


## Article

# Development of a Transient Synchronization Analysis Tool for Line-Start PM Motors

Phillip Schommarz and Rong-Jie Wang \* 

Department of Electrical &amp; Electronic Engineering, Stellenbosch University, Stellenbosch 7600, South Africa

\* Correspondence: [rwang@sun.ac.za](mailto:rwang@sun.ac.za)

**Abstract:** With more stringent IEC energy efficiency standards, electrical machine industry increasingly focuses on new motor technologies. Amongst others, the line-start permanent magnet synchronous machine (LSPMSM) is considered as an attractive alternative to induction machine, especially for low power and fixed-speed applications. However, the design of LSPMSMs is rather complex as both steady-state and transient synchronization performances need to be considered. The synchronization capability determination of a LSPMSM design usually relies on time-consuming transient finite-element simulations, which is impractical for use in an iterative design optimization process. This paper compares and evaluates various existing analytical synchronization analysis methods in an attempt to identify most suitable equations and methods for fast synchronization analysis. Using the selected methods, a software tool is developed that can seamlessly work with ANSYS Electronics Desktop to perform rapid transient synchronization analysis. Given its ability to quickly determine the critical inertia factor of a LSPMSM design, the software tool is further adapted for use in a highly iterative, multi-objective design optimization procedure. It shows that the developed software tool can be successfully used in the design of LSPMSMs.

**Keywords:** analytical modeling; finite element method; line-start motor; permanent magnet machine; software development; synchronization; transient performance



**Citation:** Schommarz, P.D.; Wang, R.-J. Development of a Transient Synchronization Analysis Tool for Line-Start PM Motors. *Energies* **2022**, *15*, 9206. <https://doi.org/10.3390/en15239206>

Academic Editor: Adolfo Dannier

Received: 8 November 2022

Accepted: 30 November 2022

Published: 5 December 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Industrial application of electrical motors consumes between 30% and 40% of generated electrical energy worldwide [1]. With the advent of the new industrial revolution brought forth by the development of e-mobility and renewable energy systems, the use of electrical motors will continue to grow at a fast pace. However, hand-in-hand with this growing demand for electrical motors comes the demand for improvement in their energy efficiency. With more stringent IEC efficiency standards, electrical motor manufacturers increasingly focus on new motor technologies as induction motors (IMs) are approaching their efficiency limits. This is especially true for low power general purpose IMs that drive fixed-speed loads such as fans, pumps and conveyors.

The line-start permanent magnet synchronous motor (LSPMSM) is an attractive alternative as it promises both higher efficiency and power factor than an IM [2]. While the efficiency of super-premium (IE4) 2.2 kW IMs has been observed to lie between 86 and 89% [3], LSPMSMs of the same power rating carry the potential to breach the 90% efficiency barrier. However, the design of LSPMSMs is rather complicated as its rotor structure contains both permanent magnets and cage windings, both of which can be configured in numerous ways. Furthermore, LSPMSMs have two distinct regions of operation, namely asynchronous operation during start-up, and synchronous operation during steady-state. The trade-off between starting/synchronization capability and higher efficiency and power factor is a key challenge for LSPMSMs [4].

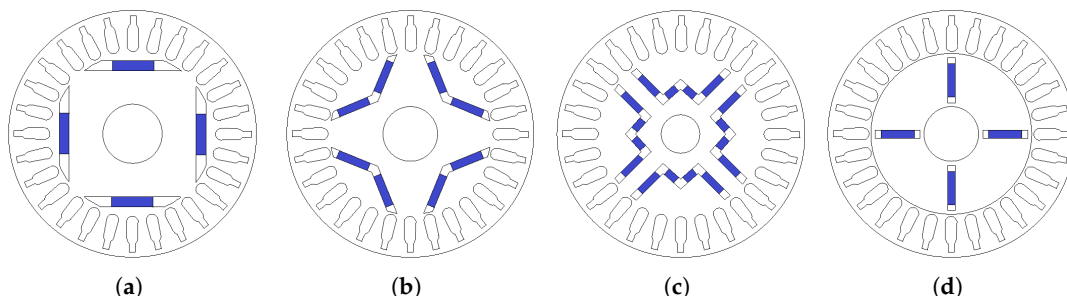
Determining whether a LSPMSM candidate design synchronizes usually requires transient finite-element (FE) analysis, which is computationally expensive and not suited

for use in any iterative design optimization procedure [5–7]. As an alternative, various fast analytical approaches for the synchronization analysis of LSPMSMs have been proposed such as energy methods [8,9], time-domain method [9], semi-numerical method [10], multi-damping-circuit model [11], reluctance network model [12], and sizing equation based mathematical model [13], which promise to provide at least a good estimate of a design’s synchronization capability. The work reported in [14] was probably the first attempt of incorporating both analytical steady-state and synchronization analysis into the multi-objective design optimization process of LSPMSMs. However, the optimization was conducted using the Taguchi method. Apart from that, these analytical approaches are only captured in literature with hardly any practical implementation in LSPMSM designs. Since both transient and steady-state operations need to be considered in the design of LSPMSMs, there is clearly a need to develop a transient synchronization analysis tool that can be integrated into mainstream electromagnetic simulation software.

This paper presents the development of a time-efficient transient synchronization analysis tool for use with ANSYS Electronics Desktop software environment, which determines the critical inertia factor ( $x_{cr}$ ), a figure of merit for the synchronization capability for a LSPMSM design. The remainder of the paper is organized as follows: In Section 2, the electromagnetic torque characteristics of the LSPMSM during transient synchronization process is first described, followed by the a discussion of recent development of analytical synchronization determination methods. In Section 3, different variants of synchronization torque equations and analytical synchronization methods are compared and evaluated in order to identify most suitable equations and methods for the software tool development. The detailed steps of software design and implementation is given in Section 4. The application example of the developed software tool is demonstrated in Section 5. Relevant conclusions are drawn in Section 6.

## 2. Synchronization Analysis of LSPMSMs

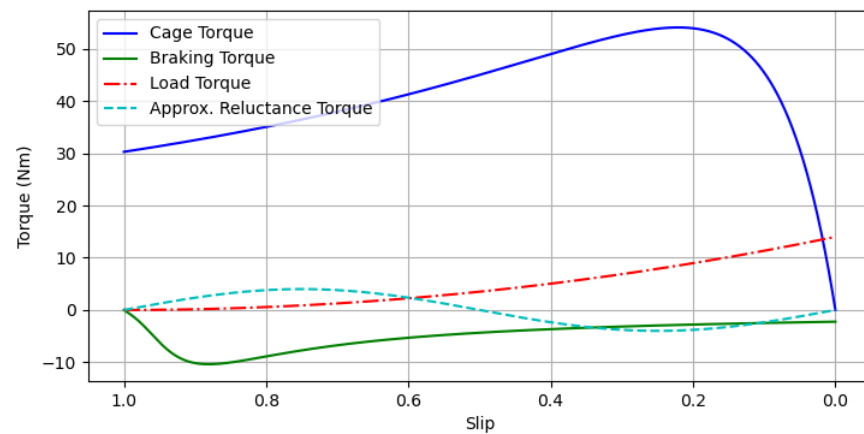
With both magnets and cage winding in the rotor of a LSPMSM, there is an inherent competition of space between them. This requires judicious arrangement of the cage winding and PM array within a rotor in order to realize a good design. As a result, a variety of rotor topologies have been created. Some common rotor designs can be observed in Figure 1. In comparison with IMs, the manufacturing of LSPMSMs are more costly, particularly for more complex topologies such as the V-type and W-type designs (Figure 1b,c). These additional costs arise from both the expensive PM materials and the added mechanical complexity of the interior ducts for hosting the PMs. However, a more prevalent problem across all rotor topologies is the synchronization capability—and related critical inertia—of a candidate design [15].



**Figure 1.** Various LSPMSM rotor layouts: (a) radial-type; (b) V-type; (c) W-type; (d) spoke-type.

### 2.1. Electromagnetic Torque Characteristics

To better understand the synchronization capabilities of a LSPMSM design, the electromagnetic torque components present at all stages of the synchronization process and their interactions need to be understood. A visual representation of these torque components during the asynchronous operation can be seen in Figure 2.



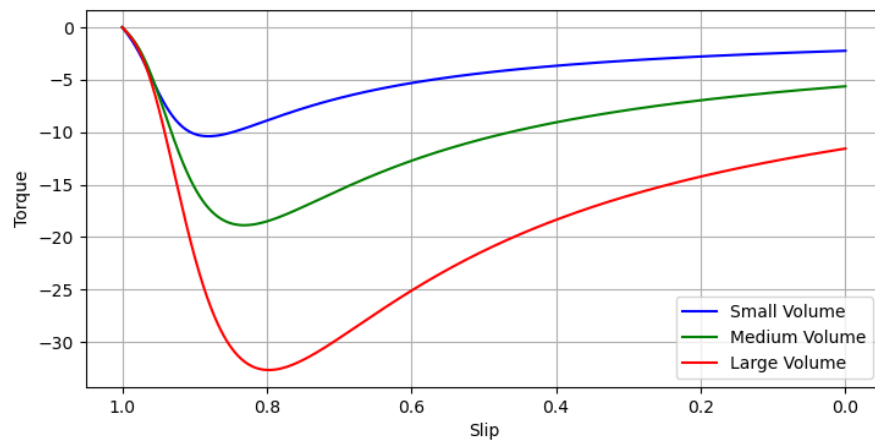
**Figure 2.** Various torque components during the asynchronous operation of a LSPMSM.

### 2.1.1. Braking Torque

The braking torque, sometimes also called the magnet torque, is the torque component contributed during start-up by the PMs. The magnets induce current in the stator at a frequency of  $(1-s)f$ , which negatively affect synchronization capabilities [16,17]. The analytical braking torque equation is given by (1).

$$T_b = -\frac{mpE_0^2R_1}{2\pi f} \cdot \frac{[R_1^2 + (1-s)^2X_q^2](1-s)}{[R_1^2 + (1-s)^2X_qX_d]^2} \quad (1)$$

where  $E_0$  is the induced back-EMF,  $R_1$  is the stator resistance, and  $X_d$  and  $X_q$  are the direct and quadrature axis synchronous reactances, respectively. While the volume of magnets impacts braking torque greatly (as shown in Figure 3), it also plays an important role to the power factor of a LSPMSM at steady-state.



**Figure 3.** Braking torque for various magnet volumes.

### 2.1.2. Cage and Reluctance Torque

In literature the cage torque of a LSPMSM is often treated the same as that of standard squirrel-cage IMs [8]. However, as explained in [17], owing to the non-symmetric magnetic circuit in a LSPMSM rotor, the rotating magnetic field created by rotor current may be decomposed into two counter-rotating fields. The positive sequence rotating field of the rotor winding results in an asynchronous torque while the negative sequence rotating field leads to a reluctance torque component, which hinders the synchronization process for  $s < 0.5$  and aids it for  $s > 0.5$  [17].

To avoid unnecessary complexity, the reluctance torque is sometimes lumped in together with the cage torque to form a combined equation [9]. Since different approaches

are used in literature to represent cage torque of LSPMSMs, it is necessary to compare and evaluate them.

### 2.1.3. Pulsating Torque

Apart from the cage torque and braking torque, which are essentially average torque components, there are other pulsating torque components that are resulted from the interaction of rotor and stator rotating fields of different speed and the slotting effects. These pulsating torque components are functions of load angle  $\delta$  and often included in the synchronization analysis [18,19]. However, there are also published work that neglects the pulsating component entirely [20]. As the oscillations contributed by the pulsating torques are known to impact synchronization capabilities [14], it seems unwise to neglect the effect of these torque components.

## 2.2. Analytical Synchronization Methods

The dynamic performance of LSPMSM is mainly concerned with the starting process of the machine. Owing to the existence of PMs in the rotor, the starting of the LSPMSM is more complicated than that of the IM. Furthermore, during the starting process the  $d$ - and  $q$ -axis armature reaction reactance vary with time due to the influence of magnetic saturation. Some researchers use a combination of analytical representation and finite element modelling for the synchronization analysis [10,21], which, while yielding more accurate solutions, is time consuming and not suitable for design optimization purposes.

Recently, there are some new developments of the analytical synchronization analysis methods, which are based on the classical approaches by Miller and Honsinger [18,22]. Amongst others, the energy-based analytical approaches by Rabbi et al. [8] and Chama et al. [9], and the time-domain method by Chama et al. [9] are the most representative. To facilitate further discussion in the paper, these methods are referred hereinafter as Energy method (Rabbi et al.), Energy method (Chama et al.), and Time-domain method (Chama et al.), respectively.

For all the three methods, the analytical representation of the instantaneous torque  $T_i$  is established from a design's parameters, which is a combination of the torque components described in Section 2.1 and represented by (2).

$$T_i(s, \delta) = T_p(\delta) + T_c(s) + T_b(s) - T_l(s) \quad (2)$$

where  $T_p$  refers to pulsating torque. Subsequently, the instantaneous torque  $T_i$  follows the equation of motion in the slip-load angle plane, captured in (3).

$$-\frac{J\omega_s^2}{p} \cdot s \frac{ds}{d\delta} = T_i(s, \delta) \quad (3)$$

Further treatment of (3) is where analytical approaches differ.

### 2.2.1. Energy Method (Rabbi et al.)

The treatment of (3) by Rabbi et al. [8] is focused around approximating the last pole slip of the instantaneous torque. Initially, slip is set to 0, with the aim of solving the equation  $T_i(0, \delta) = 0$ . For a majority of cases, this results in two solutions as shown in Figure 4, namely the load angle  $\delta'_s$  that is the load angle for which the motor reaches synchronous speed for the first time, and  $\delta_s$  that binds the domain in which the critical slip  $s_{cr}$  will be found. The critical slip is the local maximum situated between  $\delta'_s$  and  $\delta_s$ .

Then, using the value found for  $\delta'_s$ , the last pole slip is approximated as  $s = s_{cr} \sin \frac{1}{2}(\delta'_s - \delta)$  in order to find  $s_{cr}$ . Mathematically, this is conducted by solving  $T_i(s, \delta'_s - \pi) = 0$  as shown in Figure 5. With  $\delta'_s$  and  $s_{cr}$  acquired, the required kinetic energy  $E_{scr}$  to pull the motor into synchronization is calculated using (4).

$$E_{scr} = \int_{s_{cr}}^0 -\frac{1}{p} J\omega_s^2 s \, ds = \frac{1}{2p} J\omega_s^2 s_{cr}^2 \quad (4)$$

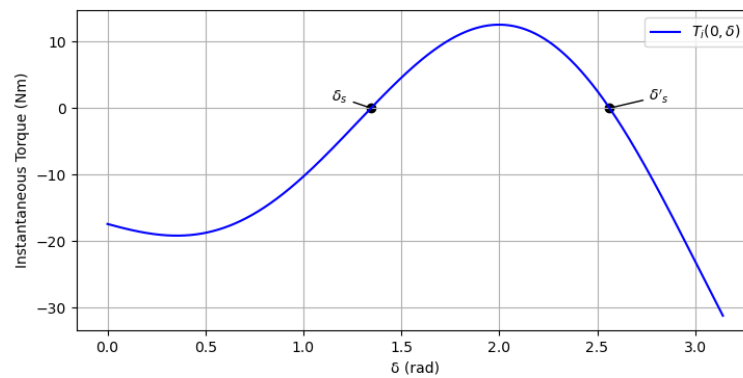


Figure 4. Finding  $\delta'_s$ .

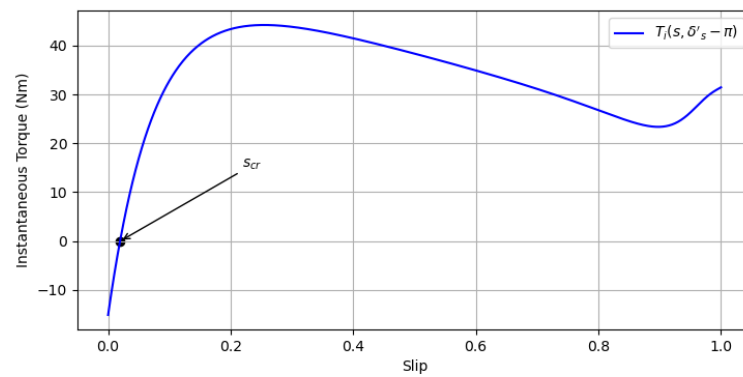


Figure 5. Finding the critical slip.

The approximated synchronization energy  $E_{syn}$ , or the energy actually provided by the instantaneous torque, is calculated using (5).

$$E_{syn} = \int_{\delta'_s - \pi}^{\delta'_s} T_i(s_{cr} \sin \frac{1}{2}(\delta'_s - \delta), \delta) d\delta \tag{5}$$

When the apparent rotor kinetic energy  $E_{syn}$  matches the required pull-in energy  $E_{scr}$ , the critical inertia  $J_{cr}$  is reached, which is the largest system inertia a motor design can successfully synchronize. Using (4) and (5),  $J_{cr}$  can be formulated as (6). Furthermore, by normalizing the critical inertia to the rotor inertia, the critical inertia factor  $x_{cr}$  can be defined, which can be very useful to quantify the synchronization capability in the design process.

$$J_{cr} = \frac{2pE_{syn}}{s_{cr}^2 \omega_s^2} \tag{6}$$

### 2.2.2. Energy Method (Chama et al.)

The energy-based method by Chama et al. [9] once again starts off by forming the equation of motion (3) from the instantaneous torque. Instead of relying on any approximations, the equation of motion is rewritten into the form seen in (7); an implicit, nonlinear ordinary differential equation (ODE).

$$\frac{ds}{d\delta} = -\frac{p}{J\omega_s^2 s} T_i(s, \delta) = f(s, \delta) \tag{7}$$

Instead of approximating the last pole slip, as proposed by Rabbi et al. [8], the energy-based method proceeds to solve the ODE using the implicit Runge–Kutta–Fehlberg method. The advantages of this ODE solver is its high order of convergence. However, the ODE can be solved by other numerical methods for ODEs as well, such as the backward differentiation formula (BDF) or the Gauss–Radau method. The choice for ODE method to be

used depends on the stiffness of the problem at hand, and thus further comparison will be performed in Section 4.2.2.

The ODE, when solved, establishes the slip as a function of load angle, an example of which can be seen in Figure 6. From this direct resolution of the ODE,  $\delta'_s$  and  $s_{cr}$  can be obtained.

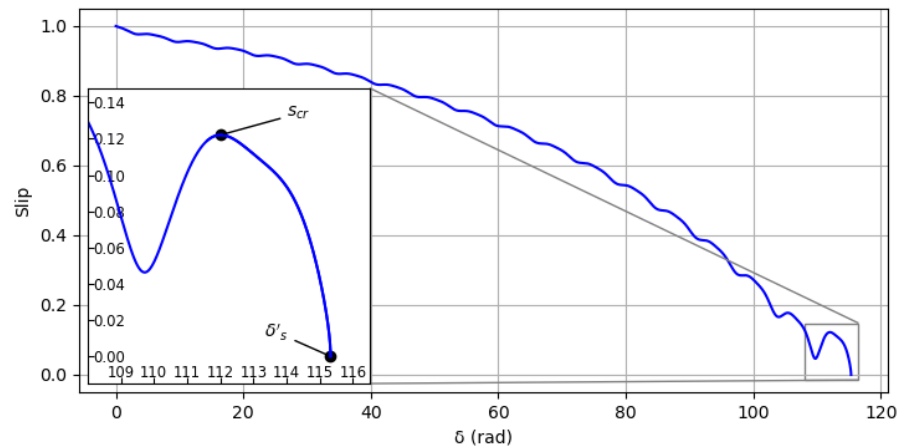


Figure 6. Slip as a function of  $\delta$ .

With these values readily available,  $E_{s_{cr}}$  and  $E_{s_{syn}}$  can be calculated and compared again. In the case of  $E_{s_{cr}}$ , the method used by Chama et al. is essentially the same as that of Rabbi et al., i.e., (4). Given the fact that the slip has been established as a function of load angle, the approximation present in (5) is not necessary anymore. Instead, (8) can be used to compute  $E_{s_{syn}}$ , where  $s_{cr} = s(\delta_{s_{cr}})$ .

$$E_{s_{syn}} = \int_{\delta_{s_{cr}}}^{\delta'_s} T_i(s(\delta), \delta) d\delta \tag{8}$$

Once again, when  $E_{s_{syn}} \geq E_{s_{cr}}$ , the candidate design synchronizes.

### 2.2.3. Time-Domain Method (Chama et al.)

The time domain model presented by Chama et al. [9] takes a slightly different approach. The equation of motion (3) is formulated into a transient variation taking the form of (9) and (10), where  $\omega_{rm}$  represents the motor speed in rad/s and  $\theta$  represents the rotor angle in radians.

$$J \frac{\partial \omega_{rm}}{\partial t} = T_i(s, \theta) \tag{9}$$

$$-\frac{1}{p} J \omega_s^2 s \frac{\partial s}{\partial \theta} = T_i(s, \theta) \tag{10}$$

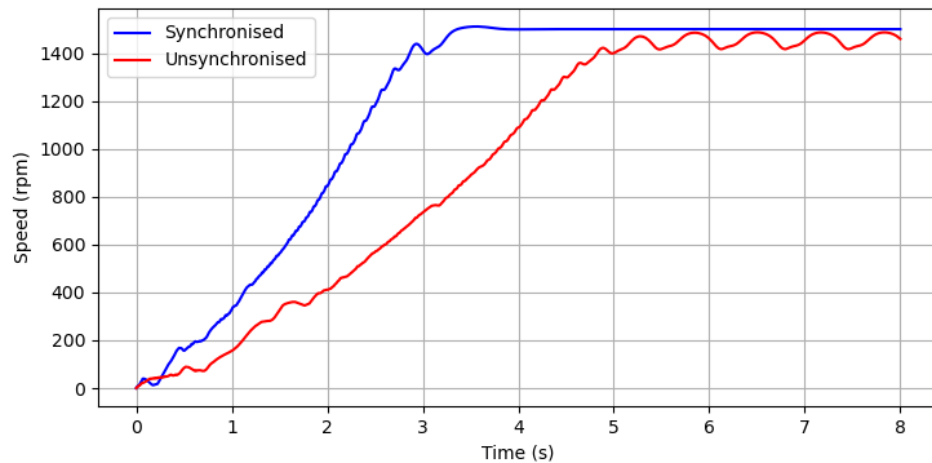
Given the relationship between motor speed and slip  $\omega_{rm} = \frac{\omega_s(1-s)}{p}$ , and expressing  $\frac{\partial s}{\partial \theta}$  as  $\frac{\partial s}{\partial t} \frac{\partial t}{\partial \theta}$ , (9) and (10) can be algebraically transformed into the initial value problem seen below. This system of equations can also be solved using any implicit ODE method over any desired period of time.

$$\frac{\partial s}{\partial t} = -\frac{p}{J \omega_s} T_i(s, \theta)$$

$$\frac{\partial \theta}{\partial t} = s \omega_s$$

Once the solution to the ODE has been computed, the motor's speed versus time graph can be extracted. Two cases can arise: The first case being when synchronization

occurs, resulting in the motor's speed settling around its rated speed  $n_{rated}$ . The second case arising when the motor fails to synchronize; the speed oscillating below rated speed. Both cases are visualized in Figure 7.



**Figure 7.** Speed vs. time graphs for a synchronized and un-synchronized motor.

To mathematically determine whether this synchronization process has occurred, the average speed  $\overline{n_{rm}}$  and gradient of the average speed  $\overline{n'_{rm}}$  of the tail end of the generated graphs can be calculated. If the following conditions hold, then synchronization occurs.

$$\begin{aligned} |\overline{n_{rm}} - n_{rated}| &\leq 10^{-2} \\ \overline{n'_{rm}} &\leq 10^{-2} \end{aligned}$$

### 3. Appraisal of Analytical Synchronization Equations and Methods

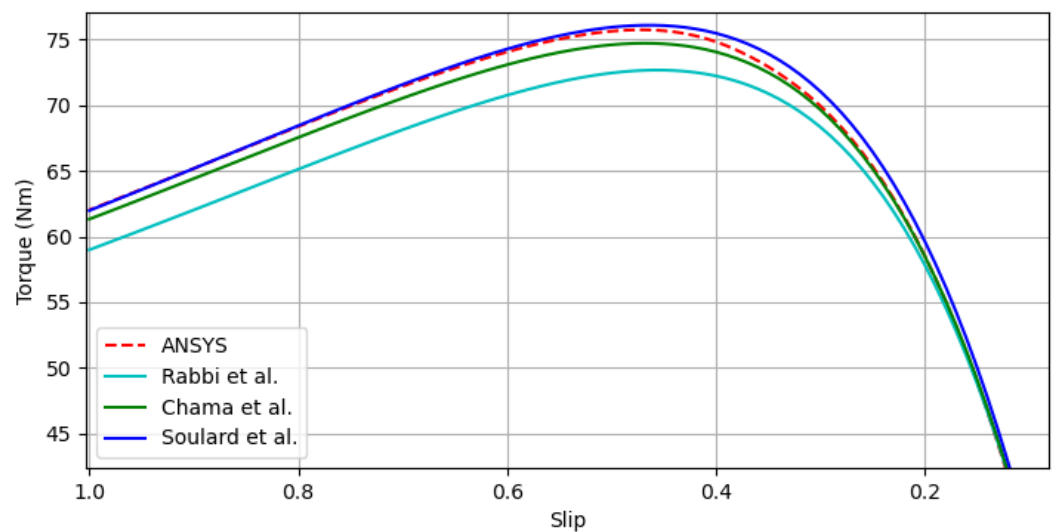
In order to identify most suitable analytical synchronization equations and methods to be implemented in the software tool, the analytical methods described in Section 2.2 are compared and evaluated in this section. To facilitate the subsequent comparison, the set of torque components will be standardized and individually examined. A number of LSPMSM candidate designs with different rotor topologies (radial, V-type, W-type and spoke-type) are created in ANSYS Electronics Desktop for test and verification purposes. The details of these designs can be found in Figure A1.

#### 3.1. Cage Torque Equations

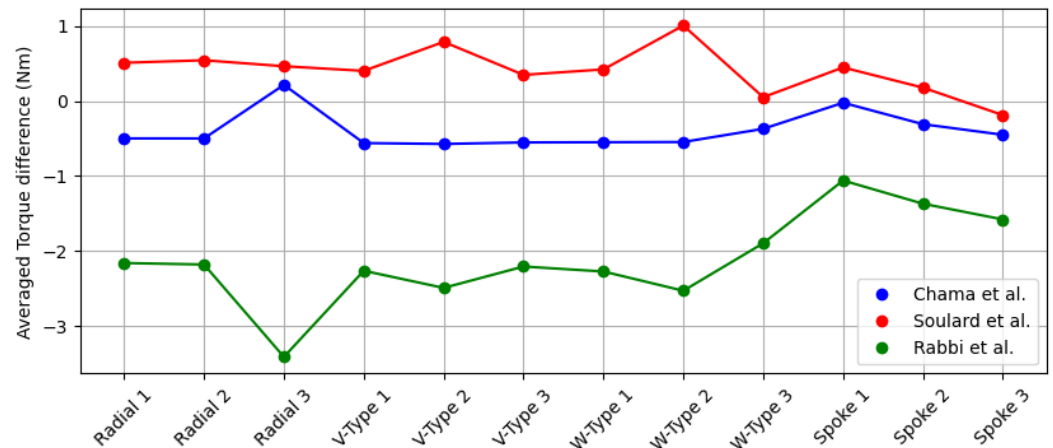
To determine the most appropriate cage torque equation for a general use case, ANSYS Electronics Desktop was used to generate the cage torque curves of the aforementioned LSPMSM designs. This provides the benchmark cage torque characteristics that various analytical formulae strive to replicate.

Three different equations selected for comparison are from Soulard et al. [19], Rabbi et al. [8], and Chama et al. [9]. The required machine parameters for the analytical cage torque curve are extracted from the ANSYS RMxprt and summarized in Table A1.

Figure 8 shows a cage torque comparison for a radial-type rotor design. To compare these cage torque curves more generally, each analytical cage torque curve was then subtracted from the ANSYS-generated benchmark cage torque curve, and the average of their discrepancy was calculated. This was conducted for each candidate machine design in the aforementioned test set. The average torque difference for the 12 test machines can be seen in Figure 9. A condensed box plot data of Figure 9 is shown in Figure 10, summarizing the medium, minimum, maximum, and interquartile range for each analytical cage torque equation.



**Figure 8.** Various (Rabbi et al. [8], Chama et al. [9], Soulard et al. [19]) analytical cage torque curves for the Radial 1 design.



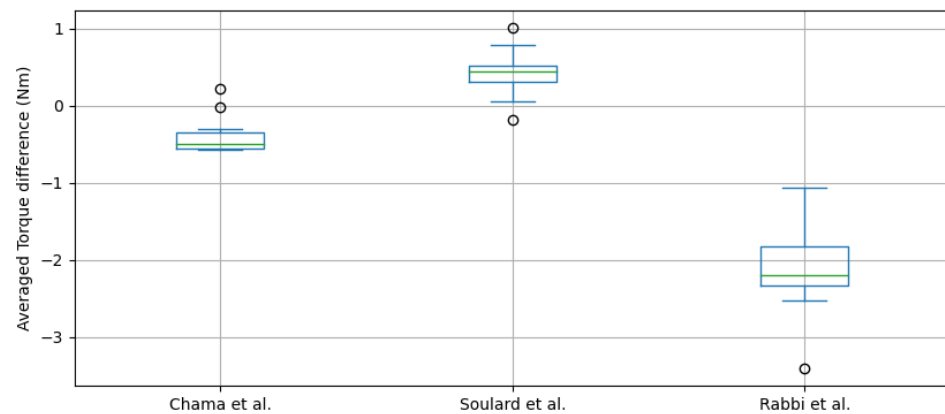
**Figure 9.** Average torque difference of various (Rabbi et al. [8], Chama et al. [9], Soulard et al. [19]) analytical  $T_c$ s from target  $T_c$  for each motor design.

It is evident that the cage torque equations presented by Chama et al. [9] and Soulard et al. [19] compare favorably with that of Rabbi et al.'s [8]. When comparing the equations by Chama et al. and Soulard et al., Figure 10 highlights that Chama et al.'s cage torque equation is much more consistent over a variety of designs, with only one outlier in the form of the Radial 3 design.

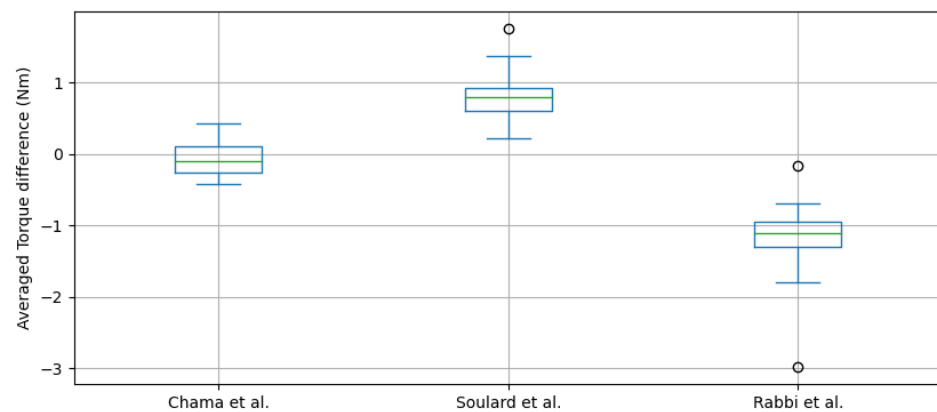
Furthermore, closer inspection of Figure 8 reveals that Chama et al.'s [9] analytical cage torque equation tends to approximate the target cage torque curve better in the region nearing  $s = 0$ ; closer to synchronization. This is confirmed by the box plot in Figure 11, which only takes the latter half of synchronization into account. Soulard et al.'s [19] equation approximates better near the region  $s = 1$ ; the start of the synchronization process.

For determining synchronization capabilities of LSPMSM designs, the region between asynchronous and synchronous operation is of utmost importance. Therefore, the analytical cage torque Equation (11) used by Chama et al. [9] appears to be a good option.





**Figure 10.** Box plot of average torque difference of various (Rabbi et al. [8], Chama et al. [9], Soulard et al. [19]) analytical  $T_c$  equations from target  $T_c$  for the design set.



**Figure 11.** Box plot of average torque difference of various (Rabbi et al. [8], Chama et al. [9], Soulard et al. [19]) analytical  $T_c$  equations from target  $T_c$  for the region  $0.5 > s \geq 0$  for the design set.

$$T_c = \frac{mp}{2\pi f} \cdot \frac{sR_2' V_{ph}^2}{(sR_1 + c_1 R_2')^2 + s^2(X_1 + c_1 X_2')^2} \quad (11)$$

where the factor  $c_1$  is given by  $c_1 = 1 + \frac{X_1}{X_m}$ .

### 3.2. Pulsating Torque

The direct comparison of various pulsating torque equations found across literature to a benchmark pulsating torque curve generated by a FE program such as ANSYS Electronics Desktop is unfortunately impossible, seeing as isolating this component is not feasible.

However, an alternative approach can be taken. Firstly, the instantaneous torque is created from the selected best suited equations, i.e., (1) for  $T_b$ , (11) for  $T_c$ , and (12) for  $T_l$  will be used to populated (13).

Since LSPMSMs are often used for constant speed operation such as in pumps, compressors, and fans [23], for the sake of simplicity, a general fan load torque curve will be used in this study, which takes the form seen in (12).

$$T_l = T_{rated}(1 - s)^2 \quad (12)$$

This leaves only  $T_p$  to be varied, where analytical pulsating torque equations from Soulard et al. [19], Rabbi et al. [8], and Tang [17] can be inserted for comparison.

$$T_i(s, \delta) = T_p(\delta) + T_c(s) + T_b(s) - T_l(s) \quad (13)$$

To form a basis for comparison, a transient FE synchronization simulation is first conducted using ANSYS Maxwell for a LSPMSM design. Then the following steps are performed:

- extract time-varied slip  $s(t)$  and rotor angle  $\theta(t)$  curves from ANSYS Maxwell;
- approximate load angle  $\delta(t)$  from rotor angle  $\theta(t)$ ;
- select  $T_p$  equation for comparison and create  $T_i(s(t), \delta(t))$  by inserting extracted  $s(t)$  and  $\delta(t)$ ;
- compare analytical  $T_i$  curve to actual ANSYS Maxwell  $T_i$  curve.

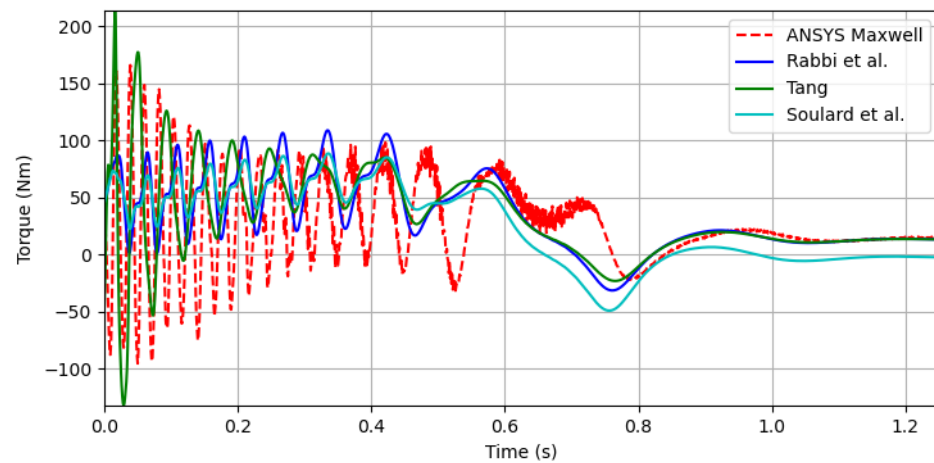
Following these steps, Figure 12 was created for the pulsating analytical equations mentioned above.

Unlike Soulard et al. [19], Tang [17] and Rabbi et al.'s [8] analytical formulae correctly meet the target curve once synchronisation has occurred and can be argued for. Tang's [17]  $T_p$  seems to result in a curve more accurately representing the magnitude of the initial instantaneous torque, while Rabbi et al.'s [8]  $T_p$  seems to be a better fit nearing the region of synchronization.

For this reason, both  $T_p$  equations will be carried forward into the analytical methods comparison in Section 3.3. To simplify notation, Tang's [17]  $T_p$  will be hence referred to as  $T_{pa}$  and Rabbi et al.'s [8] will be referred to as  $T_{pb}$ . The equations for  $T_{pa}$  and  $T_{pb}$  are given in (14) and (15), with the individual components listed in Appendix B.1.

$$T_{pa} = T_{pa1} \sin \delta + T_{pa2} \cos \delta + T_{pa3} \sin 2\delta + T_{pa4} \cos 2\delta \quad (14)$$

$$T_{pb} = T_{pb0} + T_{pb1} \sin \delta + T_{pb2} \sin 2\delta + T_{pb3} \cos \delta + T_{pb4} \cos 2\delta \quad (15)$$



**Figure 12.**  $T_i$  for various (Rabbi et al. [8], Tang [17], Soulard et al. [19])  $T_p$  cases.

### 3.3. Analytical Methods

When comparing different analytical synchronization methods mentioned in Section 2.2, two factors will be considered. Firstly, the consistency/accuracy of the method in determining the critical inertia factor, and secondly, any additional resulting outcomes which might prove meritorious to a designer.

#### 3.3.1. Critical Inertia Factor

In order to compare the critical inertia factors produced by each method, the actual critical inertia factors of the test machines need to be determined. This was conducted through a binary search approach using transient FE analysis. The values determined by this process can be seen in the column labelled "FEM" in Table 1 and represent the target which the analytical methods aim to meet. For the three different analytical methods, each method follows a somewhat different approach for the critical inertia factor ( $x_{cr}$ ) determination. Rabbi et al.'s method, mentioned in Section 2.2.1, determined  $x_{cr}$  using (6), while both Chama et al. methods used a linear search approach. The linear search approach

was implemented by simply increasing the load inertia by a factor, until this factor fails synchronization; the factor before that is  $x_{cr}$ .

The results of this comparison can be seen in Table 1. While accuracy is of some importance, a consistent trend in increase/decrease of  $x_{cr}$  seen across a cluster of machines is much more so. This is due to the fact, that when a designer makes a design change, the critical factor should accurately reflect the resulting impact on  $x_{cr}$  as well. Additionally, if the trend is not obeyed,  $x_{cr}$  would be of little use in the cost function of an iterative design optimization procedure. In Table 1, a cluster of green represents where a trend was correctly followed, while a cluster of pink represents where it was not.

**Table 1.**  $x_{cr}$  determined by various analytical methods for the machine set.

Design	FEM	Chama et al. (Time)		Chama et al. (Energy)		Rabbi et al.	
		$T_{pa}$	$T_{pb}$	$T_{pa}$	$T_{pb}$	$T_{pa}$	$T_{pb}$
Radial 1	44	110	52	111	52	213	69
Radial 2	40	112	49	113	50	225	67
Radial 3	93	0	66	0	66	556	44
V-Type 1	55	114	57	114	58	193	72
V-Type 2	39	120	40	121	40	306	58
V-Type 3	58	110	60	111	60	175	72
W-Type 1	65	121	60	122	60	218	77
W-Type 2	48	122	31	124	31	386	47
W-Type 3	164	200	158	201	158	174	120
Spoke 1	48	59	42	59	42	67	42
Spoke 2	45	42	34	43	34	38	28
Spoke 3	152	199	154	200	154	131	72

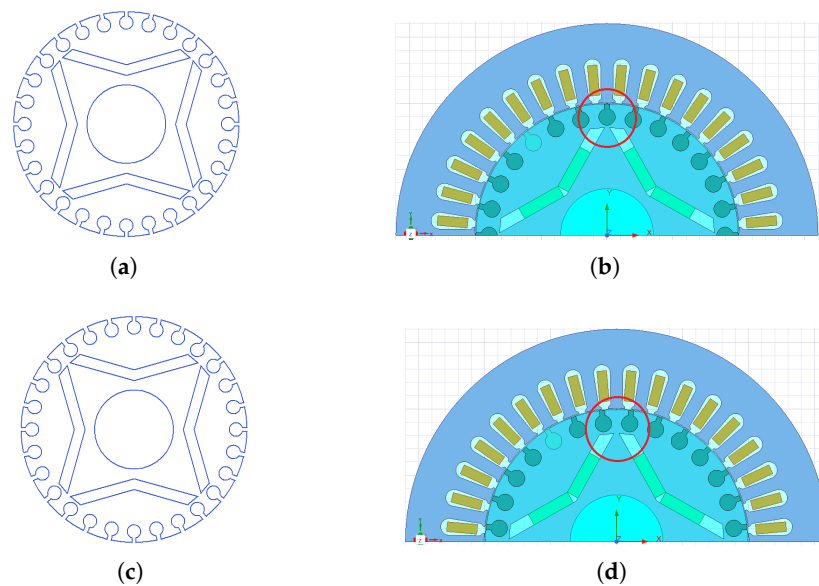
From Table 1 it is clear, that for use across a variety of different rotor topologies, both of Chama et al.'s methods with  $T_{pb}$  (Equation (15)) are the most fitting to accurately determine the trend of  $x_{cr}$ . While  $T_{pa}$  demonstrates the correct trend for some rotor design and method combinations, the predicted critical inertia values are much less comparable than when  $T_{pb}$  is used.

### 3.3.2. Influence of an ANSYS Electronics Bug to $X_{cr}$ Determination

While Table 1 demonstrates Chama et al.'s analytical methods correctly following the  $x_{cr}$  trend for the V-Types, W-Types, and Radial Types, there are cases (e.g., Radial 3) which show a large difference between the FEM and analytically predicted  $x_{cr}$  values. It was later found that a certain geometry conversion bug in ANSYS Electronics Desktop is largely responsible for this discrepancy.

One of such cases is demonstrated in Figure 13, where Figure 13a,c display two RMxprt LSPMSM designs with the only difference of 0.01 mm in PM thickness. While the RMxprt model layouts look identical, the respective Maxwell-2D FEM models seen in Figure 13b,d distinctly vary, as the red circles highlight. Clearly, for the 4.58 mm case, the cage slot was accidentally shifted by 1-slot pitch.

The time-domain method (Chama et al.) predicts a critical inertia factor of 60 for both designs from their RMxprt models. However, because of the unexpected slot shifting, the actual critical inertia factors, as determined through FE analysis in ANSYS Maxwell, differ greatly. The actual  $x_{cr}$  for the 4.59 mm case is 58, very close to the analytically predicted  $x_{cr}$ . For the 4.58 mm case on the other hand, where the alignment was shifted, the actual  $x_{cr}$  is 32, far from what was predicted for it by the analytical model and thus disturbing the trend.



**Figure 13.** Geometry conversion bug found in ANSYS Electronics Desktop. (a) ANSYS RMxprt geometry (PM thickness 4.58 mm); (b) generated ANSYS Maxwell model for (a); (c) ANSYS RMxprt geometry (PM thickness 4.59 mm); (d) generated ANSYS Maxwell model for (c).

### 3.3.3. Computational Aspects

During the creation of Table 1, the computational time to calculate the critical factors of all machines was noted, and can be seen in Table 2. The energy approach by Chama et al. is most time consuming among the three methods. The computational time of a specific method may not be a major concern for a single-run critical inertia factor analysis. However, it is critically important when the analysis is used in a multi-objective, highly iterative optimization procedure.

**Table 2.** Computational time to determine test machine's  $x_{cr}$  per method.

Type of Method	Time (s)
Time-Domain Method (Chama et al.)	33.667
Energy Method (Chama et al.)	450.852
Energy Method (Rabbi et al.)	0.500

Computationally, Rabbi et al.'s energy method outperforms both Chama et al.'s methods greatly. This is because Rabbi et al.'s method only focuses on the last pole slip and uses a sinusoidal approximation to estimate  $x_{cr}$  with (6). Chama et al.'s methods on the other hand take the entire asynchronous region into account and rely on linearly increasing the multiplier factor to determine  $x_{cr}$ , which is a far more computationally intensive approach. However, the energy method by Rabbi et al. fails to consistently predict the critical inertia factor trend (see Table 1). The time-domain method (Chama et al.) appears to be the best option, given the fact that it correctly predicts the critical inertia factor trend while at the same time not being too time demanding. Thus, it will be implemented in the multi-objective optimization procedure in Section 5. Note that all the computer simulations in this paper were conducted on a PC with Intel Core i7-3700 CPU 3.4 GHz, 16 GB RAM running 64-bit Windows 10.

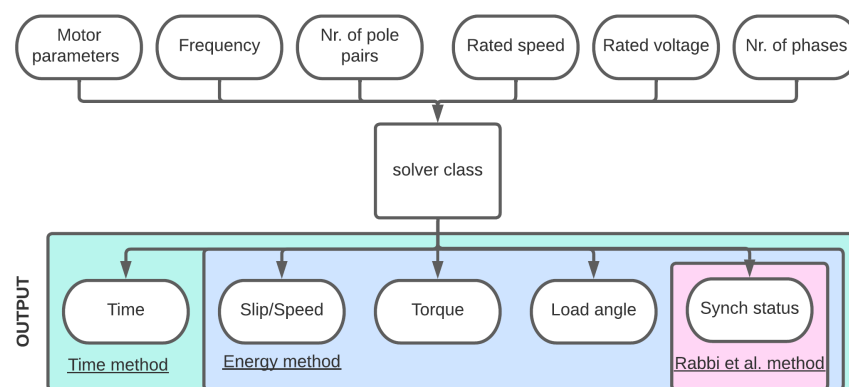
## 4. Software Design

*Python* was selected as the programming language for this development. Given its maturity and prevalence for scientific use, a variety of libraries to perform such tasks exist, the most notable being *SciPy*. In addition, a variety of graphical user interfacing (GUI) libraries exist as well, such as *Tkinter* and *PyQt*. The language also allows for compilation into executable files, thus meeting all necessary criteria for the development.

### 4.1. Overall Layout

Although the time-domain method (Chama et al.) has been identified as the most appropriate method to be implemented by the program, it would be beneficial to also include the energy-based methods in the implementation. This can be useful when a designer wants additional confirmation for a factor or graph produced by the time-based method. To accommodate these multiple methods, a facade structural pattern approach—as described in [24]—will be used. This facade will control a single “solver” class which, in turn, will interface with the various methods.

The solver class controls which methods are used, which  $x_{cr}$  approach is taken, and which parameters are passed between GUI and synchronization method. A list of all input and output parameters the solver class controls can be seen in Figure 14.



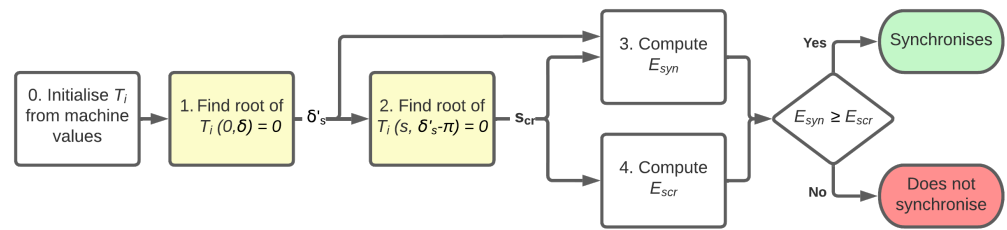
**Figure 14.** Input and output parameters managed by solver class.

### 4.2. Implementation of Methods

The methods as a whole are implemented as functions, with the required machine values being passed as arguments. Seeing as all methods require for the creation of  $T_i$ , as outlined in Section 2.2, the instantaneous torque equation is captured in a nested function, with parameters  $s$  and  $\delta$ . Implementing the instantaneous torque as a function will allow for a variety of different actions to be performed on it.

#### 4.2.1. Energy Method (Rabbi et al.)

The steps needed for the energy method (Rabbi et al.) are summarized in the flowchart below (Figure 15), where steps 1 and 2, as highlighted in yellow, require root finding algorithms. For this purpose, *SciPy*'s *root* function is used, with the  $T_i$  function as implemented in Section 4.2 passed as argument. Step 3's  $E_{syn}$  requires functional integration, which is performed using *SciPy*'s *quad* function.  $E_{scr}$  is simply computed, and the final return value of the process is the Boolean value defined by  $E_{syn} \geq E_{scr}$ . This value is passed to the “solver” class, with the facade updating accordingly.



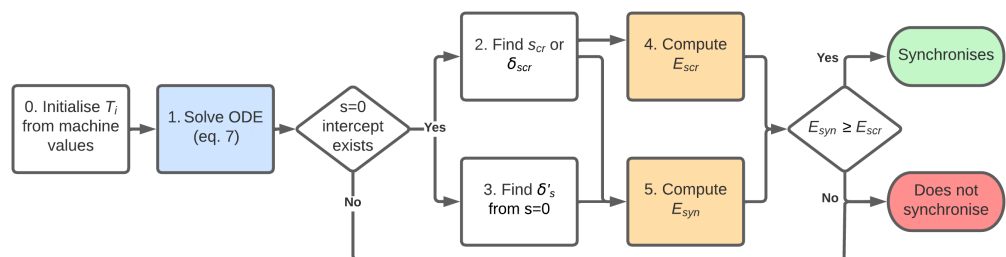
**Figure 15.** Flowchart implementing the energy method (Rabbi et al.).

#### 4.2.2. Energy Method (Chama et al.)

The steps needed for the energy method (Chama et al.) can be seen in Figure 16. Step 1 requires solving the ODE seen in (7). While Chama et al. recommend solving this with the Runge–Kutta–Fehlberg method (RK45), by choosing to use SciPy’s *solve\_ivp* function, other approaches can be considered as well.

Table 3 compares the critical inertia factors  $x_{cr}$  found by using FEM and different ODE solver methods such as the recommended RK45, Radau IIA, backward differentiation formula (BDF), and combined Adams and BDF method (LSODA). All of these are methods specifically aimed at numerically integrating implicit ODE equations.

From the table, the proposed RK45 method seems to deviate from the target  $x_{cr}$  values the most, especially for the V-Type 3 and W-Type 3 designs. While the other methods fall relatively close in terms of  $x_{cr}$  predicted, when it comes to computational time taken greater variation can be observed across the various methods. Table 4 summarizes the run-times for the different ODE methods, with the adaptive LSODA method executing noticeably faster than its peers.



**Figure 16.** Flowchart implementing the energy method (Chama et al.).

Since the ODE solver returns the slip versus load angle curve as a numerical array, steps 4 and 5—highlighted in orange in Figure 16—cannot be performed in the same manner as steps 3 and 4 were for the energy method (Rabbi et al.) in Figure 15. Thus for this numerical integration step, SciPy’s *Simpson* method is used.

However, while the energy comparison step  $E_{syn} \geq E_{scr}$  is mentioned in Chama et al.’s method, the paper also mentions that one advantage of the direct resolution of the ODE is that it allows for “easily recognizing the synchronization capability of the machine” [9] as well. This happens when the slip never reaches zero, and thus Figure 16 can be reduced to the flowchart seen in Figure 17.

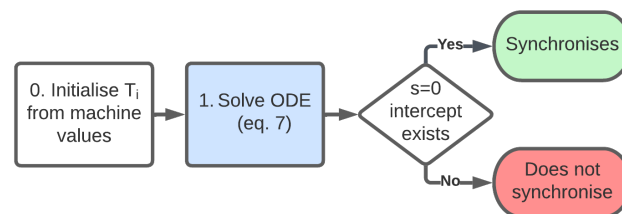
This has no impact on the  $x_{cr}$  determination, but in combination with a binary search approach, reduces the computational time of the LSODA ODE solver to 161.292 s.

**Table 3.**  $x_{cr}$  determined by the energy method (Chama et al.) with various ODE solvers compared to target  $x_{cr}$  determined through FEM.

Type of LSPMSM	FEM	RK45	Radau	BDF	LSODA
Radial 1	44	55	51	52	52
Radial 2	40	53	49	50	49
Radial 3	93	72	66	66	66
V-Type 1	55	61	57	58	58
V-Type 2	39	43	39	40	40
V-Type 3	58	81	59	59	60
W-Type 1	65	65	60	60	60
W-Type 2	48	34	31	31	31
W-Type 3	164	0	157	158	109
Spoke 1	48	60	42	42	42
Spoke 2	45	50	34	34	34
Spoke 3	152	175	154	154	154

**Table 4.** Computational time to determine  $x_{cr}$  using the energy method (Chama et al.) with different ODE solvers.

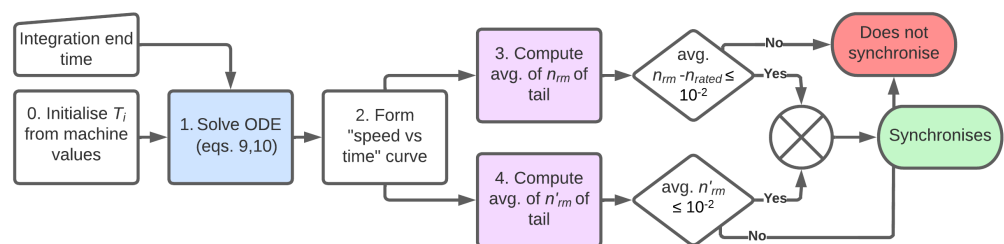
ODE Method	Time (s)
RK45	402.086
Radau	550.320
BDF	450.852
LSODA	276.545



**Figure 17.** Adapted flowchart implementing the energy method (Chama et al.).

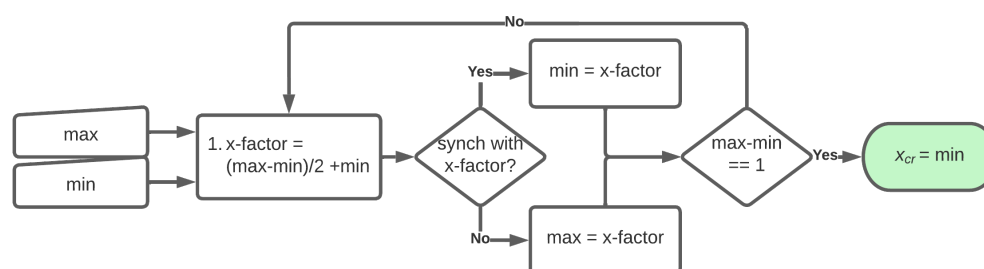
4.2.3. Time-Domain Method (Chama et al.)

The steps needed for the time-domain method (Chama et al.) are illustrated in Figure 18. Step 1, solving the ODE problem, is performed using the LSODA method, which was identified as the fastest ODE solver in Section 4.2.2. For fast computation of the averages  $\bar{n}_{rm}$  and  $\bar{n}'_{rm}$  (steps 3 and 4 in purple), the last section of the generated curve and its derivative are accumulated using the built-in Python function *sum* and then divided using the function *len*. The aforesaid derivative is calculated using SciPy’s *splev* function.



**Figure 18.** Flowchart implementing the time-domain method (Chama et al.).

To determine the critical inertia factor, a binary search approach is taken, following the logic seen in Figure 19. The minimum factor is initialized as 0, and the maximum factor as 500. This maximum value was chosen, seeing as it provides enough headroom above the commonly observed  $x_{cr}$  values, but not too far above to slow the search process down substantially. While the linear search approach used earlier in Section 3.3.3 took 33.667 s, determination of the same 12  $x_{cr}$  values was reduced to 18.822 s with the binary search method.



**Figure 19.** Flowchart implementing binary search approach to determine  $x_{cr}$ .

#### 4.3. Graphical User Interface

To implement the graphical user interface for the program, the PyQt library was used. This library was chosen, seeing as it allows for a drag-and-drop design process using the *Qt Designer* software tool. *Qt Designer* then produces a \*.ui file, which can be easily loaded with a specific PyQt function. This approach allows for independent changing of user interface elements, without needing to change the code behind them.

The interface is divided into 4 areas. The first area is used for inputting the machine's parameters needed to perform the analytical synchronization process. While these fields will be automatically filled by the script mentioned in Section 4.4, allowing the user access to these fields makes the program free-standing from ANSYS Electronics. The fields available in this area are for machine parameters  $E_0$ ,  $X_d$ ,  $X_q$ ,  $R_1$ ,  $X_1$ ,  $R_2$ ,  $X_{2d}$ ,  $X_{2q}$ , and  $J_{rot}$ . Additionally, the area adjusts dynamically to support a slider and button used for loading specific machine parameters after the additional features mentioned in Section 4.3.1 are used.

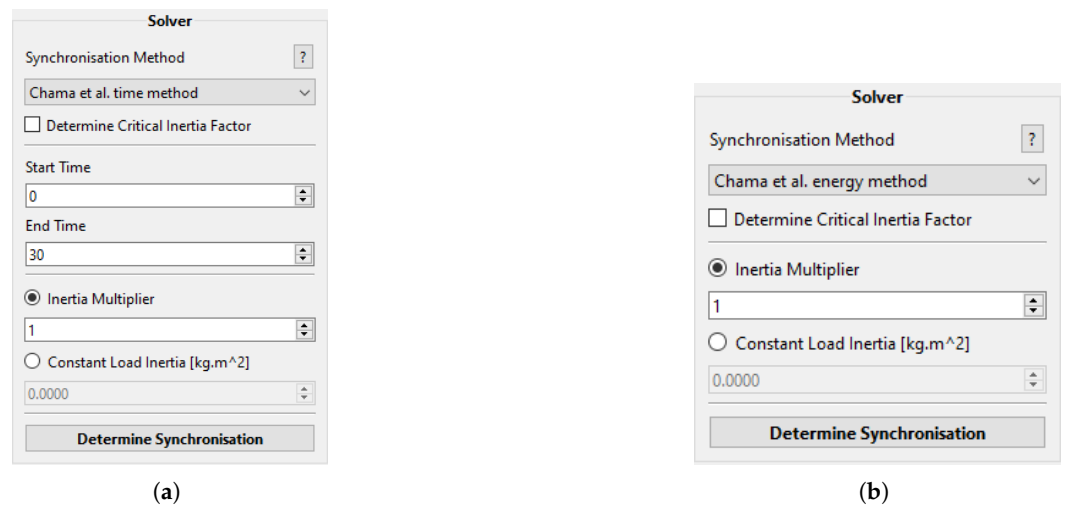
The second area contains the selection of the synchronization method to be used, along with any additional input fields. These input fields are for testing synchronization with a specific rotor inertia multiplier, testing for synchronization with specific additional load inertia, and for selecting whether the critical inertia factor should be determined. Additionally, the solver area adjusts to incorporate supplementary input fields dependent on solver type. An example of this can be seen in Figure 20.

The third area is used for displaying the various graphs generated by the methods. The selection for available graphs also adjusts according to the synchronization method selected earlier. The graphing area is implemented by using an embedded *Matplotlib* area. A "Clear Graphs" button is implemented as well, allowing for comparison of curves for different input values. The legend is auto-generated depending on whether a constant inertia value or a rotor inertia multiplier is added.

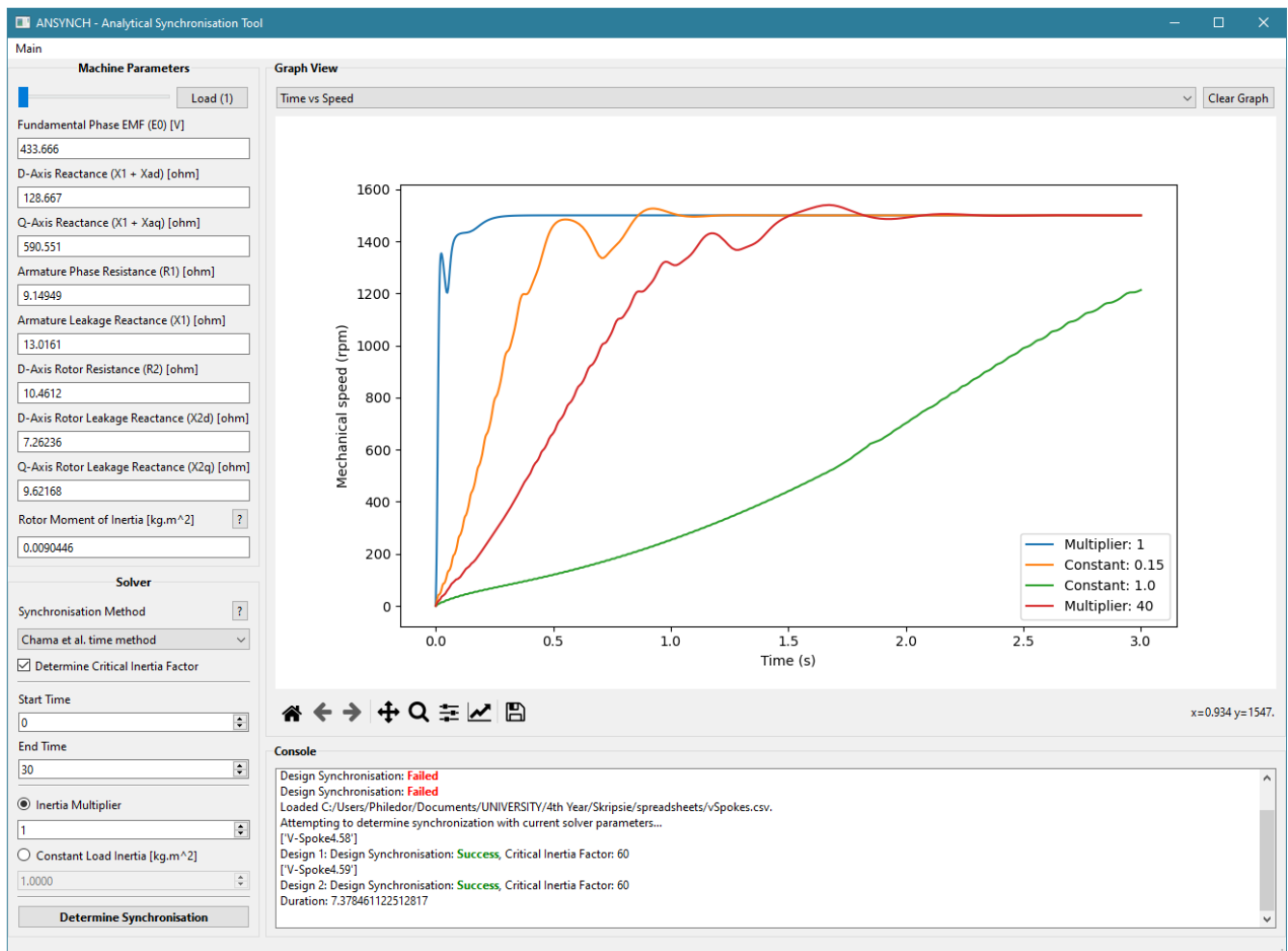
The last area is an output logging area, showing the critical inertia factor (if the option is selected) and whether synchronization has occurred. Any additional information is also displayed here.

All areas combine into the final layout seen in Figure 21. The width of the machine parameter and solver areas is fixed, while the shared space between console and graph area can be adjusted in order to meet the user's needs.





**Figure 20.** Solver UI area adjusting by solver method: (a) solver area for the time-domain method (Chama et al.), (b) solver area for the energy method (Chama et al.).



**Figure 21.** Final GUI layout of analytical synchronization program.

#### 4.3.1. Additional Features

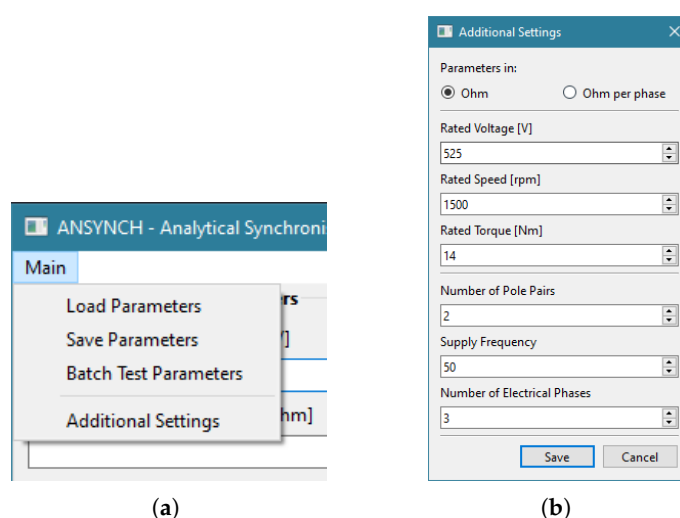
Three additional features (namely, *Loading*, *saving*, and *batch testing*) are added for convenience sake, accessible through the top left menu button as can be seen in Figure 22a. All three features use comma-separated values (CSV) files.

The “Load Parameters” feature loads a number of machines’ parameters from a CSV file into the program. Using the slider in the top right as seen in Figure 21, these various parameters can then be loaded into their respective fields for testing.

The “Save Parameters” feature appends the machine parameters currently present in the program to a CSV file of choice. If no file exists, a new one can be created.

The “Batch Test Parameters” feature uses the current solver settings to test the synchronization and—if selected—critical inertia factor of machine parameters contained within a CSV file. These machine parameters are also loaded into the program, similar to the load feature, and the output of the tests is shown in the console.

To remove clutter from the program, some of the input settings listed in Figure 14 have been extracted to an “Additional Settings” dialog. The dialog is accessible through the menu, as can be seen in Figure 22a, and opens the window seen in Figure 22b. Common machine settings occupy these fields by default.



**Figure 22.** Additional program features: (a) menu connecting to additional features and settings, (b) additional settings window.

#### 4.4. Integration with ANSYS Electronics Desktop

ANSYS Electronics Desktop offers programming capabilities through their scripting environment [25]. Further, ANSYS specific libraries are also accessible. The scripting environment implements IronPython 2.7, which implements Python 2.7 and NET framework 4.0. Thus, scripts can be coded in either Python or VBScript. Considering Python’s additional flexibility, all scripts were written in Python. These scripts can also be added to ANSYS Electronics Desktop as “Tools”, allowing them to be run from the “Tools” tab by the press of a button.

To integrate the program flawlessly into ANSYS Electronics Desktop, a script was written which:

- Extracts available machine parameters from an RMxpert design.
- Approximates inaccessible machine parameters.
- Launches and automatically populates the analytical synchronization program with the required values.

##### 4.4.1. Extraction of Available Parameters

The scripting library’s functionality is limited, seeing as it is mainly used to change ANSYS specific features. As a result, no direct function is present in the scripting environment for accessing the solution data of an RMxpert design.

In order to overcome this problem, the “output variable” commands of the scripting environment are exploited. The script contains the names of the desired machine parameters, as listed in ANSYS. Then, for each parameter, the following steps are taken:

- create temporary output variable using *CreateOutputVariable*;
- obtain output variable value using *GetOutputVariable*;
- delete temporary output variable using *DeleteOutputVariable*.

Using this approach, eight of nine parameters needed for analytical synchronization determination can be extracted from an RMxprt design. These parameters are  $E_0$ ,  $X_d$ ,  $X_q$ ,  $R_1$ ,  $X_1$ ,  $R_2$ ,  $X_{2d}$ , and  $X_{2q}$ .

#### 4.4.2. Approximation of Inaccessible Parameters

The machine parameter that cannot be extracted from RMxprt is the rotor inertia  $J_{rot}$ , which is only available after a Maxwell design was created from the RMxprt design. To make the analytical synchronization program standalone, an approximation of the combined rotor inertia is computed and then passed to the program. This approximated inertia value can still be changed by the user in the program.

The rotor's inertia can be modeled as three separate inertia values. Firstly, the rotor body inertia, which contains the cage windings, magnets, and the steel rotor core. Secondly, the end rings' combined inertia which are located at the ends of the rotor. Lastly, the shaft inertia internal to the rotor.

Using the scripting function *GetPropertyValue* allows for retrieval of the rotor inner radius  $r_i$  and rotor outer radius  $r_o$ . Using the approach mentioned in Section 4.4.1, the total cage bar mass, total magnet mass, and rotor steel mass can be extracted as well. The rotor body is then approximated as a hollow cylinder, with the cage bar mass being approximated at the edge of the rotor radius. The end rings are also approximated as two hollow cylinders. The shaft is assumed to be roughly 1.75 times longer than the rotor body length, which can be extracted from RMxprt.

The final rotor inertia approximation is then computed using (16).

$$J_{rot} = J_{body} + J_{rings} + J_{shaft} \quad (16)$$

The approximate rotor inertias are compared to the rotor inertias generated by ANSYS Maxwell in Table 5, which shows that the approximated rotor inertias are reasonably close to the ones generated by Maxwell.

**Table 5.** Approximate and Maxwell rotor inertias.

Machine	Maxwell $J_{rot}$ (kgm <sup>2</sup> )	Approximated $J_{rot}$ (kgm <sup>2</sup> )	% Difference
Radial 1	0.0090446	0.0081215	10.21
Radial 2	0.0090446	0.0081178	10.25
Radial 3	0.0090446	0.007938	12.23
V-Type 1	0.0090446	0.0081899	9.45
V-Type 2	0.0090446	0.0082991	8.24
V-Type 3	0.0090446	0.0081992	9.35
W-Type 1	0.0090446	0.0080894	10.56
W-Type 2	0.0090446	0.0082279	9.03
W-Type 3	0.0090446	0.0077131	14.72
Spoke 1	0.0090446	0.0083418	7.77
Spoke 2	0.0090446	0.0083635	7.53
Spoke 3	0.0090446	0.0077431	14.39

#### 4.5. Software Design Summary

The program described in this section is bundled into an executable using the *pyinstaller* library, after which the path to the executable is linked into the ANSYS script. The script gathers all retrievable parameters as shown in Section 4.4.1, and approximates all inaccessible parameters as described in Section 4.4.2.

The analytical synchronization program is then automatically launched with all parameters passed as system arguments. These arguments are then parsed into the program's machine parameter fields. The aforementioned script can be linked to the 'Tool' bar in ANSYS Electronics Desktop allowing for retrieval, approximation, launching, and parsing to happen with the click of one button.

## 5. Application of the Developed Software Tool

The analytical program developed in Section 4 has the ability to predict the critical inertia factor  $x_{cr}$  of a LSPMSM motor design. Using the critical inertia factor as a design objective in a multi-objective design procedure is of interest, but has so far only been achieved through use of the Taguchi method, a low-iteration multi-objective design strategy [14].

Given the developed program's integration with ANSYS Electronics Desktop, this section aims to evaluate the effectiveness of the developed program in a highly iterative, multi-objective optimization procedure using more conventional iterative strategies.

### 5.1. Software Adaptation

In order to be used for this iterative procedure, the developed software tool needs to be adapted for the task at hand: it needs to be as fast as possible. A light-weight version was thus developed, which does not feature the GUI, forces use of the time-domain method (Chama et al.), and returns the critical inertia factor upon completion. Given the simplification present here, it would have been ideal to perform this calculation in the ANSYS scripting environment for even faster speeds. However, due to the scripting environment's restricted nature, no additional libraries can be installed into it, preventing the use of *SciPy* functions.

Initially, the light-weight program was compiled in *Python* directly from its larger counterpart, and then converted into an executable file using the *pyinstaller* library. However, early testing of the program in the iterative process showed that it was still far too slow to be used for a highly iterative procedure. In an attempt to correct this, the light-weight program was rewritten into the *Go* programming language. *Go* was chosen due to the fact that it is designed around speed, easily and cleanly compiles into light-weight executables, and has a growing supplementation of user created libraries. To replace the role *SciPy*'s ODE solver played, the *Godesim* library was used. *Godesim* only employs the Runge–Kutta–Fehlberg method at the moment, but as Table 6 demonstrates, this does not cause it to differ too much from its *Python* counterpart, and the correct  $x_{cr}$  trend is obeyed, making it suitable for the task at hand.

The *Go* program's execution time proves to be far superior compared to its *Python* counterpart. Initialization scores and five iterations of the procedure described in Section 5.2 took the *Python* version 35 min and 17 s, while the *Go* version only needed 8 min and 26 s.

**Table 6.** Comparing *Go* and *Python* ODE solvers for predicting  $x_{cr}$ .

Machine	Python LSODA $x_{cr}$	Go RK45 $x_{cr}$
Radial 1	52	51
Radial 2	49	49
Radial 3	66	61
V-Type 1	57	57
V-Type 2	40	39
V-Type 3	60	59

Table 6. Cont.

Machine	Python LSODA $x_{cr}$	Go RK45 $x_{cr}$
W-Type 1	60	60
W-Type 2	31	31
W-Type 3	158	108
Spoke 1	42	42
Spoke 2	34	34
Spoke 3	154	120

### 5.2. Differential Evolution Implementation

Differential evolution (DE) was chosen as the multi-objective optimization procedure to be implemented. DE begins by initializing a starting population with a set number of members. Each member  $x$  has a number of features  $y$ . Then, three members of the population are chosen at random and each of their features are mutated following (17) to produce a mutated member.

$$y_{mutation} = y_{x1} + (y_{x2} - y_{x3}) \quad (17)$$

Crossover is then performed between the mutated member and a chosen member of the population. This involves, for each feature, generating a random value between 0 and 1 and comparing it to the crossover rate, which was set at 0.5. If the value is greater than the crossover rate, the mutated feature replaces the chosen member's feature. After crossover is completed, the new member's score is generated according to the overall evaluation criterion (OEC), and if it is greater than the chosen member's score, the chosen member is replaced. Each iteration of DE performs mutation, crossover, and comparison for each member of the population.

Ideally, DE would have been performed in ANSYS Electronics Desktop using ANSYS Optimetrics; however, interfacing with Optimetrics for each iteration of the optimization process is not possible. Instead, DE was coded in the scripting environment in Python. Figure 23 demonstrates how these various environments are iteratively used and interact.

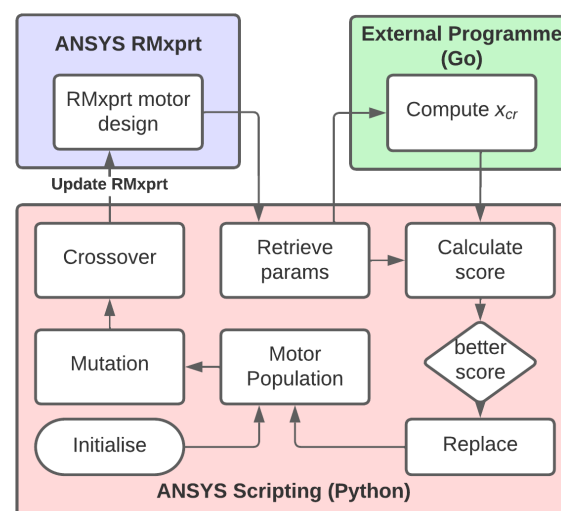


Figure 23. The differential evolution process.

### Optimization Problem

It is known that there is a competing relationship between the critical inertia factor ( $x_{cr}$ ) and the power factor ( $PF$ ) for the design of a LSPMSM [14], which means finding a design that maximizes both factors is unfeasible. Instead, a trade-off between the two factors can

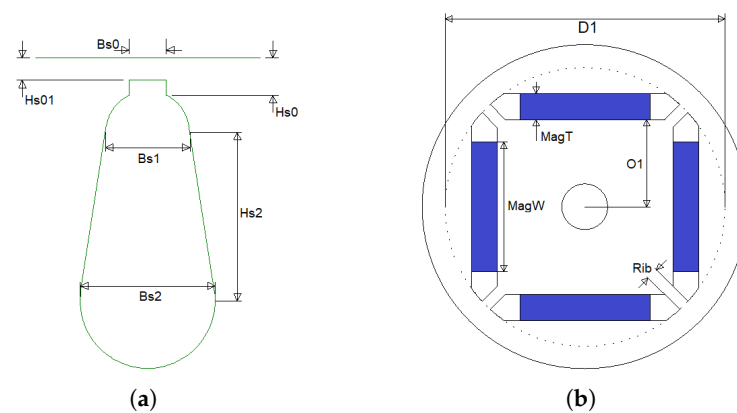
be found. Thus, the optimization problem has multiple objectives and constraints, which is formulated as follows:

$$\text{Maximize : } F(\mathbf{X}) = \sum_{m=1}^3 \omega_m f_m \quad (18)$$

where  $\mathbf{X}$  represents the vector of geometric variables illustrated in Figure 24, and  $f_m$  is populated by the objective functions seen below, which can be weighted differently through  $\omega_m$ .

$$f_1 = \frac{x_{cr}}{x_{cr \max}}; \quad f_2 = \frac{PF}{PF_{\max}}; \quad f_3 = \frac{1}{1 + (P_{out} - P_{rated})^2}$$

The output power  $P_{out}$  is included in the optimization problem to ensure a working design. Essentially,  $P_{out}$  is a constraint and not in contest with the other objectives. Its weighting is kept as 0.5 in this optimization study.



**Figure 24.** Parameters to be varied during optimization: (a) available slot parameters, (b) available magnet parameters.

### 5.3. Parameter Restrictions

Given the extreme variety of possible rotor layouts, invalid designs are more than likely to be created during crossover and random initialization. While the DE method will simply replace these designs in the next iteration, some software restrictions are implemented as well to limit the occurrence of this happening.

For the DE procedure, the radial rotor topology seen in Figure 1a was chosen, and thus the successive restrictions implemented are specifically for this type. The stator design remains constant, leaving only the slot design and rotor dimensions to be varied. Figure 24a shows the slot parameters which can be varied, while Figure 24b shows the magnet dimensions which can be varied.

Initialization begins by pre-defining a minimum and maximum value for all parameters. These initial minima and maxima are only used during the initialization process, and are then replaced per DE iteration.

#### Magnet Restrictions

While the slot parameters exhibit no complicated relationships, the magnet parameters do, and thus the order in which magnet parameters are changed matters greatly. The order is determined such that the subsequent parameter restrictions are only dependent on the ones before it. Table 7 summarizes these relationships.

**Table 7.** Restrictions for magnet dimensional parameters.

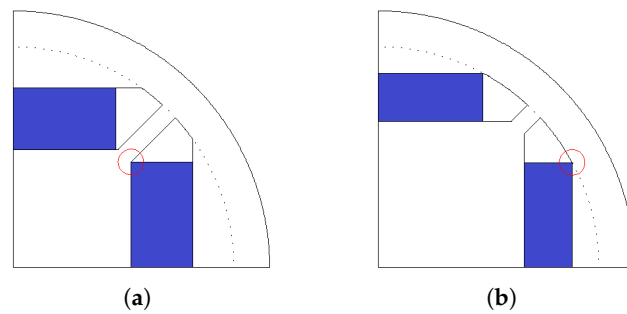
Parameters	Minimum	Maximum
1. D1	$2 r_i$	$2(r_o - \text{slot depth} - \text{CDG})$
2. O1	$r_i$	$D1/2$
3. MagT	$> 0$	$D1/2 - O1$
4. Rib	$\sqrt{2} \left( O1 - \sqrt{\frac{D1^2}{2} - O1^2} \right)$	$\sqrt{2}O1 - 1$
5. MagW	$> 0$	Equation (22)

D1’s maximum restriction is defined by the outer rotor radius, slot depth, and cage duct gap (CDG), which was set as 0.1 mm. The CDG ensures that there is at least some gap between D1 and the created slots. The slot depth is calculated using (19).

$$\text{slot depth} = Hs0 + Hs2 + \frac{Bs1}{2} + \frac{Bs2}{2} \tag{19}$$

The minimum rib restriction makes sure that the slot in which the magnet sits does not protrude D1. The maximum rib restriction ensures that MagW is not forced to be 0.

The maximum magnet width is restricted by two cases which can arise. The inner case is demonstrated in Figure 25a and thus is restricted by (20). The outer case is demonstrated in Figure 25b and as a result, restricted by (21).



**Figure 25.** Possible magnet width restrictions: (a) inner magnet width restriction, (b) outer magnet width restriction.

$$\text{innermax} = 2 \left( O1 - \sqrt{2} \frac{\text{Rib}}{2} \right) \tag{20}$$

$$\text{outermax} = 2 \left( \sqrt{\left( \frac{D1}{2} \right)^2 - (O1 + \text{MagT})^2} \right) \tag{21}$$

$$\text{MagW}_{max} = \min(\text{outermax}, \text{innermax}) \tag{22}$$

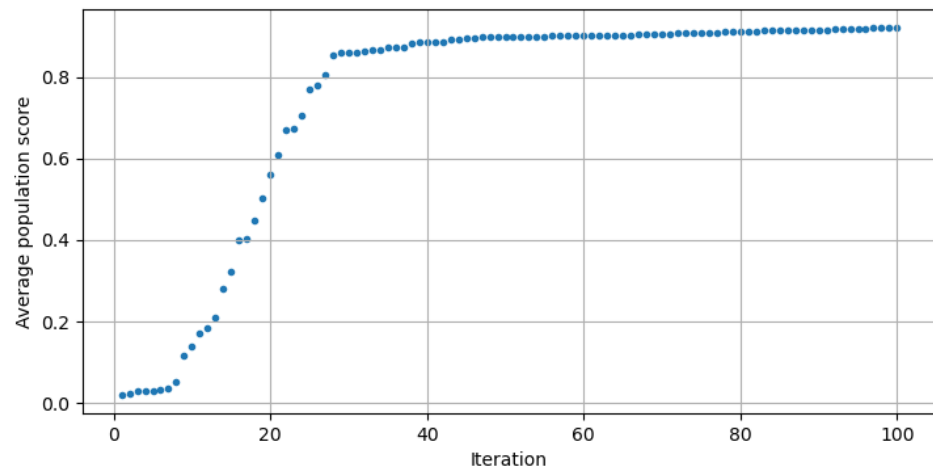
These restrictions were coded in the scripting environment. Mutation ignores the maximum and minimum values and simply creates the mutated parameters. Then, crossover is performed first for all slot properties, followed by crossover for the magnet properties in the order seen in Table 7. The minimum and maximum values for each parameter are first calculated from the preceding parameters, after which the mutated value is—if necessary—clamped into its respective range.

5.4. Optimization Procedure Results

5.4.1. Initial DE Procedure

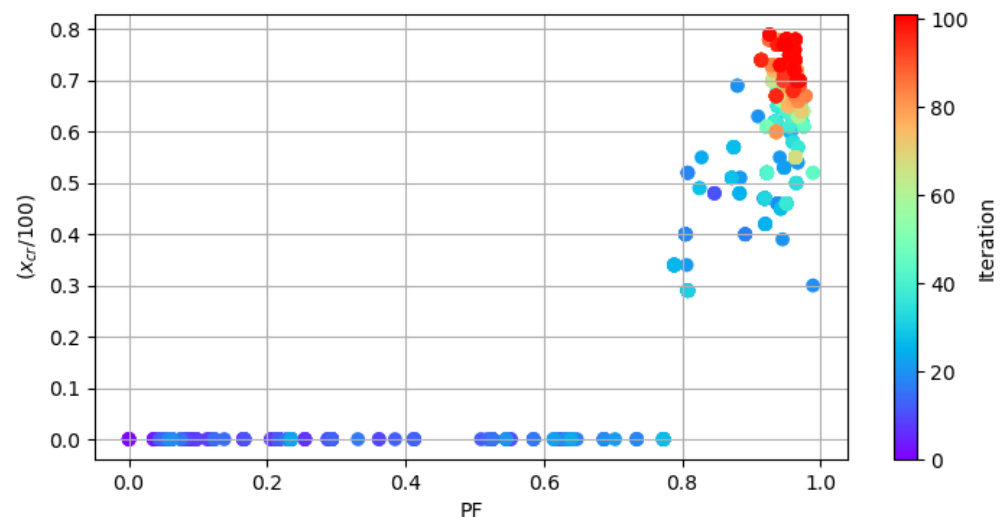
The DE procedure described in Section 5.2 was run for 100 iterations, with the OEC set as seen in (18). Given the importance of reaching the desired output power,  $\omega_3$  was set as 0.5. For initial demonstration purposes,  $\omega_1$  and  $\omega_2$  were set as 0.25 each, thus directing the DE process to find a 50/50 balance between the critical inertia factor and the power factor.

$PF_{max}$  is always 1 and from experience  $x_{cr\ max}$  was set as 100. The average population score per iteration can be seen in Figure 26, with a clear increase in average score being observable.



**Figure 26.** Average population score per iteration.

Evolution of both the power factor and critical inertia factor can be observed in Figure 27. Note that the colour map represents when last a machine design was present in an iteration's population. From Figure 27 it is clear that early iterations of the procedure produced designs which predicted a zero critical inertia factor. However, at around iteration 20, designs were created which predicted a low  $x_{cr}$ . One machine can also be seen with a full zero score during initialization due to an invalid rotor layout. Without the restrictions placed in Section 5.3, a high number of invalid initial designs could occur.



**Figure 27.** Evolution of the power factor and critical inertia factor per iteration.

#### 5.4.2. Varying Weightings

To further test the applicability of the analytical critical inertia factor program for multi-objective optimization, the weightings  $\omega_1$  and  $\omega_2$  were adjusted to various distributions in order to shift priority between the critical inertia factor and the power factor.  $\omega_3$  was kept constant at 0.5.



Figure 28 summarizes later members of the DE populations with these varied weightings. A clear competing relationship between the critical inertia factor and the power factor can be observed. By varying these weightings, designs can be created which prioritize either the critical inertia factor or the power factor. This establishes what is known as a Pareto front, a set of non-dominated solutions a designer can choose from [14]. While some outliers exist, a general trend between critical inertia factor and power factor is observable.

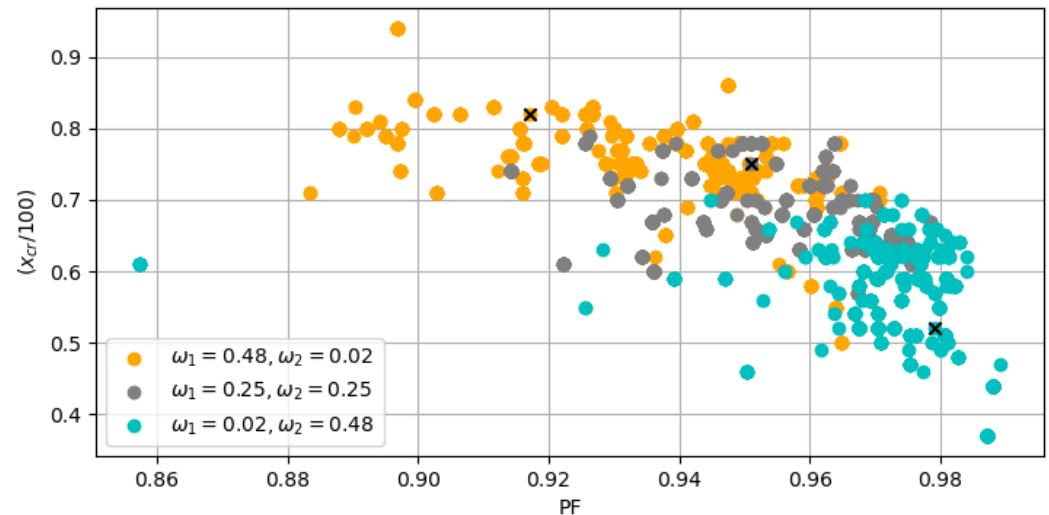


Figure 28. Late DE iterations with varied weightings.

#### 5.4.3. Confirmation of Optimized Critical Inertia Factors

To check the validity of the optimization procedure, three different LSPMSM designs (marked by crosses in Figure 28) from differently weighted populations were chosen for comparison, and their actual critical inertia factors were determined by FE analysis. The rotor layouts of the three LSPMSM designs can be seen in Figure 29.

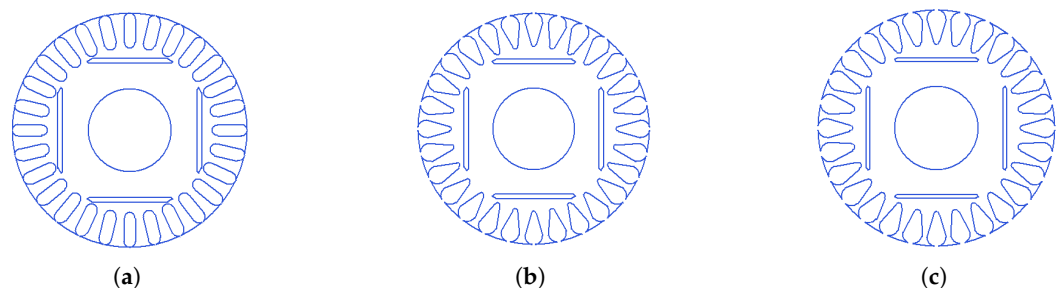


Figure 29. Three designs chosen from the DE optimization results for comparison: (a) design 1, (b) design 2, (c) design 3.

The analytically predicted and actual critical inertia factors, as well as power factors, output power, and weightings which produced the designs, are given in Table 8. As can be seen from the table, the predicted  $x_{cr}$  values fall short of the actual ones. The bug mentioned in Section 3.3.2 was present in all three designs when their Maxwell 2D models were created. While this distortion of  $x_{cr}$  values due to the geometry conversion bug is undesirable, Table 8 demonstrates that the trend set by the analytical program is still maintained, proving the validity of use for the analytical program in the procedure. It may be inferred that a better correlation between the  $x_{cr}$  values by analytical method and FE analysis can be expected without the influence of the identified ANSYS bug.

**Table 8.** Critical inertia factors, lifetimes, and scores of DE designs.

	Analytical $x_{cr}$	FEM $x_{cr}$	$\omega_1$	$\omega_2$	PF	$P_{out}$ (kW)
Design 1	63	43	0.02	0.48	0.979	2.2
Design 2	75	55	0.25	0.25	0.951	2.2
Design 3	82	57	0.48	0.02	0.917	2.2

#### 5.4.4. Computational Time

While some improvements in computational time were achieved in Section 5.1, performing the process in the ANSYS scripting environment adds to the computational cost greatly. For each machine design tested, the ANSYS model needs to be set and analyzed by ANSYS before the required parameters can be retrieved and the critical inertia factor calculated. On average, one iteration of the DE process took approximately 1.5 min or 6 s per design. Better computational time could be achieved if the program were to be merged with ANSYS Optimetrics, a task currently impossible.

## 6. Conclusions

The work in this paper began by gathering and comparing various methods and equations for determining the synchronization status of LSPMSMs. For synchronization torque equations, (1), (11), and (15) were identified as the most fitting equations. The time-domain method (Chama et al.) was determined to be the best method in terms of computational time and the ability of predicting the critical inertia factor trend.

Using the selected methods, a software tool was created which captured their individual advantages in terms of speed and visual aid. Some useful features were added to broaden the tool's use. Utilizing ANSYS scripting, the software tool was then integrated with ANSYS Electronics Desktop. The script was developed to automatically extract available parameters and to approximate inaccessible ones, providing a seamless interface between ANSYS Electronics Desktop and the analytical synchronization tool.

Given the software tool's ability to quickly determine the critical inertia factor, it was subsequently adjusted to be used in a highly iterative optimization procedure. This involved re-coding it into a programming language more suited for this recursive operation. Next, the optimization method 'differential evolution' was programmed in the ANSYS scripting environment, where the analytical tool and ANSYS RMXprt were employed recursively to successfully perform multi-objective highly iterative optimization for LSPMSMs. The results highlighted the usefulness of the tool's ability to quickly determine a machine's critical inertia factor and how it can be used in conjunction with ANSYS Electronic Desktop software suite to form a multi-objective design optimization procedure for LSPMSMs.

Although the critical inertia factors predicted by both the FEM and the analytical software tool show the same trend and consistency, there are often discrepancies between their values. It was found that a geometry conversion bug of ANSYS Electronics Desktop is most likely the cause. Better correlation between these  $x_{cr}$  values may be expected without the influence of this ANSYS bug.

**Author Contributions:** Conceptualization, P.S. and R.-J.W.; methodology, P.S. and R.-J.W.; software, P.S.; validation, P.S. and R.-J.W.; formal analysis, P.S.; investigation, P.S.; resources, P.S. and R.-J.W.; data curation, P.S.; writing—original draft preparation, P.S.; writing—review and editing, P.S. and R.-J.W.; visualization, P.S.; supervision, R.-J.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

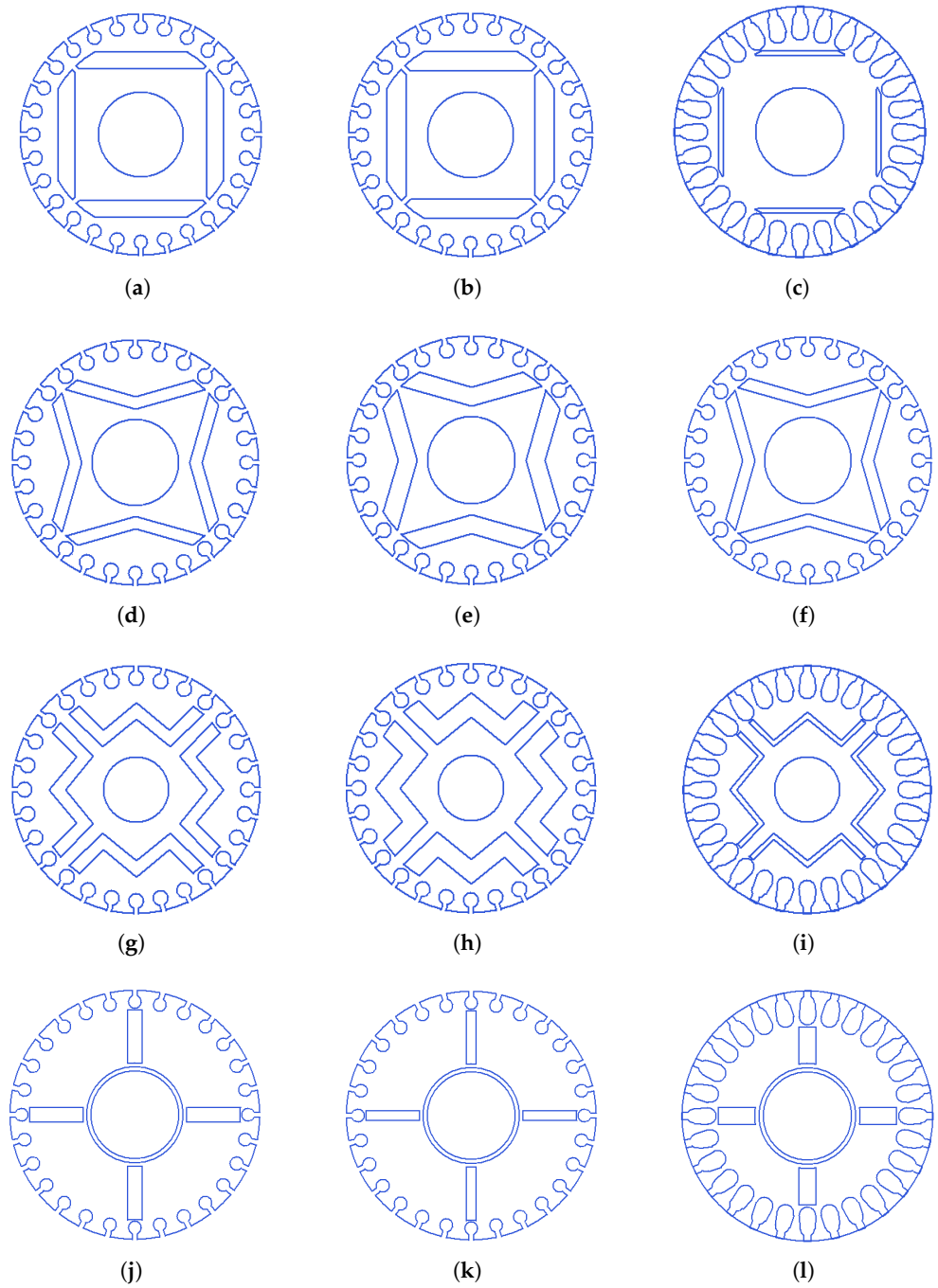
## List of Symbols

Symbols	Definition
$s$	Slip
$s_{cr}$	Critical slip
$\delta$	Load angle (rad)
$\theta$	Rotor angle (rad)
$f$	Frequency (Hz)
$\omega_s$	Electrical synchronous speed (rad/s)
$\omega_{rm}$	Motor rotational speed (rad/s)
$\bar{n}_{rm}$	Average motor rotational speed (rpm)
$n_{rated}$	Motor rated speed (rpm)
$x_{cr}$	Critical inertia factor
$J_{cr}$	Critical inertia
$T_b$	Braking torque (Nm)
$T_c$	Cage torque (Nm)
$T_p$	Pulsating torque (Nm)
$T_l$	Load torque (Nm)
$T_{rated}$	Rated torque (Nm)
$T_i$	Instantaneous torque (Nm)
$E_0$	RMS induced back-EMF (V)
$R_1$	Stator resistance ( $\Omega$ )
$X_1$	Stator reactance ( $\Omega$ )
$X_d$	Direct axis synchronous reactance ( $\Omega$ )
$X_q$	Quadrature axis synchronous reactance ( $\Omega$ )
$R_{2d}$	Direct axis rotor resistance ( $\Omega$ )
$R_{2q}$	Quadrature axis rotor resistance ( $\Omega$ )
$R'_2$	Referred rotor resistance ( $\Omega$ )
$X_{ad}$	Direct axis reaction reactance ( $\Omega$ )
$X_{aq}$	Quadrature axis reaction reactance ( $\Omega$ )
$E_{scr}$	Critical synchronisation energy (J)
$E_{syn}$	Synchronisation energy (J)
$PF$	Power factor
$P_{out}$	Output power (W)
$P_{rated}$	Rated power (W)

## Appendix A. Test Machine Set

Table A1. Test machine parameters.

	$E_0$	$X_d$	$X_q$	$R_1$	$X_1$	$R'_2$	$X_2$
Radial 1	433.666	128.667	590.551	9.14949	13.0161	10.4612	8.44202
Radial 2	449.729	125.196	590.551	9.14949	13.0161	10.4612	8.43316
Radial 3	301.781	122.485	526.218	19.542	11.4	5.46636	5.78165
V-Type 1	458.604	141.214	590.551	9.14949	13.0161	10.1826	7.56974
V-Type 2	479.15	106.323	590.551	9.14949	13.0161	10.1826	7.48063
V-Type 3	453.474	149.655	590.551	9.14949	13.0161	10.1826	7.5913
W-Type 1	445.096	136.028	590.551	9.14949	13.0161	9.85417	7.83725
W-Type 2	497.386	93.797	590.551	9.14949	13.0161	9.85417	7.72939
W-Type 3	335.239	216.471	527.534	9.14949	12.716	5.46636	7.6169
Spoke 1	436.092	206.069	590.551	9.14949	13.0161	12.2775	7.719285
Spoke 2	403.516	252.735	590.551	9.14949	13.0161	12.2775	7.83847
Spoke 3	270.129	187.855	527.534	9.14949	12.716	5.46636	7.543815



**Figure A1.** Test machine set rotor layouts: (a) Radial 1, (b) Radial 2, (c) Radial 3, (d) V-type 1, (e) V-type 2, (f) V-type 3, (g) W-type 1, (h) W-type 2, (i) W-type 3, (j) Spoke-type 1, (k) Spoke-type 2, (l) Spoke-type 3.

## Appendix B. Detailed Equations

### Appendix B.1. Pulsating Torque Equation Components

#### Appendix B.1.1. Tang

$$T_{pa1} = mp(bg' + ha' - de' - f_1c') \quad (A1a)$$

$$T_{pa2} = mp(ce' + ec' - ag' - ga') \quad (A1b)$$

$$T_{pa3} = \frac{mp}{2}(ah + bg - cf_1 - de) \quad (A1c)$$

$$T_{pa4} = \frac{mp}{2}(ce + bh - df_1 - ag) \quad (A1d)$$

$$a = K_{d1} V_{ph}$$

$$b = K_{d2} V_{ph}$$

$$c = K_{q1} V_{ph}$$

$$d = K_{q2} V_{ph}$$

$$e = \frac{V_{ph}}{2\pi f}(a_2 K_{d1} + b_2 K_{d2})$$

$$f_1 = \frac{V_{ph}}{2\pi f}(a_2 K_{d1} - b_2 K_{d2})$$

$$g = \frac{V_{ph}}{2\pi f}(c_2 K_{q1} + d_2 K_{q2})$$

$$h = \frac{V_{ph}}{2\pi f}(c_2 K_{q1} - d_2 K_{q2})$$

$$a' = -(1-s)^2 X_q \frac{E_0}{D_m}$$

$$c' = -(1-s) R_1 \frac{E_0}{D_m}$$

$$e' = \frac{X_d a' + E_0}{2\pi f}$$

$$g' = \frac{X_q c'}{2\pi f}$$

$$D_m = R_1^2 + X_d X_q (1-s)^2$$

$$K_{d1} = \frac{[R_1 - (1-2s)d_2]f_2 + (1-2s)c_2 e_2}{e_2^2 + f_2^2}$$

$$K_{d2} = \frac{[R_1 - (1-2s)d_2]e_2 - (1-2s)c_2 f_2}{e_2^2 + f_2^2}$$

$$K_{q1} = \frac{[R_1 - (1-2s)b_2]e_2 - (1-2s)a_2 f_2}{e_2^2 + f_2^2}$$

$$K_{q2} = -\frac{[R_1 - (1-2s)b_2]f_2 + (1-2s)a_2 e_2}{e_2^2 + f_2^2}$$

$$a_2 = X_d - \frac{s^2 X_{ad}^2 X_{2d}}{R_{2d}^2 + (sX_{2d})^2}$$

$$b_2 = \frac{sX_{ad}^2 R_{2d}}{R_{2d}^2 + (sX_{2d})^2}$$

$$c_2 = X_q - \frac{s^2 X_{aq}^2 X_{2q}}{R_{2q}^2 + (sX_{2q})^2}$$

$$d_2 = \frac{sX_{aq}^2 R_{2q}}{R_{2q}^2 + (sX_{2q})^2}$$

$$e_2 = R_1^2 + sR_1(b_2 + d_2) + (1 - 2s)(a_2c_2 - b_2d_2)$$

$$f_2 = sR_1(a_2 + c_2) - (1 - 2s)(a_2d_2 + b_2c_2)$$

### Appendix B.1.2. Rabbi et al.

$$T_{pb0} = \frac{mpR_1X_q}{\pi f(R_1^2 + X_dX_q)^2} \left[ (X_d - X_q) \left( \frac{V_{ph}^2}{2} - 1 + E_0^2 \right) - E_0^2 \left( \frac{R_1^2}{X_q} + X_d \right) \right] \quad (A2a)$$

$$T_{pb1} = \frac{mpE_0V_{ph}}{2\pi f(R_1^2 + X_dX_q)^2} [(X_d - X_q)(R_1^2 - X_dX_q) + (R_1^2 + X_dX_q)X_d] \quad (A2b)$$

$$T_{pb2} = \frac{mpV_{ph}^2}{4\pi f(R_1^2 + X_dX_q)^2} [(X_d - X_q)(X_dX_q - R_1^2)] \quad (A2c)$$

$$T_{pb3} = \frac{mpE_0V_{ph}R_1}{2\pi f(R_1^2 + X_dX_q)^2} [(R_1^2 + X_dX_q) - 2X_q(X_d - X_q)] \quad (A2d)$$

$$T_{pb4} = \frac{mpV_{ph}^2R_1}{4\pi f(R_1^2 + X_dX_q)^2} [(X_d - X_q)(X_d + X_q)] \quad (A2e)$$

## References

- De Almeida, A.T.; Ferreira, F.J.T.E.; Fong, J.A. Standards for Efficiency of Electric Motors. *IEEE Ind. Appl. Mag.* **2011**, *17*, 12–19. [[CrossRef](#)]
- Stoia, D.; Cernat, M.; Jimoh, A.G.; Nicolae, D. Analytical design and analysis of Line-Start Permanent Magnet Synchronous Motors. In Proceedings of the IEEE AFRICON Conference, Nairobi, Kenya, 23–25 September 2009; pp. 1–7. [[CrossRef](#)]
- Oshurbekov, S.; Kazakbaev, V.; Prakht, V.; Dmitrievskii, V.; Goman, V.V. Energy Efficiency Analysis of Fixed-Speed Pump Drives with Various Types of Motors. *Appl. Sci.* **2019**, *9*, 5295. [[CrossRef](#)]
- Palangar, M.F.; Soong, W.L.; Bianchi, N.; Wang, R.J. Design and Optimization Techniques in Performance Improvement of Line-Start Permanent Magnet Synchronous Motors: A Review. *IEEE Trans. Magn.* **2021**, *57*, 1–14. [[CrossRef](#)]
- Niaz Azari, M.; Mirsalim, M. Performance Analysis of a Line-start Permanent Magnet Motor with Slots on Solid Rotor Using Finite-element Method. *Electr. Power Compon. Syst.* **2013**, *41*, 1159–1172. [[CrossRef](#)]
- Mingardi, D.; Bianchi, N. FE-aided analytical method to predict the capabilities of line-start synchronous motors. In Proceedings of the 2014 IEEE Energy Conversion Congress and Exposition (ECCE), Pittsburgh, PA, USA, 14–18 September 2014; pp. 5123–5130. [[CrossRef](#)]
- Vannini, A.; Simonelli, C.; Marfoli, A.; Papini, L.; Bolognesi, P.; Gerada, C. Modelling, Analysis, and Design of a Line-Start Permanent Magnet Synchronous Motor. In Proceedings of the 2022 International Conference on Electrical Machines (ICEM), Valencia, Spain, 5–8 September 2022; pp. 834–840. [[CrossRef](#)]
- Rabbi, S.F.; Rahman, M.A. Critical Criteria for Successful Synchronization of Line-Start IPM Motors. *IEEE J. Emerg. Sel. Top. Power Electron.* **2014**, *2*, 348–358. [[CrossRef](#)]
- Chama, A.; Sorgdrager, A.; Wang, R.J. Analytical synchronization analysis of line-start permanent magnet synchronous motors. *Prog. Electromagn. Res.* **2016**, *48*, 183–193. [[CrossRef](#)]
- Yan, B.; Yang, Y.; Wang, X. A semi-numerical method to assess start and synchronization performance of a line-start permanent magnet synchronous motor equipped with hybrid rotor. *IET Electr. Power Appl.* **2021**, *15*, 487–500. [[CrossRef](#)]
- Zhou, Y.; Huang, K.; Sun, P.; Dong, R. Analytical Calculation of Performance of Line-Start Permanent-Magnet Synchronous Motors Based on Multidamping-Circuit Model. *IEEE Trans. Power Electron.* **2021**, *36*, 4410–4419. [[CrossRef](#)]
- Farooq, H.; Bracikowski, N.; La Delfa, P.; Hecquet, M. Modelling of Starting and Steady-State performance of Line Start Permanent Magnet Synchronous Motor using Reluctance Network. In Proceedings of the 2022 International Conference on Electrical Machines (ICEM), Valencia, Spain, 5–8 September 2022; pp. 226–231. [[CrossRef](#)]
- Palangar, M.F.; Mahmoudi, A.; Kahourzade, S.; Soong, W.L. Optimum Design of Line-Start Permanent-Magnet Synchronous Motor Using Mathematical Method. In Proceedings of the 2020 IEEE Energy Conversion Congress and Exposition (ECCE), Detroit, MI, USA, 11–15 October 2020; pp. 2064–2071. [[CrossRef](#)]
- Sorgdrager, A.J.; Wang, R.J.; Grobler, A.J. Multiobjective Design of a Line-Start PM Motor Using the Taguchi Method. *IEEE Trans. Ind. Appl.* **2018**, *54*, 4167–4176. [[CrossRef](#)]
- Hassanpour Isfahani, A.; Vaez-Zadeh, S. Line start permanent magnet synchronous motors: Challenges and opportunities. *Energy* **2009**, *34*, 1755–1763. [[CrossRef](#)]

16. Sorgdrager, A. Development of Line-Start Permanent Magnet Synchronous Machines Using the Taguchi Method. Ph.D. Thesis, Stellenbosch University, Stellenbosch, South Africa, 2017.
17. Tang, R.Y. *Modern Permanent Magnet Machines: Theory and Design*; China Machine Press: Beijing, China, 1997.
18. Miller, T.J.E. Synchronization of Line-Start Permanent-Magnet AC Motors. *IEEE Power Eng. Rev.* **1984**, PER-4, 57–58. [[CrossRef](#)]
19. Soulard, J.; Nee, H.P. Study of the synchronization of line-start permanent magnet synchronous motors. In Proceedings of the Conference Record of the 2000 IEEE Industry Applications Conference, Thirty-Fifth IAS Annual Meeting and World Conference on Industrial Applications of Electrical Energy (Cat. No. 00CH37129), Rome, Italy, 8–12 October 2000; Volume 1, pp. 424–431. [[CrossRef](#)]
20. Elistratova, V.; Hecquet, M.; Brochet, P.; Vizireanu, D.; Dessoude, M. Analytical approach for optimal design of a line-start internal permanent magnet synchronous motor. In Proceedings of the 2013 15th European Conference on Power Electronics and Applications (EPE), Lille, France, 2–6 September 2013; pp. 1–7. [[CrossRef](#)]
21. Jędryczka, C.; Knypiński, U.; Demenko, A.; Sykulski, J.K. Methodology for Cage Shape Optimization of a Permanent Magnet Synchronous Motor Under Line Start Conditions. *IEEE Trans. Magn.* **2018**, 54, 1–4. [[CrossRef](#)]
22. Honsinger, V.B. Permanent Magnet Machines: Asynchronous Operation. *IEEE Trans. Power Appar. Syst.* **1980**, PAS-99, 1503–1509. [[CrossRef](#)]
23. Behbahanifard, H.; Sadoughi, A. Line Start Permanent Magnet Synchronous Motor Performance and Design: A Review. *J. World's Electr. Eng. Technol.* **2015**, 4, 58–66.
24. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley Professional Computing Series; Pearson Education: London, UK, 1994.
25. ANSYS, Inc. *ANSYS Electronics Desktop: Maxwell Scripting Guide*; ANSYS, Inc.: Canonsburg, PA, USA, 2021.