Structural and Multidisciplinary Optimization (2022) 65:175
https://doi.org/10.1007/s00158-022-03263-4

INDUSTRIAL APPLICATION PAPER

Robust feasibility verification and region inner-point detection algorithms for geometric shape objects applied to electric machine optimization workflow

Branko Ban[1,2] · **Stjepan Stipetic**[1]

Received: 17 November 2021 / Revised: 28 February 2022 / Accepted: 26 April 2022 / Published online: 3 June 2022

Abstract
In most cases, the search for the optimal design of an electrical machine is closely related to its 2D radial cross section. When optimizing a 2D cross section, special attention must be paid to the geometry and to the definition of the parameters along with their boundaries. Even if properly bounded, complex geometries generated by optimization algorithms can lead to geometrically infeasible candidates. These cannot be manufactured because they contain generally undesirable geometric relationships between air, magnets, and steel. Different commercial and open-source finite element analysis (FEA) design tools treat the infeasible designs differently. The results vary from simulation stop to successful FEA calculation of the infeasible candidate, which wastes time by producing useless data. To prevent the infeasible designs from entering the optimization competition and possibly appearing incorrectly as optimal solutions, and to reduce optimization time, it is important to capture the infeasible designs during optimization. Moreover, the FEA tool requires a precisely determined interior point to assign the material to each closed region (air, steel, epoxy, magnet...). This can be very challenging for complex geometries. To avoid creating geometry or material regions that are not valid, this paper proposes a novel robust methods for checking feasibility and determining interior points on geometric shape objects. In this paper, the proposed method is applied to the optimization of electrical machines.

Keywords Electric machine · Design feasibility · Robust · Material · Region · Shape object

1 Motivation

Our research group is involved in the state-of-the-art design of electrical machines based on mathematical optimization. The presented methods have been proven in industrial applications and can be applied to any similar problem that can be defined with geometric shapes following the procedure described in the last sections of the paper. Considering the fact that this paper deals with a complex problem and proposes a solution that is relatively simple to implement, the reader is gradually introduced to several topics: electric machine optimization workflow, geometric feasibility, standard feasibility detection, region inner-point detection issues, and finally, the proposed robust set of solutions.

2 Introduction

The electrification of the transportation sector will be one of the biggest disruptions to market dynamics over the next two decades. This will create significant winners and notable losers among vehicle original equipment manufacturers (OEMs) (International Energy Agency 2020).

All the major OEMs have launched their electric vehicles (EVs) to the market. Despite many advantages, EVs are not cost competitive as conventional vehicles, which is the main requirement for large-scale market penetration (Sarlioglu et al. 2017; De La Parra et al. 2009). For this to become a reality, continuous research is required to develop optimized and cost-effective technologies in three main areas: battery, inverter, and electric machine.

Responsible Editor: Tae Hee Lee

✉ Branko Ban
branko.ban@fer.hr; branko.ban@torquery.com

[1] Department of Electric Machines, Drives and Automation, University of Zagreb, Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia

[2] Torquery Consulting, Göteborg, Sweden

In the design of an electrical machine, the objective is to reduce component cost while maintaining sufficient efficiency and torque density. Since these requirements are in conflict, modern electrical machine design techniques require a kind of mathematical optimization procedure (Bråmå and Enblom 1996; Zhao and Schofield 2020; Lee et al. 2021). This is particularly evident in the problem of increasing efficiency through global legislative initiatives (EU 2009; European Environment Agency 2016).

In most cases, the search for the optimal design of an electrical machine is closely related to its 2D radial cross-section. The increase in parametric complexity correlates with an increase in the performance of the optimal design, but also with a longer optimization time (Pellegrino et al. 2015; Lu et al. 2017) and a higher probability of producing a geometrically infeasible design. In addition, some FEA tools require a precisely determined interior point to assign the material to each closed region (air, steel, magnet...), which can be difficult to determine for complex geometries.

The paper provides a novel solution to both problems through a novel robust feasibility verification procedure and inner-point detection using Matlab polyshape objects. The generality of approach is demonstrated through a set of simple, but extremely robust algorithms. The approach elevates geometric design analysis from geometric primitives (points, lines and arcs) to the level of objects (shapes), applicable to any type of machine geometry as an upgrade to existing code.

3 Optimization methodology

Most of the requirements for the design of electric machines are in conflict with each other (reduction of volume or mass, increase of efficiency, etc.). Therefore, a manual design that satisfies all constraints can be an overwhelming task due to a large number of coupled parameters that affect the performance and quality of the machine. The solution is in the use of mathematical optimization.

Optimization algorithms can be divided into gradient based methods and stochastic or metaheuristic methods. Gradient-based methods converge quickly but have difficulty with global optima.

Usually they require feasible starting point which can be a problematic task in complex problems (Quasi Newton method (Kamper et al. 1996). Stochastic methods are heavily used in electrical machine optimization (Powell's method (Kamper et al. 1996). The disadvantage is that convergence can take days and the global optimum cannot be mathematically proven. On the other hand, from the engineer's point of view, these methods can find a satisfactory global result. Popular metaheuristic methods are based on natural behaviour (Genetic algorithm, Differential evolution (Lampinen 2002; Zarko et al. 2017), Particle swarm (Bramerdorfer et al. 2016), but they can also be iterative (Lee et al. 2010), or based on approximation (Lee and Jang 2021).

All methods are generally set to solve a multi-objective problem, which is mathematically defined as follows: find the vector of parameters (1), subject to D parameter boundary constraints (2) and subject to m inequality constraint functions (3), that will minimize (or maximize) n objective functions (4). A vector $\vec{x}$ of D variables specifies dimensions, dimensionless ratios, current densities, material types used, etc. The goal of design optimization is to have a chosen objective function $f(\vec{x})$ reach its minimum or maximum value while keeping other engineering indices within an acceptable range.

$$\vec{x} = [x_1, x_2, \dots, x_D], \qquad \vec{x} \in R^D \tag{1}$$

$$x_i^{(L)} \le x_i \le x_i^{(U)}, \quad i = 1, \dots, D \tag{2}$$

$$g_j(\vec{x}) \le 0, \quad j = 1, \dots, m \tag{3}$$

$$min(f_k(\vec{x})), \quad k = 1, \dots, n \tag{4}$$

3.1 Cross section parametrization

Each machine geometry is described by a set of parameters. The most influential design parameters that make up the vector $\vec{x}$ are usually identified using a sensitivity analysis tool. All parameters are bounded in predefined intervals, called boundary constraints, which define the search space or the design space.

Some authors use model parameters (stator bore diameter, stator slot depth, etc.) directly as optimization variables, while other authors use ratios of model parameters as optimization variables (Žarko et al. 2005; Zarko et al. 2017). A better approach (which can improve the efficiency of gradient-based algorithms) is to choose variables given as dimensionless ratios of related geometric parameters, e.g. ratio of stator inner diameter to stator outer diameter, ratio of tooth (or slot) width to slot pitch, magnet pole arc relative to pole pitch, etc.

Some parameters can just be considered as optimization variables directly within the given interval (stator outer diameter, stack length, slot current density). On the other hand, some parameters have practical limitations and should not be normalized. E.g. minimum permanent magnet width is tied to the magnet manufacturing process. Finally, a designer might choose a combination of normalized and model parameters which might create geometrical feasibility issues.

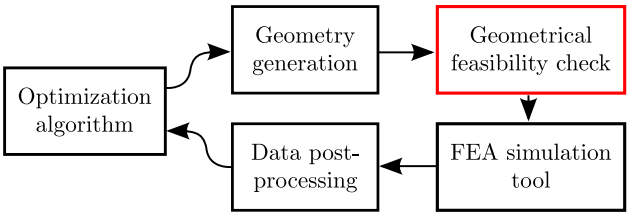


Fig. 1 FC as an optimization workflow add-on feature

3.2 Optimization workflow

A typical optimization workflow is a series of computations consisting of: optimization algorithm that generates optimization variables according to the given parameter boundaries, geometry generation block that passes the design to the selected FEA tool for performance calculations, and finally post-processing block that returns the computed data to the optimization algorithm for further evaluation (Fig. 1, black). Additionally, a geometric feasibility checking (FC) block can be part of the optimization workflow (Fig. 1, red).

3.3 Geometric feasibility

The term feasibility usually refers to the solution and means that the solution satisfies all given constraints. In other words, the region enclosed by $\forall g_j(\vec{x}) = 0$ is called a feasible region. This paper concentrates on another type of feasibility called *geometric or model feasibility*. Geometrically feasible model is valid for solving if there are no overlapping edges or non-conventional geometric relations. Infeasible candidate is the one that cannot be produced as a real machine because there are overlapping magnets, non-rectangular shaped magnets, air pockets overlapping magnets etc. Infeasible candidates should not be evaluated for performance because theoretically they can falsely appear as optimal solutions but can never be considered for manufacturing/production. It is important to note that geometric feasibility checking methods from the following sections are intended to be used only as an optimization procedure add-on (Fig. 1).

3.3.1 Optim. workflow without FC

If the optimization software does not include a feasibility check (FC), two different scenarios may occur during the FEA calculation (depending on the selected FEA tool).

In the first case, the FEA tool detects that the design is erroneous and throws an error. In this case, the designer must implement a try-catch procedure which will capture the event, otherwise the optimization procedure will fail (try-catch adds $t_{\text{extra}} \geq 4$s per design evaluation). In the second case, the FEA tool does not detect the infeasibility and a full FEA calculation is performed. This is the worst case scenario which will produce useless results and cause significant increase of optimization time (e.g., detailed transient electromagnetic simulations can take $t_{\text{extra}} \geq 15$min).

3.3.2 Optimization workflow with FC

However, if the optimization software includes a feasibility detection procedure, the designer can either attempt to correct the problematic instance or skip it. There are two typical ways to correct the optimization candidate prior to performance evaluation (passing it to the FEA calculation).

First, the entire set of optimization variables for that candidate is reinitialized randomly within the specified parameter bounds until geometric feasibility is achieved. This is a brute-force approach, but it is mandatory since evolutionary optimization algorithms (e.g. Genetic Algorithm, Differential Evolution) must have the same number of members in each population of optimization candidates. This is due to the fact that the each generation must be equal or better than the previous as the algorithm advances towards the optimum.

The alternative is forced feasibility, where each infeasible design is subjected to minimal parameter modification until feasibility is reached (e.g., the magnetic layer angle α_V is modified until w_{min} is reached, Fig. 2b). This approach can be complicated and requires an advanced parameterization with minimal feasibility constraints (Ban et al. , 2021).

Depending on the complexity of the design, both methods may increase execution time ($t_{\text{extra}} = 2 - 15$s). On the other hand, detecting and skipping the infeasible design actually does not impact execution time ($t_{\text{extra}} = 0$).

4 Standard approach for feasibility detection

The standard approach to feasibility detection is specific to each parameterized template, which typically requires immense programming/development resources for research groups engaged in electric machine design

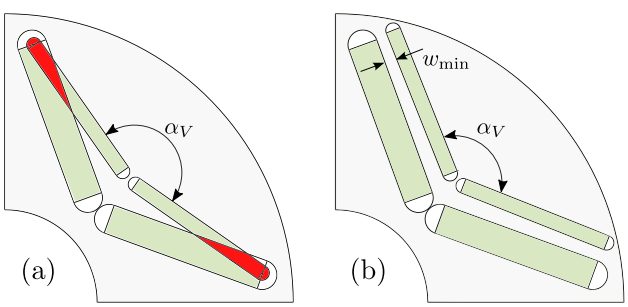


Fig. 2 Infeasible geometry (**a**) and forced feasibility (**b**)

optimization. Not only are top-level differences between templates relevant, e.g., a single-layer V-shape geometry compared to spoke interior permanent magnet machine (IPM), but also peculiarities such as air-pockets around magnets and curvature of posts and bridges have a significant impact on potential geometric feasibility. Modern high performance machines, especially in automotive traction, are all about these subtle design features. This means that an extensive study of the mathematical geometry dependencies has to be performed between arcs and lines that are either adjacent or have the possibility to touch or intersect during optimization process. All geometry primitives such as points, lines, arcs must be mathematically defined through coordinates or equations, ideally in both polar and Cartesian coordinate systems.

Depending on the definition of the optimization variables, this process can be fairly simple in some cases. For example, if the optimization variable for stator bore is defined as *ratio of stator bore diameter to stator outer diameter*

$$0.55 < D_{bore}/D_{outer} < 0.75 \tag{5}$$

then it is assured that the stator bore is always, as it should be, within the outer dimensions of the stator. A similar approach can be used for the height of the stator yoke - it must be shorter than the height of the stator annulus because both the slot and back iron must fit inside the stator. Therefore, this optimization variable is also defined as a ratio:

$$0.2 < 2h_{yoke}/(D_{outer} - D_{bore}) < 0.6 \tag{6}$$

Finally, the slot width or the tooth width, depending on the choice of stator slot type must be smaller than stator slot pitch in order to fit in the slot + the tooth inside one slot pitch.

However, when it comes to IPM rotors, especially multi-layered configurations as in Fig. 3, the mathematical description is much more complicated and, more importantly, it needs to be refined after each intervention in the template change. This is often the case in modern high performance automotive traction motors.

Typically, a single-layer, V-shape IPM rotor is described with a large number of rotor variables. The complexity can vary between different software packages or user defined templates in general-purpose FEA tools. In the case of commercial design tools, the minimum number of parameters to describe a layer is approximately 11. For multi-layer configurations, it increases linearly, so that a typical 2-layer V-shaped IPM traction motor (rotor shown in Fig. 3a) requires 22 variables just for magnets and barriers. If such a geometry is subjected to an optimization process to explore its full potential, or in other words, if the limits of the optimization parameters are set wide enough, there can be a large number of possible overlaps between air pockets, flux barriers, and magnets of a given layer or between different layers (Figs. 2, 3b–d).

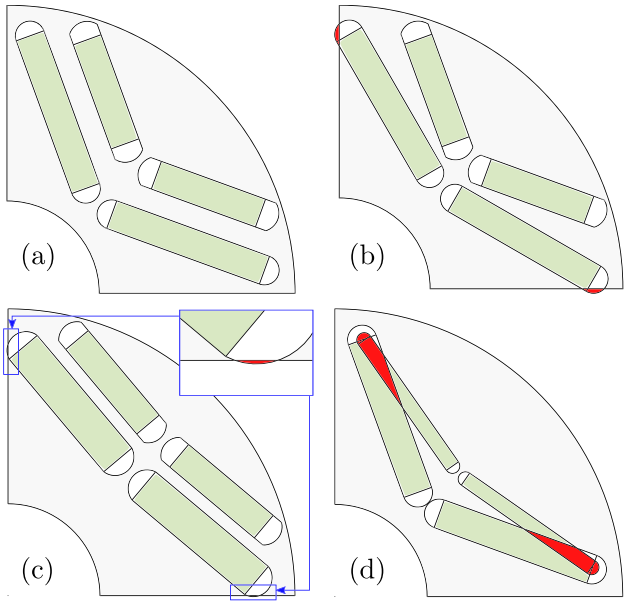


Fig. 3 Example of 2-layer V-shape IPM: **a** feasible case, **b** barrier definition point crossing polar lines, **c** barrier arc crossing polar lines, **d** barrier collision

The first step in the the standard approach would be to check whether any point (object vertex) is outside the rotor boundary: towards the stator and airgap, or inside the shaft. This is most easily achieved by checking the radial polar coordinates: any point with a radius greater than the outer radius of the rotor, or with a radius smaller than the inner radius of the rotor, means that the geometry is not feasible. A similar test exists to check if any point (object vertex) can be found outside the boundaries of each rotor pole (Fig. 3b). This is easily achieved by checking the angular polar coordinates: The angular coordinates should be within one pole. However, other tests, such as checking that barrier-limit arcs do not cross the interpolar line (Fig. 3c) and checking that there is no possible intersection between two layers (Fig. 3d), require hard-coding more complicated mathematical dependencies.

All of these calculations need to be changed, or at least reviewed, if the geometry slightly changes. The exact math can become considerably more complicated if barrier air pockets take stress relief shapes in the form of successive tangential arcs and ellipses, or if cooling channels and mass reduction holes are present in the geometry.

Algorithm 1 Feasibility verification example for 2-layer V-shape IPM geometry

```
function CHECK FEASIBILITY
    for all points do
        verify whether radial coordinates are inside rotor
        verify whether angular coordinates are inside one pole
    for all layers do
        verify whether x-coordinates of all points are sorted
        verify that arcs do not cross interpolar line
    verify layer disposition
    verify possible intersection between two layers
```

The procedures for feasibility verification are specific and different for each different geometry. The pseudocode for a 2-layer V-shape IPM is shown in Algorithm 1. It is difficult to tackle all possible geometry failure cases so sometimes the algorithm can let the infeasible geometry further down the optimization pipeline. Then it is up to the wrapper of the FEA solver to try and catch the error and ignore the solution. This invalid geometry can be examined later and the feasibility checking function can be further improved.

It can be seen that this method is not robust and needs to be modified even for small changes in the geometry features of the topology under study. Therefore, a novel feasibility handling algorithm is presented in the next chapter.

5 Robust feasibility detection

For optimization purposes, the cross-sectional geometry of the stator and rotor of the machine can be reduced to 2D Euclidean space. Geometry parameterization is performed in either Cartesian or polar coordinates, specifying all geometric primitives: Points, Lines, Arcs, and Polylines, which are sets of multiple lines that model a complex curve (polyline can be approximated with series of arcs and lines before sending a drawing *.dxf* to FEA solver). A set of connected primitives forms an object as a shape, such as a rotor barrier, which has mathematical properties: Area, Perimeter, Centroid etc. If the entire machine geometry can be defined by shapes - software objects, it would be very convenient to have some sort of functional tool to determine the correlation between different shapes. Correlations include Boolean functions, intersections, areas etc. This novel approach is made possible by the Matlab Polyshape class (Mathworks 2020), which enables elevation of primitives to an object (shape) level. Once created, the shape objects can be organized in the form of a vector. The polyshape functionality allows a quick analysis of vectorized shape objects. One of the functions is *Intersection*, which returns information when there is an intersection between members of the shape vector. This provides a robust feasibility check regardless of parameter boundaries and regardless of the complexity of the geometry shape. Considering that any electric machine topology can be reduced to geometric primitives, the proposed method can be implemented on all types of geometries as an upgrade of pre-existing code.

The main advantage of this approach is moving away from strict mathematical feasibility verification to shape object level which allows great design freedom, instead of relying on the rather complicated procedure described in the previous chapter (all functionality has been verified on Matlab releases after 2019b). Currently, there is no open-source code in Python or other interesting scripting languages that can provide the same functionality as Matlab polyshapes. Hovewer, this does not limit the presented novel approach. Considering the outlined benefits, open-source community will close this gap soon.

5.1 Polyshape function - details

Polyshape function creates a polygon defined by 2-D vertices and returns a polyshape object with properties describing its vertices, solid regions, and holes (Mathworks 2020). Essentially, polyshape is a structure that defines all vertices and holes within the 2-D shape. The syntax breakdown is given below (detailed variable description in Table 3).

$$P_{x,y} = Point(x_p, y_p) \tag{7}$$

$$L_{x,y} = Line(P_1, P_2) \tag{8}$$

$$A_{x,y} = Arc(P_{Or}, P_1, P_2) \tag{9}$$

$$M_{\text{dxf}} = \begin{pmatrix} L_1 & L_2 & \cdots & L_n \\ A_1 & A_2 & \cdots & A_m \end{pmatrix} \tag{10}$$

$$\vec{x}_s = [A_{3x}, L_{1x}, L_{3x}, A_{1x} \ldots] \tag{11}$$

$$\vec{y}_s = [A_{3y}, L_{1y}, L_{3y}, A_{1y} \ldots] \tag{12}$$

$$\text{poly} = polyshape(\vec{x}_s, \vec{y}_s) \tag{13}$$

$$\text{polyvec} = [\text{poly}_1, \text{poly}_2, \ldots] \tag{14}$$

$$C_{x,y} = Centroid(\text{poly}) \tag{15}$$

Some of the predefined functions for manipulating polyshape structures are listed in Table 1. For better

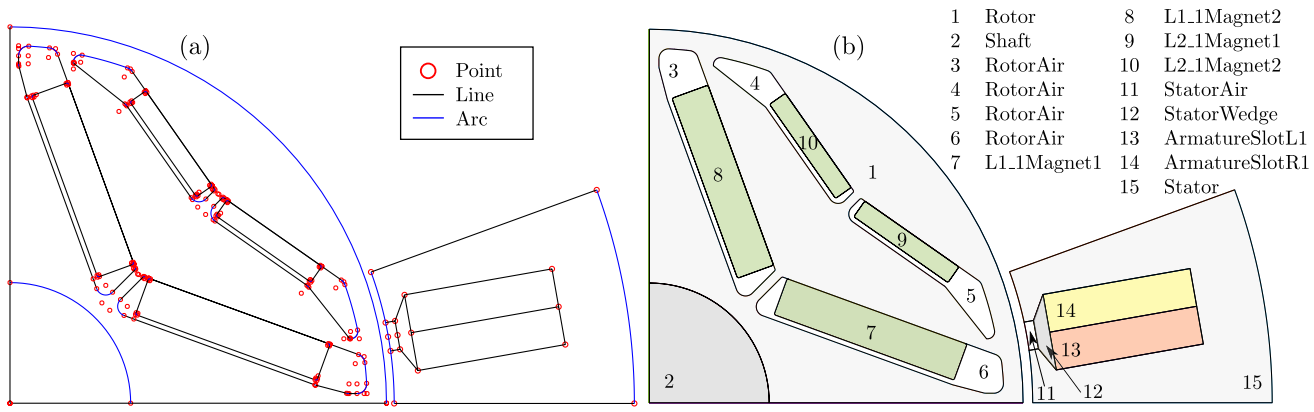


Fig. 4 All geometry points, lines and arcs (**a**); Machinecross section with all polyshapes (**b**)

Table 1 List of used primitives and polyshape functions

Function	Arguments	Return
Point	x_p, y_p	Point primitive
Line	Start point, end point	Line primitive
Arc	Origin, start, end points	Arc primitive
Polyshape	$\vec{x}_s, \vec{y}_s$	Shape object
Area	poly	Number
Centroid	poly	Centroid point ($C_{x,y}$)
Overlaps	polyvec	Intersection matrix
Polybuffer	poly, buffer	Polyshape ($\text{poly}_{\text{buff}}$)
Subtract	poly_1,	Polyshape (poly_{sub})
Union	poly_1, poly_2	Polyshape ($\text{poly}_{\text{union}}$)
Intersect	poly_1, poly_2	Polyshape ($\text{poly}_{\text{sect}}$)
Xor	poly_1, poly_2	Polyshape (poly_{xor})
Isinterior	poly, $P_{x,y}$	Boolean true or false

understanding, a pseudocode formulation is used in this paper, the functional source code is available in the (Ban 2021).

5.2 Geometry polyshape definition

The automated geometry parameterization (Fig. 4a) defines all rotor and stator point objects (7). Line objects are defined by two points (8), while arc objects are defined by origin, start and end points (9). When generated, line and arc objects contain corresponding vertex data. Shape $\vec{x}_s, \vec{y}_s$ vertex vectors are constructed by concatenating line and arc vertices in user-defined order (11), (12). Finally, $\vec{x}_s, \vec{y}_s$ are the arguments to the *polyshape* construction function (13). If there are multiple shapes (e.g., rotor of an electric machine), they can be vectorized (14), Fig. 4b.

All adjacent shapes are built from the same set of polygons with a high number of vertices, which means that there

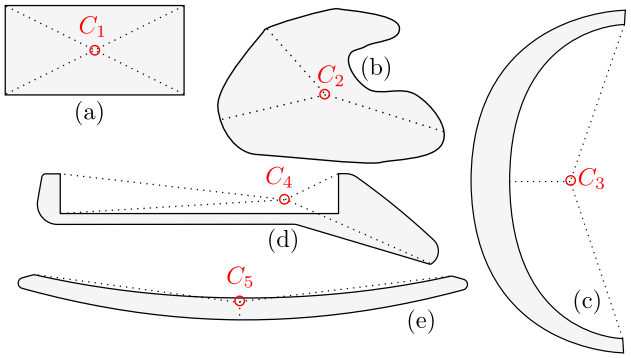


Fig. 5 Convex (**a**), general (**b**) and concave shapes (**c**, **d**, **e**)

will be no resolution problems in detecting intersections. It is important to note that when exporting the validated design for FEA calculation, the arcs do not consist of many polylines, but of *.dxf* primitives, so that the selected FEA software can optimise the arc meshing and speed up the calculation process. On the other hand, when assigning values to polyshapes, the arc and line vertices are passed as discrete vectors (11), (12). This means that the matrix of line and arc primitives (10), and the polyshape vector (14) are stored and processed separately.

5.3 Convex and concave shapes

All electromagnetic FEA tools require that a certain material is assigned to every closed region, except when boundary conditions handle holes in the model. In automated optimization procedures it is therefore necessary to define the shape region precisely, which can be a problem. The problem is even greater when complex geometries are used in the optimization of high performance machines.

In general, a generated shape can be either convex or concave (by default, shapes with holes are not allowed). A shape is said to be convex if, for any two points in the shape, the straight line segment between them lies entirely within the shape. Convex polygon centroid is always within the vertex envelope (Fig. 5a), which need not be the case for concave shapes (Fig. 5c–e). A typical example of a strictly convex shape is the permanent magnet rectangle (Fig. 5a), while the general shape may have "dents" (Fig. 5b).

On the other hand, air barriers are inherently concave because permanent magnets (PM's) are glued to the barrier surface and form separate shapes, thus creating a concave air pocket with an outer centroid (Fig. 5d). Furthermore, synchronous reluctance machine (SyRM) barriers are always concave, and depending on the curvature can have outer centroid. In the example of Fig. 5e, the centroid is within the shape limits, but very close to the edge, which can lead to problems in FEA calculations.

Overall, closed region inner-point detection by assigning the centroid point works well for convex shapes, but is unreliable for concave shapes.

5.4 Region inner-point detection

A robust region detection algorithm (Algorithm 2) was developed in combination with the existing *polyshape* functions and applied to all objects within the polyshape vector except magnets. Magnets are convex by default and require a specific magnetization direction determined by the GetMagnetAngle function (Algorithm 3).

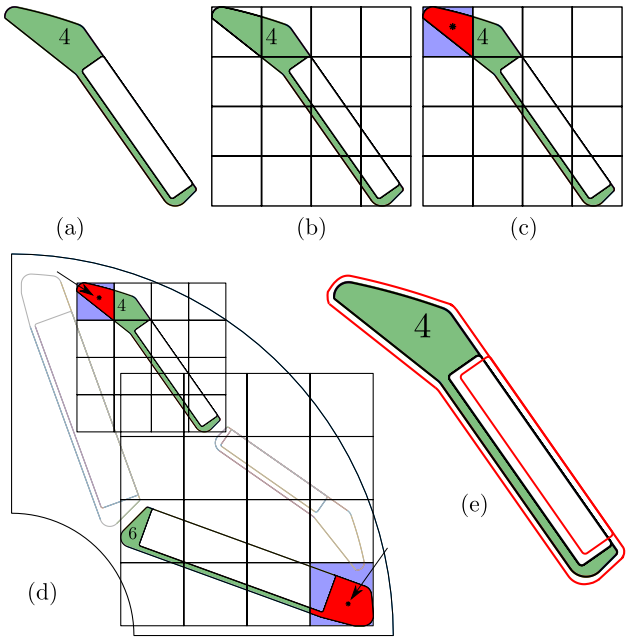


Fig. 6 Concave shapes robust region detection meshing example (**a**–**d**), functionality of *Polybuffer* function (**e**)

GetRegion function argument is polyshape structure poly_{in} (Fig. 6a green). poly_{in} $\vec{x}_s, \vec{y}_s$ vertex limits are extracted resulting in a 4x4 vectorized polyshape rectangle mesh (Algorithm 2:ln:6–7, Fig.6b). Each rectangle is intersected with poly_{in} to form an output vector of intersection points (Algorithm 2:ln:9, Fig.6b).The intersection element with the largest area is identified with the (Algorithm 2:ln:10, Fig.6c).

Algorithm 2 Region inner-point detection

```
1:  for i = 1 to numel(polyvec) do
2:      polyin = polyvec(i)
3:      if polyin != "Magnet" then
4:          function GetRegion(polyin)
5:              function GenerateMesh(polyin)
6:                  Calc. polyrec 1-16 based on polyin x,y limits
7:                  polyvecmesh = [polyrec1, ..., polyrec16]
8:                  return polyvecmesh
9:              polyvecsect = Interscest (polyvecmesh, polyin)
10:             polybig = find ( max ( Area (polyvecsect) ) )
11:             Cx,y = Centroid(polybig)
12:             if Isinterior(polyin, Cx,y) then
13:                 return Cx,y ▷ Centroid is the closed region
    inner point
14:             else
15:                 function GetRegion(polybig) ▷ Run again
16:     else
17:         function GetMagnetAngle(polyin, τ, αV)
18:             ...
19:             return Cx,y, αM
```

The next step is to compute the temporary inner-point calculation (Algorithm 2:ln:11). The point $C_{x,y}$ is checked to be inside the original poly_{in} with boolean function (Algorithm 2:ln:12). Finally, if $C_{x,y}$ is inside of the shape, it is returned as a region interior point, otherwise the procedure is repeated recursively on poly_{big} (Alg. 2:ln:15).

The proposed algorithm basically searches for a convex sector of the initial polyshape. Brute-force tests of the presented method prove that the inner-point of the region is always found within the shape boundaries, regardless of the generated geometry.

Polyshape mesh can be set to any rectangular dimension. Considering that the function execution time is proportional to the number of used mesh shapes, it is important to set the optimal mesh size. Different combinations of mesh sizes were tested on randomly generated concave shapes. In case of 2 × 2 mesh, there is a 40% probability that the inner region of concave shape is found in the first function

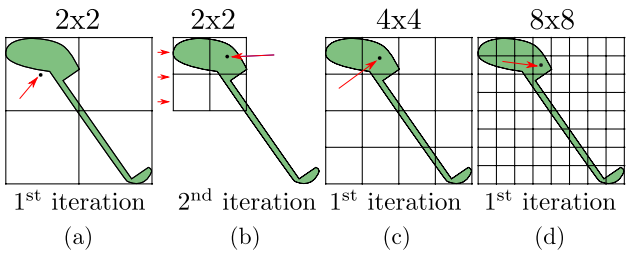


Fig. 7 Example of region detection mesh sizing

iteration (e.g. Fig 7a). The second iteration (e.g. Fig 7b) captures the inner region in 90% cases and sets the total number of mesh shapes to $2 \times 2 + 2 \times 2 = 8$ (1^{st} plus 2^{nd} iteration). On the other hand, when tested with 4×4 mesh, the inner region was in 90% cases found in the first iteration resulting in a total of $4 \times 4 = 16$ mesh shapes (e.g. Fig 7c).

Finally, 8×8 mesh finds the inner region in the first iteration in 98% cases, but with the expense of $8 \times 8 = 64$ shapes (e.g. Fig 7d). Compared with previous alternatives, the shape number is high and can potentially be doubled in nd iteration. Considering that 4×4 mesh has a rather high probability of finding the inner region in the first function iteration, 4×4 was selected as the optimal number of mesh shapes.

5.5 Magnetization direction

In addition to region inner-point, permanent magnets have magnetization direction (Fig. 8). The procedure for custom magnet shapes must be implemented individually. In this section, the procedure for V-shape rotor designs (Alg. 3) is explained. The magnet shape with the magnet angle parameter α_V and the pole step constant $\tau = \pi/p$ where p is the number of pole pairs, is passed to the function GetMagnetAngle. The function computes magnet centroid region point $C_{x,y}$ (Algorithm 3:ln:2) and the magnetization angle α_M. Algorithm 3:ln:6-12 and Fig.8 show the α_M calculation procedure which works when $\alpha_V \geq 180°$, and even in the case of square-shaped magnets.

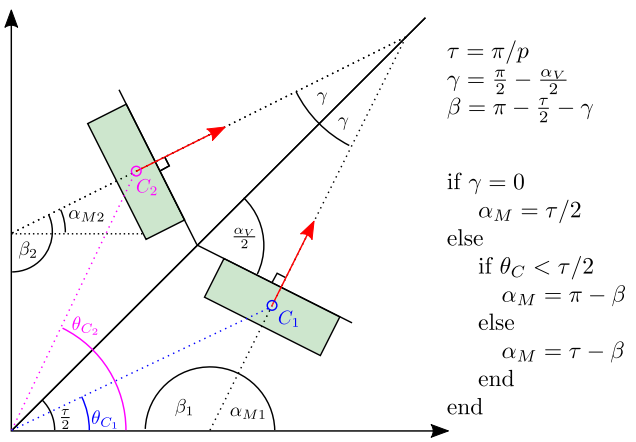


Fig. 8 Determining magnetization direction

Algorithm 3 Magnetization direction and region detection

```
1: function GetMagnetAngle(poly_in, τ, α_V)
2:     C_{x,y} = Centroid(poly_in)
3:     [θ_c, R_c] = cart2pol(C_{x,y})
4:     γ = (π − α_V)/2
5:     β = π − τ/2 − γ
6:     if γ == 0 then
7:         α_M = τ/2
8:     else
9:         if θ_C < τ/2 then
10:            α_M = π − β
11:        else
12:            α_M = τ − β
13:    return C_{x,y}, α_M
```

5.6 Feasibility check

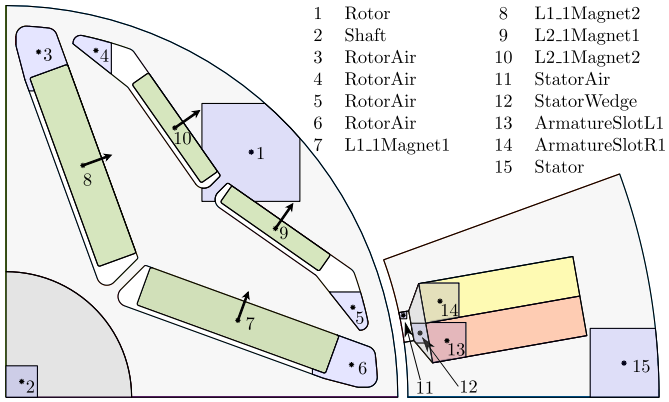
The complete machine cross section with all polyshapes and assigned material regions including the rectangular mesh intersections is shown in Fig. 9a. In this example, a total of 15 polyshapes are stored in a vector form (14). The polyshape vector is passed to the *overlaps* function which returns overlaps matrix M_{ovl} (Alg. 4:ln:3). If M_{ovl} is a unitary matrix (Algorithm 4:ln:4), the design is feasible and there are no overlaps (Fig.9a, b), otherwise the generated design is infeasible (Fig. 9c, d). If the user wants to include a limit on the distance between certain shapes (useful for detecting material elements that are too thin to be manufactured), the *Polybuffer* function can temporarily increase the borders of the shape during feasibility check (Algorithm4:ln:2, Fig. 6e). Additionally, the same function is used to avoid geometrical conflicts (11), (12) in borderline cases (e.g. two shapes are touching).

Algorithm 4 Robust feasibility check

```
1: function CheckFeasibility(polyvec)
2:     polyvec = polybuffer (selected polyvec elements, buffer)
3:     M_ovl = overlaps (polyvec)
4:     flag = isequal (M_ovl, eye ( size (M_ovl, 1) ) )
5:     if flag then
6:         disp ('Feasible geometry!')
7:     else
8:         disp ('Infeasible geometry!')
9:     return flag
```

5.7 Impact on total execution time

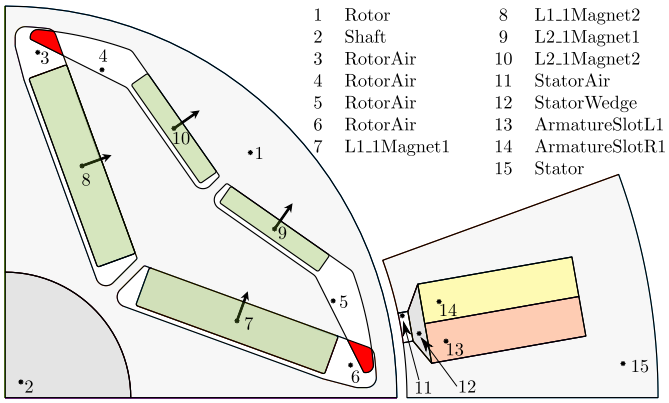
The feasibility and region detection procedure is performed only once during geometry generation, with a typical duration of 0.5–2 s. The transient calculation of a single operating point in ANSYS Motor-CAD for a two-layer IPM machine geometry typically takes between 30–60 s. If additional calculations are performed during the design



(a)

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(b)



(c)

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(d)

Fig. 9 Feasible (**a**, **b**) and infeasible geometry (**c**, **d**)

Table 2 FC impact on total optimization time

Run	1	2	3	4
Feasible designs	758	749	731	749
Infeasible designs	242	251	269	251
Total	1000	1000	1000	1000
Calc. time try-catch, no FC [h]	**64.0**	**63.2**	**61.7**	**63.2**
Calc. time with FC [h]	*63.6*	*62.8*	*61.3*	*62.8*
Calc. time worst case [h]	83.3	83.3	83.3	83.3
Time save, try-catch to worst case [%]	23.2	24.1	25.9	24.1
Time save, FC to worst case [%]	23.6	24.6	26.4	24.6

evaluation (thermal transients, drive cycle analysis, etc.), the cumulative time can increase to 10–15 min.

As an experiment, a sensitivity analysis with 4 consecutive runs of 1000 iterations was performed for a well-defined two-layer V-shape rotor geometry with 44 parameters (Table 2). The design space has been specified by defining upper and lower bounds of all design parameters. A set of parameters for each design has been generated by scanning the multidimensional space via Space filling Latin Hypercube sampling method using Ansys OptiSLang surrogate modelling tool. The average design evaluation time is 300 s (5 min). Using try-catch adds 4 s (304 s per design evaluation), while using feasibility check adds 2 s (302 s per design evaluation).

Let us first consider the worst-case scenario without try-catch or FC (FEA tool runs the entire simulation). This approach leads to a maximum duration of 83.3 h per run. On the other hand, the try-catch procedure detects all possible errors, leading to an average execution time of 63.1 h (here we assume the scenario where the FEA tool outputs an error for each infeasible design). Finally, using FC results in an average execution time of 62.6 h.

As expected, the try-catch and FC procedures result in virtually the same execution time and a significant overall time savings (25% or 20 h shorter execution time). The

conclusion is that the use of FC has no impact on the total execution time. Unlike the try-catch procedure, FC can provide the information about the colliding regions and parameters that can be exploited in surrogate (metamodel) optimization, or tools such as Ansys OptiSLang (Riviere et al. 2019, 2020).

6 Conclusion

Optimization is an important and inevitable part of the modern electric machine design process. When properly applied, optimization leads to a design that satisfies all imposed requirements.

Even when properly constrained, complex geometries generated by optimization algorithm can lead to infeasible designs that in worst case increase optimization time or propagate non-manufacturable candidates. In addition, the accurate detection of interior points for each closed region in order to assign material can be a serious challenge.

The standard procedure for solving these issues is based on hard coding of simple or complex mathematical relationships between geometric elements. This method tends to be inflexible to any major changes and leads to a complex geometry code. The paper provides a solution to both problems through a novel robust feasibility verification procedure and inner-point detection using Matlab polyshape objects. The generality of the approach allows application in any other script language. The polyshape approach elevates geometric design analysis from geometric primitives (points, lines and arcs) to the level of objects (shapes), applicable to any type of machine geometry as an upgrade to existing code. In addition to geometrical properties (vertex coordinates), elevation to shapes allows severely simplified surface, mass and Boolean calculations. When implemented, the method represents a paradigm shift in electric machine design.

The main benefits of robust feasibility checking are: preventing infeasible (non-manufacturable) candidates to be propagated or to win in the optimization competition, gathering information about which shapes and parameters cause problems and finally, greater freedom in defining geometry parameter boundaries when describing complex geometries.

Appendix

See Tables 3 and 4.

Table 3 Complete list of variables and functions

Variable	Description
α_M	Magnetization direction according to Fig. 8
α_V	V angle of the the rotor magnets (Fig. 8)
β	Angle according to Fig. 8
γ	Angle according to Fig. 8
τ	Angle of one pole step according to Fig. 8
x_p	Individual point x coordinate
y_p	Individual point y coordinate
$\vec{x}_s$	Shape x vertex vector
$\vec{y}_s$	Shape y vertex vector
$P_{x,y}$	Point primitive defined by x_p, y_p
$L_{x,y}$	Line primitive defined by two point objects
$A_{x,y}$	Arc primitive defined by three point objects
$C_{x,y}$	Polyshape centroid point
θ_C	$C_{x,y}$ angular coordinate
R_C	$C_{x,y}$ radial coordinate
D_{bore}	Stator bore diameter
D_{outer}	Stator outer diameter
h_{yoke}	Stator yoke height
e_{in}	Inner barrier eccentricity
e_{out}	Outer barrier eccentricity
M_{ovl}	Matrix of shape overlaps
M_{dxf}	Matrix of line and arc primitives
p	Number of pole pairs
$w_{\mathrm{c\ min}}$	Minimum flux carrier width
.dxf	Drawing Interchange Format
GenerateMesh	Function that generates 4x4 polyshape mesh
GetMagnetAngle	Magnetization angle calculation function
GetRegion	Polyshape inner region point calc. function
Polyshape	Polyshape constructor function
poly	Polyshape structure defined by $\vec{x}_s, \vec{x}_y$
polyvec	Vector of polyshapes
$\mathrm{poly}_{\mathrm{big}}$	Polyshape with the biggest area in $\mathrm{polyvec}_{\mathrm{sect}}$
$\mathrm{poly}_{\mathrm{in}}$	Input polyshape vector
$\mathrm{poly}_{\mathrm{rec1-16}}$	16 mesh polyshapes
$\mathrm{polyvec}_{\mathrm{mesh}}$	Vector containing all mesh polyshapes
$\mathrm{polyvec}_{\mathrm{sect}}$	Intersect. between the $\mathrm{polyvec}_{\mathrm{mesh}}$and $\mathrm{polyvec}_{\mathrm{in}}$

Table 4 List of abbreviations

Abbreviation	Description
EV	Electric vehicle
FC	Feasibility check
FEA	Finite element analysis
IPM	Interior permanent magnet
OEM	Original equipment manufacturer
PM	Permanent magnet
SyRM	Synchronous reluctance machine

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00158-022-03263-4.

Acknowledgements This work was supported by Motor Design Limited.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Replication of results Minimum example Matlab code has been published for open-access on GitLab site (Ban 2021).

References

Ban B, Stipetic S, Jercic T (2021) Minimum set of rotor parameters for synchronous reluctance machine and improved optimization convergence via forced rotor barrier feasibility. Energies 14(10):16. https://doi.org/10.3390/en14102744 (**ISSN 19961073**)

Branko B (2021) A minimum code example: robust feasibility verification and region inner-point detection algorithms for geometric shape objects in e-machine optimization, https://bit.ly/3sPutFo

Lee C, Lee J, Jang In G (2021) Shape optimization-based design investigation of the switched reluctance motors regarding the target torque and current limitation. Struct Multidisc Optim 64(2):859–870. https://doi.org/10.1007/s00158-021-02897-0

Lu C, Ferrari S, Pellegrino G (2017) Two design procedures for PM synchronous machines for electric powertrains. IEEE Trans Transp Electrif. https://doi.org/10.1109/TTE.2016.2646738

De La Parra HZ, Freddy M, Sjoerd B (2009) Challenges for electric machines and power electronics in automotive applications. International Conference on Ecological Vehicles and Renewable Energies,(EVER'09), pp 1–9

EU (2009) Commission Regulation (EC) 640/2009 Implementing directive 2005/32/EC of the european parliament and of the council with regard to ecodesign requirements for electric motors. Offic J Europ Union (640): L 191/26

European Environment Agency (2016) Electric Vvhicles in Europe. ISSN 1557-170X. https://www.eea.europa.eu/publications/electric-vehicles-in-europe

Gerd B, Ciprian ZA, Siegfried S, Edwin L, Wolfgang A (2016) Possibilities for speeding up the FE-based optimization of electrical machines-a case study. IEEE Transactions on Industry Applications. ISSN 00939994. https://doi.org/10.1109/TIA.2016.2587702

Pellegrino G, Cupertino F, Gerada C (2015) Automatic design of synchronous reluctance motors focusing on barrier shape optimization. IEEE Trans Indus Appl. https://doi.org/10.1109/TIA.2014.2345953

International Energy Agency (2020) Global EV outlook 2020. URL https://www.iea.org/reports/global-ev-outlook-2020

Lee J, Jeong HS, Kikuchi N (2010) Topology optimization of switched reluctance motors for the desired torque profile. Struct Multidisc Optim 42(5):783–796. https://doi.org/10.1007/s00158-010-0547-1

Jouni L (2002) Multi-constrained nonlinear optimization by the differential evolution algorithm. In: Soft Computing and Industry. Springer-Verlag London. https://doi.org/10.1007/978-1-4471-0123-9_26

Kamper MJ, Van Der Merwe FS, Williamson S (1996) Direct finite element design optimisation of the cageless reluctance synchronous machine. IEEE Trans Energy Convers 11(3):547–553. https://doi.org/10.1109/60.537006 (**ISSN 08858969**)

Lee C, Jang In G (2021) Topology optimization of multiple-barrier synchronous reluctance motors with initial random hollow circles. Struct Multidisc Optim 64(4):2213–2224. https://doi.org/10.1007/s00158-021-02976-2 (**ISSN 16151488**)

Mathworks (2020) Matlab polyshape function. https://www.mathworks.com/help/matlab/ref/polyshape.html

Nicolas R, Giuseppe V, Marco V, Giuseppe F, Di Leonardo L, Mircea P (2019) Design analysis of a high speed copper rotor induction motor for a traction application. In: (2019) IEEE International Electric Machines and Drives Conference, IEMDC 2019. https://doi.org/10.1109/IEMDC.2019.8785124

Nicolas R, Markus S, James G (2020) An innovative multi-objective optimization approach for the multiphysics design of electrical machines. In: 2020 IEEE Transportation Electrification Conference and Expo, ITEC 2020, pp 691–696. https://doi.org/10.1109/ITEC48692.2020.9161650

Bulent S, Casey TM, Di H, Li S (2017) Driving toward accessibility: a review of technological improvements for electric machines, power electronics, and batteries for electric and hybrid vehicles. IEEE Ind Appl Mag 23(1):14–25. https://doi.org/10.1109/MIAS.2016.2600739 (**ISSN 10772618**)

Torsten B, Roger E (1996) Extensions to structural optimization software to enable multidisciplinary applications. In: 6th Symposium on Multidisciplinary Analysis and Optimization, (X): 1202–1211, https://doi.org/10.2514/6.1996-4168

Zarko D, Stipetic S, Martinovic M, Kovacic M, Jercic T, Hanic Z (2017) Reduction of computational efforts in finite element-based permanent magnet traction motor optimization. IEEE Trans Industr Electron 65(2):1799–1807. https://doi.org/10.1109/TIE.2017.2736485 (**ISSN 02780046**)

Zhao N, Schofield N (2020) An induction machine design with parameter optimization for a 120-kW electric vehicle. IEEE Trans Transp Electrif 6(2):592–601. https://doi.org/10.1109/TTE.2020.2993456 (**ISSN 23327782**)

Žarko D, Drago B, Lipo TA (2005) Design optimization of Interior Permanent Magnet (IPM) motors with maximized torque output in the entire speed range. In: 2005 European Conference on Power Electronics and Applications. https://doi.org/10.1109/epe.2005.219269