+=frame_axial(R*2,pi*R**2,T_surface=T[8],T_air=20)
    Rmat[0,0]+=shaft_calculator(R=stator.Rint,L=stator.L1,lam=lam_r_iron,T_surface=T[0],T_air
        =Tambient)
def rotor_calculator(out,Rmat,T,Tambient):
    machine=out.simu.machine
    rotor=machine.rotor
    r_in=rotor.Rint
    r_out=rotor.Rext
    L=rotor.L1*rotor.Kf1
    lam_r_iron=rotor.mat_type.HT.lambda_x
    [R1,R2,R3]=yoke_resistance_calculator(r_out,r_in,L,lam_r_iron)
    Rmat[8,11]+=R1 if rotor.is_internal else R2
    Rmat[11,12]+=R2 if rotor.is_internal else R1
    Rmat[9,11]+=R3
    #magnet calculation
    Hmag=rotor.slot.Hmag
    lam_r_mag=rotor.magnet.mat_type.HT.lambda_x
    alphap=rotor.slot.Wmag*rotor.get_pole_pair_number()
    [R1,R2,R3]=annular_resistance_calculator(r_out+Hmag,r_out,L,lam_r_mag,alphap)
    Rmat[11,12]+=R1 if rotor.is_internal else R2
    Rmat[3,12]+=R2 if rotor.is_internal else R1
    Rmat[10,12]+=R3
```

```
    lam_a_iron=rotor.mat_type.HT.lambda_z
    lam_a_mag=rotor.magnet.mat_type.HT.lambda_z
    #[R1,R2,R3]=yoke_axial_resistance_calculator(r_out,r_in,L,lam_a_iron)
    #Rmat[9,9]=R3+R1*R2/(R1+R2)
    #[R1,R2,R3]=annular_axial_resistance_calculator(r_out+Hmag,r_out,L,lam_a_mag,alphap)
    #Rmat[10,10]=R3+R1*R2/(R1+R2)
    #Rmat[8,8]=shaft_axial() if rotor.is_internal else frame_axial()
    rm=(machine.stator.get_Rbo()+rotor.get_Rbo())/2
    g=abs(machine.stator.get_Rbo()-rotor.get_Rbo())
    rotor_area=2*alphap*L*rotor.get_Rbo()
    omega=out.simu.input.OP.N0/30*pi
    Rmat[3,12]+=airgap_calculator(rm,g,omega,rotor_area,T_surface=T[10],T_air=20)
    Rmat[8,11]+=airgap_calculator(r_out+g/2,g,omega,2*pi*(r_out+g/2)*L,T_surface=T[9],T_air
        =20)
def ThermalLPTN_SIPMSM_run(out,Tambient,iron_loss_dict,I_rms,kf=0.4):
    machine=out.simu.machine
    N0=out.simu.input.OP.N0
    P=machine.get_pole_pair_number()
    Nstator=8
    Nrotor=6
    Ntotal=Nstator+Nrotor-1
    Rmat=zeros([Ntotal,Ntotal])
    Gmat=zeros([Ntotal,Ntotal])
    Pvec=zeros([1,Ntotal])
    eye=np.identity(Ntotal)
    tamb=np.ones([1,Ntotal])*Tambient
    T=np.array([65,65,65,20,110,110,20,20,65,65,65,20,20],dtype=float)
    Node_name=["shaft","stator yoke","stator teeth","air gap","winding","end winding","node 6
        ","node 7","frame",
              "rotor yoke","magnets","node 11","node 12"]
    if type(machine.rotor.slot)!=SlotM11:
        #print("rotor geometry not supported")
        print(machine.rotor.slot)
        return Rmat,Pvec
    if type(machine.stator.slot)!=SlotW22:
        #print("stator geometry not supported")
        print(machine.stator.slot)
        return Rmat,Pvec
    for i in range(100):
        Rmat=zeros([Ntotal,Ntotal])
        Gmat=zeros([Ntotal,Ntotal])
        Told=T.copy()
        Lwinding=Copper_Loss_PMSM(stator=machine.stator,f=N0/60*P,T=T[4],I=I_rms,kf=kf)
        Lrotoryoke=iron_loss_dict['rotor']
        Lmag=0
        Lstatoryoke=iron_loss_dict['stator yoke']
        Lteeth=iron_loss_dict['stator teeth']
        Pvec=np.array([0,Lstatoryoke,Lteeth,0,Lwinding,0,0,0,0,Lrotoryoke,Lmag,0,0]).reshape
            (-1,1)
        Pvec_dict=dict(zip(Node_name,Pvec.tolist()))
        R1=frame_ambient(D=2*machine.rotor.Rext,area=2*pi*machine.rotor.Rext*machine.rotor.L1
            ,T_surface=T[8],T_air=Tambient)
        R2=frame_axial(D=2*machine.rotor.Rext,area=2*pi*machine.rotor.Rext**2,T_surface=T[8],
            T_air=Tambient)
```

```
        Rmat[8,8]=R1*R2/(R1+R2)


        stator_calculator(out,Rmat,T,Tambient)
        rotor_calculator(out,Rmat,T,Tambient)

        Gmat=np.where(Rmat==0,0,1./Rmat)
        amb=np.diagonal(Gmat)
        Gmat=Gmat-eye*amb

        Gmat=Gmat+np.transpose(Gmat)


        Gsum=np.sum(Gmat, axis=1)

        Gmat=Gmat*-1


        Gmat=Gmat+eye*(amb+Gsum)
        #np.savetxt('Gmat.csv', Gmat, delimiter=",")
        ambtamb=amb*tamb

        Pvec=Pvec+np.transpose(ambtamb)
        Ginv=np.linalg.inv(Gmat)
        np.savetxt('Rmat'+str(i)+'.csv', Rmat, delimiter=",")
        T=np.dot(Ginv,Pvec)
        #np.savetxt("Gmat.csv",Gmat,delimiter=',')
        if np.allclose(T,Told,rtol=0.001):
            break

    Temp_dict=dict(zip(Node_name,T.tolist()))
    return Ginv,Pvec_dict,Temp_dict
```

F

Thermal Model for Forced Cooling

```
"""This file creates a 14 node lumped parameter thermal network for a surface mounted
    permanent magnet machine """
"""The code can be adapted for other machines as well as more nodes"""
from re import L
from numpy import pi, r_
from numpy import zeros
from numpy import log as ln
from numpy import sqrt
from numpy import tanh
from numpy import sinh
from pyleecan.Classes.SlotW22 import SlotW22
from pyleecan.Classes.SlotM11 import SlotM11

from Copper_loss_MOO  import Copper_Loss_MOO
shaft_radiator_thickness=2.5e-3
frame_thickness=2.5e-3
lam_contact=1100
slot_insulation_thickness=0.1e-3
lam_insulation=0.6# lipo pg 341/324
lam_stator_core_contact=0.6
import numpy as np
def yoke_resistance_calculator(r1,r2,L,lam):
    """determines radial thermal resistance for cylindrical yoke for T network LPTN """
    #r1 is outer radius
    #r2 is inner radius
    #L is effective stack length
    #lam is the radial thermal conductivity
    # returns the three resistances for the T network
    R1=(1-(2*r2**2*ln(r1/r2)/(r1**2-r2**2))/(4*pi*lam*L))
    R2=((2*r1**2*ln(r1/r2)/(r1**2-r2**2)-1)/(4*pi*lam*L))
    R3=((4*r1**2*r2**2*ln(r1/r2)/(r1**2-r2**2)-r1**2-r2**2)/(8*pi*lam*L)/(r1**2-r2**2))
    return R1,R2,R3

def annular_resistance_calculator(r1,r2,L,lam,alphap):
```

```
    """determines radial thermal resistance for cylindrical yoke for T network LPTN """
    #r1 is outer radius
    #r2 is inner radius
    #L is effective stack length
    #lam is the radial thermal conductivity
    #alphap is the magnet angle times the number of pole pairs
    #thermal resistance of a cylidrical shell is known multiply by 2pi and divide by magnet
        angle
    #multiply 2p to account for parallel paths
    # returns the three resistances for the T network
    R1=(1-(2*r2**2*ln(r1/r2)/(r1**2-r2**2))/(4*alphap*lam*L))
    R2=((2*r1**2*ln(r1/r2)/(r1**2-r2**2)-1)/(4*alphap*lam*L))
    R3=((4*r1**2*r2**2*ln(r1/r2)/(r1**2-r2**2)-r1**2-r2**2)/(8*alphap*lam*L)/(r1**2-r2**2))
    return R1,R2,R3

def yoke_axial_resistance_calculator(r1,r2,L,lam):
    """determines axial thermal resistance for cylindrical yoke for T network LPTN """
    #r1 is outer radius
    #r2 is inner radius
    #L is effective stack length
    #lam is the radial thermal conductivity
    # returns the three resistances for the T network
    if lam==0:
        return 0,0,0
    R1=L/(2*pi*lam*(r1**2-r2**2))
    R2=L/(2*pi*lam*(r1**2-r2**2))
    R3=L/(6*pi*lam*(r1**2-r2**2))
    return R1,R2,R3

def annular_axial_resistance_calculator(r1,r2,L,lam,alphap):
    """determines axial thermal resistance for cylindrical yoke for T network LPTN """
    #r1 is outer radius
    #r2 is inner radius
    #L is effective stack length
    #lam is the radial thermal conductivity
    #alphap is the magnet angle times the number of pole pairs
    # returns the three resistances for the T network
    if lam==0:
        return 0,0,0
    R1=L/(2*alphap*lam*(r1**2-r2**2))
    R2=L/(2*alphap*lam*(r1**2-r2**2))
    R3=L/(6*alphap*lam*(r1**2-r2**2))
    return R1,R2,R3
def rectangular_resistance_calculator(h,A,lam):
    #h is the height of the block
    #A is the area of cross section perpendicular to heat flow
    #lam is the thermal conductivity in direction of calculation
    #returns three resistance for the T network
    R1=h/(2*A*lam)
    R3=-R1/3
    return R1,R1,R3
def shaft_calculator(Rext,L,lam):
    R1=1/(2*pi*Rext*L*lam_contact)
    R2=ln(Rext/(Rext-shaft_radiator_thickness))/(2*pi*lam*L)
```

```
    R=R1+R2
    return R
def frame_calculator(Rint,L,lam):
    R1=1/(2*pi*Rint*L*lam_contact)
    R2=ln((Rint+frame_thickness)/Rint)/(2*pi*lam*L)
    R=R1+R2
    return R
def flat_plate_forced(air_v,L,area,T_surface,T_ambient=20):
    k=0.025
    mu=1.6e-5
    rho=1.225
    cp=1.005e3
    Re=rho*air_v*L/mu
    Pr=cp*mu/k
    if Re<5e5:
        Nu=0.664*Re**0.5*Pr**0.33
    else:
        Nu=(0.037*Re**0.8-871)*Pr**0.33
    h=Nu*k/L
    R=1/area/h
    return R

def frame_axial(D,area,T_surface,T_air):
    theta=T_surface-T_air
    g=9.8

    beta=3400e-6
    cp=1.005e3
    k=0.025
    rho=1.225
    mu=1.6e-5
    ethr=0.85
    sigma_sb=5.67e-8
    Pr=cp*mu/k
    Gr=beta*g*theta*rho**2*D**3/mu**2
    Ra=Gr*Pr
    if 1e4<=Ra<1e9:
        Nu=0.59*Ra**0.25
    elif 1e9<=Ra<1e12:
        Nu=0.129*Ra**0.33
    print(Ra)
    alpha_c=Nu*k/D
    T1=T_surface+273
    T2=273+T_air
    alpha_r=ethr*sigma_sb*(T1**4-T2**4)/(T1-T2)
    R=1/(alpha_c+alpha_r)/area
    return R
def frame_ambient(D,area,T_surface,T_air):
    g=9.8
    theta=T_surface-T_air
    beta=3400e-6
    cp=1.005e3
    k=0.025
    rho=1.225
```

```
    mu=1.6e-5
    ethr=0.85
    sigma_sb=5.67e-8
    Pr=cp*mu/k
    Gr=beta*g*theta*rho**2*D**3/mu**2
    Ra=Gr*Pr
    if 1e4<=Ra<1e9:
        Nu=0.525*Ra**0.25
    elif 1e9<=Ra<1e12:
        Nu=0.129*Ra**0.33
    alpha_c=Nu*k/D
    T1=T_surface+273
    T2=273+T_air
    alpha_r=ethr*sigma_sb*((T1**4)-(T2**4))/(T1-T2)
    R=1/(alpha_c+alpha_r)/area
    return R
def airgap_calculator(rm,g,omega,area,T_surface,T_air):
    #rm is average of stator and rotor bore radius
    #g is the air gap length
    #omega is the angular velocity of the rotor
    rho=1.225#mass density of air todo implement thermal effect on properties based on
        increase in air temp
    mu=1.81e-5#dynamic viscosity of air
    lam=0.025 #thermal conductivity of air
    ethr=0.85
    sigma_sb=5.67e-8
    Ta=rho**2*omega**2*rm*g**3/mu**2
    x=(2*rm-2.304*g)/(2*rm-g)
    Fg=(pi**4*x)/(1697*(0.0056+0.0571*x**2)*(1-g/(2*rm)))
    Tam=Ta/Fg
    if Tam<1700:
        Nu=2
    elif Tam<1e4:
        Nu=0.128*Tam**0.367
    else:
        Nu=0.409*Tam**0.241
    dh=g*(8/3)**0.5
    alpha_c=Nu*lam/dh
    T1=T_surface+273
    T2=273+T_air
    alpha_r=ethr*sigma_sb*((T1**4)-(T2**4))/(T1-T2)
    R=1/(alpha_c+alpha_r)/area
    return R
def stator_calculator(out,Rmat,T,Tambient):
    stator=out.simu.machine.stator
    machine=out.simu.machine
    r_in=stator.Rint
    r_out=stator.Rext
    slot_height=stator.slot.comp_height_active()
    N0=out.simu.input.OP.N0
    RRext=machine.rotor.Rext+frame_thickness
    if stator.is_internal:
        r_yoke_in=r_in+slot_height
    else:
```

```
    r_yoke_in=r_out-slot_height
L=stator.L1*stator.Kf1
lam_r_iron=stator.mat_type.HT.lambda_x
[R1,R2,R3]=yoke_resistance_calculator(r_out,r_yoke_in,L,lam_r_iron)
Rmat[6,7]+=R1 if stator.is_internal else R2
Rmat[0,6]+=R2 if stator.is_internal else R1
Rmat[1,6]+=R3
slot_base_width=stator.slot.W2*(r_yoke_in)
Ns=stator.slot.Zs
teeth_area=(2*pi*r_yoke_in/Ns-slot_base_width)*L
[R1,R2,R3]=rectangular_resistance_calculator(slot_height,teeth_area,lam_r_iron)
Rmat[6,7]+=R1/Ns
Rmat[3,7]+=R2/Ns
Rmat[2,7]+=R3/Ns
rm=(stator.get_Rbo()+machine.rotor.get_Rbo())/2
g=abs(stator.get_Rbo()-machine.rotor.get_Rbo())
stator_teeth_area=(2*pi-Ns*stator.slot.comp_angle_opening())*stator.get_Rbo()*L
omega=out.simu.input.OP.N0/30*pi
Rmat[3,7]+=airgap_calculator(rm,g,omega,stator_teeth_area,T_surface=T[5],T_air=20)


area_copper=stator.comp_fill_factor()*stator.slot.comp_surface_active()
lam_r_copper=stator.winding.conductor.ins_mat.HT.lambda_x
lam_a_copper=stator.winding.conductor.cond_mat.HT.lambda_z

#winding calculation
R1=(+slot_insulation_thickness/((2*slot_height+slot_base_width)*lam_insulation)+1/((2*
    slot_height+slot_base_width)*lam_stator_core_contact))
Rw=slot_height/(Ns*lam_a_copper*area_copper)
Gw=1/R1
x=sqrt(Rw*Gw)
y=sqrt(Rw/Gw)
R2=y*tanh(x/2)
R3=R1*(x/sinh(x)-1)
Rmat[2,4]+=R1
Rmat[4,5]+=R2
Rmat[4,5]+=R3
lam_a_iron=stator.mat_type.HT.lambda_z
#[R1,R2,R3]=yoke_axial_resistance_calculator(r_out,r_in,L,lam_a_iron)
#Rmat[1,1]=R3+R1*R2/(R1+R2)
l_end=stator.winding.comp_length_endwinding()*2
r_sew1=r_yoke_in+slot_insulation_thickness
r_sew2=r_sew1+slot_height
r_sew=(r_sew1+r_sew2)*0.5
R3=1.5*l_end/(Ns*area_copper*lam_a_copper)
R2=1.5/(4*pi**2*(r_sew1+r_sew2)*lam_r_copper)
Rmat[4,5]+=R3
Rmat[5,5]+=R2/2
end_winding_area=pi**2*r_sew*(r_sew1+r_sew2)
R1=airgap_calculator(rm,g,omega,end_winding_area,T_surface=T[5],T_air=20)
R2=flat_plate_forced(air_v=N0/30*pi*RRext,L=l_end,area=end_winding_area,T_surface=T[5])
Rmat[5,5]+=R1*R2/(R1+R2)/2
Rshaftin=r_in-shaft_radiator_thickness
#Rmat[4,4]+=frame_axial_enclosed(R*2,pi*R**2,T_surface=T[7],T_air=20)
```

```
    Rmat[0,0]+=flat_plate_forced(air_v=N0/30*pi*RRext*2,L=L,area=2*pi*Rshaftin*L,T_surface=T
        [0])
def rotor_calculator(out,Rmat,T,Tambient):
    machine=out.simu.machine
    rotor=machine.rotor
    r_in=rotor.Rint
    r_out=rotor.Rext
    L=rotor.L1*rotor.Kf1
    N0=out.simu.input.OP.N0
    RRext=machine.rotor.Rext+frame_thickness
    lam_r_iron=rotor.mat_type.HT.lambda_x
    [R1,R2,R3]=yoke_resistance_calculator(r_out,r_in,L,lam_r_iron)
    Rmat[8,11]+=R1 if rotor.is_internal else R2
    Rmat[11,12]+=R2 if rotor.is_internal else R1
    Rmat[9,11]+=R3
    #magnet calculation
    Hmag=rotor.slot.Hmag
    lam_r_mag=rotor.magnet.mat_type.HT.lambda_x
    alphap=rotor.slot.Wmag*rotor.get_pole_pair_number()
    [R1,R2,R3]=annular_resistance_calculator(r_out+Hmag,r_out,L,lam_r_mag,alphap)
    Rmat[11,12]+=R1 if rotor.is_internal else R2
    Rmat[3,12]+=R2 if rotor.is_internal else R1
    Rmat[10,12]+=R3
    lam_a_iron=rotor.mat_type.HT.lambda_z
    lam_a_mag=rotor.magnet.mat_type.HT.lambda_z
    #[R1,R2,R3]=yoke_axial_resistance_calculator(r_out,r_in,L,lam_a_iron)
    #Rmat[9,9]=R3+R1*R2/(R1+R2)
    #[R1,R2,R3]=annular_axial_resistance_calculator(r_out+Hmag,r_out,L,lam_a_mag,alphap)
    #Rmat[10,10]=R3+R1*R2/(R1+R2)
    #Rmat[8,8]=shaft_axial() if rotor.is_internal else frame_axial()
    rm=(machine.stator.get_Rbo()+rotor.get_Rbo())/2
    g=abs(machine.stator.get_Rbo()-rotor.get_Rbo())
    rotor_area=2*alphap*L*rotor.get_Rbo()
    omega=out.simu.input.OP.N0/30*pi
    Rmat[3,12]+=airgap_calculator(rm,g,omega,rotor_area,T_surface=T[10],T_air=20)
    Rmat[8,11]+=frame_calculator(r_out,L,lam_r_iron)
    Rmat[8,8]+=flat_plate_forced(air_v=N0/30*pi*RRext,L=2*pi*RRext,area=2*pi*RRext*machine.
        rotor.L1,T_surface=T[7])
def ThermalLPTN_forced(out,Tambient,iron_loss_dict,I_rms,kf=0.4):
    machine=out.simu.machine
    N0=out.simu.input.OP.N0
    P=p=machine.get_pole_pair_number()
    Nstator=8
    Nrotor=6
    Ntotal=Nstator+Nrotor-1
    Rmat=zeros([Ntotal,Ntotal])
    Gmat=zeros([Ntotal,Ntotal])
    Pvec=zeros([1,Ntotal])
    eye=np.identity(Ntotal)
    tamb=np.ones([1,Ntotal])*Tambient
    T=np.array([65,65,65,20,110,110,20,20,65,65,65,20,20],dtype=float)
    Node_name=["shaft","stator yoke","stator teeth","air gap","winding","end winding","node 6
        ","node 7","frame",
            "rotor yoke","magnets","node 11","node 12"]
```

```
    if type(machine.rotor.slot)!=SlotM11:
        #print("rotor geometry not supported")
        print(machine.rotor.slot)
        return Rmat,Pvec
    if type(machine.stator.slot)!=SlotW22:
        #print("stator geometry not supported")
        print(machine.stator.slot)
        return Rmat,Pvec
    I_rms=I_rms/machine.stator.winding.Ntcoil
    for i in range(100):
        Rmat=zeros([Ntotal,Ntotal])
        Gmat=zeros([Ntotal,Ntotal])
        Told=T.copy()
        Lwinding=Copper_Loss_MOO(stator=machine.stator,f=N0/60*P,T=T[4],I=I_rms,kf=kf)
        Lrotoryoke=iron_loss_dict['rotor']
        Lmag=0
        Lstatoryoke=iron_loss_dict['stator yoke']
        Lteeth=iron_loss_dict['stator teeth']
        Pvec=np.array([0,Lstatoryoke,Lteeth,0,Lwinding,0,0,0,0,Lrotoryoke,Lmag,0,0]).reshape
            (-1,1)
        Pvec_dict=dict(zip(Node_name,Pvec.tolist()))

        stator_calculator(out,Rmat,T,Tambient)
        rotor_calculator(out,Rmat,T,Tambient)

        Gmat=np.where(Rmat==0,0,1./Rmat)
        amb=np.diagonal(Gmat)
        Gmat=Gmat-eye*amb

        Gmat=Gmat+np.transpose(Gmat)


        Gsum=np.sum(Gmat,axis=1)

        Gmat=Gmat*-1


        Gmat=Gmat+eye*(amb+Gsum)
        #np.savetxt('Gmat.csv', Gmat, delimiter=",")
        ambtamb=amb*tamb

        Pvec=Pvec+np.transpose(ambtamb)
        Ginv=np.linalg.inv(Gmat)
        np.savetxt('Rmat'+str(i)+'.csv', Rmat, delimiter=",")
        T=np.dot(Ginv,Pvec)
        #np.savetxt("Gmat.csv",Gmat,delimiter=',')
        if np.allclose(T,Told,rtol=0.001):
            break

    Temp_dict=dict(zip(Node_name,T.tolist()))
    return Ginv,Pvec_dict,Temp_dict
```

G

MOO Routine

```
import numpy as np
from pymoo.core.problem import Problem
from pymoo.optimize import minimize
from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.core.callback import Callback
from matplotlib import pyplot as plt
import logging


logging.disable(logging.WARNING)
from os.path import join
from pyleecan.definitions import DATA_DIR
from pyleecan.Functions.load import load
from pyleecan.Classes.Material import Material
from pyleecan.Classes.MatMagnetics import MatMagnetics
from pyleecan.Classes.MatHT import MatHT
import numpy as np


Steel1 = load(join(DATA_DIR, "Material", "M19.json"))
with open("VaCoFe.csv") as file_name:
        BH_Steel1= np.loadtxt(file_name, delimiter=",")
Steel1=Material(name='Steel1');
Steel1.mag=MatMagnetics(Wlam=0.35e-3,BH_curve=BH_Steel1);
Steel1.HT=MatHT(lambda_x=30,lambda_y=30,lambda_z=0.6)
Steel1.struct.rho=8120
Magnet3 = load(join(DATA_DIR, "Material", "Magnet3.json"))
Copper1 = load(join(DATA_DIR, "Material", "Copper1.json"))
Insulator1 = load(join(DATA_DIR, "Material", "Insulator1.json"))
Copper1.elec.alpha=0.00393
Magnet3 = load(join(DATA_DIR, "Material", "Magnet3.json"))
Magnet3.HT=MatHT(lambda_x=9,lambda_y=9,lambda_z=9)
coeff=[5E-3,2,1.00E-03,1.5,4.00E-05,2]
discrete_variables={'iron':Steel1,'copper':Copper1,'insulator':Insulator1,'magnet':Magnet3,
```

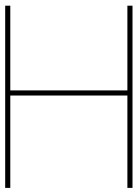
```
                    'Ns':24,'P':28,'g':.001,'N0':4300,'torque':3.45,'loss_coeff':coeff,'k_f'
                        :0.4}

def weight_calculator(machine):
    stator_weight=machine.stator.comp_masses()
    rotor_weight=machine.rotor.comp_masses()
    total_weight=stator_weight['Mtot']+rotor_weight['Mtot']
    return total_weight

from pyleecan.Functions.load import load
from pyleecan.Classes.MachineSIPMSM import MachineSIPMSM
from pyleecan.Classes.LamSlotWind import LamSlotWind
from pyleecan.Classes.LamSlot import LamSlot
from pyleecan.Classes.SlotW22 import SlotW22
from pyleecan.Classes.Winding import Winding
from pyleecan.Classes.CondType12 import CondType12
from pyleecan.Classes.EndWindingCirc import EndWindingCirc

from pyleecan.Classes.LamSlotMag import LamSlotMag
from pyleecan.Classes.SlotM11 import SlotM11
from pyleecan.Classes.Magnet import Magnet
from numpy import pi,sqrt
def machine_generator(design_variable,discrete_variables):
    RRext=design_variable[0]
    RSint=design_variable[1]
    Rratio=design_variable[2]
    L=design_variable[3]
    lm_per=design_variable[4]
    alpha_m=design_variable[5]
    h0_per=design_variable[6]#teeth
    h2_per=design_variable[7]#slot
    w0_per=design_variable[8]#opening angle
    w1_per=design_variable[9]#teeth angle
    Steel=discrete_variables['iron']
    Copper=discrete_variables['copper']
    Insulator=discrete_variables['insulator']
    magnet=discrete_variables['magnet']
    Ns=int(discrete_variables['Ns'])
    P=int(discrete_variables['P']/2)
    g=discrete_variables['g']
    kf=discrete_variables['k_f']
    end_wind=EndWindingCirc()
    stator_hieght=Rratio*(RRext-RSint)
    RSext=RSint+stator_hieght
    stator = LamSlotWind(Rint=RSint,Rext=RSext,L1=L,Nrvd=0,Kf1=0.93,is_internal=True,
        is_stator=True,mat_type=Steel)
    RRint=RSext+g
    lm=lm_per*(RRext-RRint)
    slot_pitch=2*pi/Ns

    stator.slot = SlotW22(Zs=Ns,H0=h0_per*stator_hieght,H2=stator_hieght*h2_per,W0=slot_pitch
        *w0_per,W2=slot_pitch*(1-w1_per))
    stator.winding = Winding(qs=3,Ntcoil=1,Npcp=1
                        ,p=P,type_connection=0,
```

```
                        Lewout=0.0001,Nlayer=2,Nslot_shift_wind=0,is_aper_a=True,end_winding
                            =end_wind)
    stator.winding.conductor = CondType12(Nwppc=1, Wwire=4e-3,Wins_cond=1.6e-3,cond_mat=
        Copper,
                                        ins_mat=Insulator)
    copper_area=stator.slot.comp_surface_active()*kf/2
    stator.winding.conductor.Wwire=sqrt(copper_area)
    pole_pitch=pi/P
    rotor_slot=SlotM11(W0=pole_pitch, H0=0, Hmag=lm, Wmag=alpha_m*pole_pitch, Zs=P*2)
    mag=Magnet(mat_type=magnet, type_magnetization=0, Lmag=L)
    #define rotor lamination and size
    rotor = LamSlotMag(magnet=mag,slot=rotor_slot,L1=L,Nrvd=0,Wrvd=0,Kf1=0.93,is_internal=
        False,Rint=RRint+lm,
                    Rext=RRext,is_stator=False,mat_type=Steel)
    machine=MachineSIPMSM(name="machine1",rotor=rotor,stator=stator,type_machine=1,shaft=None
        )
    return machine


from pyleecan.Classes.Magnet import Magnet
from pyleecan.Classes.Simu1 import Simu1
from pyleecan.Classes.InputCurrent import InputCurrent
from pyleecan.Classes.MagFEMM import MagFEMM
from pyleecan.Classes.OPdq import OPdq
from pyleecan.Classes.Output import Output
import logging
from numpy import sqrt
from Iron_Loss_PMSM import iron_loss_static
from numpy import cos, sin, pi,sqrt
from Copper_loss_MOO import Copper_Loss_MOO
from windage_loss import windage_calculator
from thermal_model_forced_cooling import ThermalLPTN_forced

def no_load_sim(machine,discrete_variables):
    simu_no_current=Simu1(name="machine",machine=machine)
    Na_tot=2048
    N0=discrete_variables['N0']
    torque=discrete_variables['torque']
    p=machine.get_pole_pair_number()
    qs=3
    felec = p * N0 /60 # [Hz]
    omega=felec*2*pi
    simu_no_current.mag=MagFEMM(type_BH_rotor=0,type_BH_stator=0,is_get_meshsolution=True,
        is_periodicity_a=True,Kmesh_fineness=0.5)
    simu_no_current.input=InputCurrent(Na_tot=Na_tot,angle_rotor_initial=0,Nt_tot=1)

    simu_no_current.input.OP=OPdq(Id_ref=0,Iq_ref=0,N0=N0)
    output_no_current=simu_no_current.run()
    simu_current=simu_no_current.copy()
    Phi_mag_abc=output_no_current.mag.Phi_wind["Stator-0"].values[0]
    Iq=torque*2/3/p/Phi_mag_abc[0]/sqrt(2)
    E=Phi_mag_abc[0]*omega
    for i in range(3):
        simu_current.input.OP.Iq_ref=Iq
```

```
        output_current=simu_current.run()
        T=output_current.mag.Tem_av

        delT=torque-T
        Iq+=delT*2/3/p/Phi_mag_abc[0]/sqrt(2)
        if abs(delT/torque)<0.05:
            break
    iron_loss=iron_loss_static(output_current,discrete_variables['loss_coeff'])
    Ginv,Pvec_dict,Temp_dict=ThermalLPTN_forced(output_current,0,iron_loss,Iq)
    Twind=Temp_dict['winding'][0]
    total_loss=iron_loss["stator"]+iron_loss["rotor"]+windage_calculator(output_current)+
        Pvec_dict["winding"][0]
    return Iq,total_loss,T,Twind


from Iron_Loss_PMSM import iron_loss_transient
from numpy import cos, sin, pi,sqrt
from Copper_Loss_PMSM import Copper_Loss_PMSM
from windage_loss import windage_calculator
from thermal_model_forced_cooling import ThermalLPTN_forced
def time_step_sim(machine,discrete_variables,I0_rms):
    simu=Simu1(name="Nuna_test",machine=machine)
    N0=discrete_variables['N0']
    torque=discrete_variables['torque']
    Na_tot=2048
    p=machine.get_pole_pair_number()
    qs=3
    simu.mag=MagFEMM(type_BH_rotor=0,type_BH_stator=0,is_get_meshsolution=True,
        is_periodicity_a=True,Kmesh_fineness=0.5,nb_worker=6)
    simu.input=InputCurrent(Na_tot=Na_tot)
    simu.input.OP=OPdq(N0=N0,Iq_ref=0,Id_ref=0)
    felec = p * N0 /60 # [Hz]
    omega=felec*2*pi
    theta=np.array([0,pi/12,pi/6,pi/4,pi/2,3*pi/4])
    t=theta/omega
    simu.input.time=t
    simu.input.angle=np.linspace(start=0,stop=2*pi,num=2,endpoint=False)


    rot_dir = simu.machine.stator.comp_mmf_dir()
    Phi0 =3*pi/2  # Maximum Torque Per Amp

    Ia = (
        I0_rms
        * sqrt(2)
        * cos(2 * pi * felec * t + 0 * rot_dir * 2 * pi / qs + Phi0)
    )
    Ib = (
        I0_rms
        * sqrt(2)
        * cos(2 * pi * felec * t + 1 * rot_dir * 2 * pi / qs + Phi0)
    )
    Ic = (
        I0_rms
```

```
        * sqrt(2)
        * cos(2 * pi * felec * t + 2 * rot_dir * 2 * pi / qs + Phi0)
    )
    simu.input.Is = np.array([Ia, Ib, Ic]).transpose()
    out=Output(simu=simu)
    simu.run()
    Tem=out.mag.Tem.values
    Tem_av=(Tem[0]+Tem[1]+Tem[2]+Tem[3])/4
    iron_loss=iron_loss_transient(out,discrete_variables['loss_coeff'])
    windage_loss=windage_calculator(out)
    total_loss=iron_loss['stator']+iron_loss['rotor']+windage_loss
    #machine.stator.winding.conductor.Nwppc=10
    Ginv,Pvec_dict,Temp_dict=ThermalLPTN_forced(out,0,iron_loss,I0_rms)
    total_loss+=Pvec_dict["winding"][0]
    Twind=Temp_dict['winding'][0]
    return total_loss,Tem_av,Twind

def single_evaluation(design_variable,discrete_variables):
    machine=machine_generator(design_variable,discrete_variables)
    N0=discrete_variables['N0']
    torque=discrete_variables['torque']
    weight=weight_calculator(machine)
    I,total_loss,T_em,Twind=no_load_sim(machine,discrete_variables)
    if Twind<300:
        total_loss,T_em,Twind=time_step_sim(machine,discrete_variables,I)
    return weight,total_loss,T_em,Twind


class MOO_FEMM(Problem):
    def _evaluate(self,designs,out,*args,**kwargs):
        res = []
        cons=[]
        result_set=[]
        i=0

        discrete_variables={'iron':Steel1,'copper':Copper1,'insulator':Insulator1,'magnet':
            Magnet3,
                    'Ns':24,'P':28,'g':.001,'N0':4300,'torque':3.45,'loss_coeff':coeff,'k_f'
                        :0.4,'voltage':22.5}
        for design in designs:
            np.savetxt("var.csv",design,delimiter=",")
            loss,weight,T_av,Twind=single_evaluation(design,discrete_variables)
            res.append([loss,weight])
            delT=abs(1-T_av/discrete_variables['torque'])-0.05
            delTwind=Twind-200
            cons.append([delTwind,delT])
            result_set.append([loss,weight,T_av,Twind])
        out['F']=np.array(res)
        out['G']=np.array(cons)
        Result_set=np.array(result_set)
        designs_with_result_set=np.concatenate((designs,Result_set),axis=1)
        np.save('eval.npy', designs_with_result_set)

class Mycallback(Callback):
```

```
    def notify(self,algorithm):
        plt.close('all')
        name="n_gen="+str(algorithm.n_gen)
        plt.figure()
        F=algorithm.pop.get("F")
        X=algorithm.pop.get("X")
        plt.scatter(F[:,0],F[:,1])
        np.save(name, F)
        plt.title(name)
        plt.xlabel("weight")
        plt.ylabel("losses")
        plt.savefig(name)
        plt.close()
        design_with_results_set=np.load("eval.npy")
        n_gen_array=np.ones_like(design_with_results_set[:,0])
        n_gen_array*=algorithm.n_gen
        Designs_with_result_set=np.hstack((n_gen_array[:, np.newaxis],design_with_results_set
            ))
        with open('Designs_with_results.csv','a') as csvfile:
            np.savetxt(csvfile,Designs_with_result_set,delimiter=",")

var_limit=np.genfromtxt('variable_limits.csv',delimiter=',',dtype=float,skip_header=True)
problem=MOO_FEMM(n_var=10,n_obj=2,xl=var_limit[0],xu=var_limit[1],n_constr=2)

algorithm=NSGA2(pop_size=100)

stop_criteria=('n_gen',25)

results=minimize(problem=problem,algorithm=algorithm,termination=stop_criteria,save_history=
    True,verbose=True,callback=Mycallback())

np.save("checkpoint", algorithm)
np.save("results",results)
```



Datasheet Silicon Steel Core

35WW270

B(T)	50Hz	60Hz	100Hz	400Hz	1000Hz	B(T)	50Hz	60Hz
	P(W/kg)	P(W/kg)	P(W/kg)	P(W/kg)	P(W/kg)		P(W/kg)	P(W/kg)
0.40	0.219	0.272	0.493	3.297	12.854	1.60	2.697	3.366
0.50	0.319	0.397	0.722	4.867	19.226	1.61	2.719	3.411
0.60	0.431	0.537	0.980	6.698	26.985	1.62	2.764	3.455
0.70	0.555	0.692	1.270	8.841	36.231	1.63	2.786	3.500
0.80	0.689	0.860	1.589	11.338	46.973	1.64	2.853	3.589
0.90	0.836	1.044	1.944	14.113	59.897	1.65	2.875	3.612
1.00	0.995	1.246	2.345	17.275	75.247	1.66	2.920	3.633
1.10	1.175	1.472	2.786	21.004	93.241	1.67	2.942	3.678
1.20	1.383	1.724	3.299	25.190	114.363	1.68	2.964	3.700
1.30	1.630	2.033	3.900	29.999	138.499	1.69	3.009	3.790
1.40	1.948	2.441	4.764	35.885	166.755	1.70	3.053	3.812
1.50	2.320	2.916	5.458	42.974	201.881	1.71	3.075	3.857
1.51	2.363	2.963	5.585			1.72	3.098	3.880
1.52	2.398	3.010	5.618			1.73	3.143	3.947
1.53	2.452	3.054	5.682			1.74	3.165	3.991
1.54	2.496	3.099	5.763			1.75	3.210	4.013
1.55	2.519	3.166	5.819			1.76	3.244	4.079
1.56	2.563	3.188				1.77	3.276	4.146
1.57	2.586	3.256				1.78	3.315	4.178
1.58	2.630	3.299				1.79	3.343	4.213
1.59	2.653	3.322				1.80	3.392	4.279

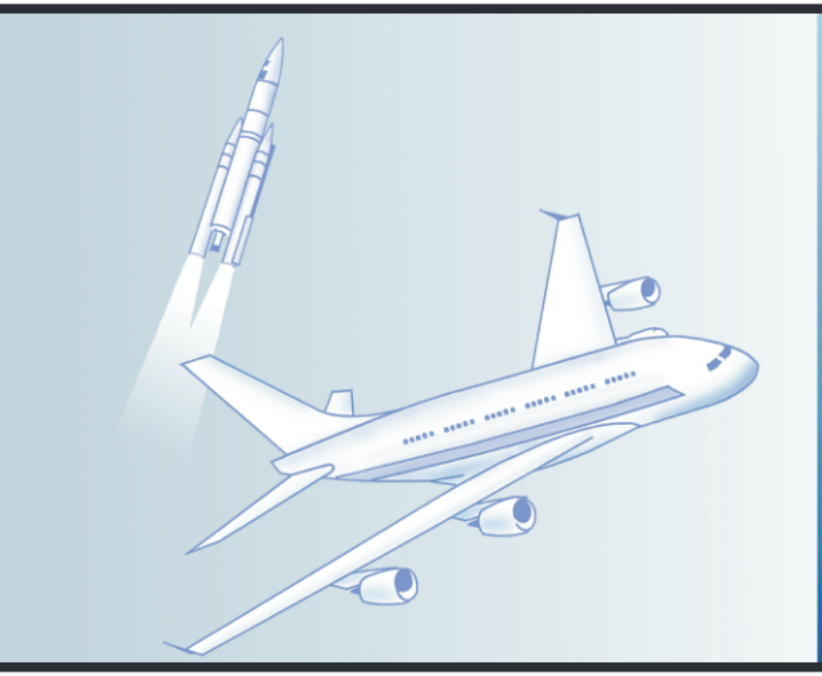
Hmax	Bmax	Hmax	Bmax	Hmax	Bmax	Hmax	Bmax	Hmax	Bmax
[A/m]	[T]	[A/m]	[T]	[A/m]	[T]	[A/m]	[T]	[A/m]	[T]
20	0.083	80	0.828	402	1.375	1508	1.511	7037	1.715
30	0.175	90	0.904	502	1.402	2011	1.541	8041	1.739
40	0.319	100	0.967	603	1.422	3017	1.588	9047	1.760
50	0.478	151	1.157	703	1.437	4020	1.628	10051	1.780
60	0.619	201	1.249	803	1.450	5025	1.661		
70	0.734	302	1.333	1004	1.471	6031	1.689		

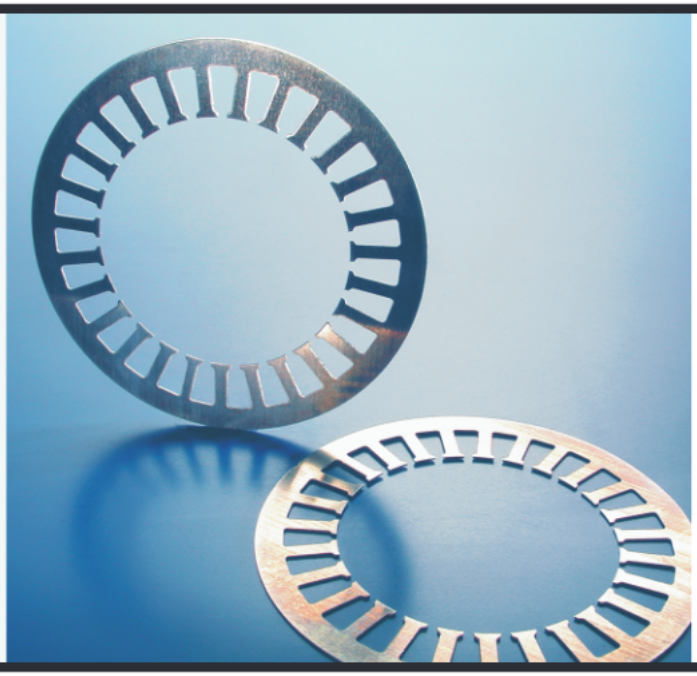
Datasheet Vanadium Cobalt Iron Core

VACUUMSCHMELZE

Soft Magnetic Cobalt-Iron-Alloys

VACOFLUX 48 • VACOFLUX 50
VACODUR 50 • VACOFLUX 17





Edition 2001

PHT-004

Soft Magnetic Cobalt-Iron-Alloys

VACOFLUX 48 • VACOFLUX 50
VACODUR 50 • VACOFLUX 17

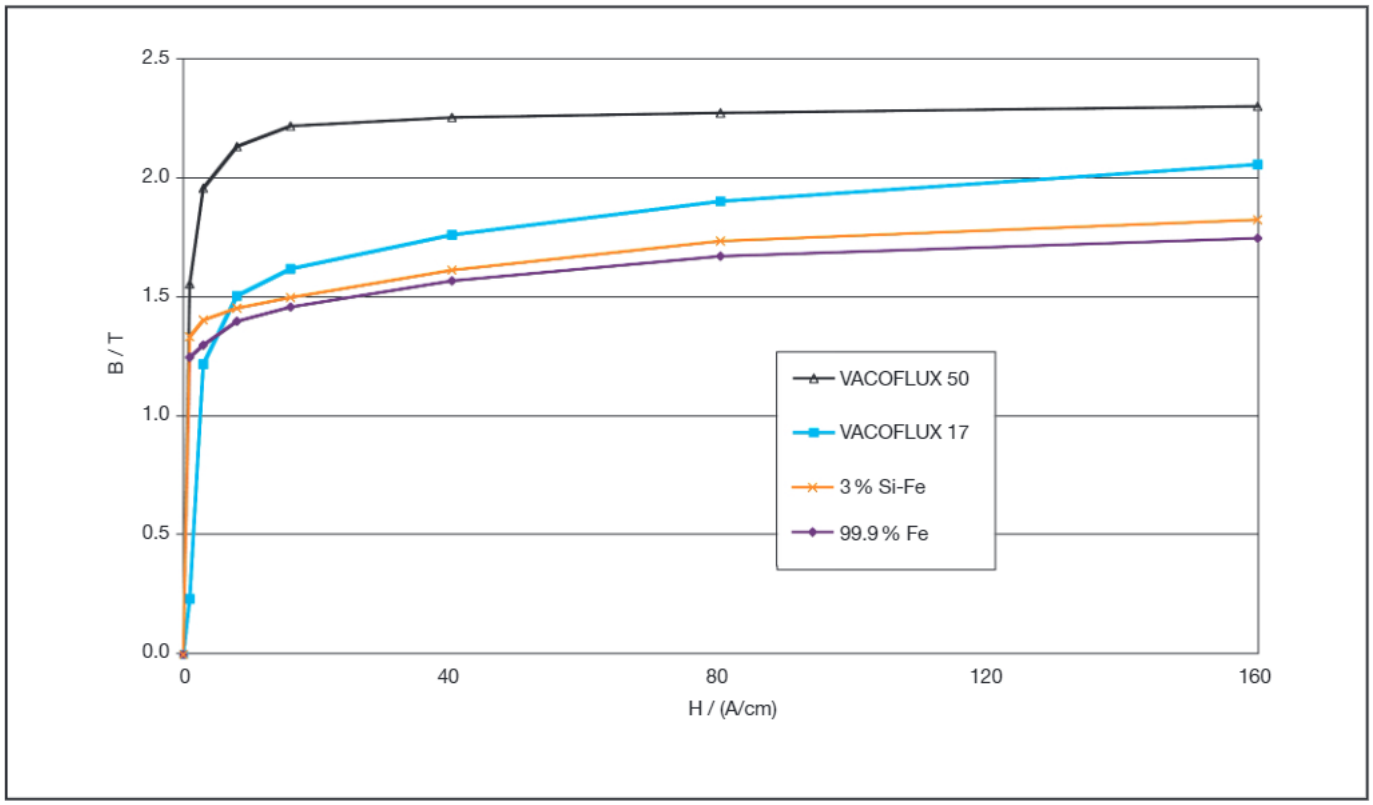


Fig. 1: Typical virgin B(H)-curves of different high saturation soft magnetic alloys for comparison
(50 % CoFe ≅ VACOFLUX 50; 17 % CoFe ≅ VACOFLUX 17;
99,98 % Fe ≅ VACOFER S1; 3 % SiFe ≅ TRAFOPERM N3)

1. Introduction

VACUUMSCHMELZE is one of the world leaders in the production of materials with special magnetic and physical properties. The product range covers soft magnetic products as well as permanent magnets and inductive components.

Our strength is the development and production of innovative materials. Especially our know-how in the field of magnetism combined with the awareness for the customer‘s requirements and visions are considerable benefits VACUUMSCHMELZE can offer. It is our aim to decisively support our partners with products providing a maximum of competitive advantages and making new and downstream solutions feasible.

VACUUMSCHMELZE‘s product range of soft magnetic materials comprises pure sintered Iron, NiFe, SiFe and CoFe alloys as well as amorphous and nanocrystalline alloys. Our CoFe alloys VACOFLUX® 48 and VACOFLUX 50 show the highest saturation magnetization and do surpass all known soft magnetic materials. Various properties and hysteresis loops can be obtained by using special compositions and selecting the optimum production procedure.

VACODUR® 50 is a further development of VACOFLUX 50 with respect to higher strength and ductility. We are also able to meet the demand for high saturation with the new developed VACOFLUX 17. For processing this alloy additionally offers remarkable features like extrusion moulding and a reduced cobalt content of only 17 % resulting in lower costs.

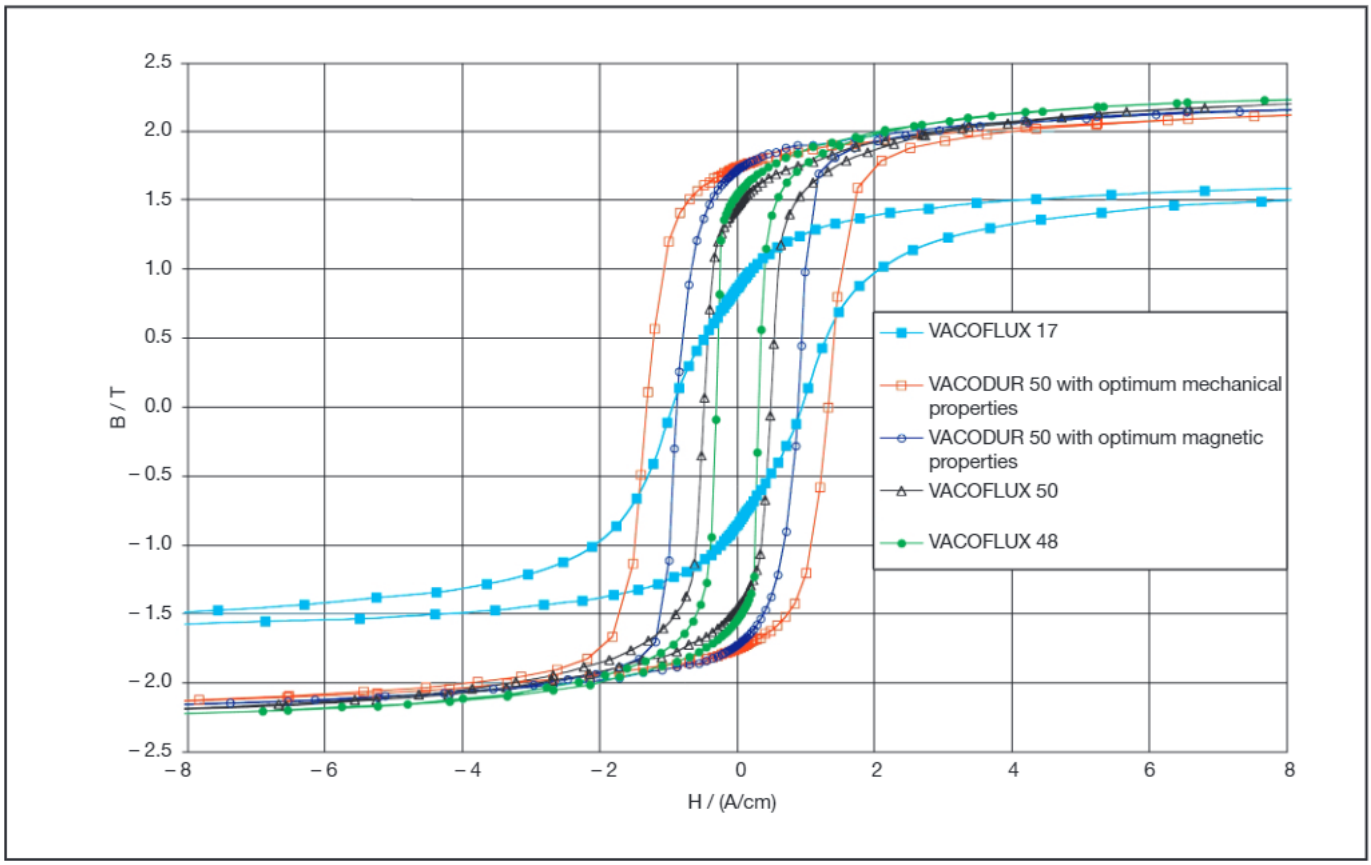


Fig. 2: Static hysteresis loops of our CoFe alloys. These are typical loops for strips with a thickness of 0.35 mm.

2. Application

	Remarks	Applications
VACOFLUX 48 (material according IEC 404-8-6 F11)	Material with a round hysteresis loop and a coercivity of $H_c \leq 0.4$ A/cm and very low losses, especially at flux densities between 1.8 T and 2.2 T.	Special transformers with low losses at very high flux densities, high performance motors.
VACOFLUX 50 (material according IEC 404-8-6 F11)	Material with a round hysteresis loop and a coercivity of $H_c \leq 0{,}8$ A/cm (up to 2 mm thickness).	Very high flux density pole-shoes, electro-magnets with maximum lifting force, magnetic lenses, needle printers, relays, motors and actuators with high torques and forces.
VACODUR 50 (material according IEC 404-8-6 F1)	A further development of VACOFLUX 50 with respect to improved mechanical properties, especially higher strength and ductility.	Alternators and generators with high rotation speed. Applications are comparable to VACOFLUX 50 with special requirements on mechanical properties.
VACOFLUX 17	Alloy with low Co-content and high saturation induction, i.e., very high magnetic force.	Devices and actuators for automotive industry and turned as well as extruded parts.

3. Magnetic Properties after Final Annealing*)

	Static Values (strip material, thickness 0.35 mm)		Static Values (solide material)		J_s (T)	Curie-Temperature (°C)	λ_s
	H_c (A/cm)	μ_{max}	H_c (A/cm)	μ_{max}			
VACOFLUX 48	≤ 0.4	15000	–	–	2.35	950	$70 \cdot 10^{-6}$
VACOFLUX 50	≤ 0.8	13000	≤ 2.4	4500	2.35	950	$70 \cdot 10^{-6}$
VACODUR 50 (with optimum magnetic properties)	≤ 1.6	10000	–	–	2.3	950	$70 \cdot 10^{-6}$
VACODUR 50 (with optimum mechanical properties)	≤ 2.0	7000	–	–	2.3	950	$70 \cdot 10^{-6}$
VACOFLUX 17	≤ 2.0	3500	≤ 2.0	2500	2.22	920	$25 \cdot 10^{-6}$

H_c = Coercivity, μ_4 = Permeability at 4 mA/cm, μ_{max} = Maximum Permeability, B_s = Saturation Polarisation, λ_s = Saturation Magnetostriction
*) Typical values for strip material

3.1 Static Values for 0.35 mm Stamped Samples*)

	B at 3 A/cm (T)	B at 8 A/cm (T)	B at 16 A/cm (T)	B at 40 A/cm (T)	B at 80 A/cm (T)	B at 160 A/cm (T)
VACOFLUX 48	2.05	2.15	2.25	2.27	2.3	–
VACOFLUX 50	1.9	2.1	2.2	2.25	2.27	2.3
VACODUR 50 (with optimum magnetic properties)	1.80	2.05	2.15	2.20	2.28	–
VACODUR 50 (with optimum mechanical properties)	1.70	2.00	2.1	2.18	2.25	–
VACOFLUX 17	1.2	1.5	1.6	1.75	1.9	2.05

B = Induction
*) Typical values

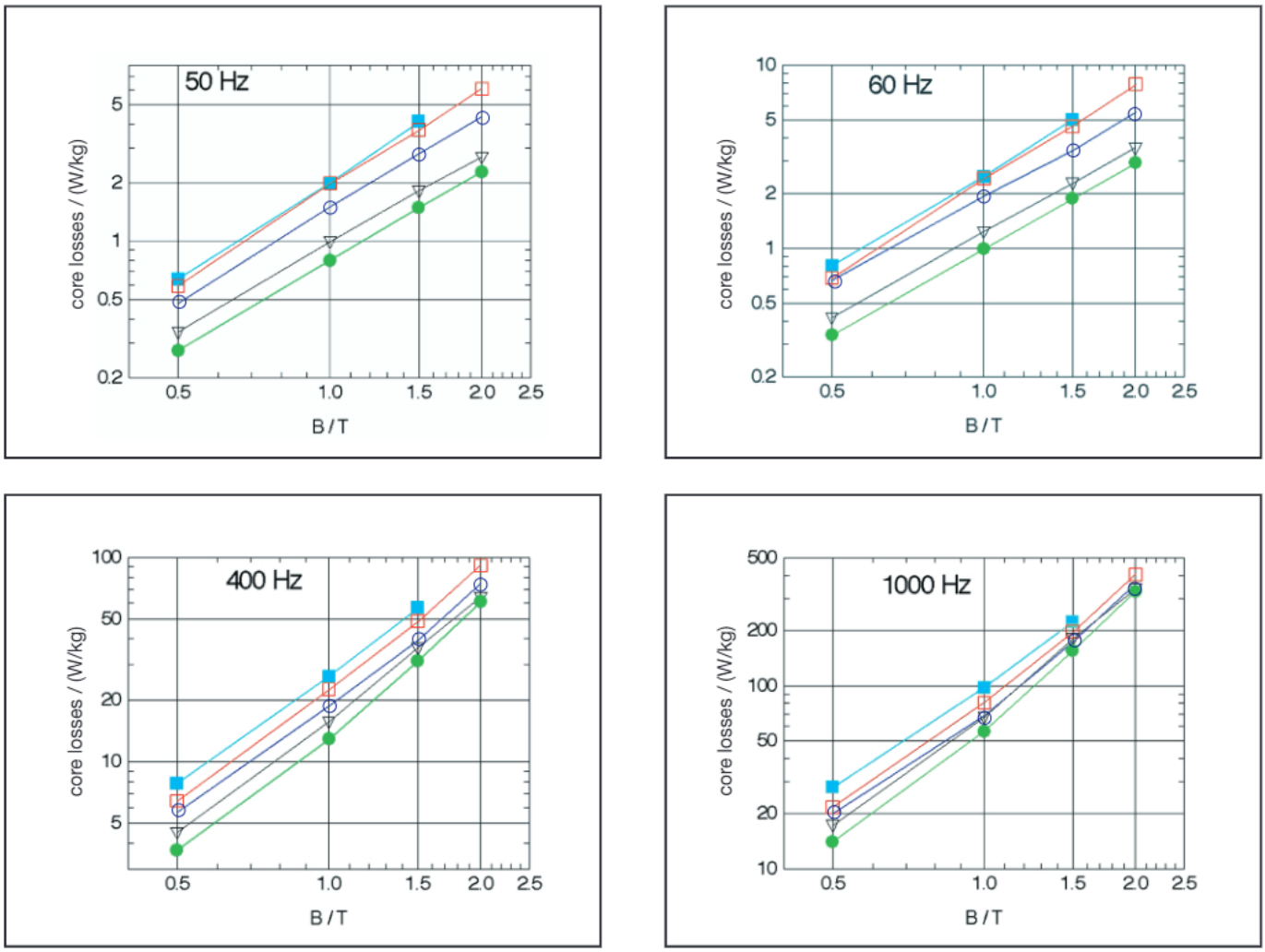


Fig. 3: Typical core losses of strips with a thickness of 0,35 mm at different frequencies.
(■ = VACOFLUX 17, □ = VACODUR 50 with optimum mechanical properties, ○ = VACODUR 50 with optimum magnetic properties, ▽ = VACOFLUX 50, ● = VACOFLUX 48)

4. Physical Properties*)

	Electrical Resistivity (after final annealing) (Ω mm^2/m)	Coefficient of Thermal Expansion (20 . . . 200°C) (10^{-6}/K)	Density (g/cm^3)
VACOFLUX 48	0.44	9.5	8.12
VACOFLUX 50	0.44	9.5	8.12
VACODUR 50 (wit optimum magnetic properties)	0.43	10.2	8.12
VACODUR 50 (with optimum mechanical properties)	0.42	10.2	8.12
VACOFLUX 17	0.39	10.8	7.94

*) Typical values

5. Final Annealing

	Temperature (°C)	Time of Annealing (h)	Atmosphere	Rate of Cooling (K/h)	Cooling until*) (°C)
VACOFLUX 48	880	10	dry hydrogen	~100	200
VACOFLUX 50	820	4-10	dry hydrogen	~100	200
VACODUR 50 (with optimum magnetic properties)	820	2-5	dry hydrogen	~100	200
VACODUR 50 (with optimum mechanical properties)	750	2-5	dry hydrogen	~100	200
VACOFLUX 17	850	10	dry hydrogen	~100	200

*) At lower temperature any cooling rate in any atmosphere is possible

6. Mechanical Properties after Final Annealing*)

	$Rp_{0,2}$ (N/mm^2)	R_m (N/mm^2)	Young's-Modulus (kN/mm^2)	Elongation until Fracture	Hardness HV
VACOFLUX 48	200	220	200	2 %	180
VACOFLUX 50[1])	250	350	210	3 %	190
VACODUR 50 (with optimum magnetic properties)	390	620	250	6 %	210
VACODUR 50 (with optimum mechanical properties)	450	720	250	6 %	230
VACOFLUX 17[1])	250	450	200	32 %	140

$Rp_{0,2}$ = Yield strength, R_m = Tensile strength
[1]) strip
*) Typical values

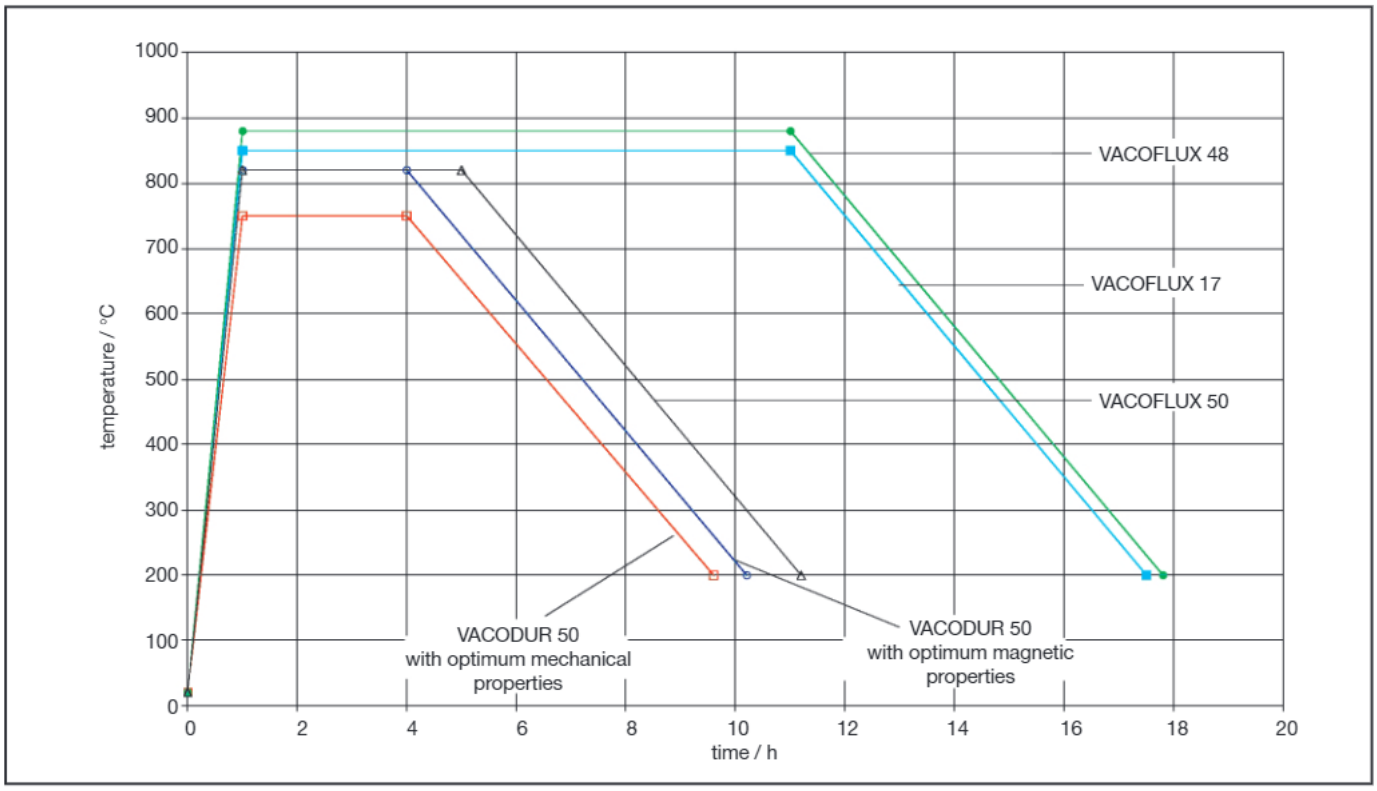


Fig. 4: Temperature profile of the magnetic final annealing.

7. Forms of Supply

	Semifinished Products		Finished Parts			
	Strips	Solid-profile-material, rods, wires	Strip-wound cores	Core laminations, stamped parts	Laminated packages, EK-cores	Solid and shaped parts
VACOFLUX 48	•	–	•	•	•	–
VACOFLUX 50	•	•	•	•	•	•
VACODUR 50	•	–	–	•	•	–
VACOFLUX 17	•	•	–	•	•	•

• available
– not available

Product Survey

Semi-Finished Products and Parts

Metallic Semi-Finished Products
Soft magnetic alloys
Magnetically semi-hard alloys
Ductile permanent magnets
Thermobimetals
Spring alloys
Glass/ceramic-to-metal sealing alloys

Parts
Stamped/bent parts
Laminations
Magnetic shielding

Superconductors

Cores and Components

Magnetic Cores
Tape-wound cores made of crystalline, amorphous and nano-crystalline alloys

Inductive Components
for xDSL, ISDN and switched-mode power supplies, for current detection and for driving power semiconductors

Rare-Earth Permanent Magnets

Magnets on Sm-Co and Nd-Fe-B Base

Polymer Bonded Magnets

Magnet Assemblies

VACUUMSCHMELZE GMBH & CO. KG

VACUUMSCHMELZE

Advanced Materials – The Key to Progress

P.O.B. 22 53
D-63412 Hanau, Germany
☏ (**49) 6181/38-0
Fax (**49) 6181/38-2065
Internet: http://www.vacuumschmelze.com
E-Mail: info@vacuumschmelze.com

052001

Published by VACUUMSCHMELZE GMBH & CO. KG, Hanau