JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

TNF

Technisch-Naturwissenschaftliche
Fakultät

Shape Optimization based on Black Box Simulations

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplomingenieurin

im Masterstudium

Industriemathematik

Eingereicht von:
Monika Kowalska

Angefertigt am:
Institut für Numerische Mathematik

Beurteilung:
A. Univ.-Prof. Dipl.-Ing. Dr. Walter Zulehner

Linz, Mai 2010

Abstract

Shape optimization is very important for many industrial applications. The typical problem is to find the optimal shape of a machine such that it minimizes a certain cost functional while satisfying given constraints, see [1].

The practical problems in this thesis are given by the ACCM (Austrian Center of Competence in Mechatronics). The task is to determine the design of an electric drive such that a cost functional is minimized. Due to the fact that the analytic definition of the cost functional is not available but all function values can be computed, this thesis considers optimization by black box simulations.

The model problem in this thesis is firstly considered with two design parameters and it is described in detail. The mathematical background for the optimization is presented. It is mainly based on the book *Practical Optimization* by Philip E. Gill, Walter Murray and Margaret H. Wright. The optimization programs are written in MATLAB. This thesis presents different numerical results by using a certain MATLAB routine. Moreover, results are shown with the availability of parallel processors and it is presented how this would reduce the number of function evaluations in each optimization step. A proposal is made at the end of the consideration of the problem with two design parameters, how the whole procedure of optimizing the motor via black box simulations can be automatized.

Finally, the gained knowledge is applied on real application problems with more design parameters given by the ACCM and the results of the optimization are presented. These results are compared to the results of a different optimization method, a genetic algorithm from [7].

Zusammenfassung

Formoptimierung ist sehr wichtig für viele Anwendungen in der Industrie. Ein typisches Problem der Formoptimierung ist es die optimale Form einer Maschine zu finden, sodass ein bestimmtes Gütefunktional mit gegebenen Nebenbedingungen minimiert wird.

Die Anwendungsprobleme in dieser Diplomarbeit werden vom ACCM (Austrian Center of Competence in Mechatronics) bereit gestellt. Das Ziel ist es das Design eines elektrischen Motors zu bestimmen, sodass ein Gütefunktional minimiert wird. Aufgrund der Tatsache, dass eine analytische Definition des Gütefunktionals nicht verfügbar ist, aber alle Funktionswerte berechnet werden können, beschäftigt sich diese Diplomarbeit mit der Optimierung mittels Black Box Simulationen.

Das Modellproblem dieser Arbeit wird zunächst mit zwei Designparametern detailliert betrachtet. Der mathematische Hintergrund für die Optimierung wird präsentiert, welcher sich hauptsächlich auf das Buch *Practical Optimization* von Philip E. Gill, Walter Murray und Margaret H. Wright stützt. Die Programme für die Optimierung sind in MATLAB programmiert. Diese Diplomarbeit liefert verschiedene numerische Ergebnisse unter Verwendung einer bestimmten MATLAB-Routine. Es werden auch Ergebnisse präsentiert, die mit Hilfe von parallelen Prozessoren berechnet wurden und es wird gezeigt, wie dies die Anzahl der Funktionsauswertungen in jedem Optimierungsschritt reduziert. Am Ende der Betrachtung des Problems mit zwei Designparametern wird ein Vorschlag dazu gegeben, wie das komplette Optimierungsverfahren mittels Black Box Simulationen automatisiert werden kann.

Schlussendlich wird das erreichte Wissen an Anwendungsproblemen vom ACCM mit noch mehr Designparametern angewendet und die Ergebnisse der Optimierungen werden präsentiert. Diese werden mit Ergebnissen einer anderen Optimierungsmethode, einem genetischen Algorithmus aus [7], verglichen.

Acknowledgements

Firstly, I want to thank my supervisor Prof. Dr. Walter Zulehner for his support and the guidance he gave me during my writing of the diploma thesis. I am also very thankful for the time he spent on helping me and answering all my questions.

I also want to thank Prof. Dr. Ulrich Langer from the Institute of Computational Mathematics for his friendly support. Moreover, special thanks go to Prof. Dr. Wolfgang Amrhein from the Institute of Electric Drives, who gave me the possibility of writing my diploma thesis on this very interesting topic, and to the ACCM (Austrian Austrian Center of Competence in Mechatronics), which enabled everything. At this, I also want to thank my colleagues from the LCM (Linz Center of Mechatronics) and the members of the Institute of Electric Drives, who have delivered me insight into the field of electric drives. It is a fascinating field, what has made my work on this topic much more easier and has grabbed my attention. Especially, I want to thank Dr. Siegfried Silber, DI Günther Weidenholzer, DI Werner Koppelstätter and Ing. Markus Aigner for all their friendly support and the time they have spent on teaching me the basics of electric drives, helping me with understanding and applying the software needed for my optimization programs and much more. Furthermore, I want to thank DI Walter Bauer, DI Florian Poltschak and DI Gerd Bramerdorfer for their constructive suggestions.

Finally, I want to especially thank my family for their mental and physical support all the years. My parents are really wonderful people, in whom I can always trust and who have supported me during my studies. I also want to thank my grandmother, who has helped me as good as possible, and all my friends, whom I could always rely on. Thanks to my fiancé Sebastian for all his encouragement and care during the last years.

This work has been carried out at the Institute of Computional Mathematics, JKU Linz, in cooperation with ACCM (Austrian Center of Competence in Mechatronics), a K2-Center of the COMET/K2 program of the Federal Ministry of Transport, Innovation, and Technology, and the Federal Ministry of Economics and Labour, Austria.

Contents

List of Figures

Chapter 1

Introduction

Optimization has always been a very important task in research and development. There are various approaches to solve optimization problems in theory as well as in application. Especially, shape optimization occurs in many application problems.

1.1 Shape Optimization

A typical shape optimization problem is to determine the optimal shape such that a certain cost functional is minimized while satisfying given constraints, see [1]. A shape optimization problem can be formulated as

$$\min_{x \in \mathbb{R}^n} f(x, u(x)) \quad \text{such that } \underline{x_i} \leq x_i \leq \overline{x_i} \quad \forall i \in \{1, 2, ..., n\},$$

where $f(x, u(x))$ is the cost functional, $u(x)$ solves an elliptic partial differential equation and x is the vector of n design parameters.

In the following, the goal of the application problems in this thesis will be to determine the optimal design of an electric motor, which minimizes a certain cost functional and satisfies some given constraints. The cost functional $f(x, u(x))$ describes mainly the power output of the electric motor, and the elliptic partial differential equation, which is solved by $u(x)$, characterizes the electromagnetic behavior. The real application problems in this thesis occur in the field of electrical engineering and are provided by the ACCM (Austrian Center of Competence in Mechatronics), where the group for electric drives has developed a special tool for the simulation of electric motors.

The optimization problems in the field of electric motors lead to cost functionals which are very complex and consist of many different factors. They can also depend on many different parameters. However, the function values are computed by simulation and due to that they are available, but

the exact analytical definition of the cost functional is not given. Therefore, one natural approach is to use a black box method to optimize the shape of the electric motor.

1.2 Black Box Optimization

Black box optimization is useful for problems, where the analytical definition of the cost functional is unknown, but the function values can be computed for all relevant parameter values. Black box methods are approximative methods.

The black box in this thesis is represented by a certain MATLAB routine, which returns function values corresponding to given parameters. Using black box optimization yields various problems and difficulties, such as the need of derivative information of the cost functional. In this thesis, the MATLAB routine *fmincon*, a built-in function of the MATLAB Optimization Toolbox, is used for optimizing the cost functional. The main idea of the MATLAB routine *fmincon* is explained in the next section.

1.3 Optimization with Matlab

The MATLAB command *fmincon* is used to solve constrained nonlinear optimization problems, see [2]. It finds the minimum of a nonlinear scalar function f, which can be multivariable, subject to given constraints. In the considered problems, the given constraints are box constraints. Lower and upper bounds have to be specified to determine the variable range. More precisely in this thesis, *fmincon* is used to find the minimum of a problem specified by

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{such that } \; lb \leq x \leq ub,$$

where x, lb and ub are vectors. The command *fmincon* tries to find iteratively a minimum starting at an initial estimate. Therefore, the user has to provide a starting value $x0$ or it is automatically suggested in the middle of the box constraints.

The MATLAB routine *fmincon* uses a gradient based method. The optimization requires that information about derivatives of the cost functional have to be available. Therefore, the user has the possibility to supply g, the

gradient of $f(x)$,

$$g = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

or to supply g and H, the Hessian of $f(x)$, which is

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}.$$

If this information is not provided, gradient and Hessian are approximated by MATLAB.

In real application problems, the cost functional f is very complex and its analytical definition is not available. The function values are computed by black box simulations. For the optimization, three MATLAB files have been written. In the main file named **OptimizationQualityFunction.m** the routine *fmincon* is called. Here, the box constraints, options and the initial value of the iterative method are specified. The cost functional named **fquality.m** has to be called in *fmincon*. The MATLAB function *fmincon* is of the form

[x,fval,...] = *fmincon*(@fquality, x0, [ ], [ ], [ ], [ ], lb, ub, [ ], options),

where the left hand side specifies the output.

In fquality.m, the function values of the cost functional are computed. Here, it is possible to supply the gradient or the Hessian as well. We do this by approximating the derivatives by difference quotients. The thesis presents various cases by supplying derivative information of the cost functional by the user. If no derivative information is supplied, MATLAB computes the gradient and the Hessian approximatively. Analogously, if only the gradient is supplied by the user, then MATLAB approximates the Hessian internally.

The function values of the cost functional are called by the function **simulate.m**, the black box. This function, simulate.m, computes the function values depending on the values of the parameters. Arbitrarily many parameters can be chosen, depending on the model problem. For the application

problems in this thesis, the engineers of the ACCM limit the number of design parameters to ten parameters.

Important optimization algorithms in *fmincon*

- Active-Set Optimization:
  A sequential quadratic programming (SQP) method is used. In this method, a quadratic programming (QP) subproblem is solved at each iteration. Using the BFGS-formula (BroydenFletcherGoldfarbShanno-formula), *fmincon* updates an estimate of the Hessian at each iteration. This method is used in *fmincon* for computing the optimum, if no information about the derivatives is provided by the user, see [2].

- Trust-Region-Reflective Optimization:
  This method is based on the interior-reflective Newton method. At each iteration, the solution of a linear system using the method of preconditioned conjugate gradients (PCG) is approximatively computed. This method is used, if the gradient is supplied and box constraints are specified by the user. In general, this method is most effective when the Hessian is computed as well, see [2].

The organization of the thesis

- Chapter 2:
  The thesis deals with a model problem which provides a first insight into the shape optimization of an electric motor. In chapter 2, this model problem is explained and also the to be optimized parameters of the cost functional.

- Chapter 3:
  The numerical background for the optimization is shown. An important part will be the approximation of gradient and Hessian by difference quotients. Moreover, a method will be considered how to find the "optimal" stepsizes for the difference quotients.

- Chapter 4:
  The model problem from chapter 2 with two parameters is considered. This problem shows the main challenges of optimizing by a black box method. One of them is that derivative information is needed. It is possible for the user to supply the MATLAB routine *fmincon* with derivative information. Therefore, numerical results of many different cases are compared and it is shown in practice how to choose "optimal" stepsizes for the difference quotients. It is explained what "optimal" in this case means and which strategies for optimizing by black box simulations are in general most efficient. There are possibilities how

to speed up the optimization by parallelizing the optimization such that in one optimization step the function evaluations are computed by multiprocessor systems. At the end of chapter 4, it is explained how the whole optimization process can be automatized.

- Chapter 5:
  The model problem is extended such that the motor is optimized with respect to five parameters. Here, the additional parameters are explained and the gained knowledge of the problem with two parameters is applied on the one with five parameters. This leads to numerical results, which are compared to the results obtained by optimizing with a genetic algorithm from [7].

- Chapter 6:
  The final problem is our model problem with eight parameters. By applying the successful machinery, which one gets from the preceding problems, especially from the problem with two parameters, it is finally revealed whether the theoretical and well tested practical results hold true for real application problems or not.

- Chapter 7:
  The conclusions and an outlook of the thesis are presented.

Chapter 2

The Model Problem

Electric motors are divided into Alternating Current (AC) types and Direct Current (DC) types. The motor considered in the model problem is an AC motor. More precisely, it is a permanent magnetic synchronous machine (PMSM). A PMSM has two characteristics. Firstly, it is a synchronous machine. Secondly, it is permanent magnetic excited. But what does these all mean?

Synchronous machine: An electric synchronous motor is an AC motor whose rotational frequency equals the air gap field frequency, i.e. the magnetic field rotates at the same speed as that created by the field current. From that comes the name synchronous motor. Ideally, a steady torque results. The air gap between rotor and stator is often larger in synchronous machines as in asynchronous. This provision is made for the protection of the permanent magnets from too large short-circuit currents, see [3].

Permanent magnetic excited: Permanent magnetic implies that the exciting field of the rotor is generated by permanent magnets. In a PMSM, the windings are located in the stator.

In the motor simulation, the magnetostatic problem is solved for every rotor position relative to the stator depending on the angle. This is done twice, once without current ($I = 0$), which is called the no-load case, and once with current ($I \neq 0$), called the load case, and it is done through 360° or less depending on the number of pole pairs. As it will be shown later on, the motor model has 2 pole pairs. Due to that it has to be done only through 180°. For every angle of the 180°, the motor is at rest and the torque and other parameters, which will be explained in the following, are computed. Moreover, either the whole motor or only half of it is simulated. This depends on whether the motor shows field symmetries or not. If the motor is field symmetric, then only half of it has to be simulated. Whether the

motor is symmetric or not depends on the design parameters. Concretely, in the problem with 2 design parameters the motor is symmetric. Therefore, only half of the motor has to be simulated. In the problems with 5 and 8 design parameters, the whole motor has to be simulated because of some of the additional parameters.

Very important for the cost functional is the torque and the power output of the machine. Torque, also called moment of force, is the tendency of a force to rotate an object about an axis. The tangential shearing force is essential for the torque. A torque can be thought of as a twist. Basically, the torque measures how strong something is rotated. The torque τ on a particle, which has the position r in some reference frame, is defined as

$$\tau = r \times F,$$

where F is the force acting on the particle, see [1].

Torque is very important for the description of an engine. It is part of its basic specification. Namely, the power output P of an engine is expressed as the scalar product of its torque and its angular speed ω

$$P = \tau \cdot \omega.$$

Power is the work per unit time. Power injected by the torque depends only on the current angular speed, not on whether the angular speed increases, decreases, or remains constant while the torque is being applied. This is equivalent to the linear case where the power injected by a force depends only on the current speed and not on the resulting acceleration, see [1].

It is important to use consistent units. The SI unit of power is watt, the one of torque is newton meter and that of angular speed is radians per second. The unit newton meter is dimensionally equivalent to joule, which is the unit of energy. In the case of torque, the unit is assigned to a vector. In the case of energy, it is assigned to a scalar, see [1].

A conversion factor must be inserted into the equation because of the different units of power, torque and angular speed. If rotational speed n (revolutions per time) is used in place of angular speed ω (radians per time), then n has to be multiplied by 2π such that it equals ω. This follows from the fact that there are 2π radians in a revolution. Therefore, power can be described by

$$P = \tau \cdot 2\pi n,$$

according to [1].

The balance of power classifies an AC machine. In [3] the following equations are described in detail. For a synchronous machine, the power P_{el}, which is electrically conducted to the stator, consists of the mechanical power P_{mech}, the stator losses P_{loss}^{stator} and of the rotor losses P_{loss}^{rotor}, i.e.

$$P_{el} = P_{mech} + P_{loss}^{stator} + P_{loss}^{rotor}.$$

A part of this sum is called the air-gap power P_δ. It consists of the mechanical power and the losses of the rotor, i.e.

$$P_\delta = P_{mech} + P_{loss}^{rotor}.$$

Hence, it follows for the electrically conducted power that

$$P_{el} = P_\delta + P_{loss}^{stator}.$$

The appearing stator losses consist of copper losses P_{loss}^{Cu} and iron losses P_{loss}^{Fe},

$$P_{loss}^{stator} = P_{loss}^{Cu} + P_{loss}^{Fe}.$$

Losses are important for the complete power output of the motor. In the model problem, a very important part of the cost functional are the copper losses P_{loss}^{Cu}, relative to the mechanical losses, but the iron losses will be not considered.

Copper losses are heat produced by electrical currents in electrical devices, i.e. ohmic resistances in the windings of an engine. Hence, copper losses are an undesirable transfer of energy, see [1].

The cost functional in the model problem is based on 4 factors:

- the maximum square deviation of the cogging torque,
- the maximum square deviation of the load torque,
- copper losses, and
- the distortion factor of the induced voltage.

By reason of solving the magnetostatic problem for every rotor position relative to the stator depending on the angle twice, the torque is also computed twice for every angle, on the one hand for the no-load case, and on the other hand for the load case. The torque for the no-load case is called the cogging torque. It is also known as "no-current" torque, see [1]. In the case $I \neq 0$, the torque is computed for every angle as well. Here, the torque consists of cogging and load torque. If we take out the cogging torque from this torque, then we derive the load torque. The following two pictures show the cogging and load torque during a simulation. The rotation angle in degree is plotted on the x-axis and the torque in newton meter is on the y-axis. Moreover, it has to be mentioned that the scaling of the two pictures is different. In fact, the load torque is higher than the cogging torque. In the optimization problems of this thesis, the task will be to minimize the maximum square deviation of the cogging and the load torque.

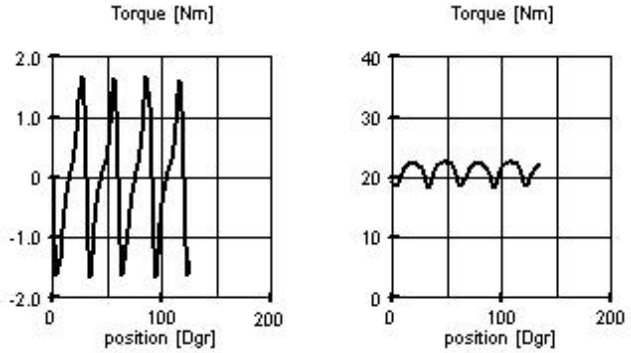


Figure 2.1: Cogging and load torque

The last part of the cost functional is the distortion factor of the induced voltage, also called the total harmonic distortion (THD) of the induced voltage. The induced voltage is the time derivative of the flux ψ, which is linked with the winding. The flux ψ depends on the rotation angle ϕ and the current I, i.e. $\psi(\phi, I)$. Its time derivative is

$$\frac{d\psi}{dt} = \frac{d\psi}{d\phi}\frac{d\phi}{dt} + \frac{d\psi}{dI}\frac{dI}{dt}.$$

The first part of the sum corresponds to the voltage in the case where $I = 0$. The second part is the additional voltage under current ($I \neq 0$). The time derivative of the rotation angle is the angular speed ω, i.e.

$$\omega = \frac{d\phi}{dt}.$$

We consider the THD of the induced voltage. The THD is a measurement of the harmonic distortion. In terms of voltages, the THD is often defined as

$$THD = \frac{V_2^2 + V_3^2 + ... + V_n^2}{V_1^2},$$

which is the ratio of the squares of the root mean square voltages V_i for $i \in \{2, ..., n\}$ to the fundamental harmonic voltage V_1, see [1].

The following two pictures show the difference between functions with a larger and those with a smaller THD.

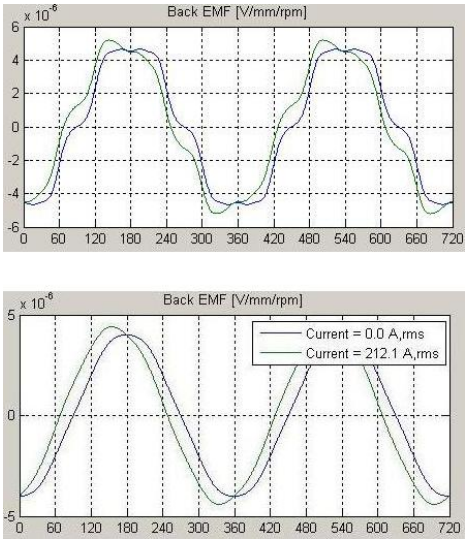


Figure 2.2: The THD of the induced voltage

This is the PMSM of the model problem:

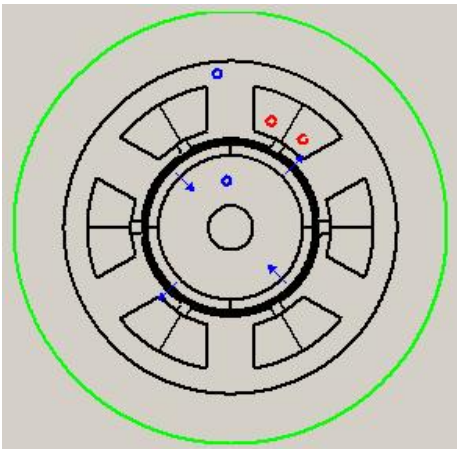


Figure 2.3: The complete motor model

The model PMSM has 6 stator cogs and 4 magnets in the rotor. Hence, the motor has 2 pole pairs. The machine has 6 windings and 3 phases, and between rotor and stator is the air gap.

In detail, the stator and the rotor are:

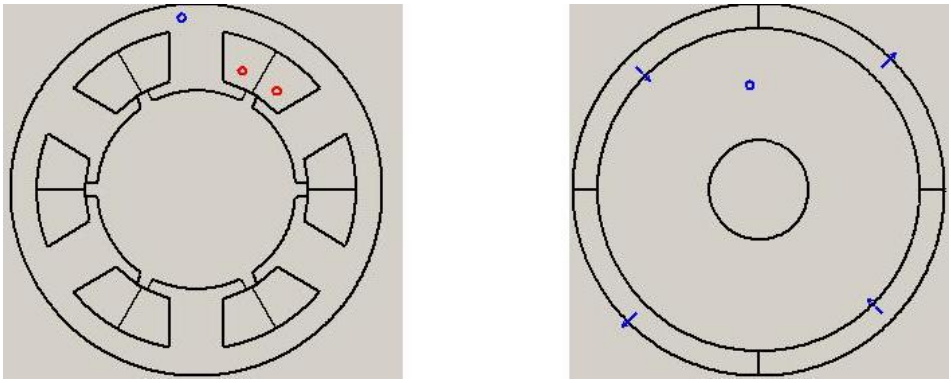


Figure 2.4: Stator and rotor of the motor model

The optimization problem is formulated as

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{such that} \quad \underline{x_i} \leq x_i \leq \overline{x_i} \quad \forall i \in \{1, 2, ..., n\}, \tag{2.1}$$

where f is the cost functional, which has to be minimized. It is given as

$$f = q^T \cdot W \cdot q, \tag{2.2}$$

where q is a vector consisting of the following 4 components: the maximum square deviation of the cogging torque τ_{cogg} relative to the mean load torque τ_{mean}, altogether called τ^{rel}_{cogg}, the maximum square deviation of the load torque τ_{load} relative to the mean load torque τ_{mean}, which is called τ^{rel}_{load}, the copper losses P^{Cu}_{loss} relative to the mechanical power P_{mech} named P^{rel}_{loss} and the THD of the induced voltage. W is a matrix of weights, such that q can be weighted by the engineer according to the significance of its components. We have

$$q = \begin{pmatrix} \tau^{rel}_{cogg} \\ \tau^{rel}_{load} \\ P^{rel}_{loss} \\ THD \end{pmatrix} \tag{2.3}$$

with

$$\tau^{rel}_{cogg} = \frac{\tau_{cogg}}{\tau_{mean}}, \quad \tau^{rel}_{load} = \frac{\tau_{load}}{\tau_{mean}} \quad \text{and} \quad P^{rel}_{loss} = \frac{P^{Cu}_{loss}}{P_{mech}}. \tag{2.4}$$

As mentioned at the beginning of this chapter, the power output P of an engine is expressed as the scalar product of its torque τ and its angular speed ω,

$$P = \tau \cdot \omega. \tag{2.5}$$

According to this equation and to [3], the mechanical power is given by

$$P_{mech} = \tau_{mean} \cdot \omega, \tag{2.6}$$

where ω is the angular speed ($\omega = 2\pi n$ with n as rotational speed).

The copper losses P_{loss}^{Cu} are specified by

$$P_{loss}^{Cu} = N_{ph} \cdot R_{ph} \cdot I_{eff}^2, \tag{2.7}$$

where $N_{ph} = 3$ is the number of phases, R_{ph} the resistance in the phase and I_{eff} equals $\hat{I}/\sqrt{2}$ with $\hat{I}$ the maximum value of the amplitudes of the phase currents.

According to the engineers, the copper losses have to be weighted higher than the other factors. Hence, the matrix of weights is given by

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.8}$$

In the following chapters, firstly a numerical background for the optimization is presented and then the problems with two, five and eight design parameters are described and results are presented.

Chapter 3

Numerical Differentiation

3.1 Approximation of Gradient and Hessian

In the optimization problems which are considered in this thesis, the function values are computed by simulation and hence they are available. For the optimization, the MATLAB routine *fmincon* is used to compute a minimum of the cost functional f, which is a scalar function. There are possibilities to supply the gradient g and the Hessian H of f by the user. Hence, we need a good strategy to approximate g and H well.

To start with, we will consider the one-dimensional case. Let $f(x)$ be a one-dimensional scalar function, which is n times continuously differentiable, i.e.

$$f : \mathbb{R} \to \mathbb{R} \quad \text{with} \quad f \in C^n. \tag{3.1}$$

Here, the gradient of f is $f'(x)$ and the Hessian is $f''(x)$. According to [9], it is possible to find a good approximation of $f'(x)$ by replacing $f(x)$ with an interpolation function $\varphi(x)$. Therefore we use $\varphi'(x)$ to approximate $f'(x)$. If we use linear interpolation through the points x and $x + h$, then we get the so called *forward difference quotient*, i.e.

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{3.2}$$

Analogously, by using linear interpolation through the points $x - h$ and x, we derive the so called *backward difference quotient*, i.e.

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}. \tag{3.3}$$

If we use quadratic interpolation through the points $x-h$, x and $x+h$, then we get the so called *central difference quotient*, i.e.

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}. \tag{3.4}$$

The next step is to analyze these derivative approximations and to look on their errors. By Taylor expansion it follows that

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f^{(3)}(x) + \frac{h^4}{4!}f^{(4)}(x) + O(h^5) \tag{3.5}$$

and

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f^{(3)}(x) + \frac{h^4}{4!}f^{(4)}(x) + O(h^5). \tag{3.6}$$

From that we get the following error estimates for difference quotients:

$$\frac{f(x+h) - f(x)}{h} - f'(x) = \frac{h}{2}f''(x) + \frac{h^2}{6}f^{(3)}(x) + O(h^3) \tag{3.7}$$

for the forward difference quotient,

$$\frac{f(x) - f(x-h)}{h} - f'(x) = -\frac{h}{2}f''(x) + \frac{h^2}{6}f^{(3)}(x) + O(h^3) \tag{3.8}$$

for the backward difference quotient and

$$\frac{f(x+h) - f(x-h)}{2h} - f'(x) = \frac{h^2}{6}f^{(3)}(x) + O(h^4) \tag{3.9}$$

for the central difference quotient. The leading term of the errors of the forward and the backward difference quotient is proportional to h, but the central difference formula yields a more accurate approximation, i.e. its leading term is proportional to h^2. Hence, we will choose the central difference quotient to approximate the gradient of f.

Before approximating the Hessian, which is $f''(x)$ in the one-dimensional case, we want to introduce some operators.

The *forward difference operator* $\triangle$ is defined as

$$\triangle f(x) = f(x+h) - f(x), \tag{3.10}$$

and the *backward difference operator* ∇ as

$$\nabla f(x) = f(x) - f(x-h). \tag{3.11}$$

The difference operators of n-th order are computed recursively by

$$\triangle^n f(x) = \triangle^{n-1}(\triangle f(x)) \tag{3.12}$$

and

$$\nabla^n f(x) = \nabla^{n-1}(\nabla f(x)). \tag{3.13}$$

According to [1] the higher-order partial derivatives are approximated by

$$\frac{\partial^n f}{\partial x^n} \approx \frac{\triangle^n f(x)}{h^n} \tag{3.14}$$

for the n-th order forward difference quotient,

$$\frac{\partial^n f}{\partial x^n} \approx \frac{\nabla^n f(x)}{h^n} \tag{3.15}$$

for the n-th order backward difference quotient and

$$\frac{\partial^n f}{\partial x^n} \approx \frac{1}{2h^n}(\triangle^n f(x - \frac{n}{2}h) + \nabla^n f(x + \frac{n}{2}h)), \tag{3.16}$$

for the n-th order central difference quotient, if n is even, and

$$\frac{\partial^n f}{\partial x^n} \approx \frac{1}{2h^n}(\triangle^n f(x - \frac{n-1}{2}h) + \nabla^n f(x + \frac{n-1}{2}h)) \tag{3.17}$$

if n is odd. It follows for the one-dimensional second-order forward difference quotient that

$$f''(x) \approx \frac{1}{h^2}(f(x+2h) - 2f(x+h) + f(x)), \tag{3.18}$$

for the one-dimensional second-order backward difference quotient that

$$f''(x) \approx \frac{1}{h^2}(f(x) - 2f(x-h) + f(x-2h)) \tag{3.19}$$

and for the one-dimensional second-order central difference quotient that

$$f''(x) \approx \frac{1}{h^2}(f(x+h) - 2f(x) + f(x-h)). \tag{3.20}$$

Taylor expansion leads to the following error estimates:

$$\frac{1}{h^2}(f(x+2h) - 2f(x+h) + f(x)) - f''(x) = hf^{(3)}(x) + O(h^2) \tag{3.21}$$

for the forward difference quotient,

$$\frac{1}{h^2}(f(x) - 2f(x-h) + f(x-2h)) - f''(x) = -hf^{(3)}(x) + O(h^2) \tag{3.22}$$

for the backward difference quotient and

$$\frac{1}{h^2}(f(x+h) - 2f(x) + f(x-h)) - f''(x) = \frac{h^2}{12}f^{(4)}(x) + O(h^3) \tag{3.23}$$

for the central difference quotient. Again the central difference formula yields a more accurate approximation of the second derivative. Therefore, we will use central difference quotients to approximate gradient and Hessian of f.

For the two-dimensional problem we have that

$$f : \mathbb{R}^2 \to \mathbb{R}. \tag{3.24}$$

The partial derivatives of the gradient g are analogously approximated by central difference quotients as the first derivative in the one-dimensional case. Therefore, the gradient is approximated by

$$g = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \approx \begin{pmatrix} \frac{1}{2h_1}(f(x_1+h_1, x_2) - f(x_1-h_1, x_2)) \\ \frac{1}{2h_2}(f(x_1, x_2+h_2) - f(x_1, x_2-h_2)) \end{pmatrix},$$

where h_1 and h_2 denote the 2 different stepsizes of the central difference quotients in the approximated gradient. The pure second partial derivatives of the Hessian H, which is

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix},$$

are approximated according to the second derivative in the one-dimensional case, namely by

$$\frac{\partial^2 f}{\partial x_1^2} \approx \frac{1}{s_1^2}(f(x_1 + s_1, x_2) - 2f(x_1, x_2) + f(x_1 - s_1, x_2)) \tag{3.25}$$

and

$$\frac{\partial^2 f}{\partial x_2^2} \approx \frac{1}{s_2^2}(f(x_1, x_2 + s_2) - 2f(x_1, x_2) + f(x_1, x_2 - s_2)), \tag{3.26}$$

where s_1 and s_2 are the 2 different stepsizes of the central difference quotients in the approximated pure second derivatives of the Hessian. Now we have to approximate the mixed second derivative of the Hessian, i.e.

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial^2 f}{\partial x_2 \partial x_1}. \tag{3.27}$$

According to [6], the mixed second derivative approximated by central difference quotients is

$$\frac{1}{4l_1 l_2}(f(x_1+l_1, x_2+l_2) - f(x_1-l_1, x_2+l_2) - f(x_1+l_1, x_2-l_2) + f(x_1-l_1, x_2-l_2)) \tag{3.28}$$

with l_1 and l_2 denoting the 2 different stepsizes of the difference quotient. By Taylor expansion and under the assumptions that the stepsizes l_1 and l_2 are around of the same magnitude, it follows that the difference between the exact derivative (3.27) and its approximation (3.28) is given by

$$\frac{l_2^2}{6}\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2) + \frac{l_1^2}{6}\frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2) + O(l_1^3) + O(l_2^3). \tag{3.29}$$

Since we have the approximations of the gradient g and the Hessian H of a scalar function f by central difference quotients for two dimensions, these approximations are derived analogously for higher dimensional cases.

For the problem with 2 design parameters there are altogether 13 function evaluations needed in each iteration step of the MATLAB routine *fmincon*: 1 for the function value $f(x_1, x_2)$, 4 additional for approximating the gradient and 8 additional for approximating the Hessian. The following complexity table illustrates the number of function evaluations, which are needed in each iteration step, depending on the number n of design parameters:

	f	g	f, g	H	f, g, H
n	1	$2n$	$1+2n$	$2n^2$	$1+2n+2n^2$
1	1	2	3	2	5
2	1	4	5	8	13
5	1	10	11	50	61
8	1	16	17	128	145

A very important part of approximating derivatives by difference quotients is a good choice of the stepsizes because it is significant in finding successfully an optimum of the cost functional. Here, it has to be mentioned that *cancelation* appears if a derivative is approximated. Hence, the stepsizes should not be chosen too small, see [9].

As next, we will present an error analysis of the given problem and afterwards we will determine through that an optimal choice for the stepsizes, which will be explained for a problem with 2 design parameters.

3.2 Error Analysis

In the error analysis of our problem we will consider the approximation error and from that we will determine the optimal stepsizes.

To start with, let $f(x)$ be the exact function value and $\bar{f}(x)$ be the perturbed function value, where the perturbance is caused by computing the function value by various methods as for example approximating an integral by numerical methods and by discretization, which is needed for the finite element method simulating the motor, and so on. In the following, we will not consider the reason for this perturbance in our model problem. For us, the function is a black box and hence we will try to derive the error between $f(x)$ and $\bar{f}(x)$ by some other techniques. The upper bound of the absolute error between the exact and the perturbed function value is called the accuracy ϵ_A, i.e.

$$\left|\bar{f}(x) - f(x)\right| \leq \epsilon_A. \tag{3.30}$$

The relative error ϵ_f is related to the accuracy by

$$\epsilon_f = \frac{\epsilon_A}{|f(x)|}. \tag{3.31}$$

In practice, the relative error ϵ_f or also the accuracy ϵ_A can be chosen in mainly three ways, i.e.

- known by the user,
- approximated by some techniques,
- computed by finite element error estimators.

In this thesis, results will be presented by either choosing the accuracy ϵ_A or estimating it by some techniques from [6]. As next, we will consider its estimation. Optimization algorithms utilize the values of the function at many points. Therefore, the concern is to obtain a good bound on the error in the computed function value for any point at which the function must be evaluated during the optimization. A significant amount of computation is needed in order to get a reasonably reliable estimate of the accuracy. Therefore, a successful way of approximating the accuracy is needed which does not require excessive computation. Such an approach is presented in [6] and has turned out to be quite successful on practical problems. It will be presented in the next subsection.

A good estimate of the accuracy for a given function f should be obtained at a certain typical point, which is usually the initial point of the optimization. If the accuracy can be estimated and a certain model of the behavior of this error can be assumed, then it is possible to estimate the accuracy of f at any point within the box constraints. It must be pointed out that the mathematical model is only a perturbation of the real problem because it is based on statistical assumptions about the error distribution. The estimate of the accuracy can effect several aspects of an optimization algorithm as the specification of the termination criteria or the minimal distinction between the points which are imposed during the step length procedure, see [6].

3.2.1 The estimation of the accuracy

The thesis considers a method from the book [6] to estimate the accuracy ϵ_A. Here, the satisfaction of the associated assumptions is a crucial factor for the quality of the estimate because the accuracy estimates rely on the behavior of the function at the selected points.

For the case that the function is twice continuously differentiable and univariate, the behavior of the function is considered naturally along the only

one direction. Additionally, if certain assumptions can be made about the behavior of the function, of its higher derivatives and about the statistical distribution of the errors in the computed values of f, then an estimate of the accuracy can be obtained.

We have to choose a set of values $x_i \in \mathbb{R}$ as

$$x_i = x_0 + i \cdot h \tag{3.32}$$

with $i \in \{0, ..., m\}$, where $m \in \mathbb{N}$. The equidistant step length $h \in \mathbb{R}$ should have a small absolute value $|h|$, but not too small because then the noise would be too strong. It is supposed that f has been computed at these values. Moreover, it is assumed that each computed value $\bar{f}_i$ is related to the exact value $f(x_i)$ by the formula

$$\bar{f}_i = f(x_i) + \delta_i \equiv f(x_i) + \theta_i \epsilon_A \tag{3.33}$$

where δ_i and θ_i are random variables with $|\theta_i| \leq 1$. As [6] states, a difference table for $\bar{f}_i$ can be set up, where the difference operator $\triangle$ is the forward difference operator. The difference table for $\triangle^k f_i$ for the numbers $i, k \in \{0, ..., 4\}$ has the form:

$$\begin{array}{lllll}
\bar{f}_0 & & & & \\
& \triangle \bar{f}_0 & & & \\
\bar{f}_1 & & \triangle^2 \bar{f}_0 & & \\
& \triangle \bar{f}_1 & & \triangle^3 \bar{f}_0 & \\
\bar{f}_2 & & \triangle^2 \bar{f}_1 & & \triangle^4 \bar{f}_0 \\
& \triangle \bar{f}_2 & & \triangle^3 \bar{f}_1 & \\
\bar{f}_3 & & \triangle^2 \bar{f}_2 & & \\
& \triangle \bar{f}_3 & & & \\
\bar{f}_4 & & & &
\end{array}$$

where each difference is computed by subtracting the two entries of the previous column. The already known values $\bar{f}_i$ build the first column of the table. From the difference table follows after k differences due to the linearity of the difference operator that

$$\triangle^k \bar{f}_i = \triangle^k f_i + \triangle^k \delta_i. \tag{3.34}$$

As it is stated in [6], it can be shown that $\triangle^k f_0 = h^k f^{(k)}(x) + O(h^{k+1})$. Furthermore, $|h^k f^{(k)}| \to 0$ for moderate values of k and for h small enough.

Then the higher order differences of the computed function values should reflect approximately the differences of the errors δ_i.

Under the assumption that the random variables $\{\theta_i\}$ are uncorrelated and have the same variance, it is possible to estimate ϵ_A, if a specific pattern can be observed in one column of the difference table, namely if the later differences of $\triangle^k \bar{f}_i$ are similar in magnitude and they alternate in sign. Typically, this pattern of behavior can be observed starting with $k = 4$ or $k = 5$. More than $k = 10$ is usually not required. Furthermore, it can be observed that the pattern of sign alternation does not have to occur in every element but can also occur in groups of four or five rather than throughout the entire column. As [6] states, experience shows that such a kind of pattern is typical in practice.

For estimating ϵ_A, [6] suggests the following formula from the k-th column of the difference table:

$$\epsilon_A^{(k)} \approx \frac{\max_i |\triangle^k \bar{f}_i|}{\beta_k} \tag{3.35}$$

with

$$\beta_k = \sqrt{\frac{(2k)!}{(k!)^2}}. \tag{3.36}$$

As [6] states, practice shows that mostly k is chosen as 4 or 5. If it is needed, then the relative error can be computed by the formula (3.31). For computing the denominator $|f(x)|$ of the relative error, we have to choose the point x well. A natural choice would be the initial value x_0. [6] states that it is an effective strategy to use this initial estimate for the error. Nevertheless, it has to be remarked that in most problems not only one estimate of ϵ_A at one point is needed but it is required at more points. In this case, [6] suggests a method for estimating the error efficiently.

In the following, we consider a two-dimensional problem. For more than one dimension, [6] suggests that the function should be considered along a normalized direction p for the accuracy error analysis, i.e. $||p|| = 1$. For two dimensions, the direction p is of the form

$$p = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \tag{3.37}$$

with $||p|| = 1$. The points

$$x_i = \begin{pmatrix} x_i^1 \\ x_i^2 \end{pmatrix}, \tag{3.38}$$

which are needed to compute the function values for the accuracy, are computed by setting

$$x_i = x_0 + i \cdot h \cdot p, \tag{3.39}$$

where x_0 has to be an arbitrary starting value for the direction p and $||h||$ should be chosen sufficiently small.

3.2.2 The total error

We will start by a consideration of the total errors of approximating the first and second derivative of a one-dimensional function f and then we will present the total errors and their upper bounds of approximating the partial derivatives of a two-dimensional function.

The total error of the first derivative $f'(x)$ approximated by central difference quotient is

$$e_x = \left| \frac{\bar{f}(x+h) - \bar{f}(x-h)}{2h} - f'(x) \right|, \tag{3.40}$$

where $\bar{f}(x)$ is the perturbed function value. Now, we use a trick by inserting 0, which leads to

$$\begin{aligned} e_x = \Bigg| & \frac{\bar{f}(x+h) - \bar{f}(x-h)}{2h} - \frac{f(x+h) - f(x-h)}{2h} \\ & + \frac{f(x+h) - f(x-h)}{2h} - f'(x) \Bigg|. \end{aligned} \tag{3.41}$$

By using the triangle inequality, we get that

$$\begin{aligned} e_x \leq & \left| \frac{\bar{f}(x+h) - \bar{f}(x-h)}{2h} - \frac{f(x+h) - f(x-h)}{2h} \right| \\ & + \left| \frac{f(x+h) - f(x-h)}{2h} - f'(x) \right|. \end{aligned} \tag{3.42}$$

If we insert the accuracy ϵ_A defined by (3.30) and use the approximation error (3.9), which follows from Taylor expansion, then we get the upper bound

$$e_x \leq \frac{\epsilon_A}{h} + \frac{h^2}{6} \left| f^{(3)}(x) \right| \tag{3.43}$$

for the total error up to higher order terms. In the following, we will always neglect the higher order terms for estimating the total errors.

Now, we want to derive an upper bound for the total error of the second derivative $f''(x)$ approximated by central difference quotient. The total error here is

$$e_{xx} = \left| \frac{1}{h^2}(\bar{f}(x+h) - 2\bar{f}(x) + \bar{f}(x-h)) - f''(x) \right|. \tag{3.44}$$

We use again the same trick as before and apply the triangle inequality, which leads to

$$\begin{aligned} e_{xx} \leq & \left| \frac{1}{h^2}(\bar{f}(x+h) - 2\bar{f}(x) + \bar{f}(x-h)) - \frac{1}{h^2}(f(x+h) - 2f(x) + f(x-h)) \right| \\ & + \left| \frac{1}{h^2}(f(x+h) - 2f(x) + f(x-h)) - f''(x) \right|. \end{aligned} \tag{3.45}$$

By inserting the accuracy ϵ_A and the approximation error (3.23) from Taylor expansion, we get that

$$e_{xx} \leq \frac{4\epsilon_A}{h^2} + \frac{h^2}{12} \left| f^{(4)}(x) \right|. \tag{3.46}$$

In the case with 2 design parameters, we get the similar upper bounds for the total errors of the approximated first partial derivatives $\frac{\partial f}{\partial x_1}$ and $\frac{\partial f}{\partial x_2}$:

$$e_{x_1} \leq \frac{\epsilon_A}{h_1} + \frac{h_1^2}{6} \left| \frac{\partial^3 f}{\partial x_1^3}(x_1, x_2) \right| \tag{3.47}$$

and

$$e_{x_2} \leq \frac{\epsilon_A}{h_2} + \frac{h_2^2}{6} \left| \frac{\partial^3 f}{\partial x_2^3}(x_1, x_2) \right|, \tag{3.48}$$

with the stepsizes h_1 and h_2. Moreover, we get for approximating the pure second derivatives $\frac{\partial^2 f}{\partial x_1^2}$ and $\frac{\partial^2 f}{\partial x_2^2}$ the following upper bounds for the total errors:

$$e_{x_1 x_1} \leq \frac{4\epsilon_A}{s_1^2} + \frac{s_1^2}{12} \left| \frac{\partial^4 f}{\partial x_1^4}(x_1, x_2) \right| \tag{3.49}$$

and

$$e_{x_2x_2} \leq \frac{4\epsilon_A}{s_2^2} + \frac{s_2^2}{12}\left|\frac{\partial^4 f}{\partial x_2^4}(x_1, x_2)\right| \tag{3.50}$$

with the stepsizes s_1 and s_2. Finally, we need the upper bound for the total error of the mixed second partial derivative approximated by central difference quotients. Here, we have the total error

$$\begin{aligned} e_{x_1x_2} = \Bigg| &\frac{1}{4l_1l_2}(\bar{f}(x_1 + l_1, x_2 + l_2) - \bar{f}(x_1 - l_1, x_2 + l_2) \\ &- \bar{f}(x_1 + l_1, x_2 - l_2) + \bar{f}(x_1 - l_1, x_2 - l_2)) - \frac{\partial^2 f}{\partial x_1 \partial x_2}(x_1, x_2)\Bigg| \end{aligned} \tag{3.51}$$

with the stepsizes l_1 and l_2. By using the same trick as before, afterwards applying the triangle inequality and then inserting the accuracy ϵ_A and the approximation error from Taylor expansion, we get that

$$e_{x_1x_2} \leq \frac{\epsilon_A}{l_1l_2} + \frac{l_2^2}{6}\left|\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2)\right| + \frac{l_1^2}{6}\left|\frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2)\right|. \tag{3.52}$$

3.3 Choice of the Stepsizes

In the two-dimensional case, 6 different stepsizes have to be chosen. For approximating the gradient, 2 stepsizes are needed, h_1 and h_2. The Hessian consists of two pure second derivatives and one mixed second derivative. The difference quotients for approximating the Hessian need 4 different stepsizes. They are denoted by s_1 and s_2 for the pure second and l_1 and l_2 for the mixed second derivatives.

The goal is to choose the stepsizes such that the derivatives are approximated as good as possible. The idea is to consider the upper bounds of the total errors and to minimize these bounds, see [6]. The upper bound (3.47) of the total error will be now denoted as

$$\hat{e}_{x_1} = \frac{\epsilon_A}{h_1} + \frac{h_1^2}{6}\left|\frac{\partial^3 f}{\partial x_1^3}(x_1, x_2)\right|. \tag{3.53}$$

Hence, this bound is a function depending on h_1, x_1 and x_2 and we can write (3.47) as

$$e_{x_1} \leq \hat{e}_{x_1}(h_1, x_1, x_2). \tag{3.54}$$

Analogously, we will denote the upper bounds (3.48), (3.49), (3.50) and (3.52).

The following table illustrates the partial derivatives, which have to be approximated, and the corresponding upper bounds of their total errors:

partial derivatives	errors
$\frac{\partial f}{\partial x_1}$	$\hat{e}_{x_1}$
$\frac{\partial f}{\partial x_2}$	$\hat{e}_{x_2}$
$\frac{\partial^2 f}{\partial x_1^2}$	$\hat{e}_{x_1 x_1}$
$\frac{\partial^2 f}{\partial x_2^2}$	$\hat{e}_{x_2 x_2}$
$\frac{\partial^2 f}{\partial x_1 \partial x_2}$	$\hat{e}_{x_1 x_2}$

3.3.1 The minimization of the total error

The upper bounds of the total errors will be now used to determine the optimal stepsizes by differentiating them with respect to the stepsizes.

Here it has to be mentioned that the two approximation error parts of the upper bound of the total error run contrary to each other. The first part is decreasing with respect to the stepsize and the second part is increasing, see [9]. The following picture is a sketch of how the errors behave.

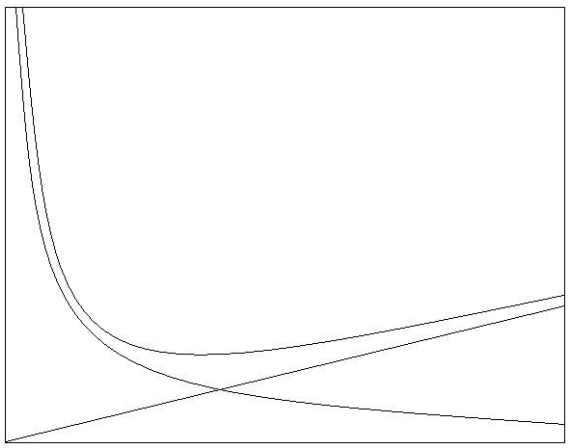


Figure 3.1: Sketch of the approximation errors and the total error

In order to get the optimal stepsizes, the upper bounds of the total errors have to be minimized with respect to the stepsizes. Starting with h_1,

$$\hat{e}_{x_1} = \frac{\epsilon_A}{h_1} + \frac{h_1^2}{6} \left| \frac{\partial^3}{\partial x_1^3} f(x_1, x_2) \right| \tag{3.55}$$

is minimized by differentiating with respect to h_1 and then setting the term equal to 0. Altogether, this leads to

$$-\frac{\epsilon_A}{h_1^2} + \frac{h_1}{3} \left| \frac{\partial^3}{\partial x_1^3} f(x_1, x_2) \right| = 0 \tag{3.56}$$

and finally

$$h_1 = \sqrt[3]{\frac{3\epsilon_A}{\left| \frac{\partial^3}{\partial x_1^3} f(x_1, x_2) \right|}}. \tag{3.57}$$

Minimizing the upper bound of the total error with respect to h_2 is done analogously and leads to the optimal stepsize

$$h_2 = \sqrt[3]{\frac{3\epsilon_A}{\left| \frac{\partial^3}{\partial x_2^3} f(x_1, x_2) \right|}}. \tag{3.58}$$

As next, the stepsizes s_1 and s_2 of the second pure derivatives are determined by firstly minimizing

$$\hat{e}_{x_1 x_1} = \frac{4\epsilon_A}{s_1^2} + \frac{s_1^2}{12} \left| \frac{\partial^4}{\partial x_1^4} f(x_1, x_2) \right| \tag{3.59}$$

with respect to s_1. As result of the minimization of this term, one gets

$$-8\frac{\epsilon_A}{s_1^3} + \frac{s_1}{6} \left| \frac{\partial^4}{\partial x_1^4} f(x_1, x_2) \right| = 0 \tag{3.60}$$

and finally

$$s_1 = \sqrt[4]{\frac{48\epsilon_A}{\left| \frac{\partial^4}{\partial x_1^4} f(x_1, x_2) \right|}}. \tag{3.61}$$

For s_2, one gets the to s_1 similar optimal stepsize

$$s_2 = \sqrt[4]{\frac{48\epsilon_A}{\left|\frac{\partial^4}{\partial x_2^4} f(x_1, x_2)\right|}}. \tag{3.62}$$

Now the determination of the last two stepsizes l_1 and l_2 has remained. The upper bound for the total error of approximating the mixed second partial derivative is

$$\hat{e}_{x_1 x_2} = \frac{\epsilon_A}{l_1 l_2} + \frac{l_2^2}{6}\left|\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2)\right| + \frac{l_1^2}{6}\left|\frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2)\right|. \tag{3.63}$$

By minimizing (3.63) with respect to l_1 and l_2 it follows that

$$-\frac{\epsilon_A}{l_1^2 l_2} + \frac{l_1}{3}\left|\frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2)\right| = 0 \tag{3.64}$$

and

$$-\frac{\epsilon_A}{l_1 l_2^2} + \frac{l_2}{3}\left|\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2)\right| = 0. \tag{3.65}$$

By transforming the equations, one gets

$$l_1 = \sqrt[3]{\frac{3\epsilon_A}{l_2\left|\frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2)\right|}} \tag{3.66}$$

and

$$l_2 = \sqrt[3]{\frac{3\epsilon_A}{l_1\left|\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2)\right|}}. \tag{3.67}$$

To solve this system of equations, one method is to insert the second equation into the first and to transform it with respect to l_1. So, one gets

$$l_1^3 = \frac{3\epsilon_A}{\sqrt[3]{\frac{3\epsilon_A}{l_1\left|\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2)\right|}}\left|\frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2)\right|}, \tag{3.68}$$

$$\sqrt[3]{\frac{3\epsilon_A}{l_1 \left| \frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2) \right|}} \left| \frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2) \right| = \frac{3\epsilon_A}{l_1^3}, \tag{3.69}$$

$$\frac{3\epsilon_A}{l_1 \left| \frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2) \right|} \left| \frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2) \right|^3 = \frac{27\epsilon_A^3}{l_1^9}, \tag{3.70}$$

and

$$l_1^8 = \frac{9\epsilon_A^2 \left| \frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2) \right|}{\left| \frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2) \right|^3}. \tag{3.71}$$

Finally, l_1 is determined by

$$l_1 = \sqrt[8]{\frac{9\epsilon_A^2 \left| \frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2) \right|}{\left| \frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2) \right|^3}}, \tag{3.72}$$

and l_2 is determined by inserting l_1 into the formula (3.67).

Now, all formulas for the optimal stepsizes are derived. For computing the stepsizes from these formulas, higher partial derivatives have to be computed. The next question is, how to get this higher derivatives. The solution is quite simple: the higher derivatives, which are of third- and fourth-order, are computed approximatively by central difference quotients.

3.3.2 Approximation of higher derivatives

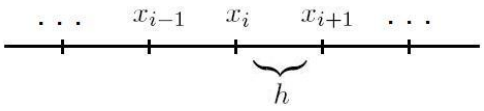
In the following, the partial derivatives of third- and fourth-order, which are needed in the stepsize formulas, are presented and how they can be practically approximated by central difference quotients. The needed derivatives are

- $\frac{\partial^3}{\partial x_1^3} f(x_1, x_2)$ and $\frac{\partial^3}{\partial x_2^3} f(x_1, x_2)$,
- $\frac{\partial^4}{\partial x_1^4} f(x_1, x_2)$ and $\frac{\partial^4}{\partial x_2^4} f(x_1, x_2)$,
- $\frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2)$ and $\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2)$.

To show the concept of deriving the central difference quotients for higher derivatives, the notation has to become more compact. This is done by using the concept described in [8]. Now, the forward difference operator $\triangle$

and the backward difference operator ∇ are used to derive the approximated higher derivatives.

For the pure partial derivatives the idea of deriving the central difference quotients can be easily shown through the one-dimensional case. If we assume equidistant points x_i with the step length h and $i \in \{0, ..., m\}, m \in \mathbb{N}$,



then the function value $f(x_i)$ is denoted as f_i. So the difference operators are now written as

$$\triangle f_i = f_{i+1} - f_i \tag{3.73}$$

and

$$\nabla f_i = f_i - f_{i-1}. \tag{3.74}$$

At the beginning of the chapter, Taylor expansion has been described and it has been shown how to determine the difference quotients for estimating the derivatives. By estimating all derivatives with the same difference quotients - in our case by the central difference quotient - leads for the n-th derivative of the function to the formulas (3.16) and (3.17).

Starting with the approximation of the third derivative by this procedure, which is important for the computation of the optimal stepsizes, we get that

$$\frac{\partial^3 f}{\partial x^3} \approx \frac{\triangle^3 f_{i-1} + \nabla^3 f_{i+1}}{2h^3}. \tag{3.75}$$

For the third forward and backward difference operator one gets

$$\triangle^3 f_{i-1} = \triangle^2(\triangle f_{i-1}) = \triangle(\triangle^2 f_{i-1}) = \triangle(f_{i+1} - 2f_i + f_{i-1}) =$$

$$= f_{i+2} - f_{i+1} - 2(f_{i+1} - f_i) + f_i - f_{i-1} = f_{i+2} - 3f_{i+1} + 3f_i - f_{i-1}$$

and

$$\nabla^3 f_{i+1} = \nabla^2(\nabla f_{i+1}) = \nabla(\nabla^2 f_{i+1}) = \nabla(f_{i+1} - 2f_i + f_{i-1}) =$$

$$= f_{i+1} - f_i - 2(f_i - f_{i-1}) + f_{i-1} - f_{i-2} = f_{i+1} - 3f_i + 3f_{i-1} - f_{i-2}.$$

Altogether it follows that

$$\frac{\partial^3 f}{\partial x^3} \approx \frac{\triangle^3 f_{i-1} + \nabla^3 f_{i+1}}{2h^3} = \frac{1}{2h^3}(f_{i+2} - 2f_{i+1} + 2f_{i-1} - f_{i-2}).$$

Hence, it follows the central difference quotient for approximating the one-dimensional third derivative

$$\frac{\partial^3 f}{\partial x^3} \approx \frac{1}{2h^3}(f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)). \tag{3.76}$$

For the two-dimensional case, it is easy to apply the formula above on a pure third derivative. This leads to the following approximations:

$$\frac{\partial^3 f}{\partial x_1^3} \approx \frac{1}{2\tilde{h}_1^3}(f(x_1+2\tilde{h}_1, x_2) - 2f(x_1+\tilde{h}_1, x_2) + 2f(x_1-\tilde{h}_1, x_2) - f(x_1-2\tilde{h}_1, x_2)) \tag{3.77}$$

and

$$\frac{\partial^3 f}{\partial x_2^3} \approx \frac{1}{2\tilde{h}_2^3}(f(x_1, x_2+2\tilde{h}_2) - 2f(x_1, x_2+\tilde{h}_2) + 2f(x_1, x_2-\tilde{h}_2) - f(x_1, x_2-2\tilde{h}_2)), \tag{3.78}$$

where the stepsizes $\tilde{h}_1$ and $\tilde{h}_2$ in here can be chosen arbitrarily.

The next step is to find an appropriate approximation of the fourth - pure and mixed - derivatives needed for the computation of the stepsizes. The derivation of the fourth pure derivative can be again shown by looking at the one-dimensional central difference quotient of fourth order. So one has

$$\frac{\partial^4 f}{\partial x^4} \approx \frac{\triangle^4 f_{i-2} + \nabla^4 f_{i+2}}{2h^4}. \tag{3.79}$$

The fourth forward difference operator in here is

$$\triangle^4 f_{i-2} = \triangle^3(\triangle f_{i-2}) = \triangle^3(f_{i-1} - f_{i-2}) = \triangle^2(\triangle f_{i-1} - \triangle f_{i-2})$$

$$= \triangle^2(f_i - f_{i-1} - (f_{i-1} - f_{i-2})) = \triangle(\triangle f_i - 2\triangle f_{i-1} + \triangle f_{i-2}))$$

$$= \triangle(f_{i+1} - f_i - 2(f_i - f_{i-1}) + f_{i-1} - f_{i-2})$$

$$= \triangle(f_{i+1} - 3f_i + 3f_{i-1} - f_{i-2})$$

$$= f_{i+2} - f_{i+1} - 3(f_{i+1} - f_i) + 3(f_i - f_{i-1}) - (f_{i-1} - f_{i-2})$$

$$= f_{i+2} - 4f_{i+1} + 6f_i - 4f_{i-1} + f_{i-2}.$$

For the backward difference operator in the formula above follows

$$\nabla^4 f_{i+2} = \nabla^3(\nabla f_{i+2}) = \nabla^3(f_{i+2} - f_{i+1}) = \nabla^2(\nabla f_{i+2} - \nabla f_{i+1})$$

$$= \nabla^2(f_{i+2} - f_{i+1} - (f_{i+1} - f_i)) = \nabla(\nabla f_{i+2} - 2\nabla f_{i+1} + \nabla f_i)$$

$$= \nabla(f_{i+2} - f_{i+1} - 2(f_{i+1} - f_i) + f_i - f_{i-1})$$

$$= \nabla(f_{i+2} - 3f_{i+1} + 3f_i - f_{i-1})$$

$$= f_{i+2} - f_{i+1} - 3(f_{i+1} - f_i) + 3(f_i - f_{i-1}) - (f_{i-1} - f_{i-2})$$

$$= f_{i+2} - 4f_{i+1} + 6f_i - 4f_{i-1} + f_{i-2}.$$

Altogether it follows that

$$\frac{\partial^4 f}{\partial x^4} \approx \frac{\triangle^4 f_{i-2} + \nabla^4 f_{i+2}}{2h^4} = \frac{1}{2h^4}(2(f_{i+2} - 4f_{i+1} + 6f_i - 4f_{i-1} + f_{i-2}))$$

and hence the central difference quotient for approximating the one-dimensional fourth derivative is

$$\frac{\partial^4 f}{\partial x^4} \approx \frac{1}{h^4}(f(x+2h) - 4f(x+h) + 6f(x) - 4f(x-h) + f(x-2h)). \quad (3.80)$$

For two dimensions, the required pure fourth derivatives $\frac{\partial^4 f}{\partial x_1^4}$ and $\frac{\partial^4 f}{\partial x_2^4}$ are approximated by

$$\frac{1}{\tilde{h}_1^4}(f(x_1+2\tilde{h}_1, x_2) - 4f(x_1+\tilde{h}_1, x_2) + 6f(x_1, x_2) - 4f(x_1-\tilde{h}_1, x_2) - f(x_1-2\tilde{h}_1, x_2)) \quad (3.81)$$

and

$$\frac{1}{\tilde{h}_2^4}(f(x_1, x_2+2\tilde{h}_2) - 4f(x_1, x_2+\tilde{h}_2) + 6f(x_1, x_2) - 4f(x_1, x_2-\tilde{h}_2) - f(x_1, x_2-2\tilde{h}_2)), \quad (3.82)$$

where the stepsizes $\tilde{h}_1$ and $\tilde{h}_2$ can be again chosen arbitrarily. For simplicity, they are chosen the same as for the pure third derivatives before.

The last part is to approximate the mixed fourth derivatives $\frac{\partial^4 f}{\partial x_1^3 \partial x_2}$ and $\frac{\partial^4 f}{\partial x_1 \partial x_2^3}$. Starting with $\frac{\partial^4 f}{\partial x_1^3 \partial x_2}$, the procedure leads to

$$\frac{\partial^4 f}{\partial x_1^3 \partial x_2} = \frac{\partial}{\partial x_2}(\frac{\partial^3 f}{\partial x_1^3}) \approx \frac{\partial}{\partial x_2}(\frac{1}{2\tilde{h}_1^3}(f_{i+2,j} - 2f_{i+1,j} + 2f_{i-1,j} - f_{i-2,j}))$$

$$= \frac{1}{2\tilde{h}_1^3}(\frac{\partial f_{i+2,j}}{\partial x_2} - 2\frac{\partial f_{i+1,j}}{\partial x_2} + 2\frac{\partial f_{i-1,j}}{\partial x_2} - \frac{\partial f_{i-2,j}}{\partial x_2}).$$

With approximating $\frac{\partial}{\partial x_2}$ again by central differences, it follows

$$\frac{\partial^4 f}{\partial x_1^3 \partial x_2} \approx \frac{1}{2\tilde{h}_1^3}(\frac{f_{i+2,j+1} - f_{i+2,j-1}}{2\tilde{h}_2} - 2\frac{f_{i+1,j+1} - f_{i+1,j-1}}{2\tilde{h}_2}$$

$$+2\frac{f_{i-1,j+1} - f_{i-1,j-1}}{2\tilde{h}_2} - \frac{f_{i-2,j+1} - f_{i-2,j-1}}{2\tilde{h}_2})$$

$$= \frac{1}{4\tilde{h}_1^3\tilde{h}_2}(f_{i+2,j+1} - f_{i+2,j-1} - 2(f_{i+1,j+1} - f_{i+1,j-1})$$

$$+2(f_{i-1,j+1} - f_{i-1,j-1}) - (f_{i-2,j+1} - f_{i-2,j-1})). \tag{3.83}$$

Analogously, follows for $\frac{\partial^4 f}{\partial x_1 \partial x_2^3}$

$$\frac{\partial^4 f}{\partial x_1 \partial x_2^3} \approx \frac{1}{4\tilde{h}_1\tilde{h}_2^3}(f_{i+1,j+2} - f_{i-1,j+2} - 2(f_{i+1,j+1} - f_{i-1,j+1})$$

$$+2(f_{i+1,j-1} - f_{i-1,j-1}) - (f_{i+1,j-2} - f_{i-1,j-2})). \tag{3.84}$$

The stepsizes $\tilde{h}_1$ and $\tilde{h}_2$ are chosen again the same as in the derivatives before because they are arbitrary.

Now we have determined all approximations for the third- and fourth-order derivatives and so everything is complete to derive the optimal stepsizes. Altogether they are computed by

$$h_1 = \sqrt[3]{\frac{3\epsilon_A}{\left|\frac{\partial^3}{\partial x_1^3} f(x_1, x_2)\right|}}, \tag{3.85}$$

$$h_2 = \sqrt[3]{\frac{3\epsilon_A}{\left|\frac{\partial^3}{\partial x_2^3} f(x_1, x_2)\right|}}, \tag{3.86}$$

$$s_1 = \sqrt[4]{\frac{48\epsilon_A}{\left|\frac{\partial^4}{\partial x_1^4} f(x_1, x_2)\right|}}, \tag{3.87}$$

$$s_2 = \sqrt[4]{\frac{48\epsilon_A}{\left|\frac{\partial^4}{\partial x_2^4} f(x_1, x_2)\right|}}, \tag{3.88}$$

$$l_1 = \sqrt[8]{\frac{9\epsilon_A^2 \left|\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2)\right|}{\left|\frac{\partial^4}{\partial x_1^3 \partial x_2} f(x_1, x_2)\right|^3}}, \tag{3.89}$$

$$l_2 = \sqrt[3]{\frac{3\epsilon_A}{l_1 \left|\frac{\partial^4}{\partial x_1 \partial x_2^3} f(x_1, x_2)\right|}}, \tag{3.90}$$

where the accuracy ϵ_A is either chosen by the user because of more information available or estimated by the method explained in this chapter.

Now, we have completed the analysis for approximating the gradient and the Hessian of f such that we can use the MATLAB routine *fmincon* and also choose the stepsizes differently. To start with, we will consider the model problem from chapter 2 with two design parameters.

Chapter 4

A Problem with 2 Design Parameters

4.1 Problem Description

The optimization problem with 2 design parameters for the model PMSM is stated by

$$\begin{aligned} &\min_{x\in\mathbb{R}^2} \quad f(x) \\ &\qquad \begin{pmatrix} \underline{x_1} \\ \underline{x_2} \end{pmatrix} \leq x \leq \begin{pmatrix} \overline{x_1} \\ \overline{x_2} \end{pmatrix}. \end{aligned} \tag{4.1}$$

The parameters

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

are the outer diameter of the rotor ($x_1 = dra$) and the width of the stator cog ($x_2 = bst$). The box constraints $\underline{x} = (\underline{x_1}, \underline{x_2})$ and $\overline{x} = (\overline{x_1}, \overline{x_2})$ are given by

$$\underline{x} = \begin{pmatrix} 23 \\ 6.5 \end{pmatrix}, \qquad \overline{x} = \begin{pmatrix} 37 \\ 10.5 \end{pmatrix}. \tag{4.2}$$

The following pictures show the whole motor, the rotor with the drawn in outer diameter of the rotor and the stator with the drawn in width of the stator cog, whereas the last two pictures have a different scaling.

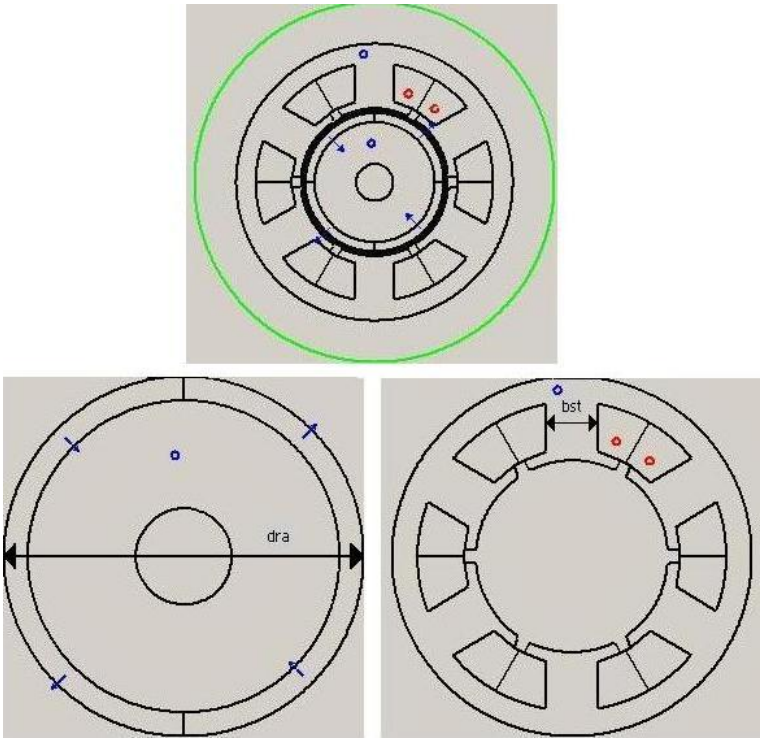


Figure 4.1: The outer diameter of the rotor and the width of the stator cog

To visualize the 2-dimensional cost functional in the feasible region, two plots were made in MATLAB. First, the surf plot shows, in which area the minimum is located.

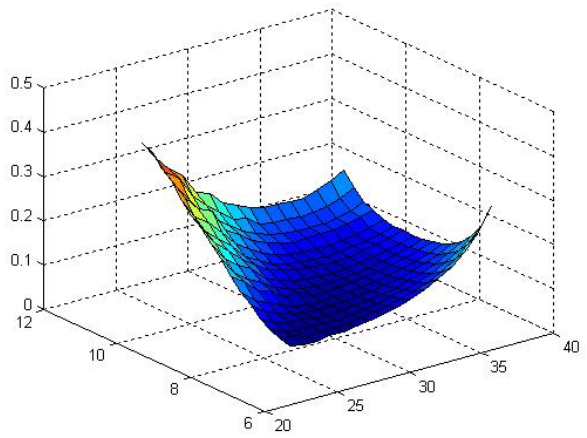


Figure 4.2: 2-dimensional surf plot of the cost functional f in the feasible region

The next figure, the pcolor plot, shows the cost functional from the bird's eye view. The darker the blue parts are, the smaller are the function values and the smaller is the distance to the minimum of the cost functional.

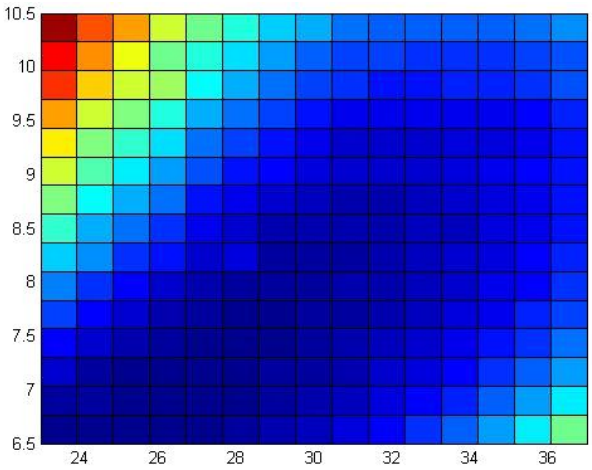


Figure 4.3: 2-dimensional pcolor plot of the cost functional f in the feasible region

The minimum is approximately in the area near the left lower corner of the 2-dimensional feasible region.

4.2 Numerical Results

4.2.1 Optimizing with the Matlab routine *fmincon*

The MATLAB command *fmincon* is of the form

[x,fval,...] = *fmincon*(@fquality, x0, [ ], [ ], [ ], [ ], lb, ub, [ ], options),

where the left hand side specifies the output. The starting value $x0$ is chosen by taking the middle point of the box, i.e.

$$x0 = \begin{pmatrix} 30 \\ 8.5 \end{pmatrix},$$

and the box constraints are

$$\text{lb} = \begin{pmatrix} 23 \\ 6.5 \end{pmatrix}, \quad \text{ub} = \begin{pmatrix} 37 \\ 10.5 \end{pmatrix}.$$

By specifying the *options*, the user has the possibility to limit the maximum number of function evaluations *MaxFunEvals* and the maximum number of iterations *MaxIter*. Moreover, there are also possibilities to vary the optimization algorithm and to specify other stopping criteria for the optimization. In the following, the values of the *options* are the default values - the shortcut dv stands for default value - besides *MaxFunEvals* and *MaxIter*, which are both chosen for all problems as 20. In all computations, more has never been necessary. Wherever the *options* are changed will be mentioned in this thesis. Furthermore and what is currently most important, the options of *fmincon* are used to supply gradient g and Hessian H of the cost functional by the user, by defining g and H in the file **fquality.m**, which computes the current function value f_i of the iteration as well. Basically, three cases arise by supplying

- only f
- f and the gradient g
- f, g and the Hessian H.

4.2.2 First results

Firstly, *fmincon* has to be used basically to get a better understanding of the problem and to get also an idea what could be improved.

In the problem with 2 design parameters, if all derivative information is supplied by the user, there are altogether 13 function evaluations needed in each iteration step. This follows from approximating gradient and Hessian by central difference quotients as it has been done at the beginning of chapter 3, where we have altogether 6 different stepsizes in the difference quotients denoted by h_1, h_2, s_1, s_2, l_1 and l_2.

The MATLAB routine *fmincon* is first run with setting the stepsizes as $h_1 = 0.3$, $h_2 = 0.2$, $s_1 = 0.7$, $s_2 = 0.5$, $l_1 = 0.4$, $l_2 = 0.3$ to get a basic idea of how *fmincon* works and which strategy is better, to supply only f, f and g or f, g and H. As the intuition leads, supplying all derivative information should have the best results. With the initial value $x0 = (30, 8.5)$, the following results are obtained, where x is the value at which MATLAB stopped the optimization algorithm and $f(x)$ is its function value. Moreover, n_{iter} is the number of iterations, *funcCount* is the number of the function evaluations f_i - without the function evaluations needed for approximating g and H in each iteration step - and *time* is the CPU (central processing unit) time of the optimization.

The first result is for the case, where g and H are not provided by the user. In this case, MATLAB has chosen the SQP (Quasi-Newton, line-search) algorithm for computing the minimum.

case	only f_i computed
x	$(30.0000, 8.5000)$
$f(x)$	0.1089
n_{iter}	1
funcCount	6
time	0.25 h
reason algorithm terminated	first order optimality measure and maximum constraint violation less than dv

The next result is for the case, where only the gradient is provided by the user. In the following, the algorithm chosen by MATLAB is the Trust-Region-Reflective algorithm, if nothing else is mentioned.

case	f_i computed and g approximated
x	$(29.8125, 8.8295)$
$f(x)$	0.1067
n_{iter}	12
funcCount	13
time	11 h
reason algorithm terminated	norm of the current step less than dv

The last result is for the case, where gradient and Hessian are supplied by the user.

case	f_i computed, g and H approximated
x	$(24.0028, 7.1505)$
$f(x)$	0.0955
n_{iter}	13
funcCount	14
time	11 h
reason algorithm terminated	relative function value changing by less than dv

It is important to describe these results. Although the first case seems to be very efficient because MATLAB's reason for terminating the algorithm was that the first-order optimality measure was less than the termination tolerance that had been specified for the function values and the one for the constraint violation, the final function value was the starting value. To the contrary, the second and the third case obtained a better result for the final function value. Especially the third case, where all derivative information had been supplied by the user, obtained a very good result according to the plots in Figure 4.2 and Figure 4.3, where the minimum is approximately

in the area near the left lower corner of the 2-dimensional feasible region. Hence, supplying all derivative information should be the best strategy.

So far, the stepsizes have been chosen, but now we want to compute the optimal stepsizes according to the formulas (3.85) - (3.90) of chapter 3, where the accuracy ϵ_A is estimated as it has been explained and the approximations of the higher derivatives are computed as well by central difference quotients.

4.2.3 Computing the minimum with optimal stepsizes

In this subsection, we will compute the minimum by estimating the accuracy, computing the optimal stepsizes and approximating gradient and Hessian of our problem with two design parameters. The options and the algorithm of *fmincon* are again the same as stated in the subsection before.

For estimating the accuracy, we choose firstly a set of values $x_i \in \mathbb{R}^2$ as

$$x_i = x0 + i \cdot h \cdot p$$

with $x0 = (23, 6.5)$, $i \in \{0, ..., 14\}$, $h = 1$ and $p = (0.96, 0.28)$. All function values $\bar{f}_i$ are computed at these values. A part of the difference table for $\bar{f}_i$ according to chapter 3 for the k-th difference with $k \in \{0, ..., 5\}$ is:

0.10123					
	0.00110				
0.10234		-0.00328			
	-0.00218		0.01205		
0.10016		0.00877		-0.02973	
	0.00660		-0.01767		0.05907
0.10676		-0.00890		0.02934	
	-0.00230		0.01166		
0.10445		0.00276			
	0.00046				
0.10491					

As it is seen in the difference table above, the later differences of $\triangle^k \bar{f}_i$ are similar in magnitude and they alternate in sign. Hence, it is possible to estimate ϵ_A. We have computed the differences for $k \in \{0, ..., 5\}$. According to chapter 3, [6] suggests the following formula from the k-th column of the difference table:

$$\epsilon_A^{(k)} \approx \frac{\max_i |\triangle^k \bar{f}_i|}{\beta_k}$$

with

$$\beta_k = \sqrt{\frac{(2k)!}{(k!)^2}}.$$

In our problem, we obtain for $k = 5$ the accuracy

$$\epsilon_A = 0.00375.$$

For the stepsizes $\tilde{h}_1$ and $\tilde{h}_2$, which we need for approximating the third and fourth derivatives in the formulas of the optimal stepsizes, we have chosen

$$\tilde{h}_1 = \tilde{h}_2 = \frac{\overline{x_1} - \underline{x_1}}{100},$$

which leads to the optimal stepsizes:

$$\begin{aligned} h_1 &= 0.5885773 \\ h_2 &= 0.1862061 \\ s_1 &= 0.4111084 \\ s_2 &= 0.1226490 \\ l_1 &= 0.5531660 \\ l_2 &= 0.0872923 \end{aligned}$$

With these stepsizes the following result is obtained:

case	f_i computed, g and H approximated with optimal stepsizes
x	(25.4556, 7.8108)
$f(x)$	0.0987
n_{iter}	16
funcCount	17
time	11.5 h

The reason the algorithm terminated was that the relative function value changed by less than the default termination tolerance on the function value. As we see in the plots of the cost functional, a good approximation of the minimum was found. In the following, we will call this case the case 1.

The next subsection presents results from different cases, which come from changing some values for computing the stepsizes and finally varying the stepsizes for computing the minimum, to see the robustness of our chosen method.

4.2.4 Robustness

One way to change parameter values of the optimization method would be to choose the starting value $x0$ for the optimization differently, but if the user does not exactly know which starting value would be good, a natural way of choosing $x0$ is in the middle of the box constraints, which has been already stated.

Besides the fact that the starting value can be chosen differently, there are other possibilities to vary the optimization method by changing some parameter values. We will take the case from the chapter above and we will change different values, which are needed for computing the minimum, to observe the robustness of our method for obtaining the minimum of the cost functional. From that arise different cases, which will be considered in this subsection. They come from the choice of the error ϵ_A and from the choice of the stepsizes $\tilde{h}_1$ and $\tilde{h}_2$ for the third and fourth derivatives in the formulas of the optimal stepsizes $h = (h_1, h_2)$, $s = (s_1, s_2)$ and $l = (l_1, l_2)$. Finally, after computing the stepsizes from that, these stepsizes will be made smaller and larger to analyze the results of our optimization method.

Firstly, we choose the accuracy as $\epsilon_A = \frac{1}{100} \cdot |f(x0)|$ with $|f(x0)| = 0.1089$, i.e. $\epsilon_A = 0.00109$, which is around one third of our estimated accuracy from case 1 in the subsection before. With the chosen accuracy $\epsilon_A = 0.00109$, the optimal stepsizes are computed the same as in case 1 and lead to the following result:

case	ϵ_A chosen as $\epsilon_A = 0.00109$
x	(28.5684, 8.1705)
$f(x)$	0.1035
n_{iter}	19
$funcCount$	20
time	13.5 h

The algorithm terminated because the norm of the current step was less than the default termination tolerance on x. The result is not as good as before.

Now, we use the estimated accuracy $\epsilon_A = 0.00375$ as before, but we vary the stepsizes $\tilde{h}_1$ and $\tilde{h}_2$. Making them smaller by a factor of 10, we get the result:

case	$\tilde{h}_1$ and $\tilde{h}_2$ smaller
x	(28.6234, 8.5287)
$f(x)$	0.1043
n_{iter}	14
$funcCount$	15
time	10.25 h

Now, we make the stepsizes $\tilde{h}_1$ and $\tilde{h}_2$ larger by a factor of 10 and obtain the result:

case	$\tilde{h}_1$ and $\tilde{h}_2$ larger
x	(26.0876, 8.0454)
$f(x)$	0.1000
n_{iter}	15
$funcCount$	16
time	11.5 h

In both cases, the algorithm terminated because the norm of the current step was less than the termination tolerance on x. The method is robust with respect to changes in $\tilde{h}_1$ and $\tilde{h}_2$.

As next, we want to consider the stepsizes h, s, l from case 1 and we want to make them smaller and larger by a factor of 10. This leads to the following results:

case	h, s, l smaller
x	(29.1991, 8.4048)
$f(x)$	0.1049
n_{iter}	15
$funcCount$	16
time	11 h

The algorithm terminated because the maximum number of function evaluations was exceeded.

case	h, s, l larger
x	(26.5018, 8.2689)
$f(x)$	0.1016
n_{iter}	20
$funcCount$	21
time	15 h

Now, the algorithm terminated because the norm of the current step was less than the specified termination tolerance on x.

The results show that supplying all derivative information and also choosing optimal stepsizes for approximating gradient and Hessian by difference quotients are a good strategy for computing the minimum of the cost functional, which is done through case 1.

As next, some pictures of the shape optimized motor of case 1 follow.

4.2.5 The shape optimized motor for the problem with 2 design parameters

The following pictures show the motor for the initial value and the shape optimized motor such that we can visualize our final result of case 1 in comparison with the motor for the initial value. Firstly, we get the following motors in the case without current, where the cogging torque is computed.

Figure 4.4: The motor for the initial value and the shape optimized motor for the problem with 2 design parameters with $I = 0$

The following picture shows the winding function and the torque of the motor for the initial value and of the shape optimized motor for the problem with 2 design parameters with $I = 0$.

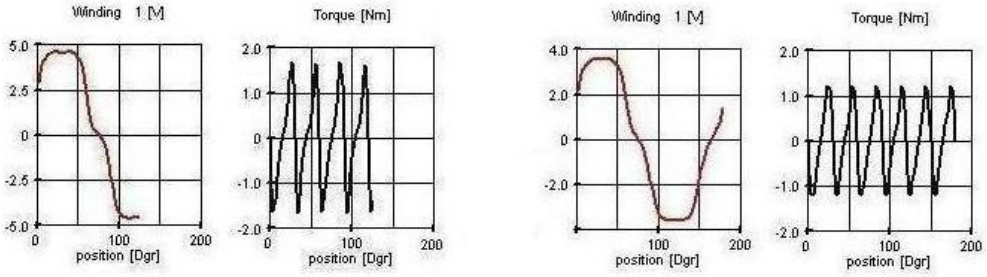


Figure 4.5: Winding function and torque of the motors above

For the shape optimized motor, the maximum square deviation of the cogging torque is smaller than for the initial value. The next picture shows

the motor in the case with current. Hence, the load torque is computed.

Figure 4.6: The motor for the initial value and the shape optimized motor for the problem with 2 design parameters with $I \neq 0$

The following picture shows the winding function, the torque and the winding currents of the motors above for the problem with 2 design parameters with $I \neq 0$.

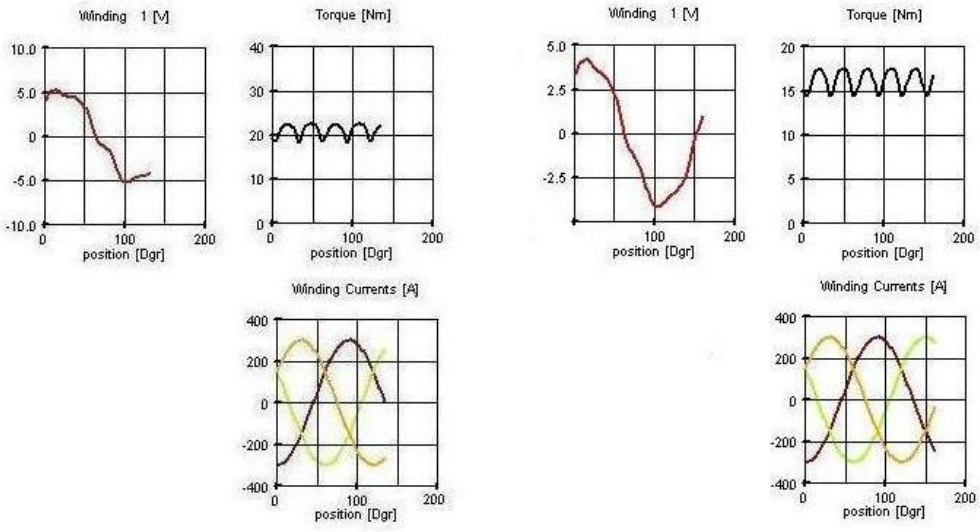


Figure 4.7: Winding function, torque and winding currents of the motors above

Now, the shape optimized motor has a smaller mean value of the load torque and also its maximum square deviation is smaller than at the beginning for the initial value. Moreover, the winding function, which is the function considered for the distortion factor and which has the title "Winding 1 [V]", has now much more the form of a harmonic function than it has had before.

4.2.6 Parallelization

Another way of reducing the computation time is to use parallel processors for computing all needed function values in every iteration step. For example in the case with 2 design parameters, if 13 parallel processors are available in every iteration step, then altogether one iteration step seems as expensive as the computation of one function value.

4.3 Automation of the Optimization

After all observations and computations, this section will recommend how shape optimization by a black box method can be automatically established according to case 1 of subsection 4.2.3.

In a MATLAB file named **precomputation.m**, a precomputation is made by approximating the accuracy and computing the optimal stepsizes, which is described in chapter 3 and in subsection 4.2.3. After the precomputation, the optimal stepsizes are inserted into the MATLAB file **fquality.m**, where all function values, which are needed in each iteration step, are computed. A function value is computed by the file **simulate.m**, the black box. The optimization is started by the file **OptimizationQualityFunction.m**, where the MATLAB routine *fmincon* is called. The initial value for *fmincon* is chosen in the middle of the box constraints. The last three MATLAB files are described in section 1.3.

Altogether the automatic optimization process consists of the four MATLAB files:

- precomputation.m, which approximates the accuracy and computes the optimal stepsizes,
- simulate.m, which is the black box,
- fquality.m, which computes all needed the function values, and
- OptimizationQualityFunction.m, where the MATLAB routine *fmincon* is called.

Of course, the user can interfere into the optimization process by choosing another initial value or the accuracy, which is needed for computing the

optimal stepsizes. Furthermore, the user can set the stepsizes for approximating gradient and Hessian as well.

In the following two chapters, problems with more design parameters will be optimized by the best working strategy, which is the one of case 1. The two problems will be one with five design parameters and then one with eight design parameters. The design parameters will be described at the beginning of each chapter. In both problems, the motor model is the same and therefore the cost functional will be the same, which is described in chapter 2.

Finally, results will be presented and they will be also compared with the results from optimizing the motor by another method, by a genetic algorithm. Usually, this method yields the global minimum of the cost functional, but there are much more function evaluations needed to reach it. After presenting both results, the advantages and disadvantages of the black box optimization method will be explained at the end.

Chapter 5

A Problem with 5 Design Parameters

5.1 Problem Description

The optimization problem with 5 design parameters is stated by

$$\begin{aligned} &\min_{x\in\mathbb{R}^5} \quad f(x) \\ &\begin{pmatrix} \underline{x_1} \\ \underline{x_2} \\ \underline{x_3} \\ \underline{x_4} \\ \underline{x_5} \end{pmatrix} \leq x \leq \begin{pmatrix} \overline{x_1} \\ \overline{x_2} \\ \overline{x_3} \\ \overline{x_4} \\ \overline{x_5} \end{pmatrix}. \end{aligned} \tag{5.1}$$

The parameters

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} dra \\ bst \\ hm \\ phiMagnet \\ exs \end{pmatrix}$$

are the outer diameter of the rotor ($x_1 = dra$), the width of the stator cog ($x_2 = bst$), the height of the rotor magnet ($x_3 = hm$), the angle of the rotor magnet ($x_4 = phiMagnet$) and the excentricity of the stator ($x_5 = exs$). The outer diameter of the rotor and the width of the stator cog are the already described parameters from the problem with 2 design parameters.

The height of the rotor magnet is illustrated by the following picture:

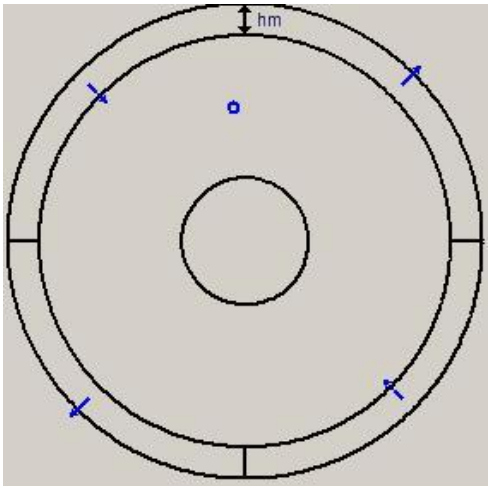


Figure 5.1: The height of the rotor magnet *hm*

The next picture illustrates the angle of the rotor magnet:

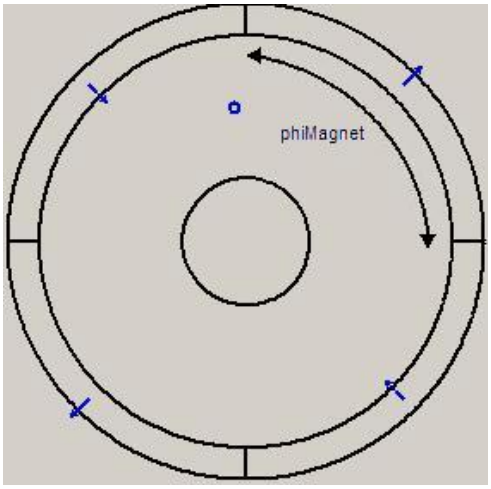


Figure 5.2: The angle of the rotor magnet *phiMagnet*

The excentricity of the stator is the distance between the center point of the stator and the center point of the inner circles of the pole shoes. It is illustrated in the next picture.

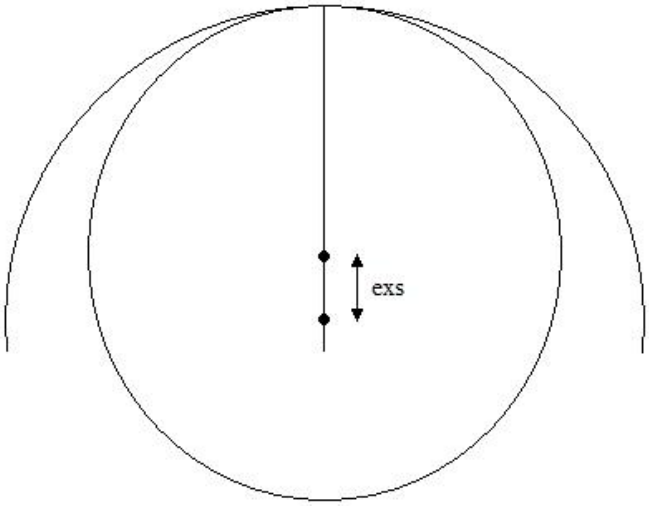


Figure 5.3: Sketch of the excentricity of the stator exs

The following picture shows the motor model with marked excentricity:

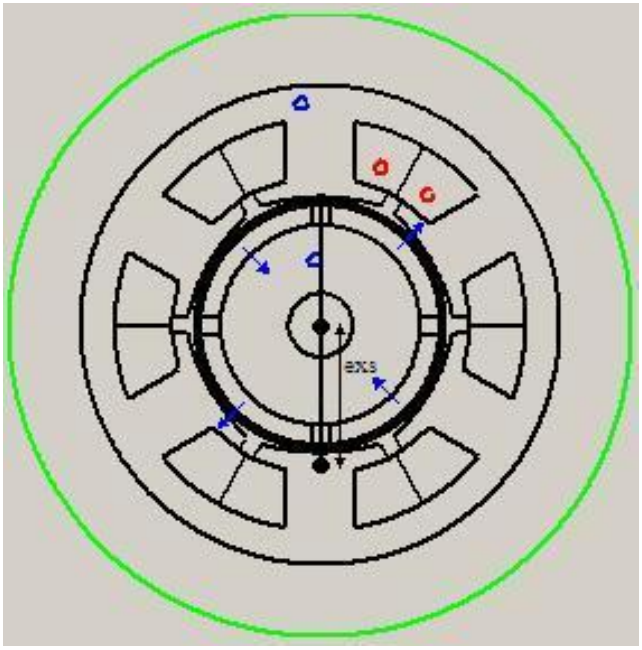


Figure 5.4: The motor model with $exs = 10$

The box constraints are given by

$$\underline{x} = \begin{pmatrix} 23 \\ 6.5 \\ 1 \\ 70 \\ 0 \end{pmatrix} \quad \text{and} \quad \overline{x} = \begin{pmatrix} 37 \\ 10.5 \\ 4 \\ 90 \\ 20 \end{pmatrix}. \tag{5.2}$$

The figure 5.4 shows the motor for the initial value right in the middle of the box constraints for the five parameters, i.e.

$$x0 = \begin{pmatrix} 30 \\ 8.5 \\ 2.5 \\ 80 \\ 10 \end{pmatrix}.$$

Moreover, it has to be mentioned that instead of the problem with 2 design parameters, the problem with 5 design parameters requires the consideration of the whole motor (360°), what comes from changing the excentricity and the angle $phiMagnet$. Nevertheless, the motor has to be turned only 180° in the motor simulation because the motor has still two pole pairs.

The following pictures show the motor for the problem with 5 design parameters for the initial value, once for the case $I = 0$ and then for $I \neq 0$.

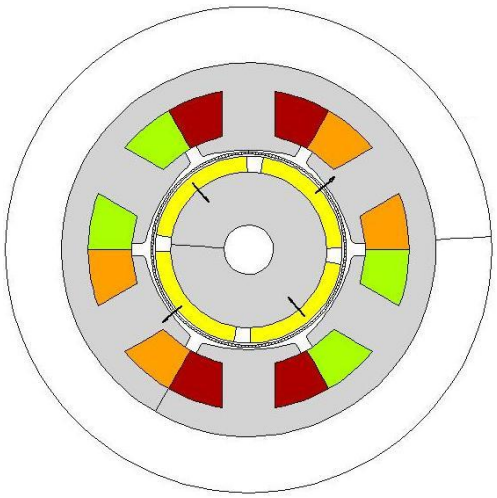


Figure 5.5: The motor for the initial value of the problem with 5 design parameters with $I = 0$

Its winding function and torque is illustrated in the next picture.

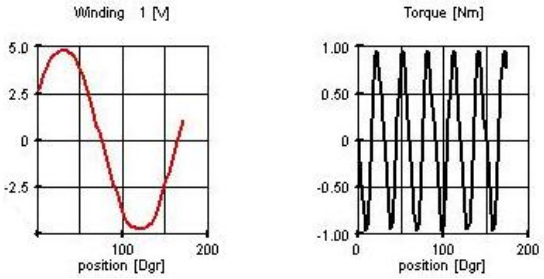


Figure 5.6: Winding function and torque of the motor above

The following picture shows the motor for the case $I \neq 0$.

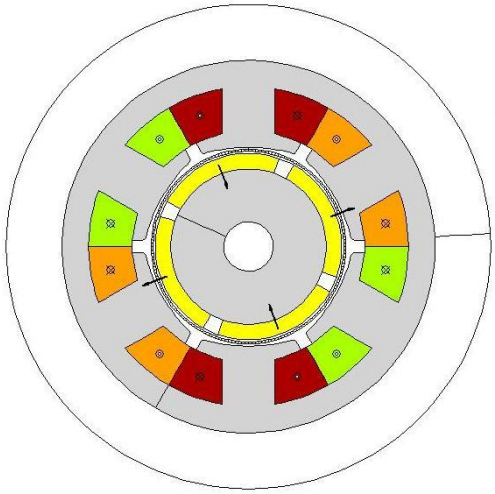


Figure 5.7: The motor for the initial value of the problem with 5 design parameters with $I \neq 0$

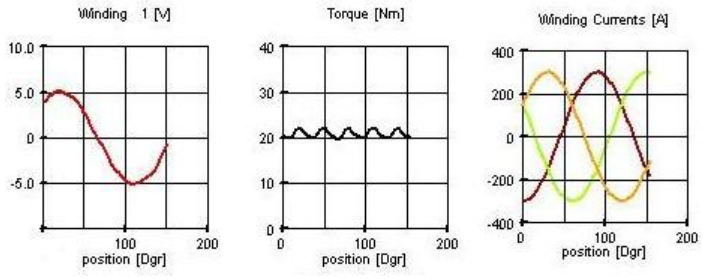


Figure 5.8: Winding function, torque and winding currents of the motor above

5.2 Numerical Results

The following results are the one using the strategy of the black box optimization, which has been described in the problem with 2 design parameters and has attained the best result. Hence, the options in *fmincon* are the same as in chapter 4. Altogether for the problem with 5 design parameters, 15 stepsizes are needed: 5 for the central difference quotients, which approximate the gradient,

$$h = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{pmatrix}$$

and analogously, there are the 5-dimensional vectors s, which contains the stepsizes for the pure second derivatives, and l containing the stepsizes for the mixed second derivatives of the Hessian. The additional shortcut time_{par} stands for the time, where 61 parallel processors are available, and the shortcut *funevals* stands for the total number function evaluations. This leads to the result:

ϵ_A	0.00545
h	(0.6854, 0.1044, 0.1744, 2.6347, 1.2715)
s	(0.6204, 0.0985, 0.1553, 0.8334, 0.6314)
l	(0.2297, 0.2081, 0.1510, 0.7385, 0.8909)
x	(30.0279, 7.9960, 3.8748, 72.1918, 10.2572)
$f(x)$	0.0273
n_{iter}	4
time	21 h
time_{par}	20 min
funevals	244

The optimization algorithm was stopped after the fourth iteration because this iteration step lasted too long.

If we assume that the optimization process is computed with parallel processors, then we can determine time_{par} by dividing time by the number of available processors. In each iteration step of the black box optimization, 61 processors are needed to compute all function values - the function value itself and the function values for the approximation of gradient and Hessian for this value - simultaneously.

To compare the results from the black box optimization, results by using other methods are needed. For that, we choose a genetic algorithm from the book [7]. This optimization process is computed with parallel processors.

By optimizing with the genetic algorithm, we have got the following results, where firstly 40 and secondly 2000 function evaluations have been obtained:

x	(25.6617, 8.8089, 1.6726, 75.4927, 2.3989)
$f(x)$	0.0325
time_{par}	30 min
funevals	40

x	(25.6617, 7.3959, 2.1001, 83.1461, 6.0644)
$f(x)$	0.0208
time_{par}	25 h
funevals	2000

Although the final value of the cost functional of the black box optimization is not as good as the one from optimizing with the genetic algorithm, compared to the 2000 function evaluations of the genetic algorithm, the number of function evaluations is only about a tenth of this number. Altogether the black box optimization has been less costly and hence this is an advantage of using this method. Moreover, if there are 61 parallel processors available, then the computation time is only about 20 minutes, which is much faster then the 25 hours of the optimization by the genetic algorithm. Compared to the genetic algorithm, where only 40 function evaluations have been computed and which have lasted only 30 min, then the black box optimization has obtained a better result in 20 min, but with more function evaluations.

Now, some pictures of the shape optimized motor for the problem with 5 design parameters follow. In comparison with the motor for the initial value, the maximum square deviation of the cogging and the load torque has again become smaller.

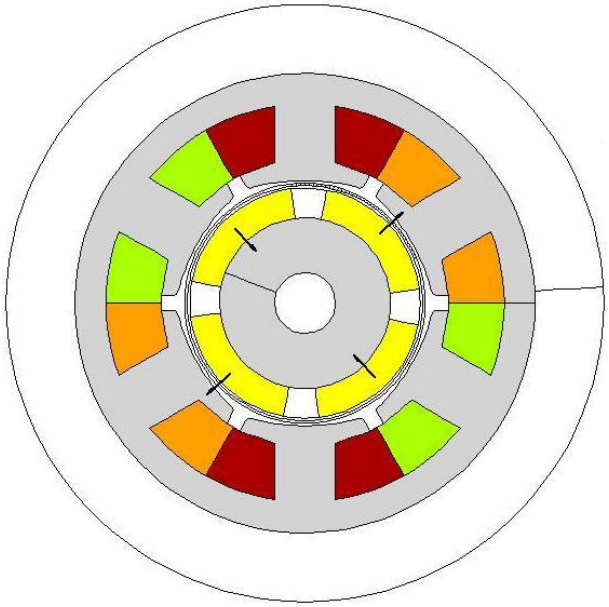


Figure 5.9: The shape optimized motor for the problem with 5 design parameters with $I = 0$

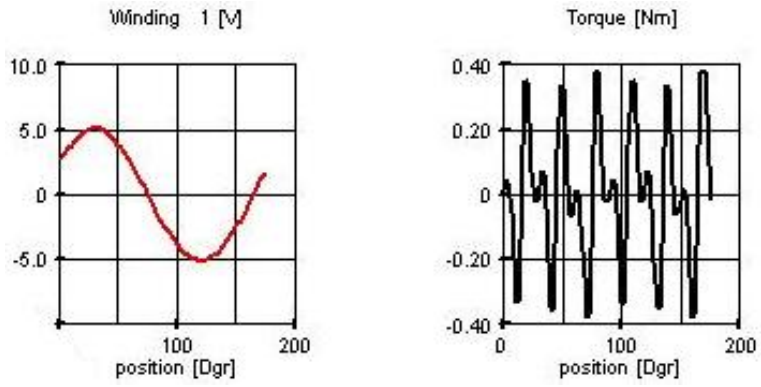


Figure 5.10: Winding function and torque of the motor above

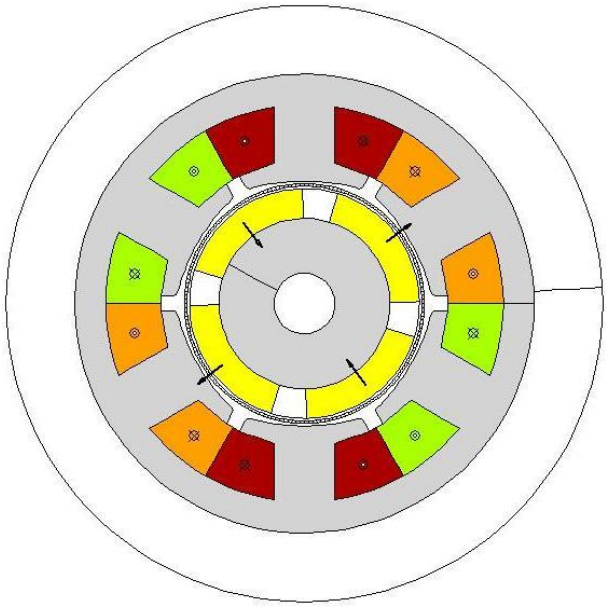


Figure 5.11: The shape optimized motor for the problem with 5 design parameters with $I \neq 0$

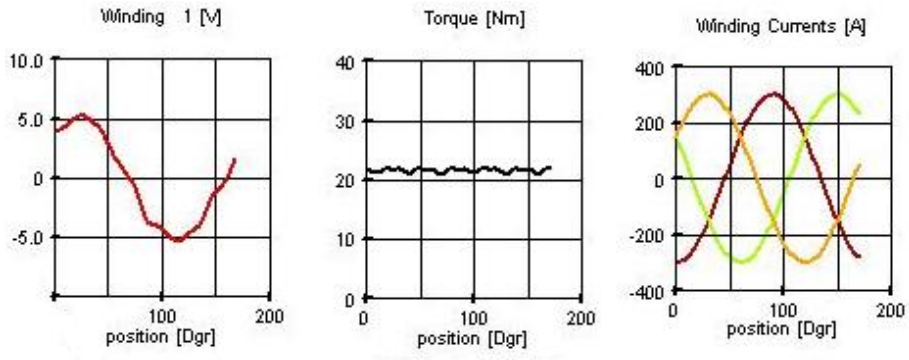


Figure 5.12: Winding function, torque and winding currents of the motor above

Finally, the problem with 8 design parameters is described and numerical results from the black box optimization are presented and are also compared to the results of optimizing with the genetic algorithm.

Chapter 6

A Problem with 8 Design Parameters

6.1 Problem Description

Finding the optimal shape of a given motor by optimizing the cost functional with respect to 8 parameters is a problem from real application. The optimization problem with 8 design parameters, which describes our model PMSM, is stated by

$$\begin{aligned} &\min_{x\in\mathbb{R}^8} \quad f(x) \\ &\begin{pmatrix} \underline{x_1} \\ \underline{x_2} \\ \underline{x_3} \\ \underline{x_4} \\ \underline{x_5} \\ \underline{x_6} \\ \underline{x_7} \\ \underline{x_8} \end{pmatrix} \leq x \leq \begin{pmatrix} \overline{x_1} \\ \overline{x_2} \\ \overline{x_3} \\ \overline{x_4} \\ \overline{x_5} \\ \overline{x_6} \\ \overline{x_7} \\ \overline{x_8} \end{pmatrix}. \end{aligned} \tag{6.1}$$

Now we have the already known 5 parameters and 3 new parameters. Altogether they are 8 parameters, with respect to whom the cost functional f is optimized. Finally, the parameters

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} dra \\ bst \\ hm \\ phiMagnet \\ exs \\ bsj \\ phisz \\ hs \end{pmatrix}$$

are the outer diameter of the rotor ($x_1 = dra$), the width of the stator cog ($x_2 = bst$), the height of the rotor magnet ($x_3 = hm$), the angle of the rotor magnet ($x_4 = phiMagnet$), the excentricity of the stator ($x_5 = exs$), the minimal yoke width of the stator ($x_6 = bsj$), the pole shoe angle of the stator ($x_7 = phisz$) and the slot height of the stator ($x8 = hs$).

The three new parameters are illustrated by the following pictures:

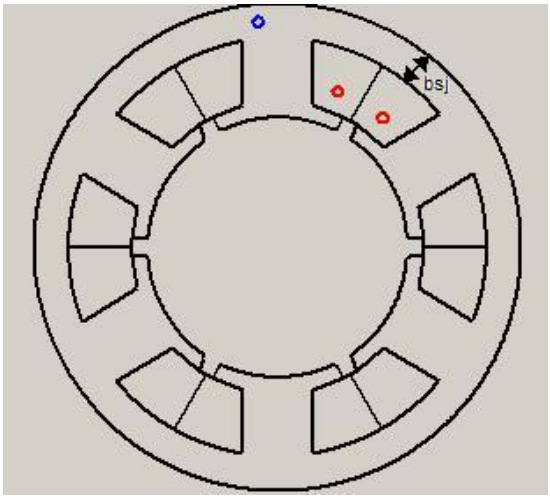


Figure 6.1: The minimal yoke width of the stator *bsj*

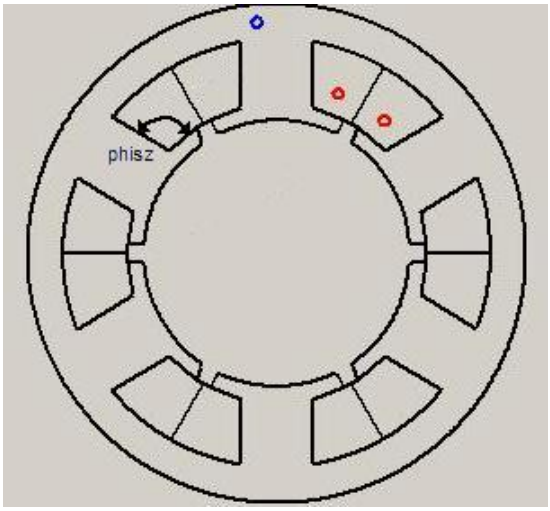


Figure 6.2: The pole shoe angle of the stator *phisz*

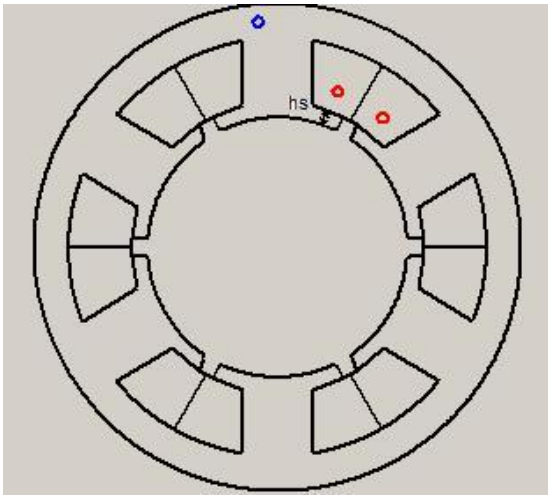


Figure 6.3: The slot height of the stator hs

The box constraints are again chosen by the engineers. They are

$$\underline{x} = \begin{pmatrix} 23 \\ 6.5 \\ 1 \\ 70 \\ 0 \\ 3.25 \\ 100 \\ 1 \end{pmatrix} \quad \text{and} \quad \overline{x} = \begin{pmatrix} 37 \\ 10.5 \\ 4 \\ 90 \\ 14 \\ 5.25 \\ 130 \\ 3 \end{pmatrix}. \tag{6.2}$$

The initial value chosen in the middle of the box constraints is

$$x0 = \begin{pmatrix} 30 \\ 8.5 \\ 2.5 \\ 80 \\ 7 \\ 4.25 \\ 115 \\ 2 \end{pmatrix}.$$

With the initial value we get the following motor, which will be shown in the next pictures.

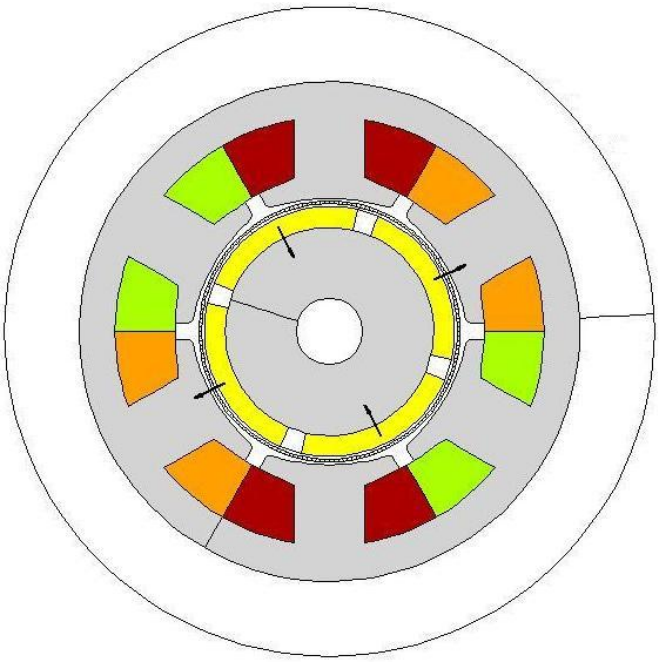


Figure 6.4: The motor for the initial value of the problem with 8 design parameters with $I = 0$

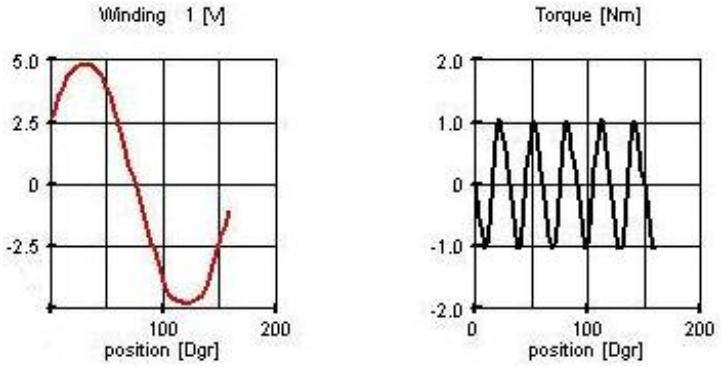


Figure 6.5: Winding function and torque of the motor above

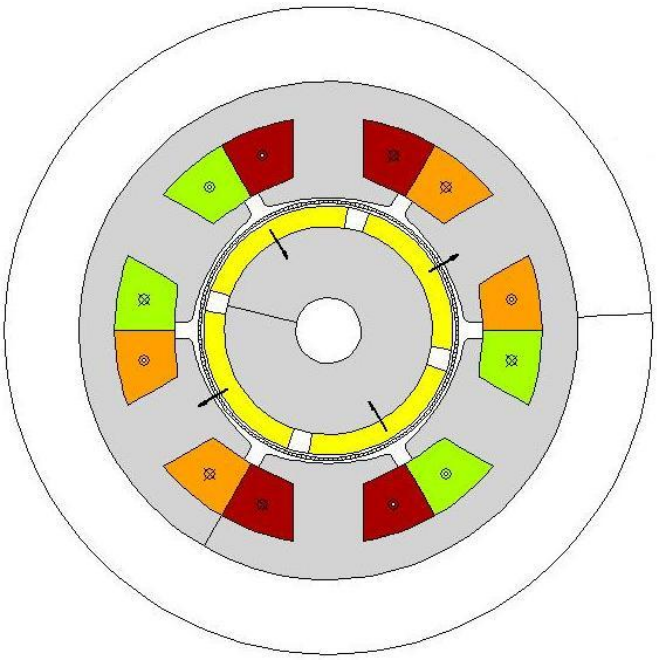


Figure 6.6: The motor for the initial value of the problem with 8 design parameters with $I \neq 0$

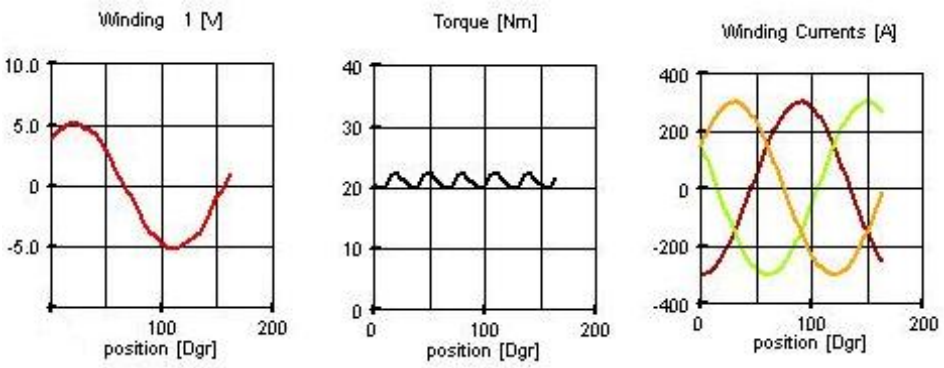


Figure 6.7: Winding function, torque and winding currents of the motor above

Now, the final numerical results of the problem with 8 design parameters are presented and they are compared to results which come from optimizing by the genetic algorithm from [7], which has been also done for the problem with 5 design parameters.

6.2 Numerical Results

Starting with the results of the black box optimization, this section considers also the results from the genetic algorithm in comparison with the results of the black box optimization. It is again important to compare the total number of function evaluations of both methods.

The optimization process of the genetic algorithm is computed parallel and therefore these results will be compared with the results from the black box optimization assuming to have the best number of parallel processors available. In the case with 8 design parameters, 145 parallel processors are the best number of available parallel processors such that in each iteration step all needed function values can be computed - also the one for the difference quotients for approximating gradient and Hessian - at the same time.

Now, we come to the black box optimization. The stepsizes were chosen again by the strategy, which has been explained in the problem with 2 design parameters, and all options in *fmincon* are set according to chapter 4 as well. Altogether for the problem with 8 design parameters, 24 stepsizes are needed to approximate the gradient and the Hessian: 8 for approximating the gradient $h = (h_1,\ h_2,h_3,\ h_4,\ h_5,\ h_6,\ h_7,\ h_8)$, 8 for the pure second derivatives $s = (s_1,\ s_2,\ s_3,\ s_4,\ s_5,\ s_6,\ s_7,\ s_8)$ and 8 for the mixed second derivatives $l = (l_1,\ l_2,\ l_3,\ l_4,\ l_5,\ l_6,\ l_7,\ l_8)$.

The options were chosen such that the optimization algorithm stops after maximal 10 iterations. Now, the shortcut time_{par} stands for the time, if we assume to have 145 parallel processors available. This strategy leads to the following result:

ϵ_A	0.00272
h	(0.7116, 0.1560, 0.2262, 0.1617, 0.1037, 0.0717, 0.2863, 0.0781)
s	(0.4098, 0.1298, 0.1037, 1.4346, 1.3824, 0.0682, 1.1687, 0.0604)
l	(1.7536, 0.0505, 0.0823, 0.5719, 0.2193, 0.0359, 0.4591, 0.0416)
x	(23.0007, 6.7256, 2.8955, 74.8980, 10.7795, 3.2518, 127.516, 2.6902)
$f(x)$	0.0146
n_{iter}	10
time	144.5 h
time_{par}	1 h
funevals	1450

The algorithm terminated because the maximum number of function evaluations had been exceeded.

By optimizing with the genetic algorithm, the following results have been obtained, where firstly 1500 and secondly 4000 function evaluations are computed:

x	(24.3335, 7.3769, 3.9965, 80.7409, 1.0108, 4.4074, 125.7691, 1.4003)
$f(x)$	0.0185
time_{par}	18.75 h
funevals	1500

x	(23.9132, 7.1836, 3.4304, 77.0381, 12.1216, 3.3553, 128.5629, 2.5719)
$f(x)$	0.0137
time_{par}	50 h
funevals	4000

Comparing the result of the black box optimization to the result of the genetic algorithm with an approximate number of function evaluations, namely 1450 and 1500, we obtain a better result with the black box optimization because its final value of the cost functional is *0.0146*, but the one of the genetic algorithm is *0.0185*. Moreover, optimizing with the genetic algorithm lasted *18.75 hours* by using parallel processors. The time of the black box optimization was *144.5 hours*, but if there were 145 parallel processors available, the optimization would last only *1 hour*. Even if there were not 145 parallel processors available but 50 - a reasonable number of available parallel processors, then the time of the black box optimization would be *3 hours*, a sixth of the optimization time with the genetic algorithm. Altogether, we have shown by comparing our results from black box optimization to the one of the genetic algorithm that the black box method is a good choice for the shape optimization of a motor.

Now, some pictures of the final results from black box optimization follow.

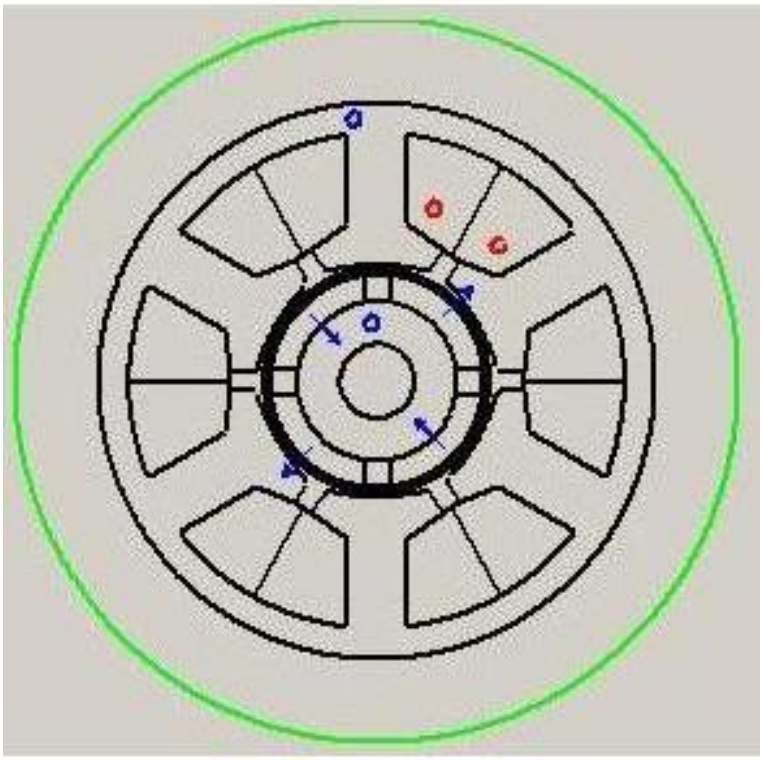


Figure 6.8: The shape optimized motor for the problem with 8 design parameters

The next picture shows the mesh:

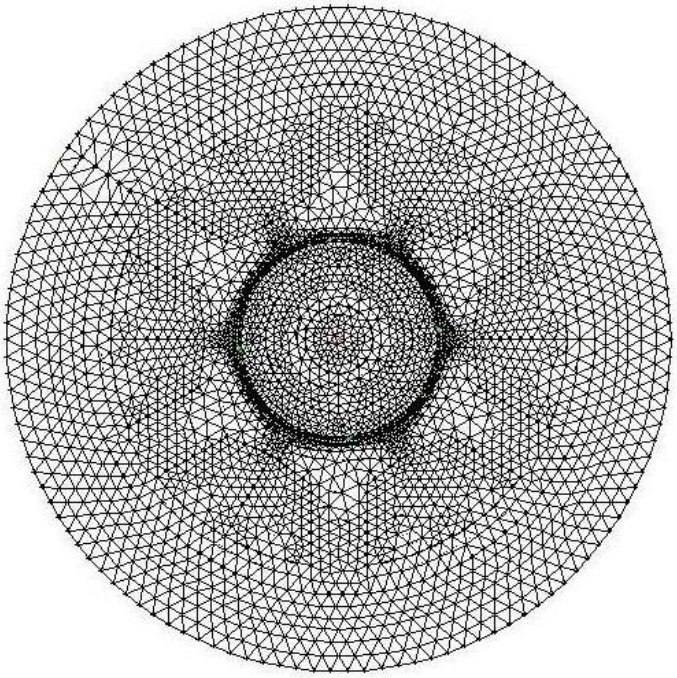


Figure 6.9: The mesh of the shape optimized motor during the FE-simulation

Every motor simulation consists of two important parts. Firstly, the magnetostatic problem is solved for every rotor position relative to the stator depending on the angle for the case $I = 0$ and then for the case $I \neq 0$. Again as for the problem with 5 design parameters, this has to be done only through 180°. The following pictures show the motor during the simulation, once in the case $I = 0$ and then in the case $I \neq 0$. Here, it is important to look again at the torque and how it has changed after comparing it to the motor at the beginning of the problem with 8 design parameters for the initial value.

In the case $I = 0$ as well as in the case $I \neq 0$, the maximum square deviation of the torque of the shape optimized motor is much more smaller than the torque of the motor for the initial value. The function seems to be almost constant. According to the engineers the result for the shape optimized motor is a reasonable one.

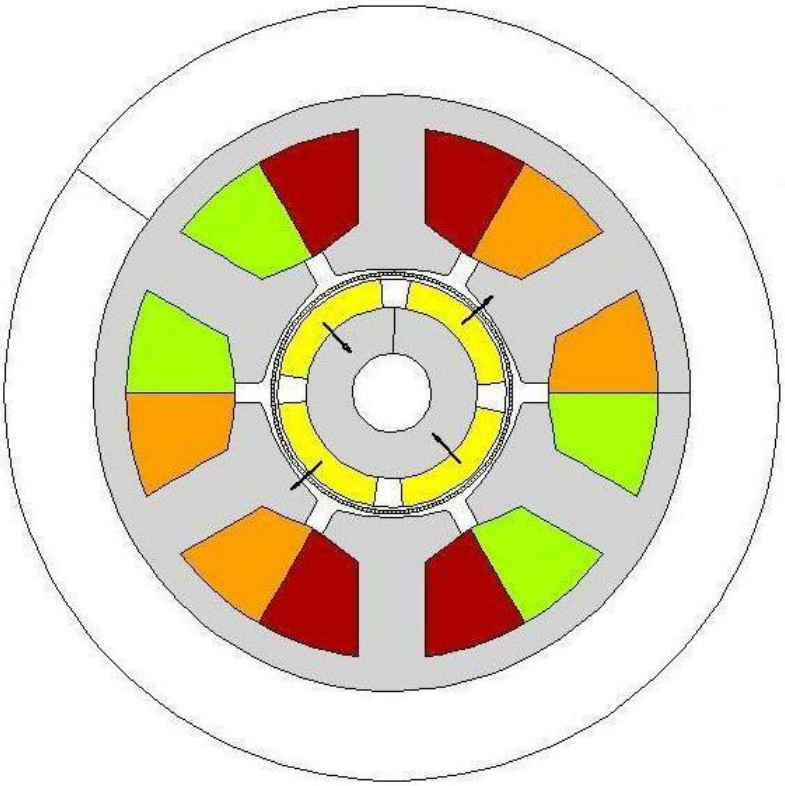


Figure 6.10: The shape optimized motor for the problem with 8 design parameters with $I = 0$

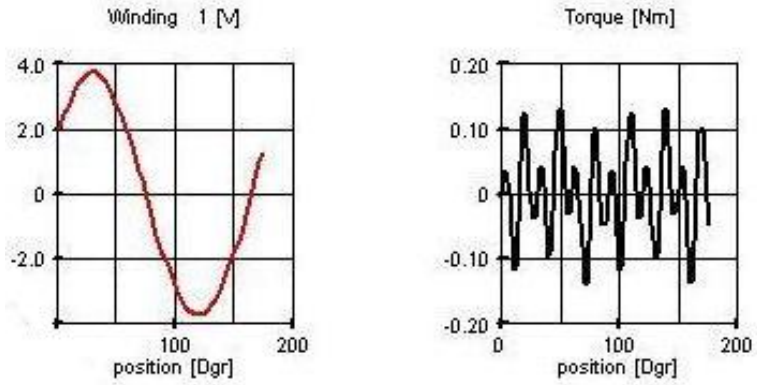


Figure 6.11: Winding function and torque of the motor above

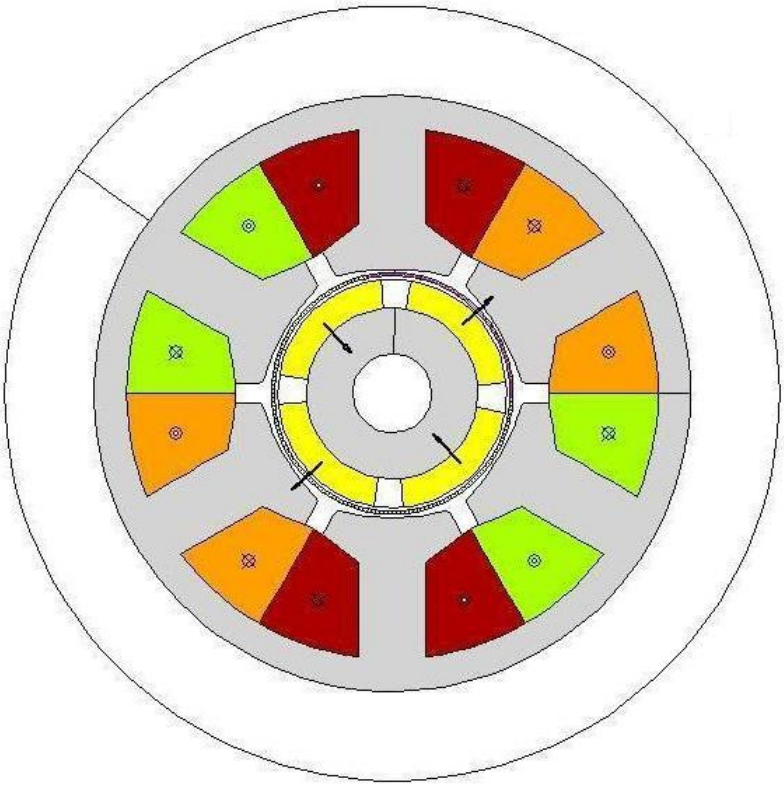


Figure 6.12: The shape optimized motor for the problem with 8 design parameters with $I \neq 0$

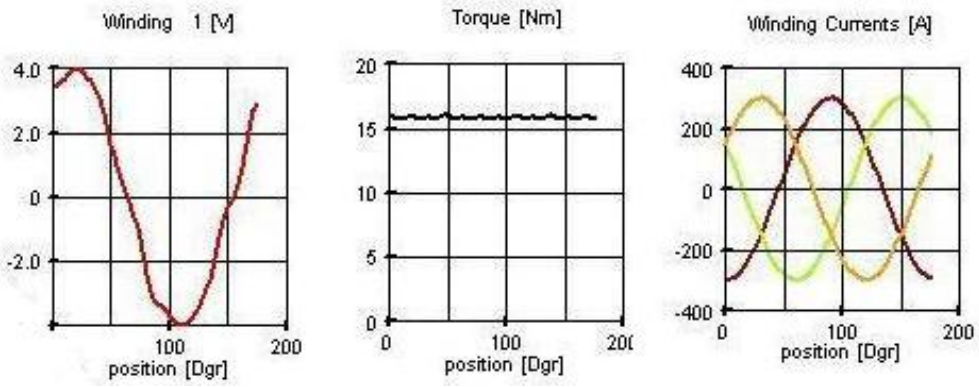


Figure 6.13: Winding function, torque and winding currents of the motor above

Chapter 7

Conclusion and Outlook

The thesis describes the shape optimization of a motor by black box simulations. The practical problems in this thesis are given by the ACCM (Austrian Center of Competence in Mechatronics), where the task was to determine the design of a motor such that a cost functional, which has been given by the engineers from the group of electric drives, is minimized while satisfying given constraints. The analytic definition of the cost functional is not available but all function values can be computed. Hence, the choice of optimizing by black box simulations is a natural one.

Firstly, the numerical background for a problem with n design parameters was described by developing it for problems with one and two design parameters. The mathematical background for the optimization was mainly based on the book *Practical Optimization* by Philip E. Gill, Walter Murray and Margaret H. Wright, which is [6], and also on [9] and [8]. For the optimization, the MATLAB routine *fmincon* was used together with a software from the group of electric drives.

As next, we tried to find the optimum of a real application problem with 2 design parameters. Various numerical results were presented, which arise from the use of *fmincon*. Results were shown with the availability of parallel processors as well. After the detailed treatment of the problem with 2 design parameters, there was made a proposal how to automatize the whole black box optimization process.

After the consideration of the problem with 2 design parameters, the shape optimization of the already given motor was extended to an optimization with respect to 5 parameters. The results were compared to results from optimizing with a genetic algorithm.

Finally, the motor was optimized with respect to 8 design parameters. The motor was again optimized by the genetic algorithm as well. Altogether for the problems with 5 and 8 design parameters, the outcome was very good. With the availability of parallel processors the time by optimizing with black box simulations would be less than by optimizing with the genetic

algorithm. If we compare the results of both methods by assuming to have around the same number of function evaluations and parallel processors, then the results of the problems with 5 and 8 design parameters are better. Nevertheless, the disadvantage of the black box optimization method in this thesis is that this method uses a Trust-Region-Reflective algorithm, which is based on the interior-reflective Newton method. That means that the optimization is gradient based and local. Although the shape optimization based on black box simulations worked quite well in this thesis and provided good results, one has never to forget that the method generally determines a local optimum.

Altogether, this thesis covers the most important parts of optimizing the shape of a motor by black box simulations. The conclusion of optimizing with a black box method is that this procedure works well and can be used for real application problems. It is a method, which can be applied on different problems and provides good results in the field of electric drives.

Bibliography

[1] http://www.wikipedia.org. *Shape optimization, Electric motor, Torque, Copper loss, Total harmonic distortion, Finite difference*, 2009/10.

[2] MATLAB 7.6 Help. *Optimization Toolbox (fmincon)*, 2009.

[3] Wolfgang Amrhein. *Lecture Notes for the course Elektrische Maschinen*. Johannes Kepler Universität Linz, 2009.

[4] Rolf Fischer. *Elektrische Maschinen*. Carl Hanser Verlag München Wien, 2006.

[5] Helmut Gfrerer. *Lecture Notes for the course Optimierung*. Johannes Kepler Universität Linz, 2007.

[6] Philip E. Gill, Walter Murray, Margaret H. Wright. *Practical Optimization*. Academic Press, Inc., San Diego, CA, 1981.

[7] Hartmut Pohlheim. *Evolutionary Algorithms: Overview, Methods and Operators Documentation for: Genetic and Evolutionary Algorithm Toolbox for use with Matlab*. http://www.geatbx.com/, 1999.

[8] Robert D. Richtmyer, K. W. Morton. *Difference Methods for initial-value problems*. Interscience Publishers a division of John Wiley & Sons, Second edition, 1967.

[9] Walter Zulehner, Ewald Lindner. *Lecture Notes for the course Numerische Analysis*. Johannes Kepler Universität Linz, 2005/06.

Eidesstattliche Erklärung

Ich, Monika Kowalska, erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, Mai 2010

_____

Monika Kowalska