Framework for automated optimization of FEM design of permanent magnet motor bearing unit
Atte Putkonen
ABSTRACT

Lappeenranta University of Technology
LUT School of Energy Systems
Electrical Engineering

Atte Putkonen

Framework for automated optimization of FEM design of permanent magnet motor bearing unit

2017

Bachelor’s Thesis.
16 p.

Examiner: D.Sc. Rafal Jastrzebski

Optimizing a motor usually requires many stages from building the model and setting the conditions to very time consuming finite element analysis of the electromagnetic behaviors. Usually FEA needs to be ran several times to the most optimized results.

The first goal of this work is to realize communication between MATLAB (MathWorks) and JMAG-Designer (JSOL Corporation) which is a major requirement for the second goal of this work. The second goal of this work is to present a framework to automatically optimize a desired motor’s parameters. Calculations and recursion are performed using MATLAB. The optimization is done using differential evolution. Electromagnetic analysis is performed using 3rd party program, which is JMAG Designer in this case study.

The framework is tested on a magnetic levitation realized a permanent magnet linear synchronous machine. The objective is to minimize thrust and normal force ripples. The parameters to optimize are motor’s end tooth’s dimensions. After running the script, the simulations show that the thrust and normal force ripples are reduced ~52-80 %.

Keywords: Automated optimization, permanent magnet, linear synchronous motor, FEM, finite element method, force ripple, minimization
TABLE OF CONTENTS

Abbreviations and symbols ........................................................................................................... 4
1. Introduction ................................................................................................................................. 5
2. Communication between JMAG and MATLAB ................................................................. 7
3. Execution chain ......................................................................................................................... 8
4. Example case study ................................................................................................................... 10
5. Simulation results .................................................................................................................... 13
6. Conclusions .............................................................................................................................. 15
References ..................................................................................................................................... 16
ABBREVIATIONS AND SYMBOLS

CAD  Computer-aided design
DE   Differential evolution
FEM  Finite element method
FEA  Finite element analysis
MS-DOS  Microsoft disc operating system
PM   Permanent magnet
LFSPM Linear Flux-Switching Permanent Magnet Levitated Motor
UI   User interface

$F$  force
$k$  weight coefficient
$w$  width

Subscripts

nAvg  normal average
nRipple  normal ripple
max  maximum
min  minimum
thAvg  thrust average
thRipple  thrust ripple
1. INTRODUCTION

Bearingless motors can offer many advantages in various industrial applications compared to traditional electrical motors with rotors on mechanical bearings. The major advantages include reduced need for maintenance, lubrication free operation, more compact machine construction compared to separate motor and AMBs, and high reliability (Tir and Mirimani, 2014). Optimization of a motor’s parameters is a central part of a motor design to improve the cost and efficiency ratio (Sashidhar and Fernandes, 2015). The parameters can be e.g. geometrical, material choices, number of turns in coils. Due the complexity of analytical methods when calculating electromagnetic fields and forces in the motor, the finite element method (FEM) is often used. For more precise results of finite element analysis (FEA) it is necessary to use an accurate model of the motor (Tir and Mirimani, 2014).

Depending on the application, different objectives in machine design are considered. In majority of applications the maximizing the performance in terms of high torque or force capacity as well as minimizing torque ripple and high efficiency in the operating conditions are required. In particular, in bearingless motor designs small variations of force amplitudes and force error angles are desired. Because of multiple often contradictory requirements the designer faces challenging design optimization. In order to alleviate the multi-objective design process post-processing of the FEA results may be needed for:

a) calculating new initial values for next iterations if desired conditions aren’t met or more precise result is needed
b) comparing the results and finding the optimized interpolated design parameter values
c) utilizing various design optimization methods that use the FEM obtained results.

The goal of this bachelor’s thesis is to implement a framework for automated optimization of FEM design for permanent magnet (PM) motor bearing unit. One of the research questions is as follows: “Can FEM software be controlled from MATLAB to automate bearingless motor design?” If the optimization process is done manually it would require all steps, such as FEA, importing and processing FEA results to MATLAB, possible re-running of FEA, to be done separately, which is why the process should be automated.

The research method is based on FEM simulations of the electromagnetic behavior of a test motor. The simulation of the motor and the electromagnetic behavior results are calculated
in a FEA software “JMAG-Designer” provided by JSOL Corporation. The rest of the functionality is done with MATLAB provided by MathWorks. In this work the main parts of the MATLAB script are also presented based on the framework. The script’s functionality is demonstrated with an example motor. In this thesis, the example motor is chosen to be a linear permanent magnet motor to keep simplicity of the optimization problem within limits.
2. COMMUNICATION BETWEEN JMAG AND MATLAB

In this section the basic procedure of software communications is presented. As stated before MATLAB is intended for using as operating software and user interface (UI) in this work whereas JMAG-Designer (version 15.0, published in 2016) is used as FEA tool. Since MATLAB and JMAG are provided by separate corporations, a communicating tool between these software doesn’t exist.

JMAG has a feature for using user-defined scripts for processing from model creation to results display. JMAG supports three different script languages: VBScript, Python or Jscript. The script can be generated in MATLAB dynamically based on the desired cases. JMAG can then be executed and ran with the script’s commands through MATLAB’s dos-command which executes an MS-DOS command for Windows platforms.

For importing the results JMAG provides a console application “jquery.exe” for extracting the result data from binary to text (*.csv or *.txt). For it being a console application, it can be executed with the dos-command. For input variables “jquery.exe” requires the path of the JMAG project file and component to be extracted. An export path is also recommended. MATLAB can then be used to import the data from the exported file. Alternatively, the results can be exported by scripting.

The communication of MATLAB and JMAG is illustrated in Fig 2.1.

![Fig. 2.1. A representation of the objective function running in MATLAB, which also illustrates the communication between MATLAB and JMAG. The blocks represent which software has the control at which stage of the execution.](image-url)
3. EXECUTION CHAIN

In this section a general presentation of the framework stages is introduced. More detailed stages are shown and explained in the next section. It should be noted that the upcoming presentation of the execution chain is developed with JMAG so it may vary with other FEA software tools. This section assumes that the FEM model is already built correctly with all required conditions.

The execution chain consists of several stages (of which some cannot be automated). At first, the variable parameters need to be exported to a file and then imported into MATLAB. This is necessary because certain parameters, especially CAD parameters, have variable names. That said, it’s recommended to check that the CAD parameter names are correct. The limits for the parameters need to be defined.

The basic info about the JMAG project file, such as model name and study name, needs to be defined for the python script. In addition, dos command needs the path of the ‘JMAG-designer.exe’ file which starts the JMAG application. Different cases are built by choosing random values from within the limits of the parameters. The script writing function adds cases to the script file dynamically every iteration. The most time-consuming part user-wise is the initialization of objective function (and the possible plotting function). Since it is dependent on the application it needs to be done by hand.

After initialization, the rest of the stages are automated. MATLAB uses dos command to run JMAG’s FEA with the initial cases. The results are imported with JMAG’s own ‘jquery.exe’ tool which is again ran with dos command. The result files in JMAG are binary files so the jquery.exe converts them to numbers and saves them to a *.csv-file. The optimization loop compares the results with a user-defined performance function and sets new parameters from within the limits for FEA until conditions are met.

So, assuming the model is correctly built in JMAG, the execution chain of framework can be summarized in 7 main stages.

1. Initialize parameters and set limits for optimized variables and build the performance function for minimization
2. Run FEM simulations
3. Import results from JMAG to MATLAB
4. Evaluate weighted performance function
5. Set new values to parameters based on optimization routine
6. Repeat 2-5 until limit of iterations or weighted performance function < limit
7. Generate a final report

The execution chain’s flowchart is illustrated in Fig 3.1.
4. EXAMPLE CASE STUDY

In this section the framework of this study is demonstrated on an example case motor which is also introduced. The example case for this framework is linear flux-switching permanent magnet levitated motor (LFSPM). However, the script is easily extendable for different types of motors. LFSPMs are similar to a corresponding rotating motor that is “rolled” open. LFSPMs are applicable to any applications where linear movements are needed.

For the FEA, a 2D model is built up. The initial cross section of the LFSPM is shown in Fig. 4.1. More precise dimensions of the core can be found in (Jastrzebski, Jaatinen and Pyrhönen, 2016). The main constant values are shown in table 4.1.

![Initial structure of the example LFSPM. The letters above coils represent different phases of the corresponding coil. Arrows represent the magnetization direction of the corresponding magnet.](image)

**Table 4.1.** Constant parameters of motor’s dimensional values and circuit conditions.

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of phase</td>
<td>3</td>
</tr>
<tr>
<td>Current amplitude of normal force simulations (A)</td>
<td>±2</td>
</tr>
<tr>
<td>Phase θ of normal force simulations</td>
<td>233.63°</td>
</tr>
<tr>
<td>Current amplitude of thrust force simulations (A)</td>
<td>4</td>
</tr>
<tr>
<td>Phase θ of thrust force simulations</td>
<td>143.63°</td>
</tr>
<tr>
<td>Frequency (Hz)</td>
<td>1/46.4e-3</td>
</tr>
<tr>
<td>Mover’s constant velocity (m/s)</td>
<td>1</td>
</tr>
<tr>
<td>Air-gap length (mm)</td>
<td>2.5</td>
</tr>
<tr>
<td>Total height (mm)</td>
<td>85.8</td>
</tr>
<tr>
<td>Mover core material</td>
<td>SURA M270-35A</td>
</tr>
<tr>
<td>Mover core height (mm)</td>
<td>61.3</td>
</tr>
<tr>
<td>Mover core length (mm)</td>
<td>644.35</td>
</tr>
<tr>
<td>Stator material</td>
<td>SURA M270-35A</td>
</tr>
<tr>
<td>Stator length (mm)</td>
<td>696</td>
</tr>
<tr>
<td>Stator teeth height (mm)</td>
<td>12</td>
</tr>
<tr>
<td>Stator teeth width</td>
<td>16.3</td>
</tr>
<tr>
<td>PM material</td>
<td>NMX-S34GH</td>
</tr>
<tr>
<td>PM thickness (mm)</td>
<td>10.85</td>
</tr>
<tr>
<td>PM height (mm)</td>
<td>41</td>
</tr>
<tr>
<td>Number of turns per coil</td>
<td>90</td>
</tr>
<tr>
<td>Resistance per coil (ohm)</td>
<td>2.125</td>
</tr>
</tbody>
</table>
Force ripple results from electromagnetic effects, which cause periodic variations to the force (Bianchi and Bolognani, 2003). The goal is to optimize dimensions of end tooth of the mover core in respect of force ripple minimization but also to keep thrust/normal force average values in mind. The variable dimensions are shown in Fig. 4.2. The other dimensions remain unchanged.

![Diagram of variable dimensions of the end tooth.](image)

**Fig 4.2.** Variable dimensions of the end tooth. Other end of the motor has same dimensions mirrored. Height of the block is constant 45 mm. Initial dimensions are \( w_1 = 39.55 \text{ mm}, w_2 = 0 \text{ mm} \) and \( w_3 = 10.7 \text{ mm} \).

Widths in Fig 4.2 have limitations: \( w_1 \) is limited between 31.55 – 39.55 mm, \( w_2 \) between 1.0 – 9.08 mm and \( w_3 \) between 12.7 -21.4 mm. The optimization is done with classical differential evolution (DE) optimizer proposed in (Price, Storn and Lampinen, 2005). Maximum iteration for the DE is limited to 10. The DE uses population size of 5 so the script chooses the best dimensions from 50 different cases. Parameters and cases for J MAG of one function evaluation is shown in table 4.2.

<table>
<thead>
<tr>
<th>Case</th>
<th>width1 [mm]</th>
<th>width2 [mm]</th>
<th>width3 [mm]</th>
<th>Current phase [*]</th>
<th>Current amplitude [A]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36.09</td>
<td>5.12</td>
<td>16.36</td>
<td>233.63</td>
<td>-2</td>
</tr>
<tr>
<td>2</td>
<td>36.09</td>
<td>5.12</td>
<td>16.36</td>
<td>233.63</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>36.09</td>
<td>5.12</td>
<td>16.36</td>
<td>143.63</td>
<td>4</td>
</tr>
</tbody>
</table>
Cases 1 and 2 simulate two opposite armatures with opposite current values so the total normal force is sum of those cases. (4.1) is the cost function for the DE to minimize.

$$J = [k_1 \ k_2 \ k_3 \ k_4] \star \begin{bmatrix} F_{\text{thRipple}} \\ F_{\text{nRipple}} \\ F_{\text{thAvg}} \\ F_{\text{nAvg}} \end{bmatrix}$$

(4.1)

where \( k_1 = k_2 = 0.45 \), \( k_3 = 0.06 \) and \( k_4 = 0.04 \). Ripples are calculated with (4.2). \( F_{\text{thRipple}} \) is sum of the thrust force ripples of all cases, \( F_{\text{nRipple}} \) is ripple of sum of normal forces of cases 1 and 2 plus ripple of case 3, \( F_{\text{thAvg}} \) is average value of thrust force of case 3 and \( F_{\text{nAvg}} \) is average value of normal force of sum of cases 1 and 2.

$$F_{\text{Ripple}} = |F_{\text{max}} - F_{\text{min}}|$$

(4.2)
5. SIMULATION RESULTS

In this section the framework presented in the section 3 is applied to the LFSPM presented in the previous section. Step number is set to 30 and the end time is set to 0.0522 s in JMAG. The cost function (4.1) values on every function evaluation after running the framework script are shown in Fig 5.1.

![Cost function J vs function evaluation graph](image1)

**Fig 5.1.** The cost function’s values visualized on every function evaluation. The minimum value is gotten on evaluation number 42.

From Fig 5.1 at the evaluation 42 the dimensions of the end tooth are $w_1 = 34.4940$ mm, $w_2 = 2.7889$ mm and $w_3 = 17.0190$ mm. The motor’s structure with the optimized values is shown in Fig 5.2. The optimized motor’s average thrust force is approximately same as in the initial case but force ripple is reduced significantly, as shown in Fig. 5.3. (a)-(c). However, the absolute value of the average normal force is increased and ripple is also reduced very significantly, as shown in Fig 5.3 (d)-(e).

![Motor structure](image2)

**Fig 5.2.** The cross section of the optimized motor.
Fig 5.3. Force ripple comparisons between the initial and the optimized case. (a) Thrust force ripple of the upper armature is reduced ~59%. (b) Thrust force ripple of the lower armature is reduced ~64%. (c) Thrust force of the machine when $I = 4$ A is reduced over ~58%. (d) Sum of the upper and lower armature’s normal forces when $I = \pm 2$ A is reduced ~80%. (e) Normal force ripple is reduced ~52%. 
6. CONCLUSIONS

In this work, a framework for automated optimization of FEM design of permanent magnet motor bearing unit is presented and tested on LFSPM presented in section 3. The simulations show that the framework script has improved the operation of the motor with respect to the force ripples’ minimization. However, more precise results might have gotten if the step number and maximum iteration was set higher. Other operational problems of the motor are not considered since the optimization problem is kept simple. Also, the motor’s optimization hasn’t been the main goal of this work, therefore the number of iterations of the FEM simulations has been kept reasonably low.
REFERENCES


