CODEN:LUTEDX/(TEIE-5269)/1-56/(2010)

Industrial Electrical Engineering and Automation

Finite-Element Based Shape-Optimization of Electric Motors

Tobias Månsson

Dept. of Industrial Electrical Engineering and Automation
Lund University

ABB	Technical Report Corporate Research	SECRC/PT/TR-10/009

Issued by department	Last edit date	Lang.	Revision	Page
ZCRD/SECRC/PT	January 14, 2010	Eng	01	1/56

Customer	SECRC/PT	Classification	Public
Doc. title	**FEM-based shape-optimization of electric motors**		
Project name/ID	Thesis	Status of document	Final
Main Author	Tobias Månsson	Project ID (CRID)	CRIDxxxxx
Additional Author(s)		CR Program	EMMC
Distribution	Pär Ingelström		
Keywords	Electrical motor, FEM, geometry, optimization, rotor		
Attested by Reviewer	Pär Ingelström		
Approved by	Heinz Lendenmann	Final Approval Date	2010-01-14

Summary:

By introducing spectral or polynomial functions to perturb the shape of the rotor in an electrical machine that was previously optimized by conventional means, it was shown that performance parameters can be further improved, in some cases significantly.

This was accomplished by developing and applying an optimization method designed for use with a commercial finite element simulation software (Maxwell). This report describes the method and its development. Also issues the user must consider when applying the method, e.g. related to accuracy or choice of objectives, are discussed. Further, some results from the optimizations are presented - even though not all details can be published for confidentiality reasons.

The results shows a success in improving the objectives (e.g. torque ripple and/or power factor) while keeping other parameters of the motor unchanged (e.g. inductances). The method can take different load points into consideration, and works for different kinds of motors. The improvement can be significant, for one motor design, torque ripple and power factor were simultaneously improved: reduced by 13 %-units and increased by 2.5 %-units, respectively.

This is a public version of an internal ABB report, and hence some sources of information and results are confidential.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

CONTENTS

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

1 INTRODUCTION

1.1 Thesis Specification

The more powerful computers and finite element tools are getting, the more opportunities there are to utilize finite element based method for design and optimization of electrical machines.

Traditionally, most designs are made up by straight lines and arcs around the origin partly because this is what could be analyzed with available methods, and partly because these geometries are easier to construct. However, in some cases manufacturing is not restricted to simple geometrical shapes. The idea with this thesis work is to investigate how machine performance can be improved when more arbitrary geometric shapes using multiple variables are allowed for the machine design.

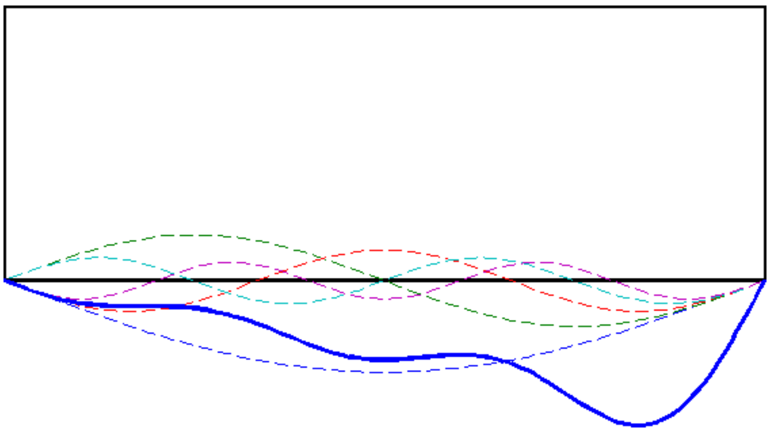


Figure 1.1: Perturbing a line by adding spectral functions

There are several questions to be answered:

- How should geometries be represented and parameterized?
- What optimization methods are good to use?
- What objective function(s) are good to use?

These questions likely have both practical and theoretical answers. The goal here is to find first experiences and rules of thumb. Another goal is to create a generalized program structure for applying this method on different kinds of motors.

1.2 Purpose

The purpose of this report is to gather all information necessary to investigate an optimization model for, to limit the scope, the shape of a rotor.

The target group for this report is varied, however motor design engineers already in possession of a by conventional means previously optimized motor model is an important group.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

In this case, finite element analysis with parameterized geometries is required to calculate objective functions. The objective functions are calculated using a secondary software, in this case MATLAB, while the simulations are run by a stand-alone FEM software, in this case Maxwell 2D. The choice of software is free and will only affect the syntax used, methodology is the same.

1.3 Scope

The thesis has the aim to create a model for how free form optimization can be performed, which factors are important and what early mistakes are possible to avoid. The paper will not go in to detail about the inner workings of electric motors and will not make any assumptions on what result such an optimization would give, but provide a method to find a well-grounded sound result based on user input. To accomplish this several measures are being taken to perform validity checks, like adding boundaries and limitations. An open mind is however crucial when doing this optimization, it's important to remember that the goal is to find geometries not before considered.

The final performance of the machine cannot be determined solely from electromagnetic simulations. Thermal properties, mechanical limitations and (often) converter properties also affect the performance significantly. It is however desirable to reduce the complexity of the (finite element) analysis to save both man and computer time, therefore this work is limited to electromagnetic simulations. It is up to the user to set-up up the simulation conditions in a way that is physically possible.

Other than creating a model following the set objectives, it's also desirable to build a method that can be used on different kinds of motors and with different objective functions.

1.4 Past Work

The theoretical background for analytically modeling a electric motor is extensive and not something that will be covered in this report. Optimized motor geometries will be used, all based on previous conventional optimizations. Considerations will be given to what assumption will have to be made to simplify the calculations, and how they affect the result. Some understanding of the motor's physical behavior is required, to make sure the adjustments are made where they can affect the targeted objective. Most of the information used in this thesis comes from ABB internal reports and by using the know-how available in-house, but also from [4] and [1].

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

2 PROBLEM DESCRIPTION

A lot of work has already been put into calculating the optimum shape and size of many rotors in use today, and extensive parameter studies have been conducted in internal ABB reports. These reports are used as a basis for models used in this thesis. These reports and hence also the background studies for this thesis can not be covered here, because of classifications and restrictions given by ABB.

Understanding the results of previous work is crucial to be able to set up intelligent objectives for optimization. If one is able to make these choices correctly there is a lot that can be gained in terms of accuracy and computation time. There is however another side to it, that is important to remember, and that is that this is an computer aided optimization and the more freedom is given to the simulations, the more interesting the result should be.

The result of the optimizations will have to be able to cover several objective functions, given by the user. Any calculation that needs to be done should have to be handled, either within the simulation software or by a external controlling software. This is to ensure no limitation are put on how complex the objective could be.

The simulation should also be able to take into consideration and weight-in different work points for the motor. This since different conditions will likely have different optimum geometries. Finding a geometry that suits the most conditions is desirable.

2.1 Using Analytical Model and Adding Free Form

The number of variables necessary and possible needs to be estimated for ensuring that the results are reliable and sufficient. Using a to high level of freedom may lead to unpredictable results that are difficult to interpret, while to few might limit the optimizer. By adding this ”free form” to the design, additional design variables are available compared to conventional methods.

The generation of geometries will be investigated, in detail how to represent geometries for easy manipulation, by functions as polygons, splines, bezier curves and equations. It will also be investigated how to parametrize the shape of interesting parts, by using individual coordinates, global/local ’offsets’ described by series expansions or polynomials/sinusoids.

Free form might lead to geometries that are difficult to put into practical use. Limitations need to be constructed to avoid impossible geometries being generated, only giving inaccuracy and taking up simulation time. This issue is resolved by adding penalties for geometries that are stepping out of a valid geometric variable range, defined by the user. When invalid geometries are being pushed onto the simulation software, an error usually occurs. This error will have to be handled for the optimization to be able to continue, adding a penalty when this happens.

2.2 Optimization Methods

First and foremost a understanding of the system that is being optimized is required; whether it has one or several local minimums, how ’deep’ are these possible solutions and is the solution linear or not. The motors used are already optimized to best effort. This fact makes

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

the use of a deterministic algorithm the best choice, since a search for a global minimum has already been performed.

2.3 Accuracy of Solutions

A high accuracy in the simulation result is important to be able to achieve a consistency in the results. Inconsistent results will lead to failure in the optimization, because it will generate false values for different search points. So even if the simulation are done faster with a lower resolution, a higher resolution (and longer simulation time) will likely lead to faster convergence.

Mesh/angular resolution versus simulation time will be considered, together with the required time resolution. Several ways of creating the geometry and the mesh will be tried and evaluated using an unmanipulated geometry.

2.4 Program Design

A high priority for the program design is to write an easily manipulated, structured code and a overall generalizing program, making it easier to change model and the optimizing method. This is making the process of applying this method more efficient.

A successful program is easy to understand, powerful in functionality and stable enough to run for hours or days without failing. To ensure this testing is required and the program structure and functionality will evolve continuously, making is important to document the process.

When building the program many different pieces are required. The main program will be written in pure MATLAB-code, as well as any directly supporting sub-programs. The connection to the simulation software will be performed with a modified Visual Basic language (applied in the MATLAB environment), with some limitation in functionality compared to using the GUI of Maxwell 2D. Finally the geometries will be written using C++, then compiled into dll-files which makes the geometry generation fast. The use of coded geometries enables a high degree of freedom, compared to using the GUI in Maxwell 2D.

Guidelines on how to design the model will be provided, as well as how to incorporate it into the optimizer code.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

3 OBJECTIVE FUNCTIONS

Several objective function will be investigated, and how they are correlated to each other during optimization. Interesting measurements are; voltage, current, flux linkage, torque and losses.

After finding how each objective is affected by the geometry, a weighted mix of different important objective can be created. Some factor needs to weight in to ensure the solution is valid out of a practical point of view, for example limiting voltage and reviewing mechanical properties.

3.1 Torque

Since torque is the product of a motor and is the parameter that governs how well it will act as a machine, its characteristic is undoubtedly an important factor to consider when tuning the rotor design. Torque curve shape and size is affected by many factors in the motor, making it difficult to foresee and control.

It's closely connected to the voltage and current of the stator, since this is where the torque gets its energy.

Mean value

Mean torque (T_{mean}) is used as a measure of how much actual torque is produced during one period of simulation. The calculations are done by a mean function, 3.1:

$$T_{mean} = \frac{1}{n}\sum_{i=1}^{n} T_i \tag{3.1}$$

n = Total number of simulation steps.

Ripple

Torque ripple (T_{ripple}) is an important factor to ensure that the machine delivers a constant torque without too much distortions, enabling a smoother drive. The ripple factor is also important to reduce losses, since a high ripple means higher fluctuations in the energy flowing trough the core. If the same mean torque is to be produced, then a high ripple gives high power transients and internal energy flow, that the machine must be dimensioned for.

$$T_{ripple} = \frac{T_{max} - T_{min}}{T_{mean}} \tag{3.2}$$

3.2 Inductance

Different motors have different sensitivity to the inductance factor, and this objective function is not always needed to be considered. In the case of some machines however, it is an important objective to investigate. Below follows how this calculation is done. A Discrete Fourier Transform (DFT) is performed on the current and the flux vectors. This result will then be multiplied with a correction constant. The constant is needed because there is a

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

phase shift from the load angle of the stator current. When looking at the inductance, you want to divide it into different component to show inductance in different parts of the rotor[1]. Here they are called d and q axis. This can be done by shifting the coordinate system of the DFT-result, so that the real part corresponds to d-axis and imaginary part to q-axis.

$$I_f = DFT(I)_{fundamental}, \; F_f = DFT(F)_{fundamental} \tag{3.3}$$

$$I_d = Re(I_f), \; I_q = Im(I_f) \tag{3.4}$$

$$\lambda_d = Re(F_f), \; \lambda_q = Im(F_f) \tag{3.5}$$

$$L_d = \frac{\lambda_d}{I_d}, \; L_q = \frac{\lambda_q}{I_q} \tag{3.6}$$

3.3 Voltage

Voltage is calculated as the derivative of flux linkage in the stator windings. The voltage has two main components, the fundamental frequency, that is the same as the current, and higher harmonics. The fundamental frequency is used to calculate the power factor while harmonics are used as a way to optimize the losses.

For reducing the harmonics of the voltage, a DFT operation is carried out on the simulation results. The amplitude of the harmonics of frequencies other than the fundamental are summed together and used as a objective function.

$$V_f = DFT(V) \cdot 2/N \tag{3.7}$$

$$V_{fundamental} = |V_f(1)| \tag{3.8}$$

$$V_{harmonics} = |V_f(i)|, i = 2, 3, ..., N/2 \tag{3.9}$$

3.4 Power Factor

The power factor (pf) of a machine is defined as the phase difference between current and voltage.

$$\phi_V = angle(V_f(1)), \; \phi_I = angle(I_f(1)) \tag{3.10}$$

$$pf = cos(\phi_V - \phi_I) \tag{3.11}$$

[1] Further shifting needs to be done depending on stator/rotor relationships

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

3.5 Losses

Core losses are mainly produced in teeth of the stator and rotor or a area close to the air-gap, when high amounts of energy is transported from the stator winding to the rotor, and vice-versa. This factor is affected by the overall efficiency, but other objectives can have a contribution to this value. This is an important objective to monitor.

For solid rotors, ohmic losses are introduced by voltages harmonics in the tip of the rotor.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

4 OPTIMIZATION

4.1 Optimization Algorithm

The information in the following chapter is based on a book on optimization [2], published by Lund University and used in optimization courses. This part will briefly discuss different algorithms and choose a suitable for this purpose.

There are several important factors to consider when choosing an optimization algorithm. One of these are if you are able to configure it to get the results you are looking for in the region you want to analyse.

The possibility to define which point to examine in the first optimization sweep is important to get a fast and accurate solver. While the starting position (which is very important and a topic in itself) will always be chosen by the user, the search area is often chosen by the algorithm. While some algorithms have a built-in function to choose the search area, in some the users themselves have to provide the function via rescaling the target variables.

This optimization problem is a function with constraints, but because the function is complex, multi-variable and non-analytical it's difficult to use any built in MATLAB algorithm. Because of this a boundary function, with penalties for stepping outside a certain area, has to be written for use with an unconstrained algorithm. Which also leads to a higher freedom from the user side. The downside is that the code gets more complicated, and that more customization needs to be done at the change of each geometry model. Here it is assumed that the model is not changed often, and that the increased required time is beneficial as understanding increases in the process.

4.1.1 Initial Search

Each algorithm begins its search by creating a initial mapping of the differential in and around the selected starting point. The actual optimization is then based on the results of these mappings, making it an important part of the process. The direction the algorithm starts to search in, is most likely the direction where it will find it's local minimum as well.

To ensure that this initial search is sufficiently big or accurate to reach the minimum there are two factors to consider. One is to have a big initial search area to cover as diverse solutions as possible, enabling a greater differential between values. The second is to make a good first estimation of where the global minimum could be. A combination of these two is of course desirable, but making a big first search is the easiest to implement. Caution should be put into not stepping out of a valid geometric area.

4.2 Nelder-Mead Method

This algorithm is the one used in this thesis and it is a deterministic algorithm which creates a solution simplex[2], and by the use of this steps toward a (local) minimum. The simplex can however contain several local minimums and move towards the most optimum one, all depending on the size of the first simplex. One might say it creates an initial rough landscape

[2]An n-simplex is an n-dimensional polytope with n + 1 vertices, of which the simplex is the convex hull. A single point may be considered a 0-simplex, and a line segment may be viewed as a 1-simplex.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

map of the function solution and moves towards the lowest value, shrinking the simplex as it moves along.

This algorithm uses a group of points instead of one single point when stepping through the function landscape. In this way, a better weight-in stepping manner can be performed. This leads to faster convergence when provided with accurate results, [3].

For a detailed view of the stepping procedure for this algorithm, see [2], chapter 3.7.

4.3 2-dimensional Example

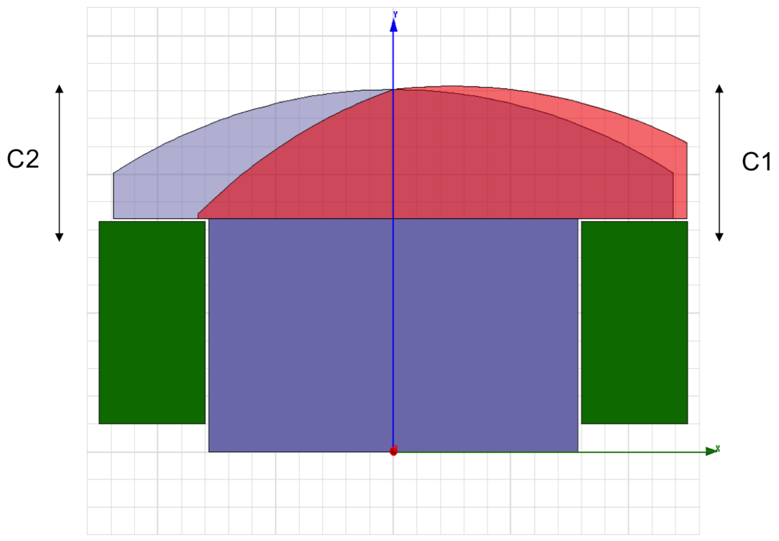


Figure 4.1: Parametric study geometry

When optimizing, the algorithm works to find a minimum in a space determined by which objective functions are used and what variables are available. In Fig 4.2 such a space is visualized. The maps calculated for this example is from motor B. Only two design variables, called C1 and C2, are used to perform a parametric study, visualizing a two-dimensional optimization map. Four different objectives are calculated; voltage inductance, voltage harmonics, torque ripple and mean torque. Relative values are used.

The algorithm works in such a manner that it traverses through this function landscape, following a path of negative derivative. This is why it is so important to chose a initial point which covers multiple local minimums, or one that starts inside the well of a local minimum. In this example only the ”torque ripple” objective has more than one local minimum, which makes is an interesting map to study closer.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

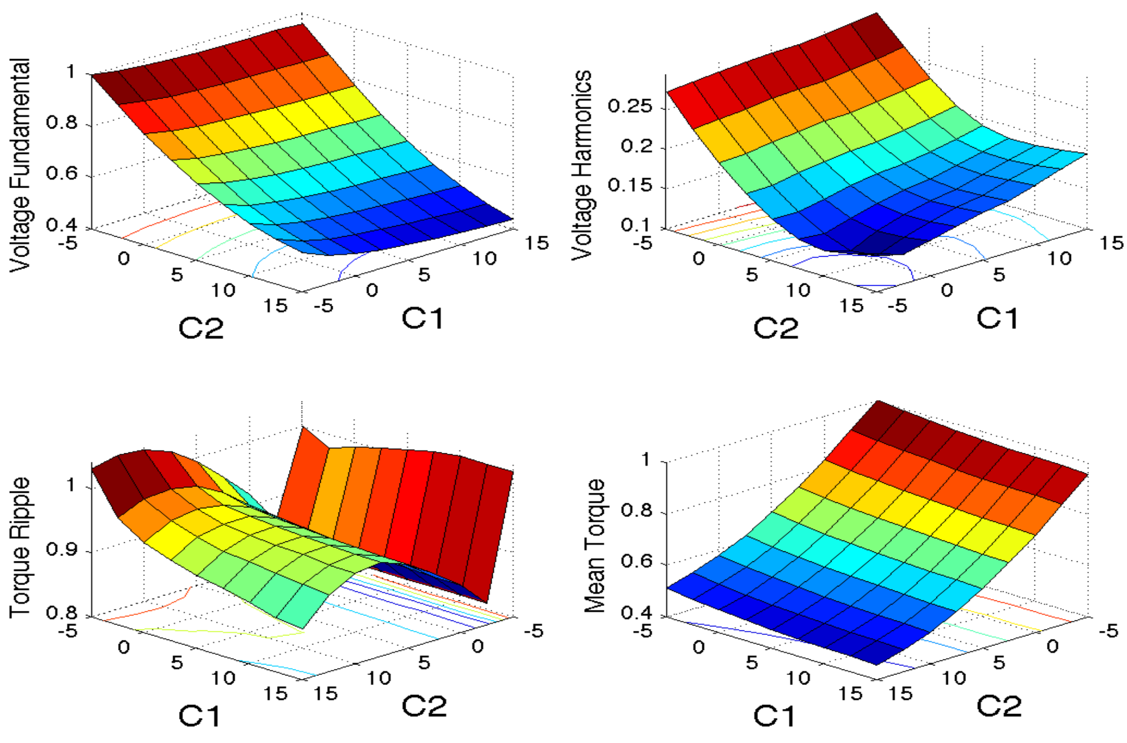


Figure 4.2: Parametric study

The ”Torque ripple” map is shown in greater detail in Fig. 4.3. Here one can clearly see the two local minimum areas, one at low C2-values and one at high C2-values. In this case the algorithm would start in the 0-0 position, making the traverse towards the global minimum ridge at $C2 = 2$. This clearly shows that a well based guess on a starting position can be very important to gain the best results possible. It also shows that this design is fairly optimized to start with, when it comes to torque ripple.

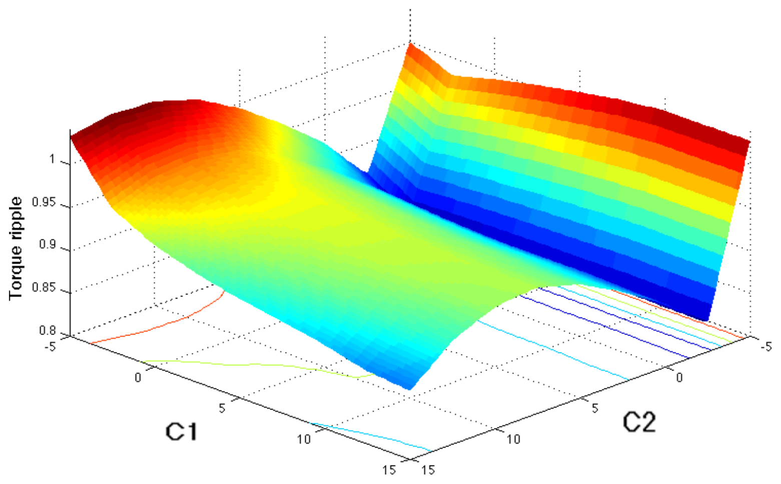


Figure 4.3: Torque ripple example

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

4.3.1 Combining Objectives

Returning to Fig. 4.2 again, we will look at the next difficult issue. What happens when you combine several objective functions? This is an area where effort can be saved by knowing a little about the correlation between different objectives. One example is in the voltage maps, which have a strong correlation. They are both slopping in the same C2-direction, but have a minimum in a different C1-direction. Since it is desirable to keep the fundamental voltage constant while decreasing it's harmonics, weighting of the objectives comes into play. This is mentioned in detail in section 7.6. In this case the harmonics has a greater slope and its minimum in a different position as the fundamental, making it possible to optimize on one while avoiding to lower the other to much.

Fig. 4.4 shows a combination of fundamental and harmonic voltage, by the code:

```
val = Vh_val + sign(Vf_val).*Vf_val;
```

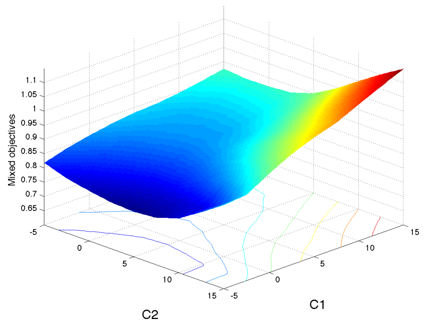


Figure 4.4: Voltage Combination

Fig. 4.4 shows what results is to expect when objectives are combined. Since the Vf part of the objective is designed so that any change from the original value will give a penalty, the map could be slightly difficult to picture. The results it clear enough though, there is a distinct minimum within the search area.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

5 GEOMETRIC REPRESENTATION

There are some different ways available in generating geometries, some more versatile then others. Most of the shape is being controlled by analytical equations derived in previous publications. But some parts of the geometry need a higher degree of manipulative freedom than an equation can provide. This can be accomplished by using for example so called spline and bezier curves, where curves will be automatically generated, based on input points and different criteria. Even if this is easy to implement and gives satisfying results, it can be difficult to foresee the result, giving an unwanted degree of uncertainty to the model. Instead of using those, a high resolution, line segment model provides predictability as well as a high level of control over the shape.

There are several ways of creating a desired segment form by constructing a linear combination of functions. In this work it been found most suitable to use Fourier series and polynomial series. When deciding what functions to use, an evaluation of the wanted shape for that type of rotor will have to be done. Below follows an example on how to perform this and possible result.

5.1 Creating Free Form With Perturbation

Enabling a free form is accomplished by adding a perturbation function to the existing geometry generating code. The alternation section for this kind of pertubation looks as follows:

```
for (m=0; m < M; i++) {
  ee = 0;
  pe = m*pi / M;
  for (i=0; i < length(C); i++) {
    ee += C[i]*sin((2*i + 1)*pe);
  }
  pos[m].x = rr*cos(theta) + ee;
  pos[m].y = rr*sin(theta) + ee;
}
```

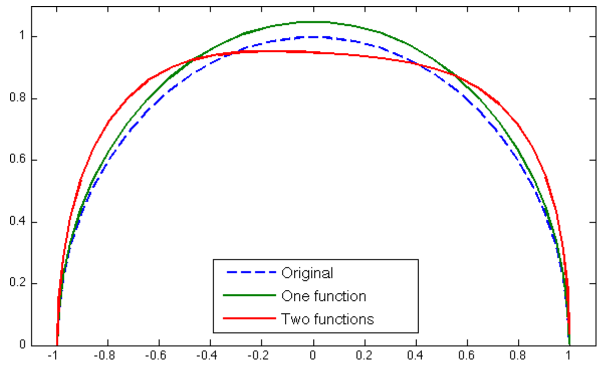


Figure 5.1: The effect of perturbations

More complex shapes could be added to the shape if necessary. This section of the method is not part of the main program, but is done in beforehand during the building of the FEM-model. Several options are available for customizing this perturbation. Sometimes it is important to limit the change to a certain part of the rotor, to avoid invalid geometries. Care has to be taken to the rotational direction of the rotor, this affects if the shape should be symmetrical or not.

The user needs to remember how to incorporate this into the program code though. The variables should have a similar weight to them, to make to solution more transparent. For example, changing one variables with 1 point, should have a similar effect (in size) to changing a second by 1.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

6 SIMULATION

Most information has been gathered from Ansoft Maxwell built-in help function and by reading supporting document to the same software, all available on-line as a support feature when purchasing the software.

6.1 Optimizing the Solver

The optimization uses numerical values for calculating the objective function. Hence it is also the numerical value that are optimized, and not necessarily the best geometric solution. This fact have two important implications. These are to find the right resolution, for both time and mesh, and whether the high resolution results is actually needed for the optimization to work.

Finding the right resolution is all about deciding on what degree one can be willing to sacrifice speed for accuracy, this depends on the optimization target function and the expected simulation time against available time. Different parameters of the motor characteristics have different wave forms. Torque contains high frequency information important for post-processing, while flux mainly has a fundamental component of interest when analyzed separately. A comparison between torque, voltage and flux is shown i figure 6.1. This leads to the fact that depending on what parameter is of interest, a different time resolution is required.

Torque: High resolution required for ensuring all information is gathered.

Voltage: Depends on objective function, higher harmonics require higher resolution.

Flux linkage: Voltage is derived from this parameter, but itself has a low requirement.

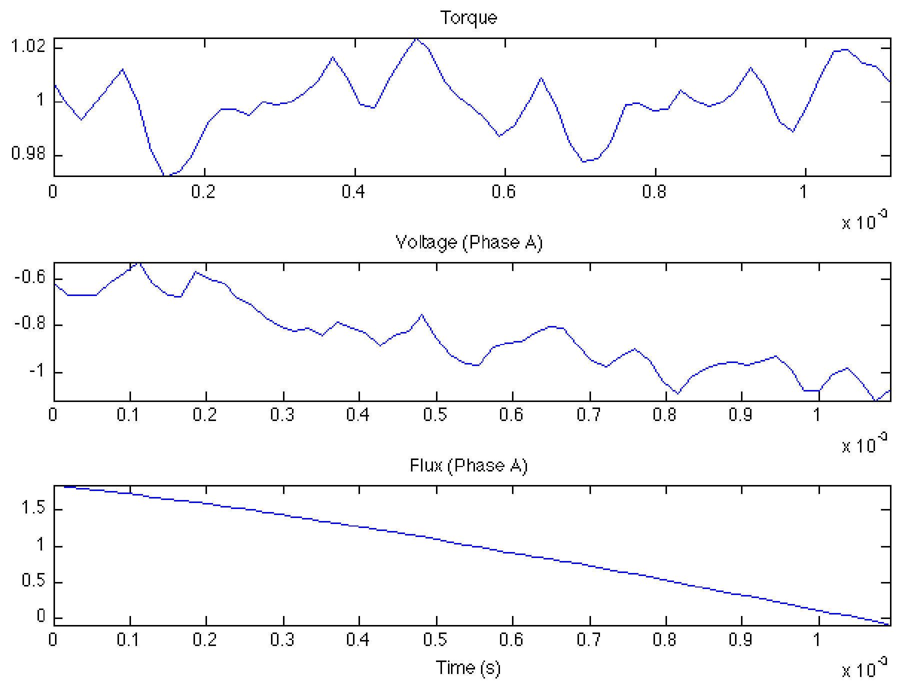


Figure 6.1: Waveform comparison

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

The second and equally important factor to consider is whether the numerical value received from the simulation software needs to have complete accuracy, or if the relative accuracy between iterations is sufficient to get a correct optimization result. This is a tough question to answer since it is completely dependent on which objective function is used. Intensive studies will have to be done to investigate the effect of the resolutions on the relative result. This means running one simulation on very high resolution and manually evaluate the need, depending on objective functions. Since the gains are often quite small and this is done on a case to case basis, a higher resolution to start with is the best way to go. Alternatively running many different functions on low resolution, and re-running interesting ones on higher later.

6.2 Time Resolution Evaluation

When evaluating which time resolution to use, the simulations shown in Fig. C.1 were used. Here simulations start of with a high value, in this case 100 steps per 1/6 period. When the resolution is then lowered, the result changes. By using the first high resolution result as a base value, the percentage change will show how much the results differ from what would be considered as a ”correct” value. Here it is also shown how different objectives change with resolution.

Conclusion out of the tests performed where that 20 steps per 1/6 period is a minimum, to ensure no characteristics are missed. Using a lower resolution, 10 steps per 1/6 period, means high frequency dependant objectives as torque ripple will be reported close to zero.

6.3 Mesh Resolution

The mesh resolution determines how much details are available when performing the simulations. A high resolution is mostly important in areas where flux shape is changing quickly, for example the rotor tip. In areas with a constant flow, less details are required. If a to course mesh is used, many important factors can be missed. Such factors as losses would not be possible to compute accurately without a fine enough mesh.

Because of confidentiality, not all applicable meshes are shown. In Fig. 6.2 however, a typical set-up is shown. Here the stator and rotor tips are having a high resolution, while the interior is more course.

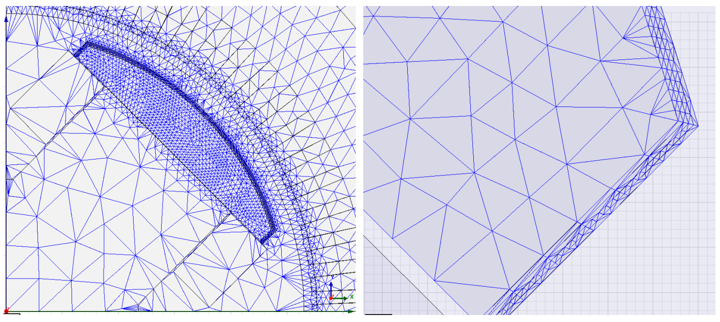

(a) Full mesh (b) Rotor tip mesh

Figure 6.2: $Meshplot$

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

6.4 Simulation Setup

6.4.1 Current Source

To avoid initial transients in the solution, a current source is used for excitation. When a current source is used, voltage is derived to fulfil the needs of the load. A correction $-\frac{\pi}{N}$ comes from the way voltage is calculated in Maxwell, by taking the derivative of the flux, introducing a shift. Since current is given and have values set at specific times while voltage has an offset, the voltage will have to be corrected to match this by adding the factor in eq. 6.1.

$$e^{-j\frac{\pi}{N}} \tag{6.1}$$

6.4.2 1/6 of a Period → 1 Period

Because of symmetry in a three phase system one can use the information available in one 1/6 of a period, over all three phases, and piece them together to form one full period. This is also possible due to the use of numerical calculations. For the internal electro-magnetic properties of the motor, every 1/6 of a period the waveforms repeat.

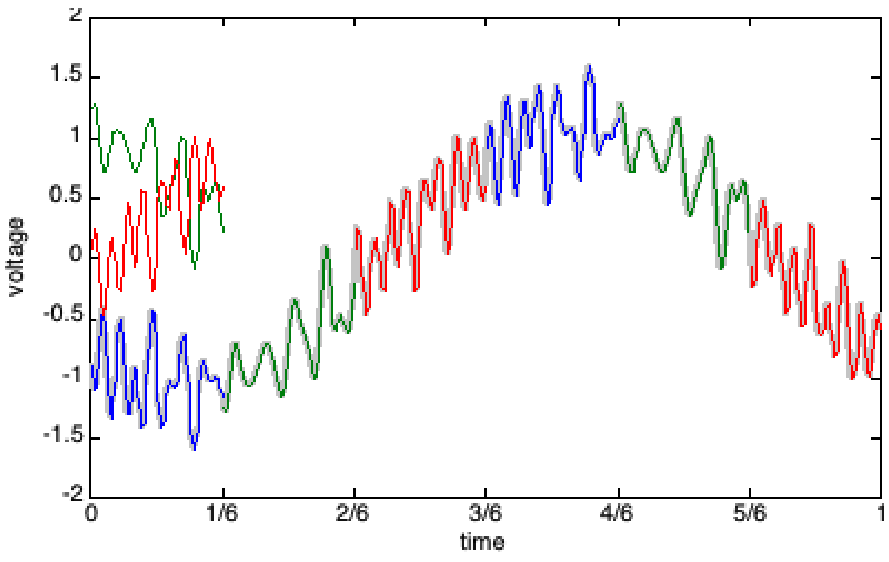


Figure 6.3: Symmetric three phase system

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

7 EXECUTING PROGRAM

In the initial runs the connection between MATLAB and Maxwell was simple and involved having to restart and reload Maxwell after every iteration. This was a solution that was not desirable since the model took as long to load as running the simulations. By incorporating the script in MATLAB, Maxwell could run in the background and then be called by a ”run script” feature. The method controls Maxwell in the same manner a user interface would.

Shown in this part and also in the appendix is the code for executing simulation for one type of motor. The design for different motors will vary slightly, but not so much that it requires a separate chapter. Whenever there is a major difference or something important to note, it will be mentioned.

The final design is almost entirely run from MATLAB, with only FEM-calculations and data extraction still done in Maxwell. All post-processing, storage and displaying is done in the MATLAB environment. This was found to be the most flexible, stable and fast solution.

The full executing program structure, with its support programs, is shown in appendix D.

7.1 Initial

In the first part of the program structure global variable declarations are made, and the connection to the simulations software is established. Some important variable as number of time steps, persistent objective function values and an iteration counter are all created, to make sure these are not removed accidentally. Since that would lead to a program failure it is beneficial to have these separate from frequently modified code.

The connection between MATLAB and Maxwell is enabled through a ActiveX scripting engine implementation. Via this engine, Maxwell functions are called using a modified version of the visual basic (VB) language. The connections works well, but some functions are lacking compared to running a system consisting of VB-files straight to Maxwell. Using VB-files is however not as easily manipulated and requires the user to understand the VB-language, which is not required in this method since MATLAB-commands are used and then MATLAB works with the scripting engine to communicate with Maxwell.

A restart function is also implemented (shown below), enabling a purge of Maxwell’s memory. It is needed since the simulations slow down considerably over time if this is not performed.

```
if isequal(mod(counter,100)-1,0) && counter ~= 1
    oProject = invoke(oDesktop,'OpenProject', ...
        'C:\Documents and Settings\SETOMAN\My Documents\matlab\Sim\Sim.mxwl');
else
    oProject = oDesktop.SetActiveProject(project_name);
end
```

7.2 Modify Design

The design modification code contain three major parts; initial position offset, diff(x) manipulator and the variable declaration.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

The default initial optimization point is set according to previous studies mentioned in the beginning of the report. If the user wishes to alter this point, it can be easily done by just inputing the new position as an offset. This is by default a vector of zeros, the same length as the input vector.

The diff(x) manipulator is important because of the fact that most optimization algorithms in MATLAB uses a predetermined step size for searching through the function space. This step size could be very low for some applications, or just desirable to manipulation by the user. What it does is to multiply the initial step size of the algorithm, and hence also all the following ones. In this way the user is in full control of how big the initial simplex will be. This is a requirement to ensure the entire variable space of interest can be covered, minimizing the possibility for the algorithm to converge into local minimums, instead of a global minimum.

The variable declaration is a straight-forward process, containing a sweep over the entire input-vector provided by the user, manipulated by the optimization algorithm. Maxwell is then called through the scripting engine, altering the geometrical design of the rotor as well as any project variables. There is however one important note to recognize while sending the new variables to Maxwell. That is to send all changes in one cell-vector, and not to perform the changes one variable at a time, since changing one variable takes the same time as changing all at once. This is due to the way Maxwell builds the geometries, calling the dll-file previously mentioned in section 2.4.

```
offset = zeros(length(in),1);
n = 1; var = cell(length(in),1); in = 1e4.*in;
for i=1:(length(in)-3)/6;
    k = num2str(i);
    var{n} = {['NAME:' ['C1L' num2str(k)]], 'Value:=', num2str(offset(n)+in(n))};
    n++;
end
```

The code for changing the variables is dependent on how many variables are present, what names are given to them, how the values are represented and whether all are zero-centered or mixed with actual values. This is the section, together with objective function calculation, where most work needs to be done by the user. It is also hard to provide a fixed solution beforehand, since it is so strongly model dependent.

7.3 Solver

This code section is small and only contains two parts; the solve-call, which performs the simulations covered in section 6, and an error catching routine in case Maxwell throws an error. There are several possible errors that can occur, covered in detail in section 7.9.

There is one error, which can be handled, that happens so often that a catch-routine is needed. It is due to a quite frequent license server fault, which is solved by entering a delay and retry after 60 s.

A second error related to creating an invalid geometry is also frequent. In this case nothing can be done to solve the problem, instead it will be fixed at the next iteration when the

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

geometry changes again. What is important to do here is to make sure there is a penalty for stepping outside of the range of a valid geometry. This can be done in several ways, depending on how important it is to stay within the boundaries. If the penalty is set to high, the end gradient will be very high, pushing the algorithm inward to much. The global minimum might be close to the boundary of a valid geometry.

There is a possibility to expand this section to handle multiple work points, like different load angles. By doing this a more general optimization is performed, valid for more than one work point. Two ways of doing this has been tested, by an internal loop and by distributed computing.

The loop is performed by adding a for-loop to the later sections of the program, and saving the results in a matrix instead of just a vector. When all variations are done, the matrix will be weighted together to form only one return value, this is also up to the user to perform since it is part of the objective function handling.

The second method is distributed computing which is preferable since that includes running all simulation variations simultaneously, saving a lot of time compared to the loop technique. This does however require an additional license to be purchased and setup of distributed computers. It was tested briefly at a late stage with mixed results. It worked but the syntax is somewhat different compared to running a single simulation, making it difficult to test extensively. If there would have been a possibility to do this at an earlier stage of the development, more success would likely have been gained since it provides a good opportunity for more generalized solutions.

7.4 Read Simulation Output

This section contains scripts for storing nominal variation, available reports and current solutions.

Acquiring the solution from the simulation software is done by first reading what reports are defined in Maxwell, done according to the documentation in appendix 9. These reports are then saved to their respective file. This file is then read by MATLAB and stored in local variables. This procedure is quite cumbersome, but the fastest and easiest way when using Maxwell.

7.5 Interpret Data

Here the acquired data is processed and formated in a way that is useful for the user, preparing project variables for writing to file and creating local variables.

The vectors needed for calculation of the objective functions are created here. Including time and current vectors. Both 1/6-period and full-period time vector are created. The reason why a time-vector has to be created using the t_final variable, is that Maxwell reports the unit of the output variables is not set. This means that time could be returned in for example milliseconds for a low value of t_final and seconds for a higher value. By creating the time-vector here this is avoided, however an exported time-vector would be preferable for synchronization reasons.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

The first simulation step is removed by the idx-variable, due to the transient solution. At the start of the simulation, the voltage is defined to be zero leading to false results in the first time step.

```
speed = oDesign.GetVariableValue('rpm_speed');
speed = str2double(speed(1:length(speed)-3))*(2*pi/60);

I1 = oDesign.GetVariableValue('I1');
Is = str2double(I1(1:length(I1)-1))*sqrt(2);

efreq = oDesign.GetVariableValue('efreq');
efreq = str2double(efreq(1:length(efreq)-2));

phiA = oDesign.GetVariableValue('phiA');
phiA = str2double(phiA(1:length(phiA)-3))*(pi/180);

t_final = 1/efreq; time = (0:t_final/(6*steps):t_final/6)';
idx = 2:1:length(time);
time4 = (1:(length(time)-1)*6)*(t_final/(6*steps));

I = Is*sin(2*pi*time4*efreq + phiA);
```

Next voltage, flux linkage, torque and losses are formatted into vectors with a format that is useful for calculating the objective functions. FFT is also performed on the voltage vector, to be able to calculate the frequency content of it. Voltage and flux linkage is created by adding together the three phase 1/6-periods content, creating a full period for easy visualization and FFT calculation.

```
V6 = [Va; -Vb; Vc; -Va; Vb; -Vc]';
F6 = [Fa; -Fb; Fc; -Fa; Fb; -Fc]';
Cl = 1e-6.*coreLoss.data(idx,2);
T = T.data(idx,2);
Vfft = (2/length(V6))*fft(V6);
```

7.6 Calculate Objectives

The objective are calculated according to the equations defined in section 3, and will not be covered here.

When calculating the value to return to the optimization algorithm, it is a good thing to use per unit values instead of actual values. By doing this the different objectives will weight equally (although still dependent on the rate of change). This method is shown below, using the power factor value as the objective. The variable pf_{val} will be given a value of zero in the first run, and pf_{nom} will be saved as a persistent variable in the MATLAB workspace, for use during the entire optimization. Future values will then be compared to pf_{nom}, by executing the second part every iteration. The values will all have a value corresponding to the change

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

compared to the initial geometry, making it easy to see how the different objectives are responding to the changes in geometry.

```
if counter == 1
    pf_nom = pf;
end
pf_val = (pf-pf_nom)/pf_nom;
```

When putting several objective functions together some weighting of the variables could be required. When handling just rate of change of the variable, all that is required is to multiply the value with a constant reflecting this objectives rate of change, or importance.

Often there is a wish from the user to optimize one parameter but keeping another at a fixed value, for example reducing torque ripple but keeping the mean value at the same level. This issue can be handle by adding one of the weighting functions shown below. Using the sign function which keeps track of the sign off the variable, enabling the weight value to be positive at all times. This enables the objective to add a penalty to the return value whenever it differs from the original value. Each function below has a different rise curve useful for how strict the value needs to be held, Fig. 7.1. These functions can of course also be modified with a constant to increase or decrease the features provided.

```
sign(T_val).*T_val

exp((sign(Vf_val).*Vf_val))-1

pow((sign(Vf_val).*Vf_val),2)

sqrt((sign(Vf_val).*Vf_val))
```

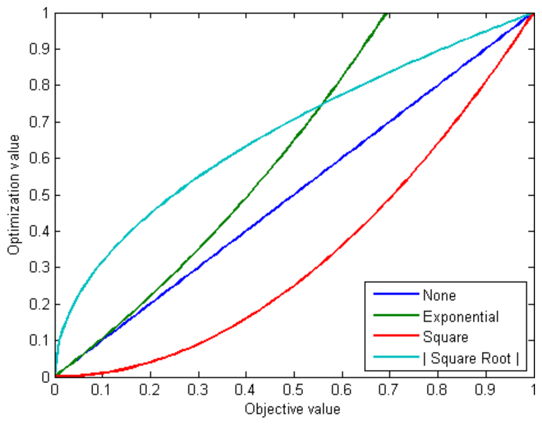


Figure 7.1: Weighting functions

When there is a mix of objective functions in the optimizer value and the weighting has not worked properly, one objective might be dominant at the start just to be taken over by another at a later stage. This can be seen in Fig. A.2 where the torque ripple drops rapidly, this affects other objectives which will then only be considered for optimization after the ripple objective has settled. If ripple would have been weighted differently, other results could have been achieved.

7.7 Save To File For Post-Processing

All vectors and results are saved to text-files for post-processing and evaluation. This is also required to monitor the progress of the optimization, since no data is stored locally in

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

Maxwell. Generally it is good to save as much data as possible, since you never know what you might need to re-run the same optimization a second time. Both geometrical model, objective function and program structure might change during the process. Without proper documentation and data storage it could prove impossible to compare old results with new at a later stage.

7.8 Finish

The last section contains the clean-up functions needed to keep the simulations running efficiently. First the current variation is deleted, freeing up memory. Secondly, depending on how many iterations have been performed, the project is saved or restarted.

Saving the project executes some cleaning commands initialized when deleting a variation, it also clears the undo history. This is performed every tenth iteration and is done very fast. Restarting the Maxwell application has also proven to be effective to handle memory-leaks in the software. This however takes some time to perform, and is only performed every hundred iteration.

7.9 Problems

7.9.1 Increasing Simulation Time

When running the optimization, Maxwell gathered solution data and ran very slow after about a hundred iterations. Deleting solutions and minimizing the output variables lowers memory consumption and makes Maxwell more responsive, as mentioned in 7.8. Instead of containing the solution in Maxwell, necessary parameters where extracted and post-processed in MATLAB. They where then saved to disk as formated text data. This was a lot less time consuming and required less disk space. The user will have to actively save relevant data, or data that could be relevant for future reference.

	200 solutions, with plot	200 solutions	No solutions
Change shape	200	40	2.6
Solve	450	250	140
Get results	10	3.3	3

Table 7.1: Time consumption per simulation, with three different conditions (s)

”With plot” means that the results are represented in Maxwells build-in report generator, which updates the plots in real time. This of course takes time to do and is not necessary or even possible to do during optimization, but can be valuable to be aware of for reference purposes.

7.9.2 Scripting Limitations

Some functionality have not been implemented in the scripting engine that is used to communicate with Maxwell through the MATLAB interface. The user should be avare of these limitation and know how to work around them.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

One important limitation is the lack of being able to declare an entire vector by using a single sign, like (:) is used in MATLAB. Instead each element has to be declare individually, or saved to a vector like in section 7.4. This would not be such a big problem if all variation could be retrieved in this way, but only the current (nominal) variation can. The reason why this is such a big limitation is in the way the optimization is done. If it would be possible to save the results from the initial differential/simplex steps for future optimization, which is always the same, much time would have been saved, see section 4.1.1.

Another issue is the lack of a direct and easy way of importing results straight to MATLAB, so a work-around had to be created. Instead of importing directly, the results will be exported from Maxwell to a file, and then MATLAB reads the content of the file. This is not a big issue since the files are small, but it would improve performance to import directly.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

8 POST-PROCESSING

Output from the FEM-simulations consist of voltage, torque, flux and losses, while current is given in advance. By using these values interesting parameters of the motor performance can be calculated during post-processing in the same way as it is used for optimization. The post-processing is a section that is highly dependent on what type of motor is being simulated and for what objectives. The calculations implemented in this program will be covered here.

8.1 Data Extraction and Variable Creation

First of all, the txt-files created during optimization are imported into MATLAB and then converted to MAT-files for easy use and data protection in the future. By using MAT-files instead, you are ensured the raw txt-files are not altered and that they can be stored in a different location.

Secondly, the variables and vectors used in the executing program are recreated. This is to enable the objective functions to be calculated once more, to investigate their individual propagation and maybe the effect of different weighting conditions.

High resolution time and current vectors will be created to prepare for interpolation of torque and voltage vectors, and their corresponding objectives.

8.2 Objective Functions Visualization

Below typical code for showing the propagation of the objective function per iteration is shown, iterations are represented by the k-variable. The same is done for all objectives, giving detailed information easy accessible.

```
for i = 1:size(F,2)
    [Id(i),Iq(i),Ld(i),Lq(i)] = induct(F(:,i),I1);
end
Ld_nom = Ld(1); Lq_nom = Lq(1); Ls_nom = Ld_nom/Lq_nom;

figure(4)
subplot(2,2,1); hold off
plot(k,Ld); title('Ld'); axis tight
subplot(2,2,2); hold off
plot(k,Lq); title('Lq'); axis tight
subplot(2,2,3:4); hold off
plot(k,Ld./Lq); title('Saliency ratio'); axis tight
```

8.3 Motor Performance Visualization

Some of the objectives are made up of individual parts that are also useful to monitor, for example voltage frequency content and the actual shape of the torque curves over time. This part handles these features. In the code the objectives and motor performance sections are

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

nested together, calculations are made before plotting to enable the calculations to be used in relations with each other.

8.4 Concluding Results

In the last part relevant information is shown about the optimal solution. Usually this is the same as the final iteration, but sometimes there might be several optimum sections in the iterations. The result depends highly on the weight of objective functions and an analysis of all iterations are required, but the conslusion is useful for a quick evaluation of the results.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

9 RESULTS

The optimizations where performed on confidential rotor structures currently in development. Because of this no practical results or geometrical changes will be presented in the public version of the report.

The number of geometrical variables used for these optimizations ranged from 16 to 27.

9.1 Objectives Presentation

The results of the optimizations will be represented by four main post-processing categories. These are explained below, together with plots showing a parametric study of an important design feature for motor A. The value of this parameter is swept over, giving different performance values of the motor. Again the MATLAB colormap jet is used to show change. Low and high in this case means the iteration numbers used, -9 to 23, which represents a design variable value.

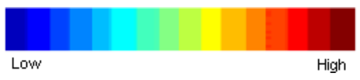


Figure 9.1: Colormap ”Jet”

9.1.1 Torque

This plot is to show how the torque curve changes with the iterations, an important visualization to understand objectives like ripple and mean torque.

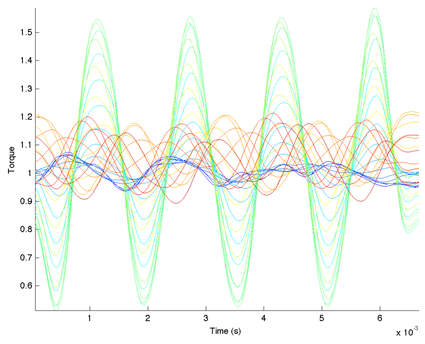


Figure 9.2: Torque

9.1.2 Voltage

With the voltage plot, the user will also get a visual representation of the voltage curve in the top figure. In the lower figure, the different frequency content is shown. Blue line is the fundamental frequency, important for power factor and torque. Red line shows the voltage harmonic content, important for ripple and losses.

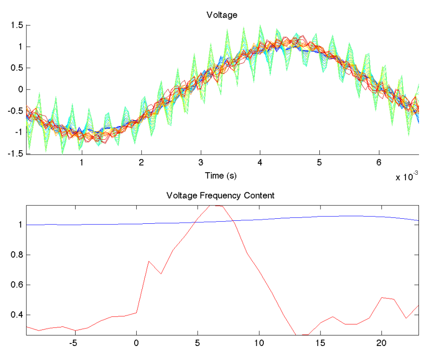


Figure 9.3: Voltage

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

9.1.3 Inductance

The inductance plot will give the user knowledge about the magnitude of inductance in different (user defined) portions of the rotor. The top plots are inductance in different coordinates. The lower plot shows the ratio between these values.

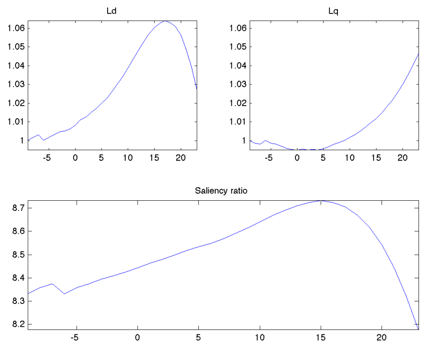


Figure 9.4: Inductance

9.1.4 Objectives

This figure shows a summary of important objective in the optimization. The top right plot shows torque ripple, the lower left shows mean torque and lower right shows power factor. The plot in top left is the sum of objective functions used, and because of this changes depending on user definition. In this case it is a combination of all objectives;
"- PFs + Rs - Ms - Ld + Lq + Cm - Vh" [3].

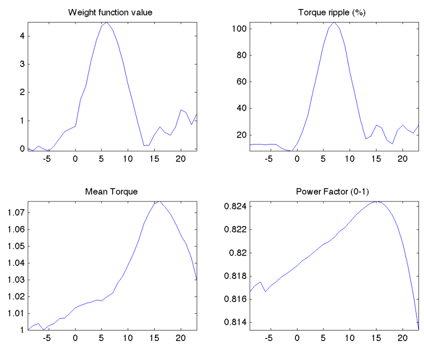


Figure 9.5: Objectives

9.2 Shape Presentation

When the optimization is done and post-processing is finished, it can be good to see how the optimization moved through different geometries to find the most optimum one. This can be done by using the short code shown in appendix D. What this code does is to save each iterations geometry to a file. These files can then be put together to create a movie showing the progress, or used to look at individually to see if there are interesting results along the way.

9.3 Objectives

To show the success rate of this method, some important results obtained during the development will be shown. In the end a mixed objective optimization with good results will be shown, for two different load angles.

The lower axis always show the progress in iterations, for the current optimization.

[3]PFs = power factor, Rs = torque ripple, Ms = mean torque, L = inductance (d/q), Cm = loss, Vh = voltage harmonics

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

9.3.1 Torque Ripple

Few optimizations, using 120 steps per period, for torque ripple were performed. Fig. A.3 contains ripple optimization, but that was in combination with inductance. Since ripple requires a high time resolution, only 120 steps simulation are relevant to analyse. However, the results are clear that this is a factor that can be highly improved by this method. Together with inductance it is the most successful objective, which is also shown in the later mixed objectives section. In Fig. 9.6 ripple is reduced from 18 % to 5 %, a large difference. The cost is a reduction of mean torque by 8 %, which is mostly due to elimination of torque peaks in the waveform and not necessarily a bad thing.

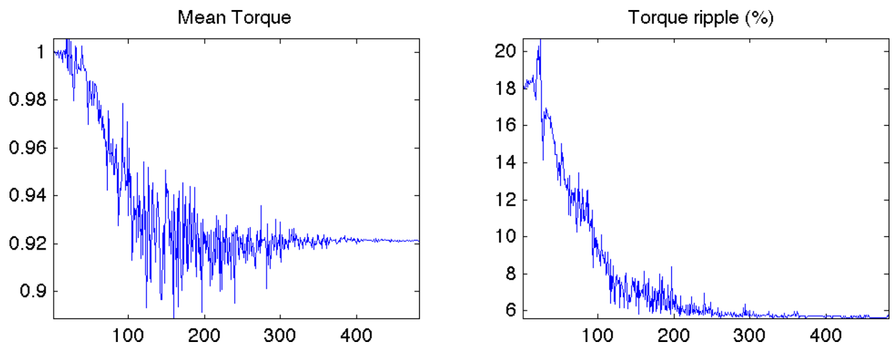
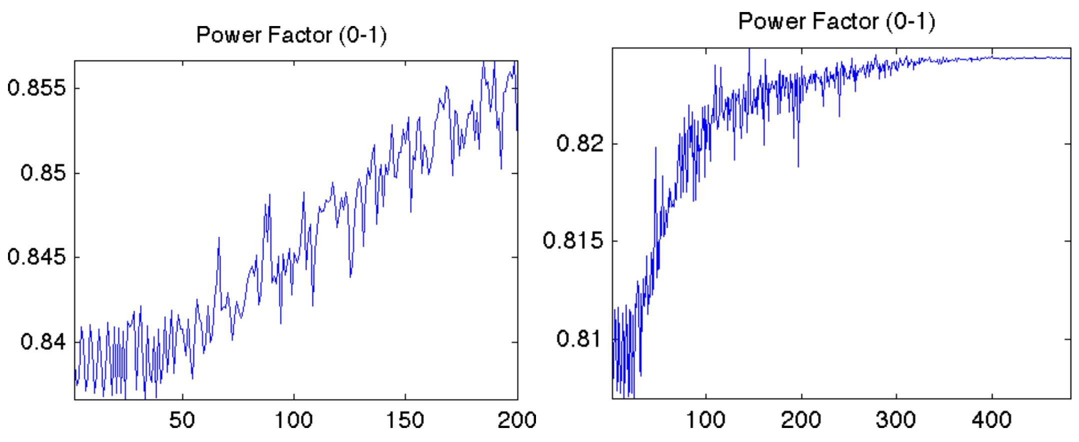


Figure 9.6: Looking at ripple optimization

9.3.2 Power Factor / Inductance

Power factor is closely related to inductance in the motor type used for these evaluations, hence this section merges these together. Even though the torque curves and overall characteristics are different in Fig. A.3 and Fig. A.1, the link between power factor and d-inductance is clear. In the best runs, power factor was improved by 2.5 percentage points, when considering a power factor of 1 to be 100 %.



(a) Power Factor objective (b) Inductance objective

Figure 9.7: Optimizing power factor and d-inductance shows similar results for power factor objective

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

9.3.3 Voltage Harmonics

For testing the ability to affect the voltage harmonics by changing the geometric shape, a second motor type was introduced. Here harmonics where added in the source, simulating a noise common when using power electronics. The result is shown in Fig. B.1. Also here, the results are satisfactory, with a decrease by 12 %, as shown in Fig. 9.8. And with minor effect on torque values, as shown in Fig. 9.9.

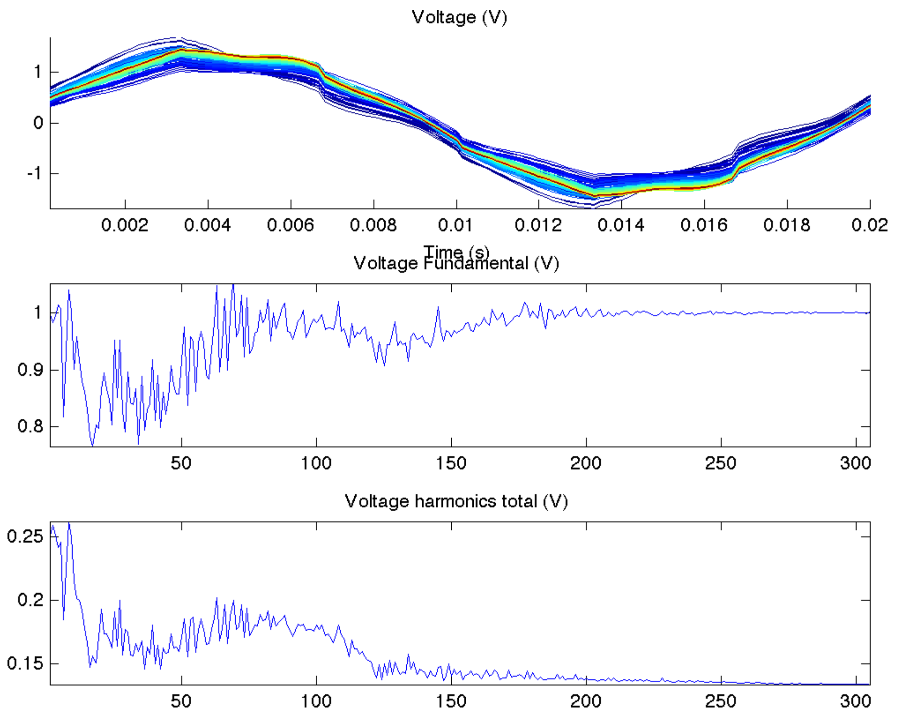


Figure 9.8: Voltage harmonics optimization

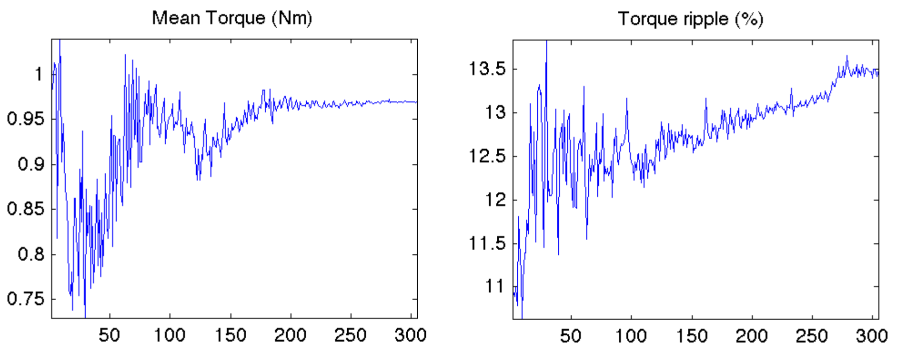


Figure 9.9: Voltage harmonics optimization has a minor effect on torque values

9.3.4 Mixed Objectives

After performing optimization for each of the set-up objective functions, and evaluating how each respond to the method, a mixed objective was created. Many variations in weighting

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

was tested, but in the end a final mixed objective optimization was performed. It used 20 steps per 1/6 period and was performed in two load angles, on with high torque and one with low. The result is shown in Fig. A.5 and A.6.

Notable in the results are that all set-up objectives has improved by some extent, in some cases very much. A summary of important results are shown in table 9.1. Most notable are; high change in torque ripple, improved power factor, stable mean torque and fundamental voltage.

A second important results is also the difference in result for different load angles. It seems as using a low torque work point leaves more room for improvement. This could be explained by the lower amount of energy flowing inside the rotor, but further analysis need to be done.

	High torque	Low torque
Torque ripple [4]	- 14.0 %	-12.7 %
Mean torque	- 0.15 %	0.1 %
Power factor [4]	0.39 %	2.58 %
d-inductance	- 0.28 %	- 1.0 %
q-inductance	- 2.7 %	- 11.4 %
Fundamental voltage	- 0.5 %	- 3.4 %
Voltage harmonics	- 1.0 %	- 6.4 %

Table 9.1: Mixed objectives, results

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

10 CONCLUSIONS

Many method-related issues have been mentioned during this report and those problems and solutions are in the area of program structure and general methodology. It is hard to describe a method in a complete manor, but the points mentioned in the report gives the user a good basis to stand on and to proceed from.

10.1 Objective Correlation

This factor depends on what kind of motor is used for the optimization, but can affect the result if it is not considered when setting up the objective functions. Correlation can be seen in the results of motor A, especially saliency ratio/power factor and torque/voltage/d-inductance.

10.2 Objective Weighting

Different objectives will react differently to optimization. Some will show a high percentage of change (torque ripple), while others will only change by a few percent (power factor). This is hard to know before performing the simulations, but is important to handle to make sure optimization is performed on the objective that is most important. By having a too low weight put on an important objective, this can be completely shrouded by a less important factor with a high rate of change.

10.3 Resolution

Resolution is always important to ensure the results gathered are correct and represent what is investigated. The difference the time resolution makes is shown when comparing Fig. A.2 and A.3. The objectives used are the same (except weighting of Ld-inductance), but the results show a clear difference. This has been covered before in section 6, and the result is seen in the torque plot.

Finding a good mesh resolution is a harder task, since different geometries will have different areas in need of high resolution. Often a basic understanding of the motor functionality is enough to increase mesh density in relevant areas. Increasing the mesh often has a lower impact on simulation time than time resolution, but a lower density will often lead to shorter simulation times. Mesh resolution will give a base simulation time, which is then multiplied by the number of time steps.

10.4 Load Angle

Work points will have a high impact on the results of an optimization and are crucial for whether the results are relevant for practical use or not. In the optimizations performed here there is a clear difference between high load and medium load. No in-depth studies have been done to find out exactly why. One reason is however that the higher the load is on the rotor, the more iron is required to conduct, leading to less freedom in geometrical shape.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

11 FUTURE WORK

A few things need to be tested before this method should be considered consistent enough to provide results on which one would base actual motor design.

11.1 Investigate Mechanical and Thermal Properties

This method only involves simulations in electromagnetic properties. To make sure the geometry generated is usable in practice, mechanical and thermal calculations are necessary.

11.2 Work Points

Further testing of different work points is necessary, only two where tested on motor A in this case.

11.3 Evaluate Results on Actual Machine

The optimizations performed for this work where done in such a way as to test the software and for error checking primarily, and not for the purpose of actually optimizing the motor. Because of this, there is no real consistency in the results and it's hard to make any well-grounded conclusions out of them. When applying to a real product, clearer goals will have to be defined.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

12 REFERENCES

[1] Karlsson P Alakula M. *Power Electronics - Devices, Circuits, Control and Applications*. Industrial Electrical Engineering and Automation, Lund Institute of Technology, Lund, Sweden, 2006.

[2] L-C. Biers. *Lectures on Optimisation*. Lund Institute of Technology, Lund University, Lund, Sweden, 2007.

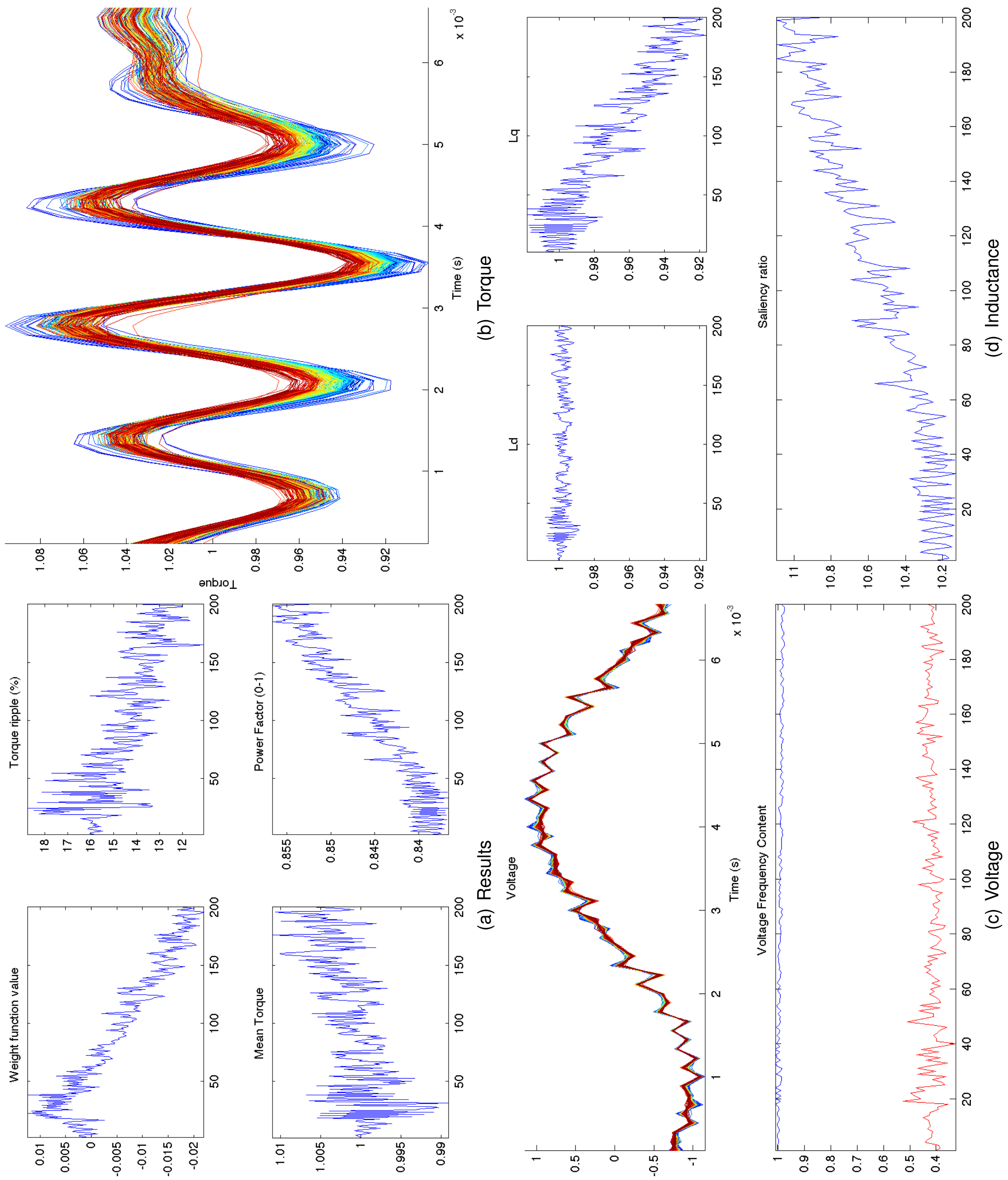
[3] Margaret H. Wright Paul E. Wright Jeffrey C. Lagarias, James A. Reeds. *Convergence properties of the Nelder-Mead simplex method in low dimensions*. 1998. PII. S1052623496303470.

[4] T.A. Lipo. *Introduction to AC Machine Design*. Departement of Electrical and Computer Engineering, University of Wisconsin, Madison Wisconsin, USA, 1996.

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

A OPTIMIZATION RESULTS, MOTOR A

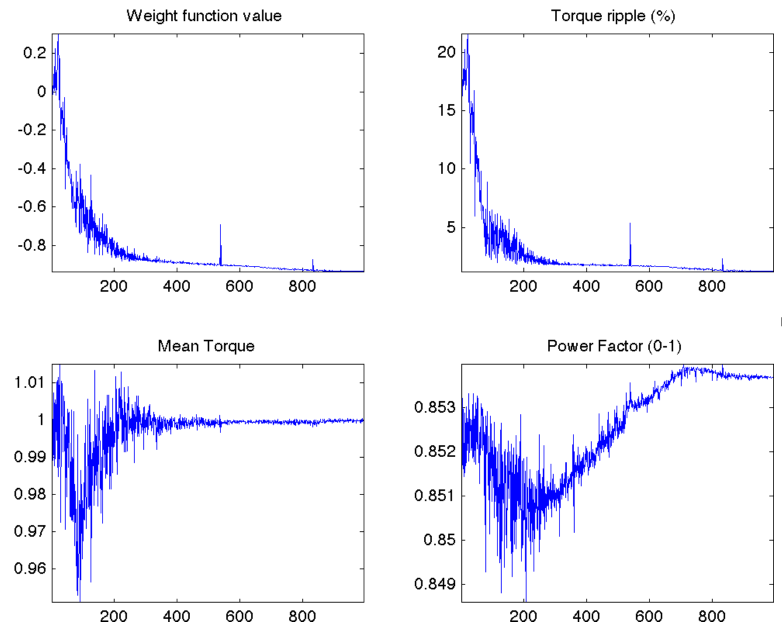
We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.



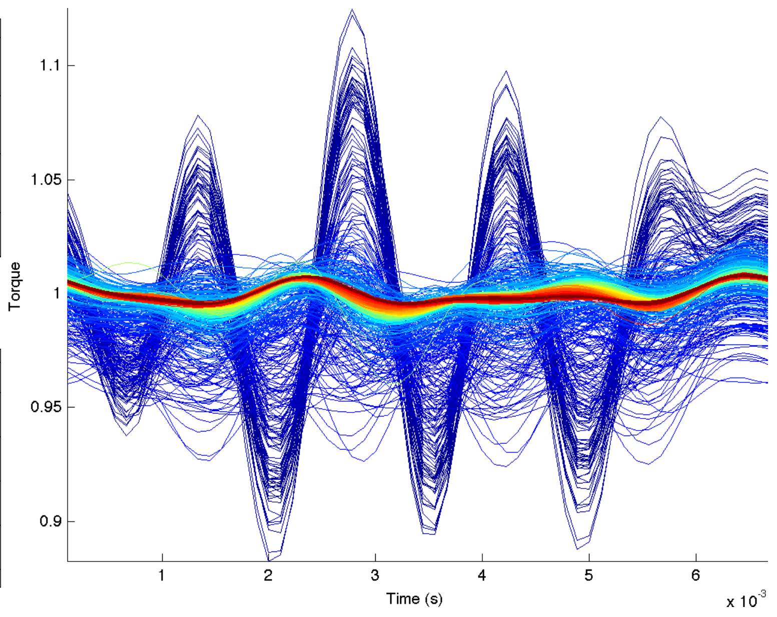
(a) Results (b) Torque (c) Voltage (d) Inductance

Figure A.1: Low torque, 10 steps, $-pf_{val} + exp(sign(Ld_{val}).*Ld_{val}) - 1$

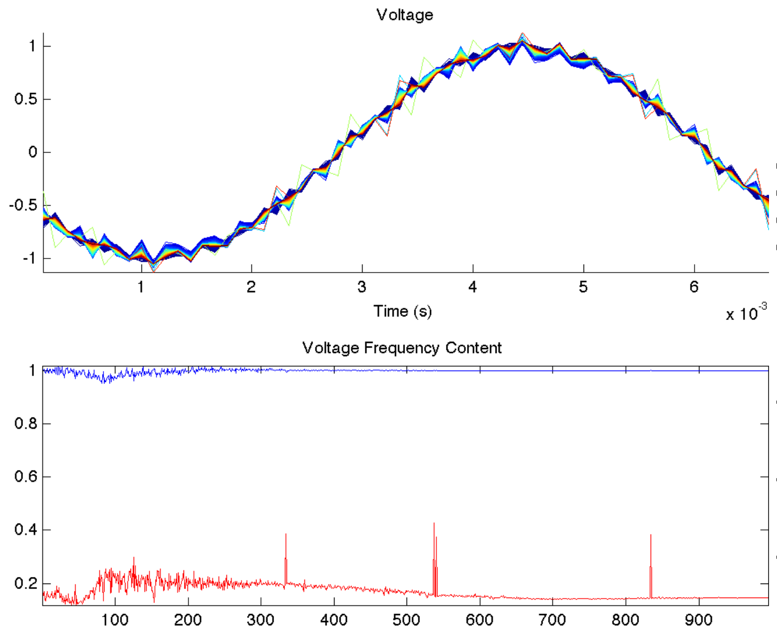
We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.



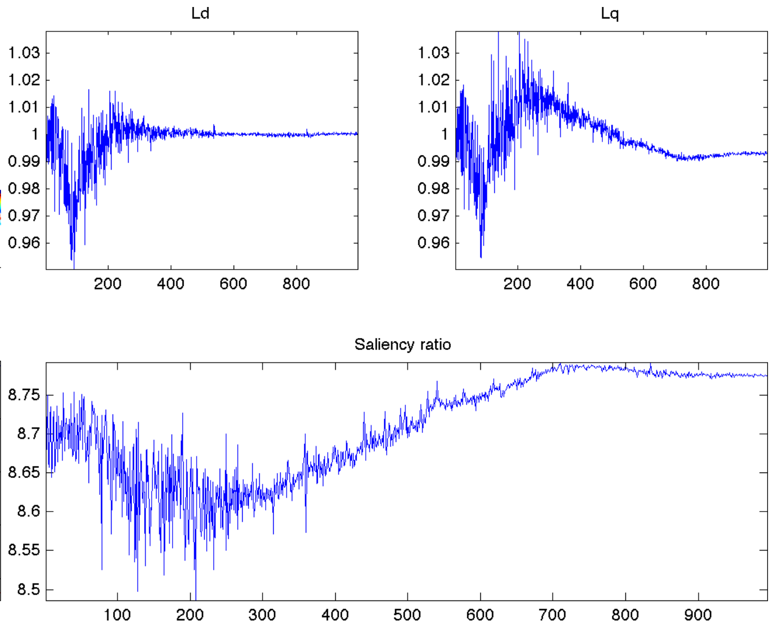
(a) Results



(b) Torque



(c) Voltage



(d) Inductance

Figure A.2: High torque, 10 steps, $Lq_{val} + 3 * sign(Ld_{val}) .* Ld_{val} - Ls_{val} + ripple_{val}$

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

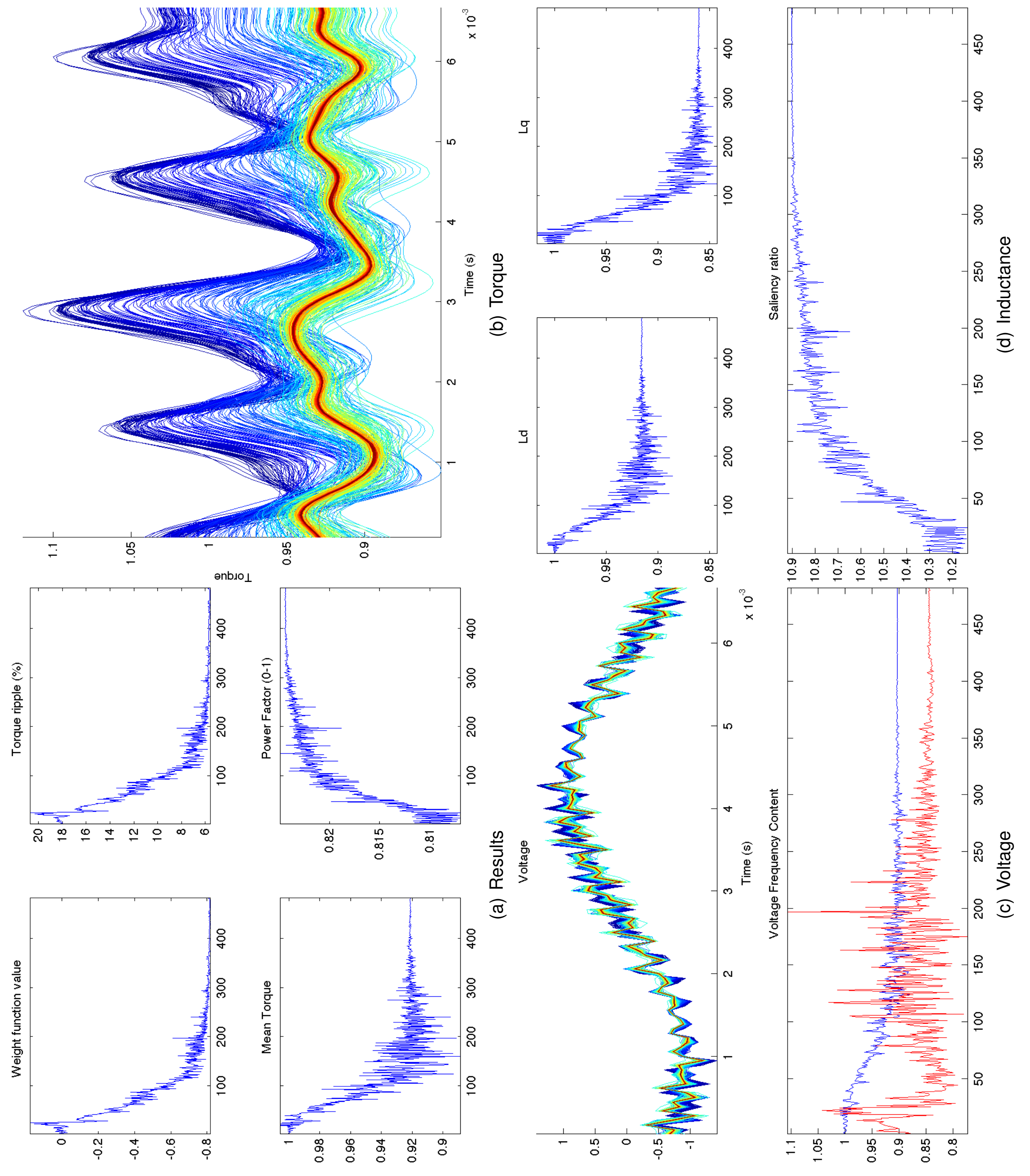
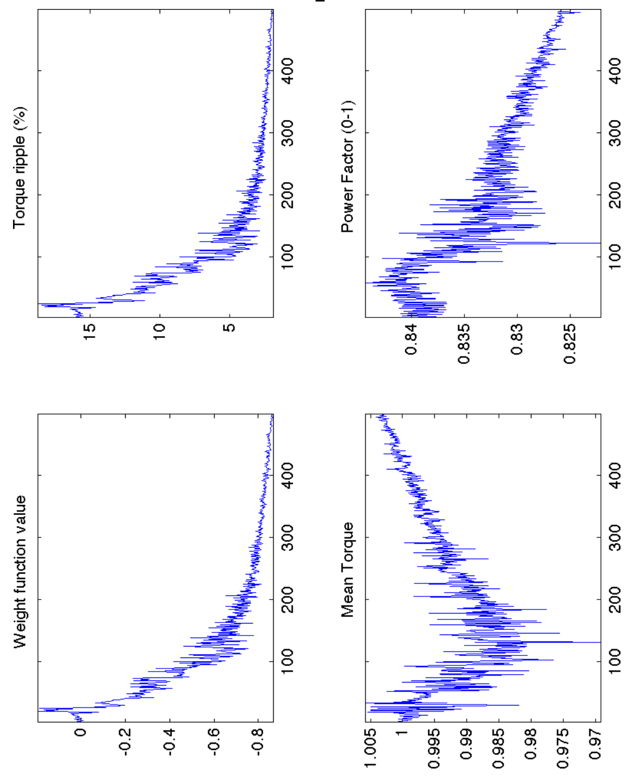
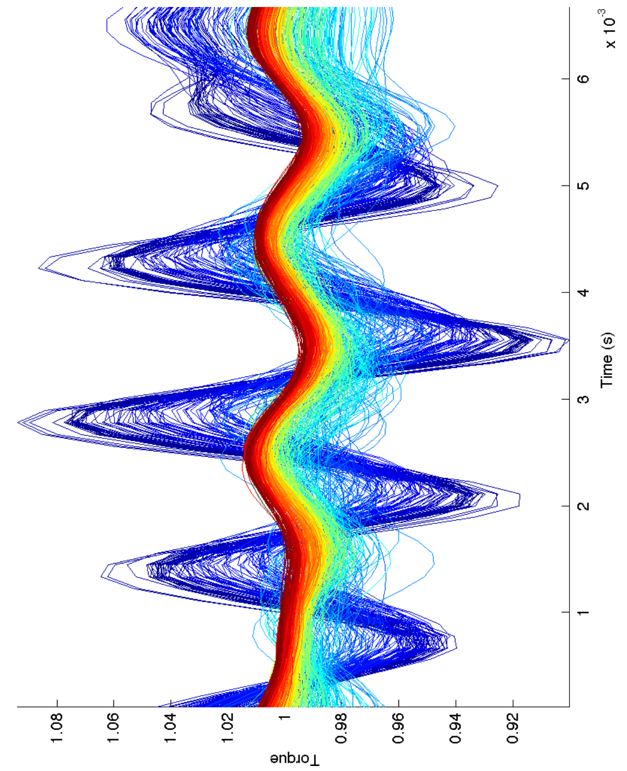


Figure A.3: Low torque, 20 steps, $Lq_{val} + exp(sign(Ld_{val}).*Ld_{val}) - 1 - Ls_{val} + ripple_{val}$

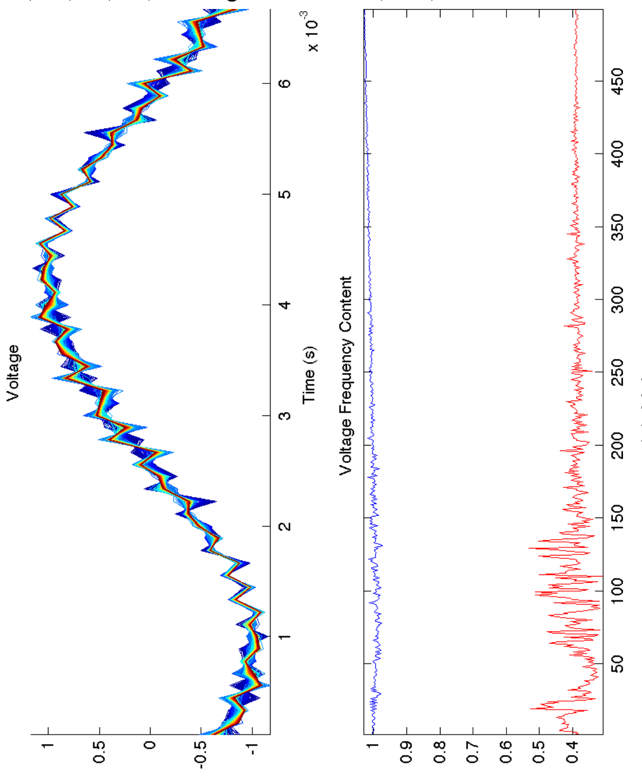
We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.



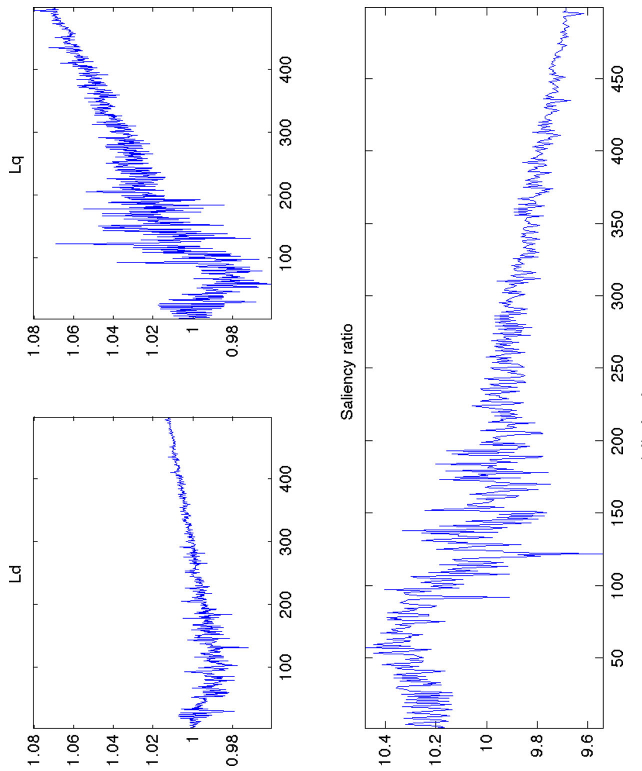
(a) Results



(b) Torque



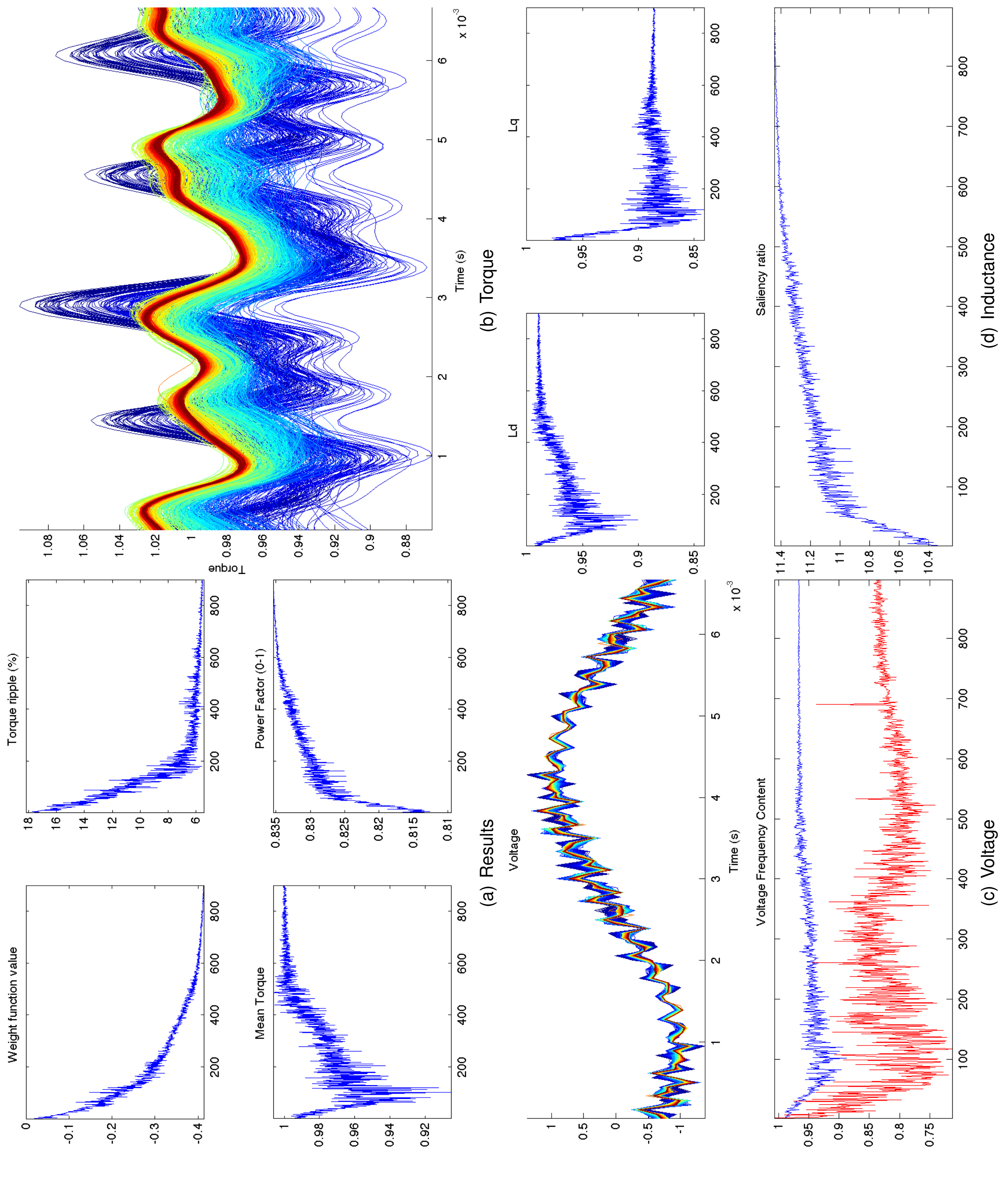
(c) Voltage



(d) Inductance

Figure A.4: Low torque, 10 steps, $ripple_{val} - T_{val}$

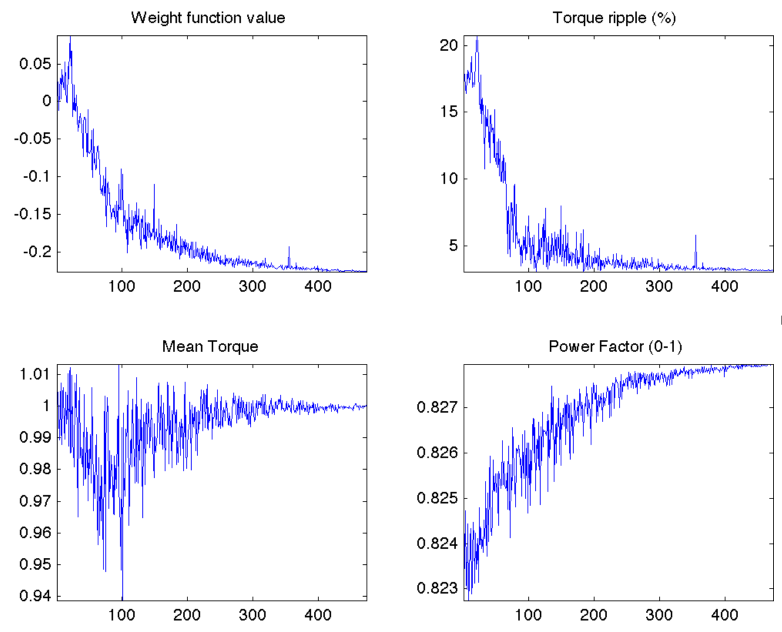
We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.



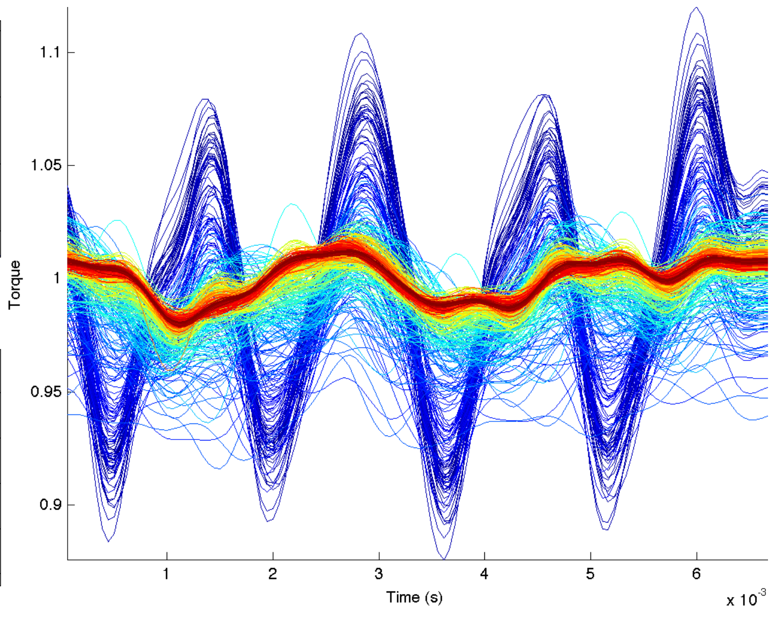
(a) Results (b) Torque (c) Voltage (d) Inductance

Figure A.5: Low torque, 20 steps, $-pf_{val} + ripple_{val}/5 + exp(sign(T_{val}).*T_{val}) - 1 + exp(sign(Ld_{val}).*Ld_{val}) - 1 + Lq_{val} - Ls_{val}$

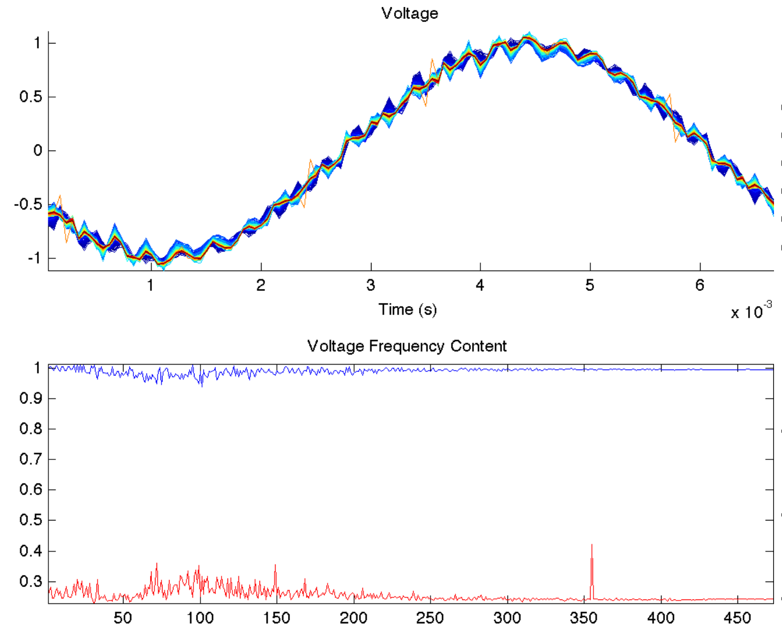
We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.



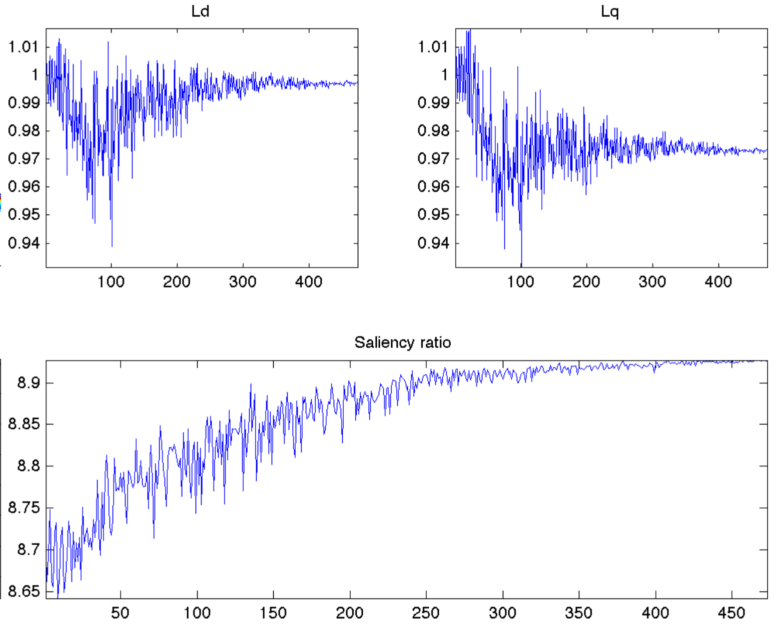
(a) Results



(b) Torque



(c) Voltage



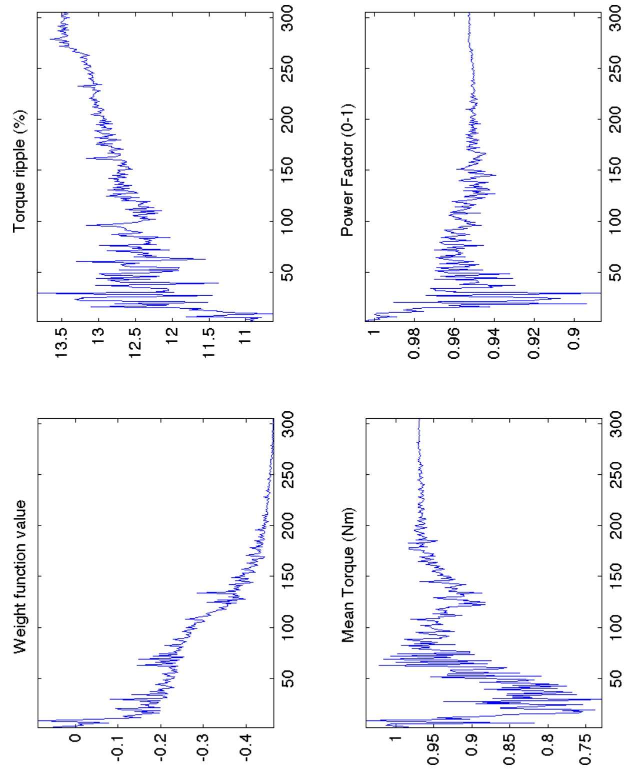
(d) Inductance

Figure A.6: High torque, 20 steps, $-pf_{val} + ripple_{val}/5 + exp(sign(T_{val}).*T_{val}) - 1 + exp(sign(Ld_{val}).*Ld_{val}) - 1 + Lq_{val} - Ls_{val}$

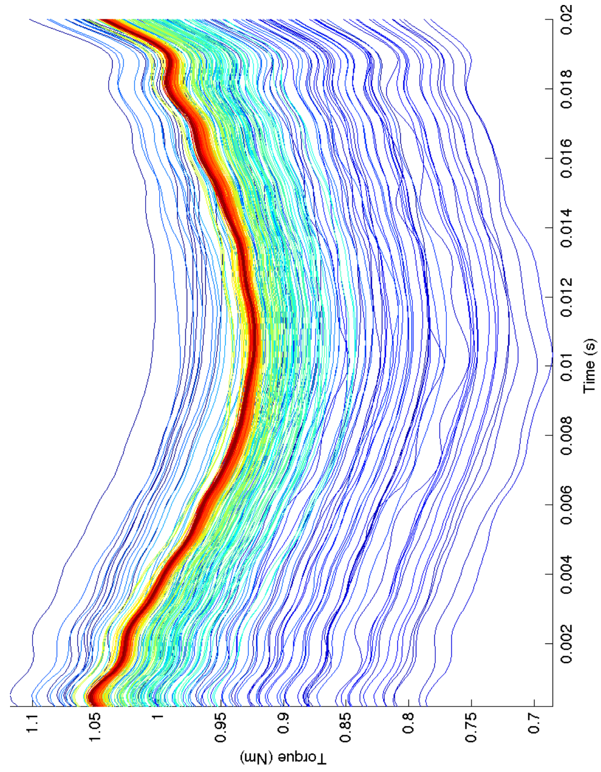
We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

B OPTIMIZATION RESULTS, MOTOR B

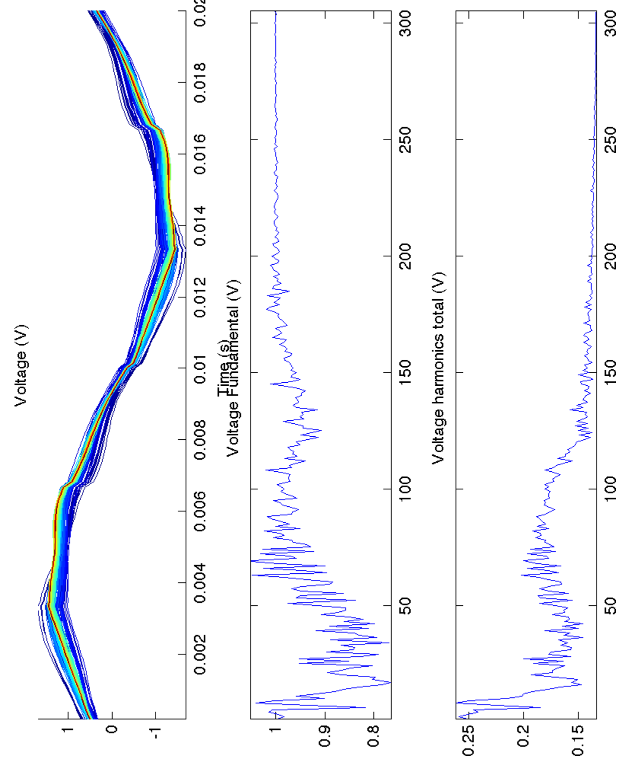
We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.



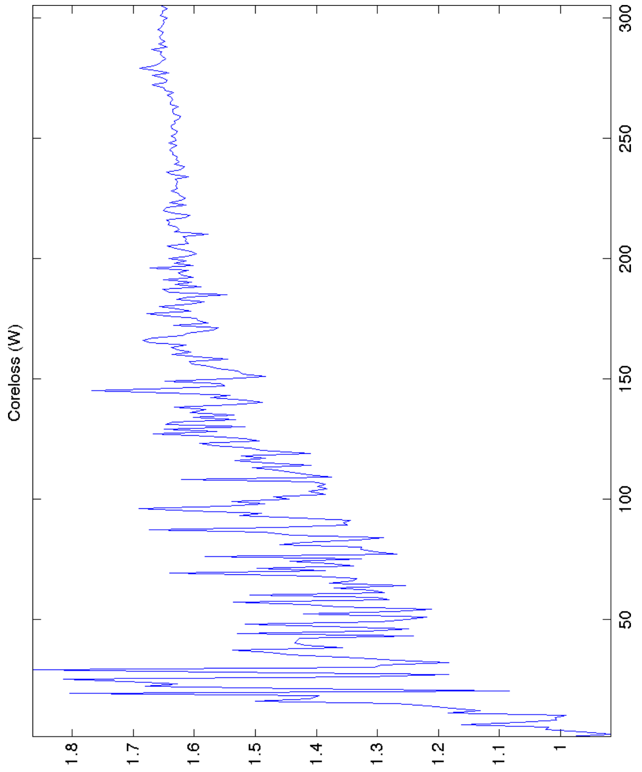
(a) Results



(b) Torque



(c) Voltage

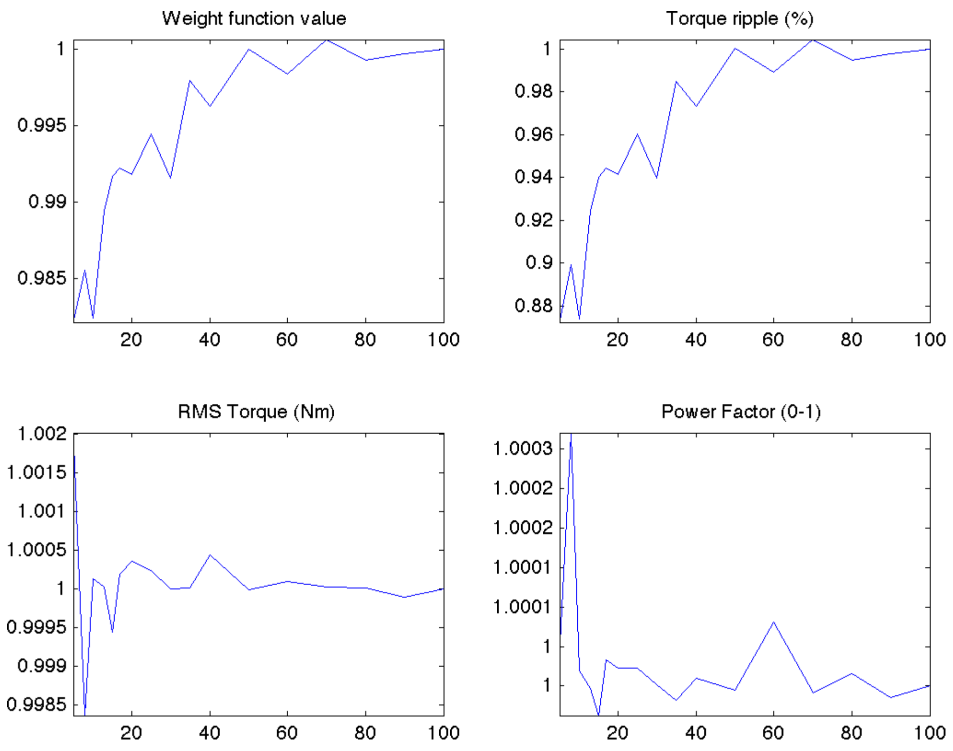


(d) Ohmic Loss

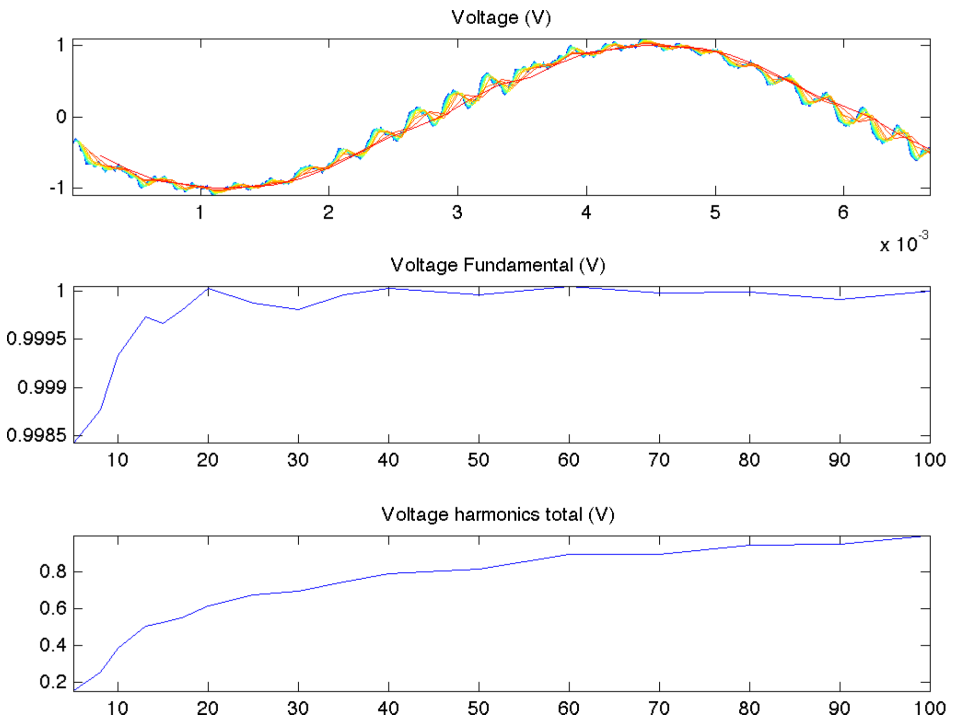
Figure B.1: 20 steps, $Vs_{val} + (sign(Vf_{val}).*Vf_{val})$

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

C TIME RESOLUTION TESTING



(a) Results



(b) Simulation time

Figure C.1: *time4*

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

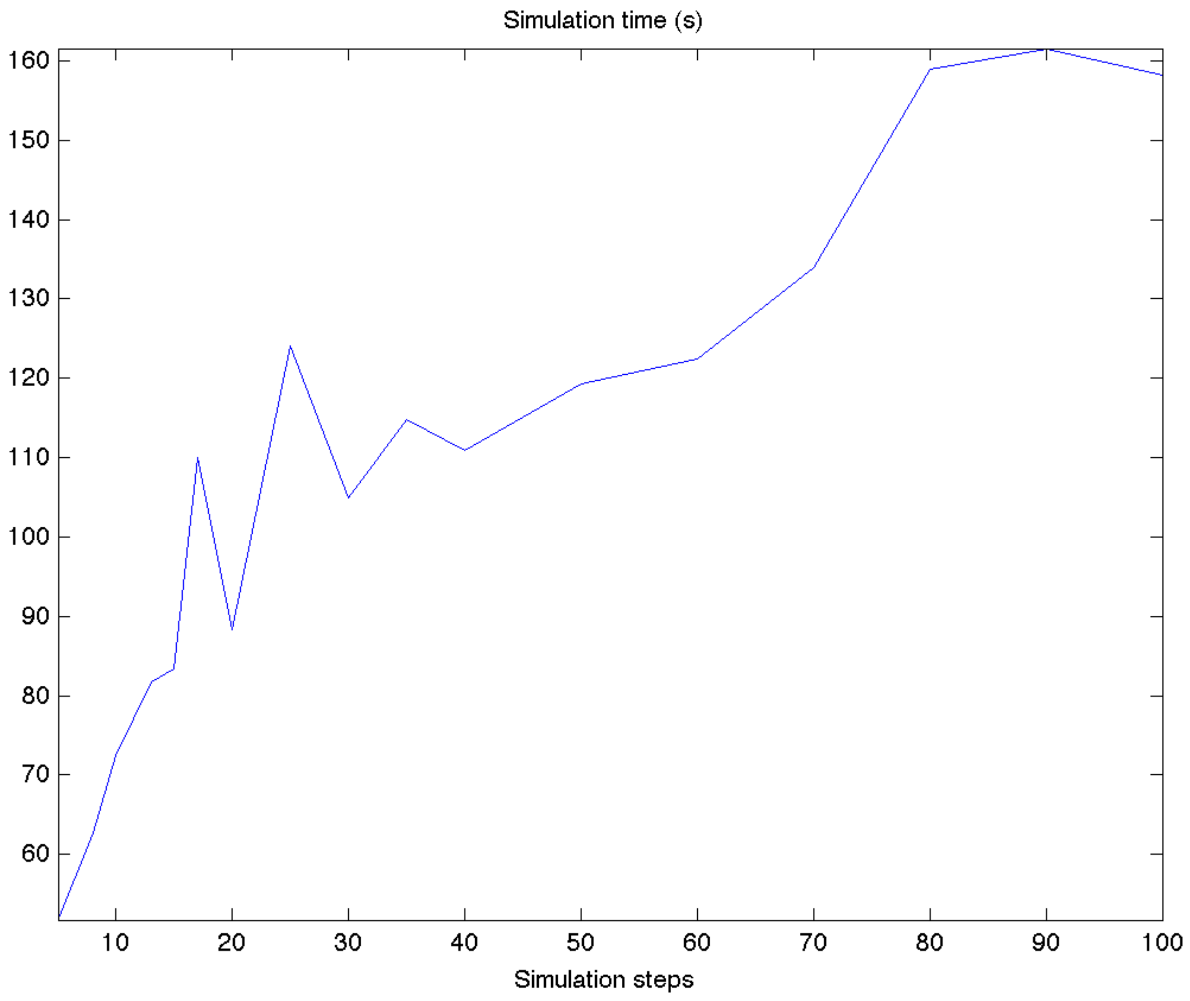


Figure C.2: Simulation time

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

D PROGRAM CODE

Simulation

```
function val = Sim(in)%,steps)
steps = 30;

% initializing Maxwell
persistent pf_nom;
persistent T_nom;
persistent ripple_nom;
persistent Ld_nom;
persistent Lq_nom;
persistent Ls_nom;
persistent Vf_nom;
persistent Vh_nom;
persistent Cls_nom;
persistent counter;
if isempty(counter)
    counter = 1;
end
project_name='Sim';
design_name='Transient';
project_path='\matlab\Sim\';

% Define the activex scripting engine
iMaxwell = actxserver('AnsoftMaxwell.MaxwellScriptInterface');

% Define the desktop object
oDesktop = invoke(iMaxwell,'GetAppDesktop');

% Open the project
if isequal(mod(counter,100)-1,0) && counter ~= 1
    oProject = invoke(oDesktop,'OpenProject', ...
        '\matlab\Sim\Sim.mxwl');
else
    oProject = oDesktop.SetActiveProject(project_name);
end
oDesign = oProject.SetActiveDesign(design_name);

% Modify design

try

    offset = zeros(length(in),1);

    n = 1; var = cell(length(in),1); in = 8e3.*in;
    for i=1:(length(in)-3)/6;
        k = num2str(i);
        var{n} = {['NAME:' ['$C' num2str(k) 'L1']], 'Value:=', num2str(offset(n)+in(n))};
        var{n+1} = {['NAME:' ['$C' num2str(k) 'L2']], 'Value:=', num2str(offset(n+1)+in(n+1))};
        var{n+2} = {['NAME:' ['$C' num2str(k) 'U1']], 'Value:=', num2str(offset(n+2)+in(n+2))};
        var{n+3} = {['NAME:' ['$C' num2str(k) 'U2']], 'Value:=', num2str(offset(n+3)+in(n+3))};
        n = n+4;
    end
    for i=1:(length(in)-3)/6;
        k = num2str(i);
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

```
            var{n} = {['NAME:' ['$CrB' num2str(k) '_1']], 'Value:=', num2str(offset(n)+in(n))};
            var{n+1} = {['NAME:' ['$CrB' num2str(k) '_2']], 'Value:=', num2str(offset(n+1)+in(n+1))};
            n = n+2;
        end
        var{n} = {'NAME:$Ccr', 'Value:=', num2str(in(n))};
        var{n+1} = {'NAME:$Cc_1', 'Value:=', num2str(in(n+1))};
        var{n+2} = {'NAME:$Cc_2', 'Value:=', num2str(in(n+2))};

        invoke(oProject,'ChangeProperty',{'NAME:AllTabs', {'NAME:ProjectVariableTab', ...
            {'NAME:PropServers', 'ProjectVariables'}, {'NAME:ChangedProps', ...
            {'NAME:$k', 'Value:=', num2str(5)}}}})

        invoke(oProject,'ChangeProperty',{'NAME:AllTabs', {'NAME:ProjectVariableTab', ...
            {'NAME:PropServers', 'ProjectVariables'}, {'NAME:ChangedProps', var{:}}}})

        oModule = oDesign.GetModule('AnalysisSetup');
        stepi = ['t_final/' num2str(6*steps)];
        invoke(oModule,'EditSetup','SetupShort', ...
            {'NAME:SetupShort', 'StopTime:=', 't_final/6', 'TimeStep:=', stepi});

    catch

        disp('Fail to initiate')

    end
    try

        % Solve the nominal problem
        tic
        invoke(oDesign,'Analyze', 'SetupShort')
        usedtime = toc;

    catch

        try
            T = timer('StartDelay',60,'TimerFcn','disp(''Restart Analyze'')');
            start(T); wait(T);
            invoke(oDesign,'Analyze', 'SetupShort')
            usedtime = toc;
        catch

            val = 0;
            disp('Fail to simulate')
            usedtime = toc;
            dlmwrite('results/fail.txt', [counter usedtime-60 in], '-append', 'delimiter', ' ')
            delete(iMaxwell);
            return
        end

    end
    try % Read output Variables

        var = oDesign.GetNominalVariation();

        oModule = oDesign.GetModule('ReportSetup');
        reportNames = oModule.GetAllReportNames;
        for i = 1:length(reportNames)
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

```
    oModule.ExportToFile(reportNames{i}, ...
        ['/matlab/Sim/results/' ...
            reportNames{i} '.tab']);
end

for i = 1:length(reportNames)
    eval([reportNames{i} ...
        ' = importdata([''results/'' reportNames{i} ''.tab''],''\t'');']);
end

% Interpret data

speed = oDesign.GetVariableValue('rpm_speed');
speed = str2double(speed(1:length(speed)-3))*(2*pi/60);

I1 = oDesign.GetVariableValue('I1');
Is = str2double(I1(1:length(I1)-1))*sqrt(2);

efreq = oDesign.GetVariableValue('efreq');
efreq = str2double(efreq(1:length(efreq)-2));

phiA = oDesign.GetVariableValue('phiA');
phiA = str2double(phiA(1:length(phiA)-3))*(pi/180);

t_final = 1/efreq; time = (0:t_final/(6*steps):t_final/6)';

idx = 2:1:length(time);
time4 = (1:(length(time)-1)*6)*(t_final/(6*steps));
%time2 = (0:t_final/(360):t_final); idx2 = 2:1:length(time2); time2 = time2(idx2);

I = Is*sin(2*pi*time4*efreq + phiA);

Va = V.data(idx,2);
Vb = V.data(idx,3);
Vc = V.data(idx,4);

Fa = F.data(idx,2);
Fb = F.data(idx,3);
Fc = F.data(idx,4);

V6 = [Va; -Vb; Vc; -Va; Vb; -Vc]';
F6 = [Fa; -Fb; Fc; -Fa; Fb; -Fc]';

Cl = 1e-6.*coreLoss.data(idx,2);

T = T.data(idx,2);

Vfft = 2*fft(V6)/length(V6);

% Calculate objective function

[Id,Iq,Ld,Lq] = induct(F6,I);

T_rms = norm(T)/sqrt(length(T));
ripple = (max(T)-min(T))/mean(T); % ripple

Vh = sum(abs(Vfft(3:end/4)));
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

```
Vf = abs(Vfft(2));

pf = powerFactor(V6,I);

Cls = sum(Cl);

if counter == 1
    pf_nom = pf;
    T_nom = mean(T);
    ripple_nom = ripple;
    Ld_nom = Ld;
    Lq_nom = Lq;
    Ls_nom = Ld/Lq;
    Vh_nom = Vh;
    Vf_nom = Vf;
    Cls_nom = Cls;
end

pf_val = (pf-pf_nom)/pf_nom;
ripple_val = (ripple-ripple_nom)/ripple_nom;
T_val = (mean(T)-T_nom)/T_nom;
Ld_val = (Ld-Ld_nom)/Ld_nom;
Lq_val = (Lq-Lq_nom)/Lq_nom;
Ls_val = ((Ld/Lq)-Ls_nom)/Ls_nom;
Vh_val = (Vh-Vh_nom)/Vh_nom;
Vf_val = (Vf-Vf_nom)/Vf_nom;
Cls_val = (Cls-Cls_nom)/Cls_nom;

val = Lq_val + 3*(sign(Ld_val).*Ld_val) + ripple_val;

% Write results to file
str = '_L_r30';

if counter == 1
    dlmwrite(['results/data' str '.txt'], 'Lq + 3*(sign(Ld)) + ripple', ...
        '-append', 'delimiter', '')
    dlmwrite(['results/data' str '.txt'], time', '-append', 'delimiter', ' ')
    dlmwrite(['results/data' str '.txt'], speed, '-append', 'delimiter', ' ')
    dlmwrite(['results/data' str '.txt'], Is, '-append', 'delimiter', ' ')
    dlmwrite(['results/data' str '.txt'], efreq, '-append', 'delimiter', ' ')
    dlmwrite(['results/data' str '.txt'], phiA, '-append', 'delimiter', ' ')
end
dlmwrite(['results/torque' str '.txt'], T', ...
    '-append', 'delimiter', ' ')
dlmwrite(['results/V' str '.txt'], V6, '-append', 'delimiter', ' ')
dlmwrite(['results/F' str '.txt'], F6, '-append', 'delimiter', ' ')
dlmwrite(['results/coreLoss' str '.txt'], Cl', ...
    '-append', 'delimiter', ' ')
dlmwrite(['results/input' str '.txt'], [in val usedtime], ...
    '-append', 'delimiter', ' ')
disp(sprintf('%d %0.1f %0.4f',counter,usedtime,val))

% Finish iteration

invoke(oDesign,'DeleteFullVariation', var, false);

if isequal(mod(counter,10),0)
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

```
            invoke(oProject,'Save')
            disp('save')
            if isequal(mod(counter,100),0)
                oProject.Close
                oDesktop.QuitApplication
                disp('restart')
            end
        end
        counter = counter + 1;

    catch

        val = 0;
        disp('Fail to extract value')

    end

    delete(iMaxwell);
```

Objectives

Inductance

```
function [Id,Iq,Ld,Lq] = induct(F,I)
N = length(I);

corr = (2/N)*exp(-1i*(15+360/N)*pi/180);
If = fft(I)*corr;
Ff = fft(F)*corr;

Id = real(If(2));
Iq = imag(If(2));

lambdad = real(Ff(2));
lambdaq = imag(Ff(2));

Ld = lambdad/Id;
Lq = lambdaq/Iq;
```

Power Factor

```
function pf = powerFactor(V,I)

II6 = fft(I);
phiI = angle(II6(2));

VV6 = fft(V)*exp(-j*(pi/length(V)));
phiV = angle(VV6(2));

pf = cos(phiV-phiI);
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

Visualization

```
clear
str = 'beta';
eval(['load ' str])

torque = interpft(torque,size(V,1));
time = data.data(1,:)'; rpm = data.data(2,1); Is = data.data(3,1);
efreq = data.data(4,1); phiA = data.data(5,1); timeTaken = input(end,:);
t_final = 1/efreq; k = 1:size(V,2); k = (1*(k-1))-9;

time2 = linspace(time(2)-time(1),t_final,6*(length(time)-1))';
I1 = Is*sin(2*pi*time2*efreq + phiA);

[opt_y opt_i] = min(input(size(input,1)-1,:));

col = colormap(jet(length(k)));

% Calculating objective function and visualization

N = size(V,1); %corr = sqrt(2)*exp(-j*(1+N/6)/180*pi)/N;
Vfft = 2*fft(V,N)/(N);
f = ((efreq*N)/2)*linspace(0,1,N/2);
Vh = sum(abs(Vfft(3:end/2+1,:)));
Vf = abs(Vfft(2,:));

for i = 1:size(F,2)
    [Id(i),Iq(i),Ld(i),Lq(i)] = induct(F(:,i),I1);
end

Cm = mean(coreLoss);

for i = 1:size(torque,2)
    Ms(i) = mean(torque(:,i));
    Rs(i) = (max(torque(:,i))-min(torque(:,i)))/Ms(i);
end

for i = 1:length(k)
    PFs(i) = powerFactor(V(:,i),I1);
end

Ws = input(size(input,1)-1,:);

% ----------- Create p.u. values

PFs_val = PFs./PFs(1); %(PFs-PFs(1))/PFs(1);
Rs_val = Rs./Rs(1); %(Rs-Rs(1));/Rs(1);
Ms_val = Ms./Ms(1); %(Ms-Ms(1))/Ms(1);
Vf_val = Vf./Vf(1); %(Vf-Vf(1))/Vf(1);
Vh_val = Vh./Vf(1); %(Vh-Vh(1))/Vh(1);
Ld_val = Ld./Ld(1); %(Ld-Ld(1))/Ld(1);
Lq_val = Lq./Lq(1); %(Lq-Lq(1))/Lq(1);
Ls_val = (Ld./Lq)/(Ld(1)./Lq(1)); %(Ld./Lq - Ld(1)./Lq(1))./Ld(1)./Lq(1);
Cm_val = Cm./Cm(1); %(Cm-Cm(1))/Cm(1);

% ----------- Plotting motor performance
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

```
figure(2); cla; hold on
for i = 1:size(torque,2)
    plot(time2,torque(:,i)./Ms(1),'color',col(i,:))
end
ylabel('Torque'); xlabel('Time (s)'); axis tight

saveas(gcf,['pic/' str '_torque.png'],'png')

figure(3)
subplot(2,1,1); cla; hold on
for i = 1:size(V,2)
    plot(time2,V(:,i)./Vf(1),'color',col(i,:))
end
title('Voltage'); xlabel('Time (s)'); axis tight
subplot(2,1,2)
ax = [k(1) k(end) min([Vf_val Vh_val]) max([Vf_val Vh_val])];
plot(k,Vf_val,k,Vh_val,'r'); title('Voltage Frequency Content'); axis(ax)
%subplot(3,1,3)
%plot(k,Vh_val); title('Voltage harmonics total (V)'); axis(ax)

saveas(gcf,['pic/' str '_voltage.png'],'png')

figure(4)
subplot(2,2,1); hold off
ax = [k(1) k(end) min([Ld_val Lq_val]) max([Ld_val Lq_val])];
plot(k,Ld_val); title('Ld'); axis(ax)
subplot(2,2,2); hold off
plot(k,Lq_val); title('Lq'); axis(ax)
subplot(2,2,3:4); hold off
plot(k,Ld./Lq); title('Saliency ratio'); axis tight

saveas(gcf,['pic/' str '_L.png'],'png')

figure(5); hold off
subplot(1,1,1)
plot(k,Cm_val); title('Losses'); axis tight

figure(6); cla; hold on
for i = 1:size(V,2)
    plot(abs(Vfft(1:end/2,i))./Vf(1),'color',col(i,:))
end
title('Voltage FFT'); xlabel('Frequency (Hz)'); axis tight

figure(7); hold off
plot(k,timeTaken,[k(1) k(end)],[mean(timeTaken) mean(timeTaken)],'r')
title('timeTaken')
xlabel(['Mean: ' num2str(mean(timeTaken)) ' s , Total: ' ...
    num2str(int16(sum(timeTaken)/3600)) ' h'])
axis tight

% ----------- Plotting objective functions

figure(1)

subplot(2,2,3); hold off
plot(k,Ms_val)
title('Mean Torque'); axis tight;
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

```
subplot(2,2,2); hold off
plot(k,100*Rs)
title('Torque ripple (%)'); axis tight;

subplot(2,2,4)
plot(k,PFs)
title('Power Factor (0-1)'); axis tight;

Ws = - PFs_val + Rs_val - Ms_val - Ld_val + Lq_val + Cm_val - Vh_val;

subplot(2,2,1); hold off
plot(k,Ws)
title('Weight function value'); axis tight;

saveas(gcf,['pic/' str '.png'],'png')

% ----------- Concluding Results

T_opt = Ms(opt_i);
Rs_opt = Rs(opt_i);
PFs_opt = PFs(opt_i);
Ls_opt = Ld(opt_i)/Lq(opt_i);
t_avg = median(input(size(input,1),:));

val = Vh_val + (sign(Vf_val).*Vf_val).^(1/2);

dlmwrite(['pic/' str '_text.txt'], ['Original solution (' num2str(1) ')'], ...
dlmwrite(['pic/' str '_text.txt'], ['Mean Torque = ' num2str(Ms(1)) ' Nm'], ...
dlmwrite(['pic/' str '_text.txt'], ['Ripple = ' num2str(100*Rs(1)) ' %'], ...
dlmwrite(['pic/' str '_text.txt'], ['Power Factor = ' num2str(PFs(1))], ...
dlmwrite(['pic/' str '_text.txt'], ['Voltage Factor = ' num2str(Vf(1)/Vh(1))], ...
dlmwrite(['pic/' str '_text.txt'], ['Saliency = ' num2str(Ld(1)/Lq(1))], ...

dlmwrite(['pic/' str '_text.txt'], ' ', '-append', 'delimiter', '')
dlmwrite(['pic/' str '_text.txt'], ['Objective: ' data.textdata{1}], ...
dlmwrite(['pic/' str '_text.txt'], ['Optimal solution (' num2str(opt_i) ')'], ...
dlmwrite(['pic/' str '_text.txt'], ['Mean Torque = ' num2str(T_opt) ' Nm'], ...
dlmwrite(['pic/' str '_text.txt'], ['Ripple = ' num2str(Rs_opt) ' %'], ...
dlmwrite(['pic/' str '_text.txt'], ['Power Factor = ' num2str(PFs_opt)], ...
dlmwrite(['pic/' str '_text.txt'], ['Voltage Factor = ' num2str(Vf(opt_i)/Vh(opt_i))], ...
dlmwrite(['pic/' str '_text.txt'], ['Saliency = ' num2str(Ls_opt)], ...
dlmwrite(['pic/' str '_text.txt'], ['Optimal change = ' num2str(100*opt_y) ' %'], ...
dlmwrite(['pic/' str '_text.txt'], ...
['phiA = ' num2str(phiA*(180/pi)) ', Time steps = ' num2str(size(time,1))], ...
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.

Make GIF output

```
function dxf_generator(input)

% initializing Maxwell
if 24==length(input)
    project_name='Variables';
else
    project_name='Variables_cutoff';
end
design_name='Maxwell2DDesign';

% Define the activex scripting engine
iMaxwell = actxserver('AnsoftMaxwell.MaxwellScriptInterface');

% Define the desktop object
oDesktop = invoke(iMaxwell,'GetAppDesktop');

% Open the project
oProject = oDesktop.SetActiveProject(project_name);
oDesign = oProject.SetActiveDesign(design_name);
oEditor = oDesign.SetActiveEditor('3D Modeler');

% Modify design
var = cell(size(input,1),1);
for p = 1:size(input,2)
    in = input(:,p);
    n = 1;
    for i=1:4;
        k = num2str(i);
        var{n} = {['NAME:' ['$C' num2str(k) 'L1']], 'Value:=', num2str(in(n))};
        var{n+1} = {['NAME:' ['$C' num2str(k) 'L2']], 'Value:=', num2str(in(n+1))};
        var{n+2} = {['NAME:' ['$C' num2str(k) 'U1']], 'Value:=', num2str(in(n+2))};
        var{n+3} = {['NAME:' ['$C' num2str(k) 'U2']], 'Value:=', num2str(in(n+3))};
        n = n+4;
    end
    for i=1:4;
        k = num2str(i);
        var{n} = {['NAME:' ['$CrB' num2str(k) '_1']], 'Value:=', num2str(in(n))};
        var{n+1} = {['NAME:' ['$CrB' num2str(k) '_2']], 'Value:=', num2str(in(n+1))};
        n = n+2;
    end
    if length(in)>24
        var{n} = {['NAME:' ['$Ccr']], 'Value:=', num2str(in(n))};
        var{n+1} = {['NAME:' ['$Cc_1']], 'Value:=', num2str(in(n+1))};
        var{n+2} = {['NAME:' ['$Cc_2']], 'Value:=', num2str(in(n+2))};
    end

    invoke(oProject,'ChangeProperty',{'NAME:AllTabs', {'NAME:ProjectVariableTab', ...
        {'NAME:PropServers', 'ProjectVariables'}, {'NAME:ChangedProps', var{:}}}})

    invoke(oEditor,'Export', {'NAME:ExportParameters', 'File Name:=', ...
        ['/matlab/SiM/pic/pic' num2str(p) '.dxf']})
end

delete(iMaxwell);
```

We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.
©2009 ABB Ltd.