NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

NTNU

Kunnskap for en bedre verden

MASTER THESIS

Name of the candidate : Asgeir Leite

Main topic : **ELECTRICAL POWER ENGINEERING**

Title : Use of FEM in design program for hydropower generators

Problem description:

Generators are normally designed using a calculation program. The designer gives all geometrical and physical parameters of the machine, and then the program calculates the performance and machine parameters. All hydropower machines in Norway are designed by programs of this kind. At our department, we are developing a program of this kind based on open and free sources, not using in-house manufactures "secret" formulas. In this project the student will expand the accuracy and quality. The main goal is to include COMSOL for more accurate calculations of parameters and increased understanding of the design process. The program is planned to be used in various courses in electric machines and also to be made public as part of the results from the research center HydroCen. PhD-students have been using this program for research purposes also.

In more detail, the work will comprise:

- Familiarize with the design of hydro power generators
- Export the geometry to COMSOL for field calculation
- Calculate various design parameters

Project starting date : 17. January 2022
Project submission deadline : 27. June 2022
Carried out at : Dep. of Electric Power Engineering/NTNU
Responsible supervisor : Professor Arne Nysveen

June, 2022

Preface

This master thesis is written at the Department of Electrical Power Engineering at the Norwegian University of Science and Technology (NTNU) during spring semester 2022.

Generators are normally designed using a calculation program. The department of Electric Power Engineering are developing a program of this kind based on open and free sources. The program is planned to be used in various courses in electric machines and also to be made public as part of the results from the research center HydroCen. It is therefore important to develop a user friendly software, while also increasing accuracy and quality.

I am grateful to get the opportunity to participate in research in this field. I want to thank my supervisors for the involvement in this project, Professor Arne Nysveen at NTNU.

Abstract

Design of hydropower generators is a difficult optimization problem. Generators are normally designed using a calculation program. The designer gives all geometrical and physical parameters of the machine, and then the program calculates the performance and machine parameters. For the most part, the dimensions have to be guessed and adjusted until an analysis gives the desired result. Companies that have a lot of experience with generator design often develop a so called in-house manufactures "secret" formulas used to ease the design procedure.

The department of Electric Power Engineering are developing a program of this kind based on open and free sources called GenProg. The core script first developed almost 10 years ago is based on a design procedure of a typical salient pole generator ranging from 10 to 50 MVA. GenProg uses several key parameters obtained from the customer needs to return every parameter describing a complete generator.

This project will expand the accuracy and quality of the program. The main goal is to include COMSOL for more accurate calculations of parameters and increased understanding of the design process. A considerable effort has been made to understand the core script in order to adapt and create the COMSOL models incorporated with the program. The program is planned to be used in various courses in electric machines and also to be made public as part of the results from the research center Hydrogen.

The newest version of GenProg numerically solves for to conditions, Dynamic and Static. The Dynamic condition is time dependent and simulates a rotating machine that results in a study of flux density and voltages. The Static condition simulates for two time instances, maximised MMF in d-axis and q-axis. This results in a study of flux density and synchronous reactances for both d-axis and q-axis current.

GenProg creates a user friendly learning environment with visual interactions required for this trial and error analysis that is needed in order to achieve desired results. The numeric solutions calculated from COMSOL are more detailed and accurate than the analytical part of GenProg. Flux density values are quite similar, but the visual map of flux density from COMSOL offers far more information for the user. Simplifications in the GenProg script creates a big difference in reactance values, thus indicating more accurate value from COMSOL. There are still several analytical parameters to numerically calculate with COMSOL, however with the COMSOL models as groundwork simply time is needed before more are implemented.

Table of Contents

List of Figures

List of Tables

1 Introduction

Generators are normally designed using a calculation program. The designer gives all geometrical and physical parameters of the machine, and then the program calculates the performance and machine parameters. All hydropower machines in Norway are designed by programs of this kind. Design of hydro power generators is a difficult optimization problem. Construction of an electric machine is a process that only partly consists of putting values into formulas and getting dimensions out. For the most part, the dimensions have to be guessed, more or less justified. The construction must then be analyzed, and the dimensions adjusted until the analysis gives the desired result. Due to this nature, companies develop a so called in-house manufactures "secret" formulas from years of experience that is not shared with outsiders. The department of Electric Power Engineering are developing a program of this kind, referred to as GenProg, based on open and free sources in hopes that it can be utilized as a learning tool for students or as a professional tool for research.

This master thesis will expand the accuracy and quality of the program. The main goal is to include COMSOL for more accurate calculations of parameters and increased understanding of the design process. The program is planned to be used in various courses in electric machines and also to be made public as part of the results from the research center HydroCen. In more detail, the thesis will familiarize with the design of hydro power generators and incorporate COMSOL for numerical analysis of various design parameters.

It is important to note that the reader is assumed to have prior knowledge of COMSOL, MATLAB and LiveLink. The thesis focuses on how the softwares are used to accomplish the thesis goals.

2 Theory

2.1 Introduction to Hydropower Generator Design

After a thorough analysis of hydrological conditions, operating conditions and economic assessments, the customer will decide on the number of aggregates in his power station and determine the generator's performance and power factor. The frequency is given by the network the generator is to work on, and in Europe is usually 50 Hz. The most economical speed is determined on the basis of unit size and head, in consultation with the turbine and generator supplier. Furthermore, the turbine supplier will state the unit's rush speed and the necessary inertia torque for the generator. The moment of inertia is determined by the permissible pressure rise and speed rise by load reduction, and by consideration of the stability conditions of the turbine control. The choice of generator voltage is often left to the generator supplier. The voltage often plays little role for the customer, especially if the generator is to work in block connection with a transformer, while it is a great advantage for the generator designer to be free at this point.

In reality, the sizing of an electric machine is a very difficult optimization problem. The construction must meet all the requirements set by the customer and the norms, and at the same time be done in the cheapest way.

The lower limit for the short-circuit ratio or the upper limit for the synchronous reactance x_d is usually specified by the customer. The value of x_d is important for the static stability, and also for how large a capacitive load the generator can have in the form of a remote charging line (line charging capacity). The upper limit of the transient reactance x'_d is specified less frequently. It is important for the transient stability and for the voltage regulation.

Construction of an electric machine is a process that only partly consists of putting values into formulas and getting dimensions out. For the most part, the dimensions have to be guessed, more or less justified. The construction must then be analyzed, and the dimensions adjusted until the analysis gives the desired result. This is also the way to proceed when the construction is carried out automatically by means of a calculation program.

The customer can refer to Norwegian standards for rotating electrical machines, or equivalent foreign standards, for general requirements that must be met during construction. The windings must withstand voltage tests far above the rated voltage. The machine must be able to operate at rated performance against voltage variation $\pm 5\%$ from the rated value, and the voltage curve of the voltage at idle must not deviate from the sine shape by more than 5% of the maximum value.

2.2 Synchronous Generator

Synchronous generators are synchronous machines used to convert mechanical power to ac electric power. In a synchronous generator, a dc current is applied to the rotor winding, which produces a rotor magnetic field. The rotor of the generator is then turned by a prime mover, producing a rotating magnetic field within the machine. This rotating magnetic field induces a three-phase set of voltages within the stator windings of the generator.

Two terms commonly used to describe the windings on a machine are field windings and armature windings. In general, the term "field windings" applies to the windings that produce the main magnetic field in a machine, and the term "armature windings" applies to the windings where the main voltage is induced. For synchronous machines, the field windings are on the rotor, so the terms "rotor windings" and "field windings" are used interchangeably. Similarly, the terms "stator windings" and "armature windings" are used interchangeably.

The rotor of a synchronous generator is essentially a large electromagnet. The magnetic poles on the rotor can be of either salient or nonsalient construction. The term salient means "protruding" or "sticking out," and a salient pole is a magnetic pole that sticks out from the surface of the rotor. A salient-pole rotor is shown in Figure 1. Salient-pole rotors are normally used for rotors with four or more poles. Because the rotor is subjected to changing magnetic fields, it is constructed of thin laminations to reduce eddy current losses.

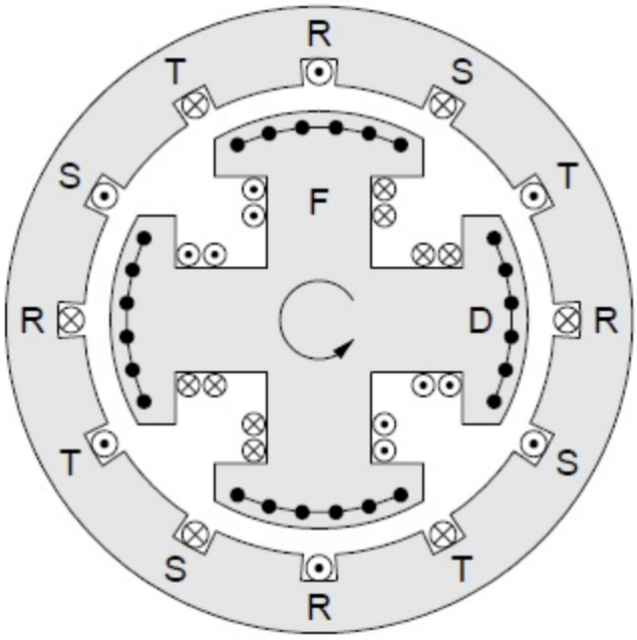


Figure 1: Salient pole generator construction

2.3 The Calculation Program

GenProg is based on a design procedure written by Professor E. Westgaard in 1964. This compendium describes the design of a typical salient pole generator ranging from 10 to 50 MVA. The document gives a detailed walk-through of a typical design procedure based on several key parameters obtained from the customer's needs and returns every parameter used to describe a complete generator. A basic understanding of how the core script operate is necessary for expanding accuracy and quality.

The script starts by calculating the external and internal dimensions of the stator. External dimensions of the stator include inner diameter, number of slots, nominal current, voltage and many other related parameters. After the external stator dimensions are determined the script calculate slot and armature dimensions from target current density. This step is also the basis for air gap flux density and stator resistance. The dimensions can at any time be defined by the user.

From the previous stator parameters the initial rotor pole dimensions are calculated, which in turn give the corresponding magnetic parameters for the entire generator. Field windings are dimensioned from the magnetic parameters. At this point if the flux values exceeds allowed limits, the script redimensions the rotor.

Finally the program proceeds to calculate values for losses, thermal, mechanical, reactances and time constants.

2.4 Analytical vs Numerical solutions

One goal for this project is to determine the accuracy of COMSOL when calculating parameters by comparing it with analytical solutions. One of the benefits of using COMSOL is its ability to model certain physical systems more accurately than analytical solutions can. In order to demonstrate this point, I will derive an analytical solution for magnetic flux density in a generator pole core, and compare it with calculations performed using COMSOL. The flux density of a magnetic field can be calculated by first determining the total flux generated from the magnetic field and then measuring the total area on which it acts. After dividing the total flux by the area, one arrives at the flux density. This is an example of an analytical solution, in which one value is found using pure math equations.

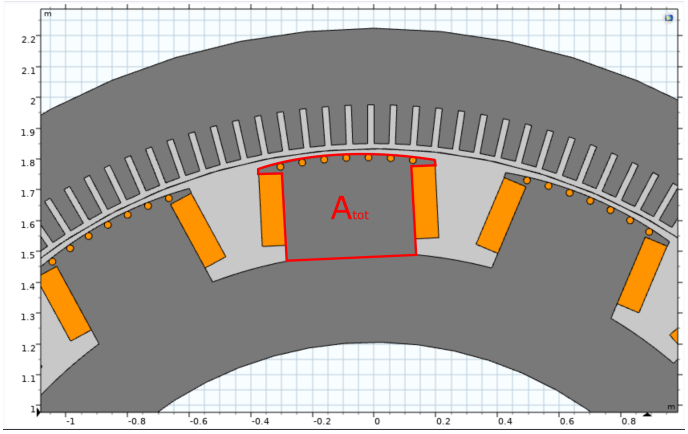


Figure 2: Pole core total Area

The problem with this approach is that it gives the average value over the area, which gives very little information about how the flux density is distributed. This is not optimal for both a design and educational tool. For a more accurate representation of flux density, we need to subdivide the larger system into smaller, simpler parts as seen in Figure 3.

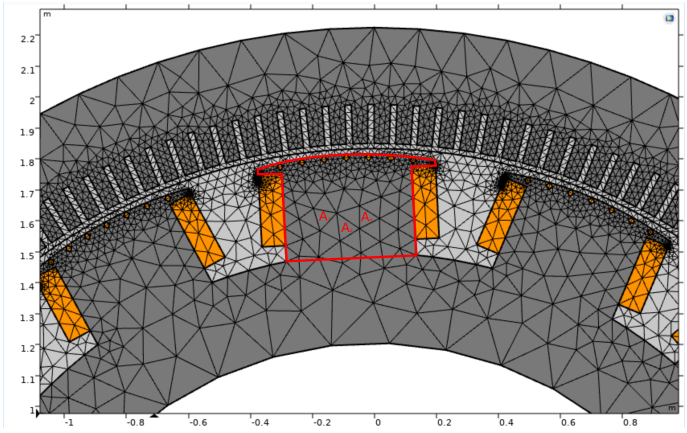


Figure 3: Pole core subdivided Area

These smaller, simpler parts generates a system of algebraic equations which impose relations between them, in other words a partial differential equation (PDE). The solutions to partial differential equations are usually impossible to write down explicitly. There are however methods to numerically approximate a solution by using computers. One of those methods is the finite element method (FEM) used by COMSOL. See Figure 4.

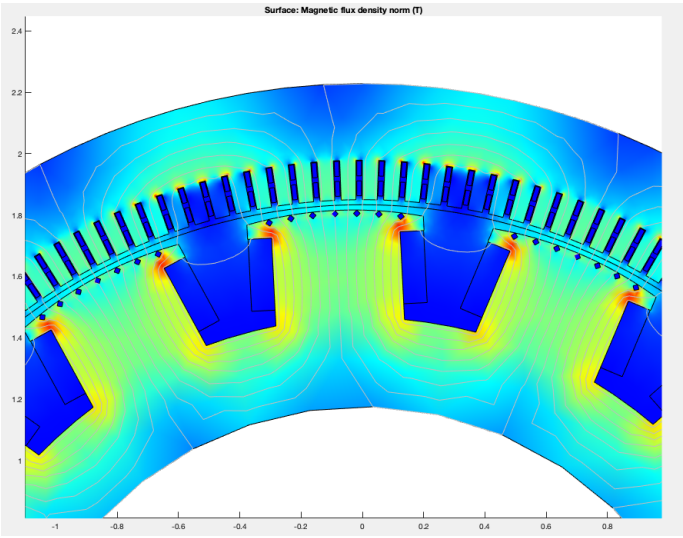
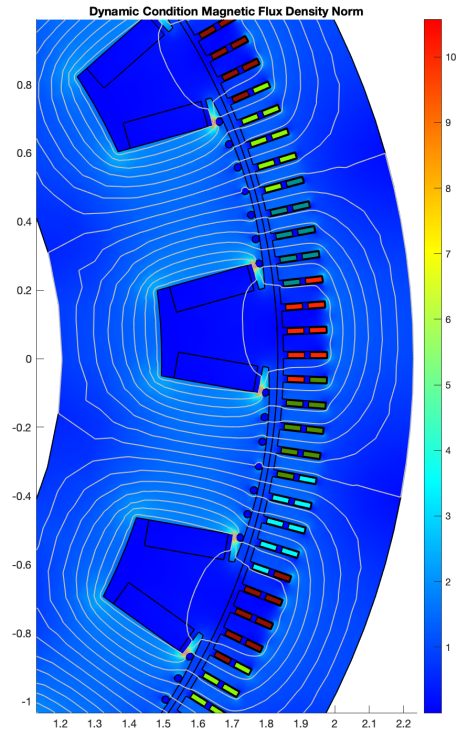


Figure 4: Finite element analysis (FEA) of flux density in generator

Comparing flux density in Figure 5 and 6 is a great representation of how useful a numeric analysis is. Changing the pole-shoe height gives almost no changes to the analytical calculations of flux density, however the numerical solutions are night and day. The generator in Figure 5b have unfeasible values and without the numeric analysis one might overlook this problem. After changing the parameter in Figure 6, we observe that the values are feasible.

Air Gap	0.9783	T
Stator Core	1.3	T
Stator Tooth	1.678	T
Pole Core	1.858	T
Rotor Ring	1.455	T

(a) Analytical Solution

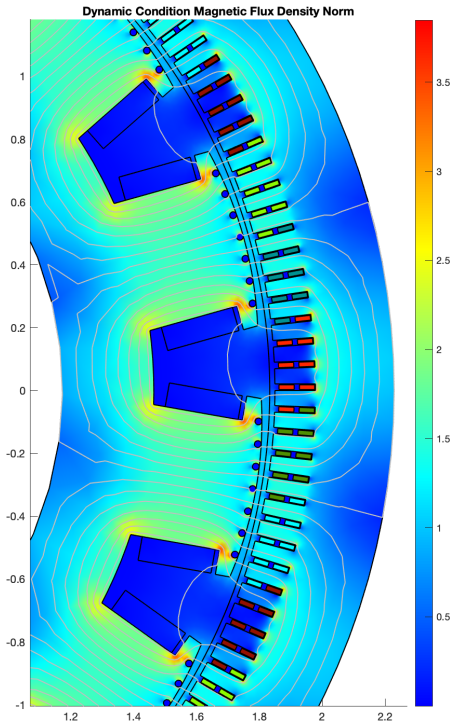


(b) Numerical Solution

Figure 5: Flux density with no defined pole-shoe height

Air Gap	0.9783	T
Stator Core	1.3	T
Stator Tooth	1.678	T
Pole Core	1.886	T
Rotor Ring	1.477	T

(a) Analytical Solution



(b) Numerical Solution

Figure 6: Flux density with user defined pole-shoe height

2.5 Short-Circuit Transients in Synchronous Generators

By far the severest transient condition that can occur in a synchronous generator is the situation where the three terminals of the generator are suddenly shorted out. Such a short on a power system is called a fault. There are several components of the current present in a shorted synchronous generator. It can be divided into roughly three periods. During the first cycle or so after the fault occurs, the ac current is very large and falls very rapidly. This period of time is called the subtransient period. After it is over, the current continues to fall at a slower rate, until at last it reaches a steady state. The period of time during which it falls at a slower rate is called the transient period, and the time after it reaches steady state is known as the steady-state period.

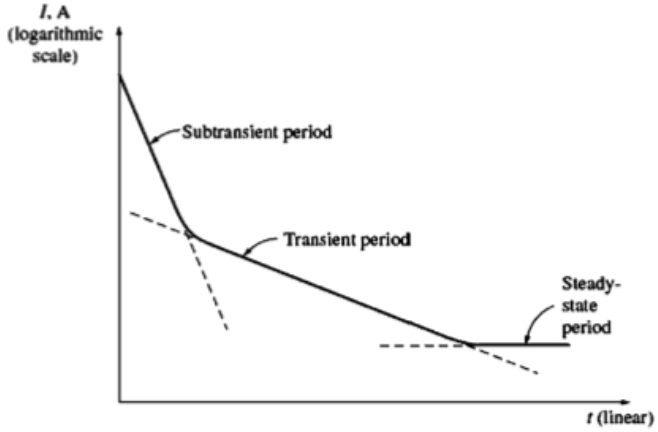


Figure 7: A semilogarithmic plot of the magnitude of the ac component of fault current as a function of time. The subtransient and transient time constants of the generator can be determined from such a plot.

The ac rms current flowing in the generator during the subtransient period is called the subtransient current and is denoted by the symbol I''. This current is caused by the damper windings on synchronous generators. The time constant of the subtransient current is given the symbol T'' and it can be determined from the slope of the subtransient current in the plot in Figure 7. This current can often be 10 times the size of the steady-state fault current.

The rms current flowing in the generator during the transient period is called the transient current and is denoted by the symbol I'. It is caused by a dc component of current induced in the field circuit at the time of the short. This field current increases the internal generated voltage and causes an increased fault current. Since the time constant of the dc field circuit is much longer than the time constant of the damper windings, the transient period lasts much longer than the subtransient period. This time constant is given the symbol T'. The average rms current during the transient period is often as much as 5 times the steady-state fault current.

After the transient period, the fault current reaches a steady-state condition. The steady-state current during a fault is denoted by the symboll I_{ss}. It is given approximately by the fundamental frequency component of the internal generated voltage E_A within the machine divided by its synchronous reactance X_s:

$$I_{ss} = \frac{E_A}{X_s} \tag{1}$$

The Synchronous Reactance X_s is the imaginary reactance employed to account for the voltage effects in the armature circuit produced by the actual armature leakage reactance and by the change in the air gap flux caused by the armature reaction.

The rms magnitude of the ac fault current in a synchronous generator varies continuously as a function of time and can be written as:

$$I(t) = (I'' - I1)e^{-t/T''} + (I' - I_{ss})e^{-t/T'} + I_{ss} \tag{2}$$

It is customary to define subtransient and transient reactances for a synchronous machine as a convenient way to describe the subtransient and transient components of fault current. The subtransient reactance of a synchronous generator is defined as the ratio of the fundamental component of the internal generated voltage to the subtransient component of current at the beginning of the fault. It is given by

$$X'' = \frac{E_A}{I''} \tag{3}$$

Similarly, the transient reactance of a synchronous generator is defined as the ratio of the fundamental component of E_A to the transient component of current I' at the beginning of the fault.

$$X' = \frac{E_A}{I'} \tag{4}$$

For the purposes of sizing protective equipment, the subtransient current is often assumed to be E_A/X'', and the transient current is assumed to be E_A/X', since these are the maximum values that the respective currents take on.

3 Constructing the generator in COMSOL

3.1 Generator Geometry

The project has several objectives, including exporting the geometry to COMSOL for field calculations. A simplified generator geometry model is constructed in COMSOL with an emphasis on automating the geometry. Automation is crucial when used in a design software like GenProg.

3.1.1 Stator

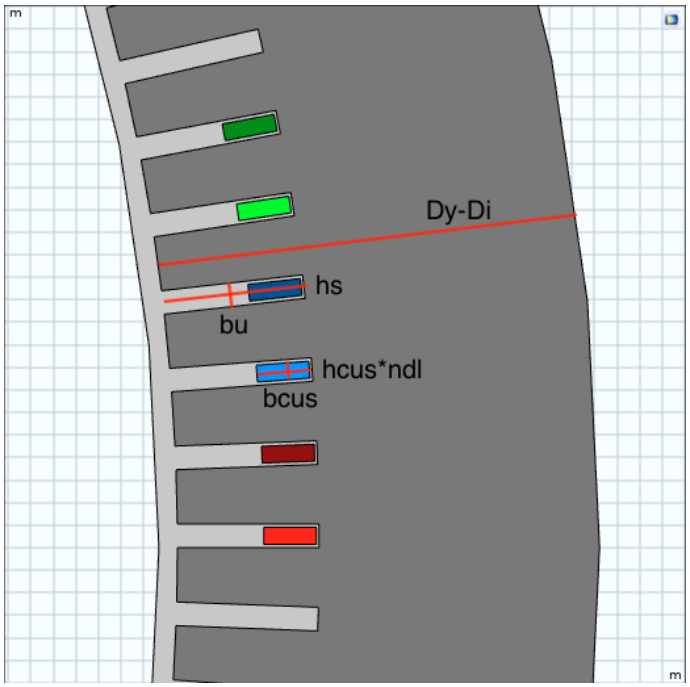


Figure 8: COMSOL Stator Geometry

The stationary part of the generator, shown in Figure 8, is constructed in COMSOL using the main parameters calculated by GenProg and are listed in Table 1.

The model includes six bars representing the armature windings. The armature windings can become complex, which is not possible to automate using only COMSOL. Each color represents a different phase and also the direction of current. This makes it possible to later generate the correct winding layout for each phase using MATLAB, and exporting it to COMSOL as an array of rotational values.

Table 1: Stator Parameters

Inner Diameter	Di
Outer Diameter	Dy
Slot Height	hs
Slot Width	bu
Number Of Strands Per Bar	ndl
Width Of Strand	$bcus$
Height Of Strand	$hcus$

3.1.2 Rotor

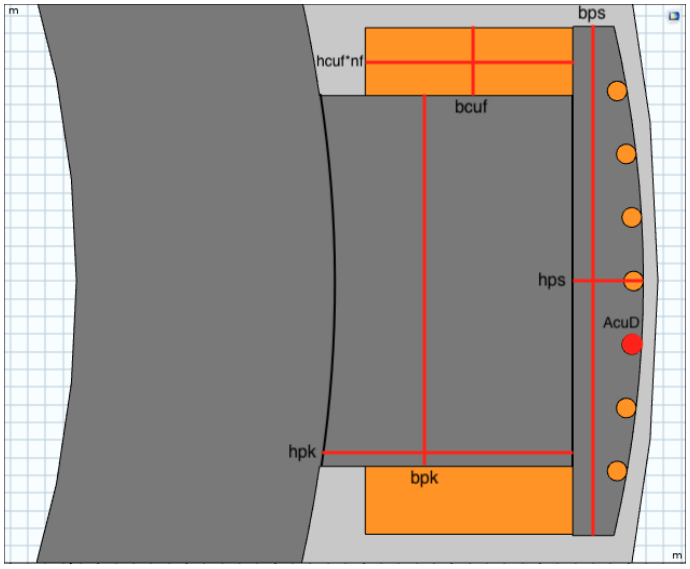


Figure 9: COMSOL Rotor Geometry

Figure 9 shows the rotating part of a generator model constructed in COMSOL. It uses parameters calculated by GenProg and can be observed in Table 2. The operations used to build this model are quite simple and can be studied further in their corresponding COMSOL files.

Table 2: Rotor Parameters

Pole Shoe Width	bps
Pole Shoe Height	hps
Pole Core Width	bpk
Pole Core Height	hpk
Field Winding Width	$bcuf$
Field Winding Height	$hcuf$
Number Of Turns Per Pole	nf
Damper Bar Cross Section	$AcuD$

3.2 Generator Materials

When calculating field strength, it is important to choose magnetic materials that are appropriate for both parts of the stator and rotor. In this model, the rotor yoke is set as iron with a constant permeability, while the stator yoke is set as a material similar to SURA M300-35A. A non-constant B-H curve could make the yoke saturated, which gives more realistic values. Windings and damper bars are set as copper.

4 Communication between COMSOL and MATLAB

4.1 LiveLink

In order to integrate the COMSOL Multiphysics file with GenProg, we must first create a communication channel between the two programs. Fortunately, COMSOL has already developed the necessary tool known as LiveLink. This software allows us to connect COMSOL Multiphysics with the MATLAB scripting environment.

For each operation you do in the COMSOL Desktop, there is a corresponding MATLAB command that you enter at the prompt. It is a simplified syntax based on Java and does not require any knowledge of Java programming. The easiest way to learn this syntax is by looking through the LiveLink users guide.

A key function of LiveLink is the possibility to change model properties with a MATLAB function. This is exactly how parameters are exported to COMSOL. The parameters are for the most part exported in the *parComp.m* script. *parComp.m* is the script where GenProg compresses parameters into their corresponding vectors before they are displayed to the app designer. Read *Modeling with a Parameterized Geometry* in the LiveLink users manual for more information.

As mentioned in subsection 3.1.1, the winding layout for the model is generated in a MATLAB script and exported to COMSOL using syntax for changing operation values. Read appendix A for more information.

GenProg run a study of the exported data and plot the results using the *mphplot* function documented in the user manual.

4.2 Navigating GenProg

After learning how to navigate the core script and the app designer, three buttons related to numeric analysis were added. The three buttons implemented at this stage were Plot Geometry, Simulate Numeric, and Plot model. As described, these plot the generator geometry, simulate the COMSOL model, and plot the corresponding results. The buttons are observed in Figure 10.

5 Computation of Inductance

A condition drop down was implemented to GenProg. The conditions are; Dynamic and Static. The static conditions gives us a numeric calculation of reactances and time constants. Currently only synchronous reactances x_d and x_q are computated and are found by exporting corresponding phase currents to COMSOL.

5.1 Direct-Axis

The polar axis in the middle of the rotor pole is called the direct axis. or d-axis.

In the computation of the direct-axis inductance, L_d, the magnetizing current is set to be zero and the stator windings are fed in such a way as to obtain a MMF distribution with the maximum value coincident with the polar axis. The phase currents with reference to the time instant t are defined as

$$i_a(t) = I_n sin(\omega t)/pnr \tag{5a}$$

$$i_b(t) = I_n sin(\omega t - \frac{2\pi}{3})/pnr \tag{5b}$$

$$i_c(t) = I_n sin(\omega t + \frac{2\pi}{3})/pnr \tag{5c}$$

where I_n is nominal current and pnr is the number of parallel circuits

Once the distribution of the winding is fixed, the three-phase currents are chosen so that the maximum of the MMF distribution coincides with the polar axis (d-axis). This corresponds to setting $I_d = I_n$ and $I_q = 0$. With the configuration of d-axis on the a-axis, the time instant $\omega t = \pi/2$ has been chosen, so that

$$i_a = I_n/pnr \tag{6a}$$

$$i_b = -\frac{I_n}{2}/pnr \tag{6b}$$

$$i_c = -\frac{I_n}{2}/pnr \tag{6c}$$

If W_{md} is the total magnetic energy stored in the synchronous generator with d-axis currents only, than the synchronous d-axis inductance L_d is

$$L_d = \frac{2}{3}\frac{2W_{md}}{I_n^2} \tag{7a}$$

and d-axis reactance X_d is

$$X_d = 2\pi f L_d \tag{7b}$$

5.2 Quadrature Axis

The interpolar axis is 90 electrical degrees in advance of d-axis with respect to the rotating direction and is called the quadrature axis, or simply q-axis.

The computation of the quadrature axis inductance, L_q, is carried out in the same way as the computation of the L_d inductance. The phase currents and the winding distribution must be chosen so as to have an MMF distribution with the maximum value coincident to the interpolar axis, i.e., the q-axis. This corresponds to setting $I_d = 0$ and $I_q = I_n$. With the configuration of d-axis on the a-axis, the time instant $\omega t = 0$ has been chosen, so that

$$i_a = 0 \tag{8a}$$

$$i_b = -\frac{\sqrt{3}I_n}{2}/pnr \tag{8b}$$

$$i_c = \frac{\sqrt{3}I_n}{2}/pnr \tag{8c}$$

The q-axis inductance and reactance is then obtained as follows.

$$L_q = \frac{2}{3}\frac{2W_{md}}{I_n^2} \tag{9a}$$

$$X_q = 2\pi f L_q \tag{9b}$$

6 How to use GenProg

6.1 Preparations

The app currently depends on both COMSOL and MATLAB, so there are some preparations needed before using the app.

Step 1 - Open the software *COMSOL with MATLAB* and follow the system command prompt instructions. You might need to connect to the COMSOL server.

Step 2 - Locate the installation file *GPapp.mlappinstall* in the source-code and follow the installation instructions. This will add the GPApp to MATLABs app tab.

Step 3 - To start the app simply launch it from the app tab within MATLAB. Make sure the current folder selected contains all the Excel files the user wishes to read from, and save to. Change the directory in *GPListRead.m*.

6.2 Results

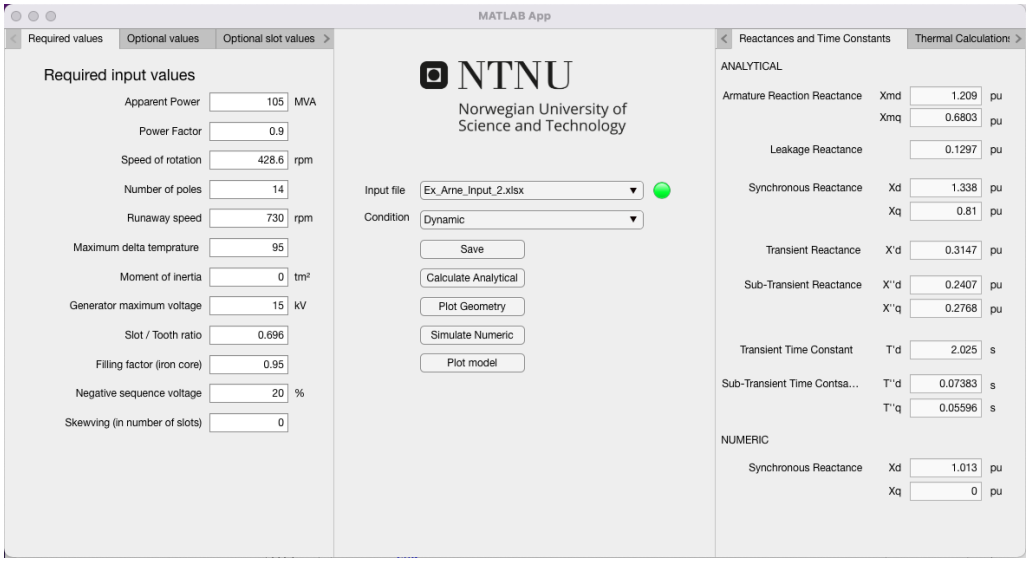


Figure 10: GenProg Graphical User Interface (GUI)

Figure 10 shows what the app looks like. Input values are shown on the left while output values are shown on the right. The input values can either be written manually or imported from an excel file ending with *Input*.

The first step is to choose a condition and calculate the analytical values. The analytical values are exported to COMSOL at this stage. This will also output a report showing errors with the design.

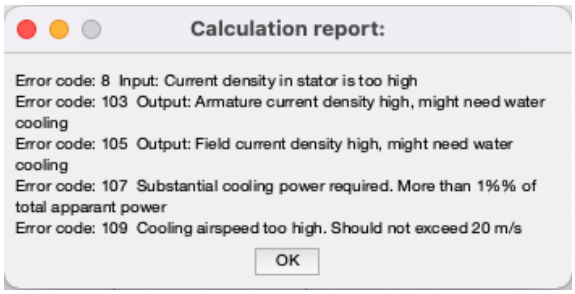


Figure 11: Errors for *Ex_Arne_Input_2.xlsx*

Pressing the *Plot Geometry* button outputs a model geometry similar to Figure 12.

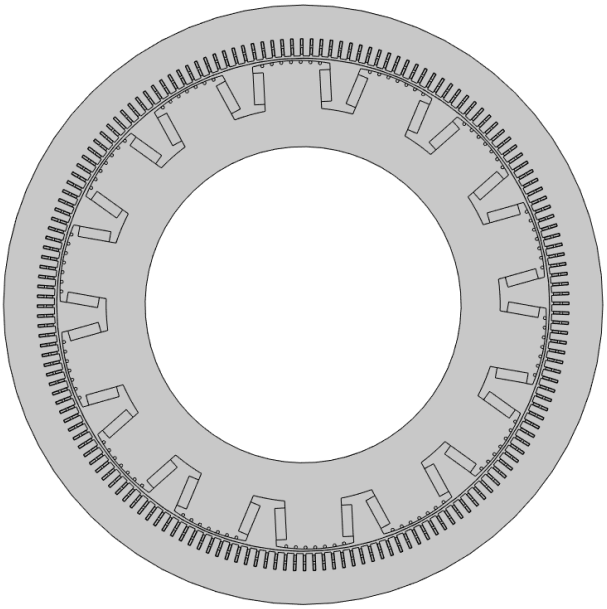
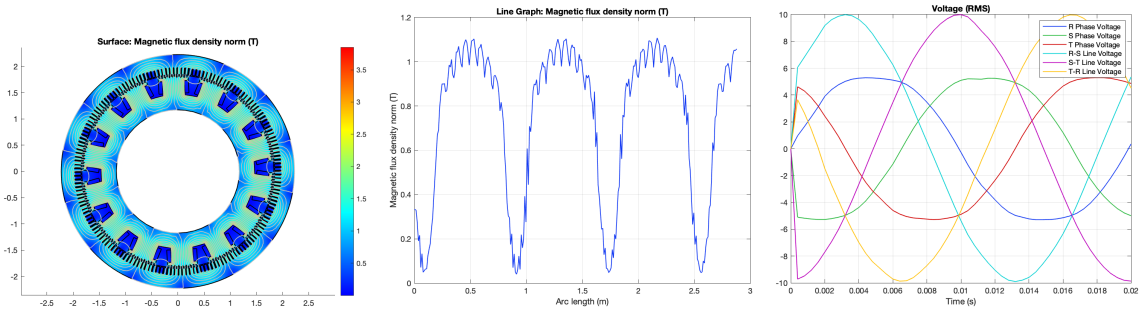


Figure 12: Model Geometry

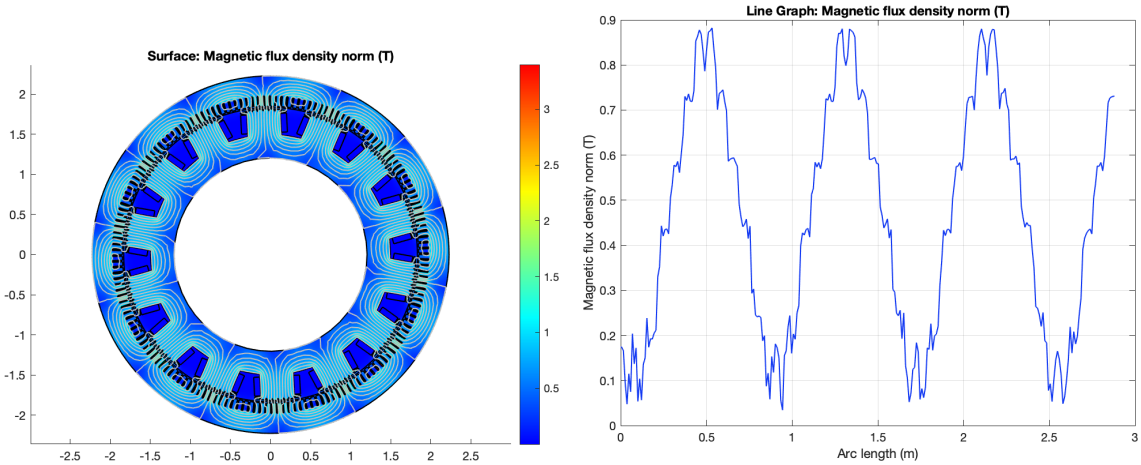
The model can then be simulated and results plotted. Currently the model simulate flux density, voltages and synchronous reactances. The resulting plots are presented in a new window, this makes it easier to interact with. As mentioned there are currently two conditions that give different results;



(a) Flux Density in generator (b) Flux Density in Air-gap (c) Voltage Generated

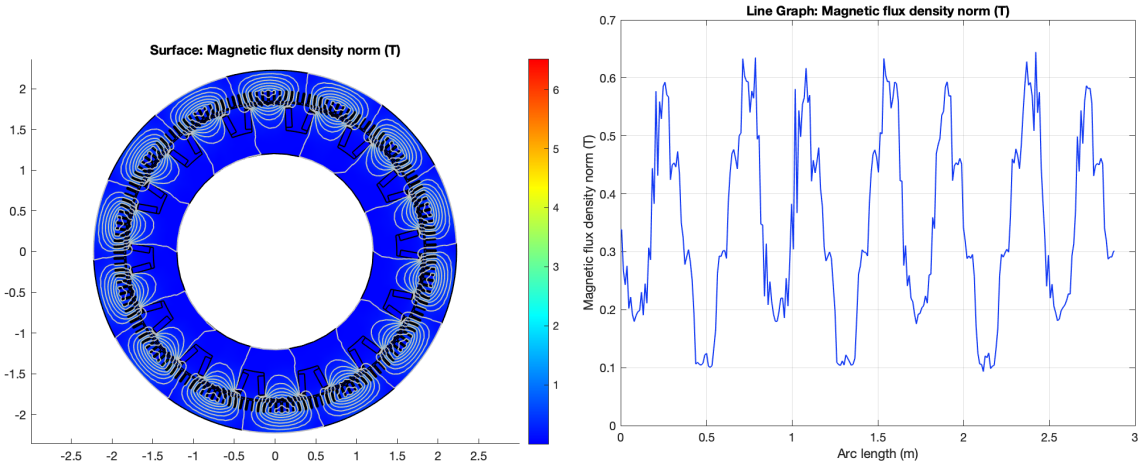
Figure 13: Simulation of Dynamic Condition

The Dynamic condition is time dependent and simulates a rotating machine that results in a study of flux density and voltages.



(a) Flux Density in generator (b) Flux Density in Air-gap

Figure 14: Simulation of Static, d-axis current only



(a) Flux Density in generator (b) Flux Density in Air-gap

Figure 15: Simulation of Static, q-axis current only

The Static condition simulates for two time instances, maximised MMF in d-axis and q-axis. This results in a study of flux density and synchronous reactances for both d-axis and q-axis current.

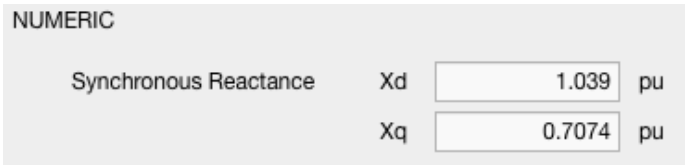


Figure 16: Numeric Synchronous Reactance

As this is a learning tool, the inputs should be played with until satisfactory results are obtained.

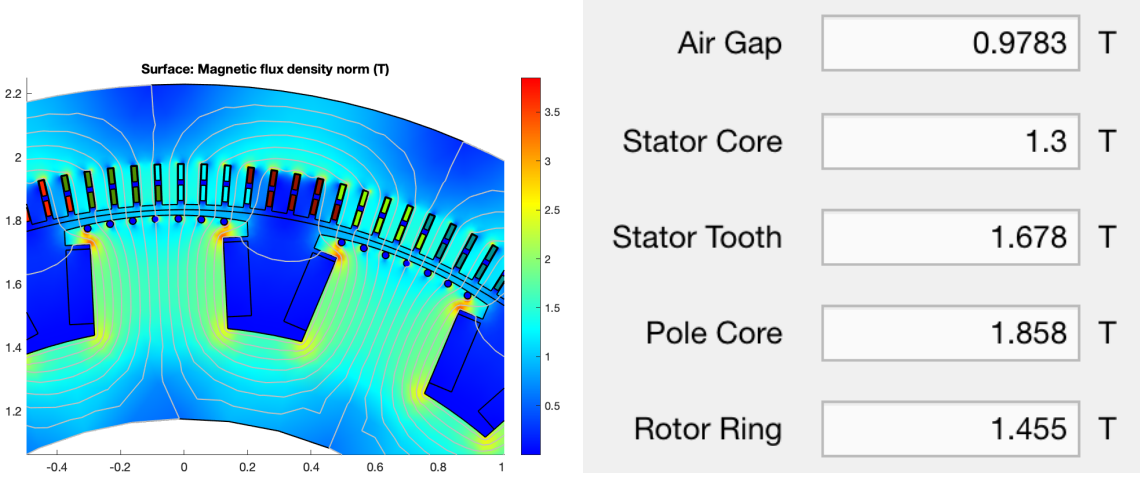
7 Discussion

When I began working on this project, I noticed that there are few sources of information available to those interested in design programs enabling the use of numerical analysis. Therefore, I began focusing on the main objective for this thesis, COMSOL. In order to use the model in COMSOL Multiphysics, the geometry of a salient pole machine needed to be constructed and automated from a few chosen parameters. The salient pole machine geometry was compared with GenProg and the pertinent parameters from Table 1 and Table 2 were chosen.These are the minimum parameters required in order to recreate a simplified 2D model of the generator. Additional work will be required to create a more detailed geometry. The model is further simplified by simulating the whole generator. While it is easier to implement it this way, ideally only the minimal sections are needed to cut down on simulation time. More automation of the COMSOL models are also possible, but time and LiveLink skills are needed. Although there may be some inaccuracies in calculating, the program nevertheless has a visual representation and understanding of the design procedure. The program is planned to be used as an educational tool, and by giving the user access to geometry plots and field calculations, it creates an environment that can be used to truly understand the design process.

The model had to be incorporated with GenProg, but in order to do so I first needed to understand how the program was put together. The script is overwhelming, however there are clues to be found when reading Jostein Hovde Aarvoll's preceding work. In his master thesis *The development of calcGenProg and GenProgApp* he divides the original script into 23 different functions, each described in a table with its name, description, and parent function. This overview along with a complete list of variable names and descriptions, makes it possible to efficiently navigate the GenProg script. This is crucial information for succeeding work on GenProg or for users wanting to dissect and understand the underlying code. During my research, I noticed flaws in the GenProg core script. For instance, some optional inputs were required, despite being marked as optional. Developing a fix for this is beyond the scope of my thesis work, but I recommend fixing the core code as part of a separate project or thesis.

To implement COMSOL with GenProg, we use LiveLink to communicate between the two software platforms. LiveLink establish a connection to the COMSOL Server, but this currently affects the app performance. MATLAB apps typically allow users to share them with other people as either desktop apps or web apps; however, as LiveLink uses COMSOL operations that are not recognized by the app compiler, users must open the app designer before running GenProg.

One of the tasks in this thesis was to expand accuracy for various calculated design parameters. This thesis focused on voltage, flux density and reactances.



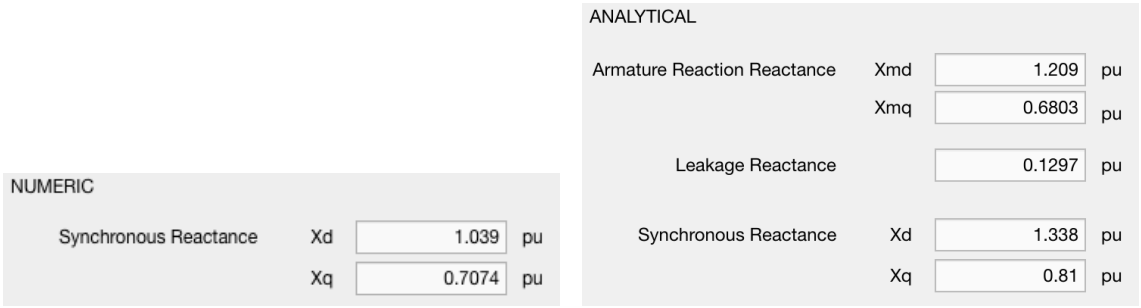
(a) Flux Density Numerical

(b) Flux Density analytical

Figure 17: Flux Density Comparison

Flux values when compared are quite similar. As mentioned, analytical values are average while numerical values gives a detailed overview. Deviations might occur due to different materials between analytical and numerical. More specific the B-H curve and if the materials are saturated or not. In this example it seems like the materials are unsaturated.

Due to lack of information on how to numerically calculate reactances, most of the time spent on this went to literature study. Thus only synchronous reactances where found numerically. However, the groundwork is laid and with more time the transients and time constants will follow.



(a) Reactance Numerical

(b) Reactance Numerical

Figure 18: Reactance Comparison

These values are quit different. This difference is presumably due to how the script analytically calculate armature reaction reactance. After calculating the armature reaction reactance, the program decides the corresponding d- and q-axis reactance simply by multiplying it with constants called d-axis factor and q-axis factor. These constants give no explanation and the simplified values are most likely inaccurate. Unlike the analytical solution, the numerical solution separately calculate the d-axis and q-axis reactances by maximizing MMF in the corresponding axis. Therefore I believe it is valid to conclude that the numerical simulation give more accurate values.

There are still design parameters to be implemented such as loss, thermal, mechanical and harmonics. With the COMSOL models as groundwork, only time is needed for these to be implemented. I believe the app should first focus on simplicity and functionality before diving into more detailed geometry and calculations. Hopefully my work and information will make it easier for future students to evolve the software. The program works great as a learning tool; with a user friendly interface and interactive results, it creates an intuitive understanding of the design process. It would be great if future students could make the program even more user friendly by implementing descriptions for every input and output value. For instance, hovering over "pole shoe height" would show a picture of where this parameter is used.

8 Conclusion

The program works great as a learning tool; with a user friendly interface and interactive results, it creates an intuitive understanding of the design process. Design of hydro power generators is a difficult optimization problem. For the most part, the dimensions have to be guessed, more or less justified. The construction must then be analyzed, and the dimensions adjusted until the analysis gives the desired result. GenProg creates a fun learning environment with visual interactions required for this trial and error approach.

COMSOL solutions are more detailed and accurate than the analytical part of GenProg. Flux density values are quite similar, but the visual map of flux density from COMSOL offers far more information. Simplifications in the GenProg script creates a big differences in reactance values, thus indicating more accurate value from COMSOL. There are still several parameters to calculate with COMSOL and I would have preferred to implement more myself, but with the COMSOL models as groundwork only time is needed before more are implemented.

References

[1] Jostein Hovde Aarvoll. The development of calcGenProg and GenProgApp (NTNU). 2021.

[2] Proff E. Westgaard and O.W. Andersen. Dimensjoneringsesempel for synkronmaskin. 1965.

[3] Nicola Bianchi. Electrical Machine Analysis Using Finite Elements. 2005.

[4] Stephen J. Chapman. Electric Machinery Fundamentals. 2005.

Appendix

A LiveLink Code For Winding Layout

```
% LiveLink Revolver

function LiveLink()


clc,clear

global model
%model = mphload('Salient Pole Synchronous Machine Final');

p = str2double(model.param.get('p'));  % number poles
Qs = str2double(model.param.get('Qs')); % number slots

pp = p/2;                       % pole pairs

%sec = gcd(Qs,pp);
sec = 1;                        % number of sectors
nrPpPS = pp/sec;                % number of Pole-pairs per sector
drad = (sec/Qs)*360*nrPpPS;         % degree step length

eldeg = 0:drad: nrPpPS * 360 - drad;   % degree vector with all slot in a sector and
↪  their corresponding electrical degree

degL = [0:360/Qs:360/sec-360/Qs];     % slots degree


% Empty vector for RST phase
windLayRplusUp = zeros(1 , Qs/sec);    % first layer
windLayRminusUp = zeros(1 , Qs/sec);
windLayRplusDown = zeros(1 , Qs/sec);  % second layer with phase shift
windLayRminusDown = zeros(1 , Qs/sec);

windLaySplusUp = zeros(1 , Qs/sec);
windLaySminusUp = zeros(1 , Qs/sec);
windLaySplusDown = zeros(1 , Qs/sec);
windLaySminusDown = zeros(1 , Qs/sec);

windLayTplusUp = zeros(1 , Qs/sec);
windLayTminusUp = zeros(1 , Qs/sec);
windLayTplusDown = zeros(1 , Qs/sec);
windLayTminusDown = zeros(1 , Qs/sec);


% function for returning R S T phase of the winding layout for a given electrical
↪  angle
for j = 1:length(eldeg)

   deg = rem(eldeg(j),360);

   % R
   if deg >= 0 && deg < 60
      windLayRplusUp(j) = 1;
      windLayRminusDown(j) = 1;
```

```
    elseif deg >= 180 && deg < 240
        windLayRminusUp(j) = 1;
        windLayRplusDown(j) = 1;

    % T
    elseif deg >= 240 && deg < 300
        windLayTminusUp(j) = 1;
        windLayTplusDown(j) = 1;
    elseif deg >= 60 && deg < 120
        windLayTminusUp(j) = 1;
        windLayTplusDown(j) = 1;

    % S
    elseif deg >= 120 && deg < 180
        windLaySplusUp(j) = 1;
        windLaySminusDown(j) = 1;
    else
        windLaySminusUp(j) = 1;
        windLaySplusDown(j) = 1;


    end
end

% Setting Rplus
RplusUpzero = windLayRplusUp.*degL;                % Giving winding coorect slot
↪  degrees
RplusUp = [0;nonzeros(RplusUpzero)];               % Removing zeros
model.geom('geom1').feature('rot3').set('rot',RplusUp); % Gives the corresponding
↪  slotbar domain in comsol the correct rotation

RplusDownzero = windLayRplusDown.*degL;
RplusDown = [0;nonzeros(RplusDownzero)];
model.geom('geom1').feature('rot13').set('rot',RplusDown);

% Setting Rminus
RminusUpzero = windLayRminusUp.*degL;
RminusUp = [0;nonzeros(RminusUpzero)];
model.geom('geom1').feature('rot5').set('rot',RminusUp);

RminusDownzero = windLayRminusDown.*degL;
RminusDown = [0;nonzeros(RminusDownzero)];
model.geom('geom1').feature('rot12').set('rot',RminusDown);

% Setting Tplus
TplusUpzero = windLayTplusUp.*degL;
TplusUp = [nonzeros(TplusUpzero)];
model.geom('geom1').feature('rot6').set('rot',TplusUp);

TplusDownzero = windLayTplusDown.*degL;
TplusDown = [0;nonzeros(TplusDownzero)];
model.geom('geom1').feature('rot15').set('rot',TplusDown);

% Setting Tminus
TminusUpzero = windLayTminusUp.*degL;
TminusUp = [nonzeros(TminusUpzero)];
model.geom('geom1').feature('rot7').set('rot',TminusUp);
```

```
TminusDownzero = windLayTminusDown.*degL;
TminusDown = [0;nonzeros(TminusDownzero)];
model.geom('geom1').feature('rot14').set('rot',TminusDown);

% Setting Splus
SplusUpzero = windLaySplusUp.*degL;
SplusUp = [nonzeros(SplusUpzero)];
model.geom('geom1').feature('rot8').set('rot',SplusUp);

SplusDownzero = windLaySplusDown.*degL;
SplusDown = [0;nonzeros(SplusDownzero)];
model.geom('geom1').feature('rot17').set('rot',SplusDown);

% Setting Sminus
SminusUpzero = windLaySminusUp.*degL;
SminusUp = [nonzeros(SminusUpzero)];
model.geom('geom1').feature('rot9').set('rot',SminusUp);

SminusDownzero = windLaySminusDown.*degL;
SminusDown = [0;nonzeros(SminusDownzero)];
model.geom('geom1').feature('rot16').set('rot',SminusDown);

end
```