NTNU
Innovation and Creativity

2D Finite Element (FE) modeling and simulation of the stator winding of synchronous machines in adjustable degree of detailing

Håkon Hole

Master of Science in Energy and Environment
Submission date: May 2007
Supervisor: Robert Nilssen, ELKRAFT

Norwegian University of Science and Technology
Department of Electrical Power Engineering

Problem Description

Electromagnetic simulation of synchronous machines with fractional slot windings requires consideration of quite big periodic machine-segments. In order to study loss distribution and fields in the two-layer stator windings, detailed calculations are required for the strands of the Roebel bars or diamond coils. Preferably, only selected slots should be modelled with a high degree of detailing, whereas the remaining slots should be simplified. This will be sufficient for the reproduction of the global machine behaviour.

During the master thesis, an ANSYS-based 2D simulation environment for synchronous machines, which is available at Voith Siemens, shall be extended to the above mentioned adjustable degree of detailing. The full process shall be automated by programming and implementing the necessary software packages.

After this preparative work, the parametric simulation tool shall be applied and tested in order to perform comparative calculations of eddy current losses in the strands of the stator winding, and to optimize the slot - and the bar/coil - design for selected machines. To do this optimization efficiently, it is necessary to separate the various field components as well as the corresponding eddy current components in the slot from a theoretical point of view, and apply this knowledge in the FE post processing. More specifically, components dependant of load must be separated from those independent of load, radial components must be separated from tangential, and homogenous from inhomogenous.

Assignment given: 12. November 2006
Supervisor: Robert Nilssen, ELKRAFT

Master Thesis

2D Finite Element (FE) modeling and simulation of the stator winding of synchronous machines in adjustable degree of detailing

by Håkon Hole

Acknowledgements

I would especially like to thank Dr. Stefan Lahres for the guidance and help during this project. His explanation of important principles, and supervision underways, has been very helpful for my understanding and made me able to carry out this work.
I would also like to thank Mr. Diego Aranda for great help with software issues and tips about APDL, ANSYS in general, and many other things concerning the GP2DET-tool.
Great thanks goes to everyone in the R&D department where I have carried out my work. I truly appreciate the way I was welcomed and taken care of.
Finally I would like to thank Voith Siemens Hydro for letting me carry out the work at their facilities in Heidenheim, and how they greatly contributed to the feasibility of the stay.
This work has been carried out with support from the Leonardo programme.

Contents

List of Figures

1 Introduction

Calculation of losses in electrical machines is in general important to increase the efficiency. An accurate determination of where the losses are created, what they are caused by and how big they are, is very useful in the overall evaluation of the machine design. The ideal design is a perfect balance between the production costs and the machine abilities. Simulation of various designs is very important for producers of large electrical machines in the pursue of such a balance, as it helps putting a price on the losses as well as the cost of reducing them.
The strength of the magnetic fields in the stator slots vary with the position. In the height direction, the field strength is generally highest closest to the slot opening, since this is closest to the poles. However, the strength can also vary in the width direction, dependent on amongst others the pole position relative to the slot and the load situation for the machine. An example of magnetic flux lines in the slot of a loaded machine can be viewed in figure 1.

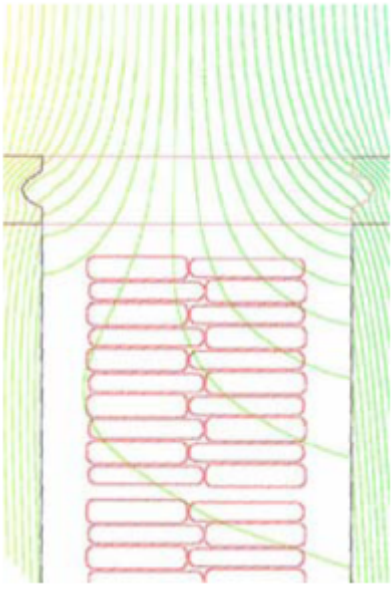


Figure 1: Example of magnetic field lines in the stator slot of a loaded machine. (Seyler [7])

Each strand in the stator winding is electrically insulated from the others throughout the slot, but in many machines they are all soldered together and shorted at the ends. Since

the magnetic field strength varies in the slot, different voltages are induced for different strands. Further, a voltage difference between the strands leads to currents flowing between them. These circulating currents only produce losses, since they can not be used to perform any work. It can intuitively be assumed that changing the position of the stator conductor in the slot will result in a change of circulating currents. However, a change of position might affect the use of iron in the stator core, and hence influence the iron core losses. An interesting question is therefore how much each of these two types of losses are affected when the position of the stator conductor is changed. Would for example a 1 centimeter displacement in height lower the copper losses in the conductor more than it increases the iron losses in the stator core? Correct simulation of the problem would probably give the answer, and could be used to find an optimal placement.

Voith Siemens Hydro developes at the moment a simulation tool for electrical calculations of large hydro power generators. This tool is named General Purpose 2D Electromagnetic Tool, GP2DET, and is based on the Finite Element method. In GP2DET each stator conductor is modelled as one solid conductor, and the stator current is applied as current density. This excludes amongst others the possibility of calculating losses from circulating currents. In order to calculate these, a fully detailed modelling of all the strands in the conductor is needed, and the currents must be induced rather than impressed as current density.

2 Stator conductor modelling basis

2.1 Losses in stator conductors

2.1.1 Skin effect

Skin effect is by Bergmann and Schaefer [2] described as a frequency dependent and inductance caused resistance that makes alternating currents in a conductor displace to the outer borders of the conductor. For applicable frequencies in electrical machines, the skin effect resistance can be expressed as:

$$R_{HF} \approx \frac{l}{2r}\sqrt{\frac{\mu_0 \rho f}{\pi}} \tag{1}$$

In equation (1), l is the length of the conductor, r is the radius, μ_0 is the permeability in air, ρ is the specific resistance of the conductor material, and f is the frequency. The so called skin depth, the depth below the conductor surface where the current density decays to $\frac{1}{\mathrm{e}}$ of the current density at the surface, can be expressed as:

$$d = \sqrt{\frac{\rho}{\pi \mu_0 f}} \tag{2}$$

The symbols in equation (2) have the same meaning as in equation (1). Calculating the skin depth for copper with a conductivity of $\rho = 0.017\mu\Omega m$ gives $d \approx 8.5$mm by 60 Hz, and $d \approx 9.5$mm by 50 Hz. This is in an order of magnitude that makes the effect noticable in a stator conductor in a large electrical machine. Such a stator conductor may typically have a width of 15 mm.

2.1.2 Circulation currents

The strength of the magnetic field at a given position in the stator slot depends on factors like load situation for the machine and position of the poles. Loading of the machine will affect the pole angle, which displaces the magnetic field in the stator compared to the rotor. This means the magnetic flux lines can be ”dragged” more out of the stator teeth and into the slot at some positions, as was shown in figure 1 in the introduction. This type of field displacement primarily applies to the regions closest to the slot opening. Further, a slot located between two poles at one given time instant, will experience a different magnetic field than a slot located outside the pole middle. Generally, more fluxlines are likely to

cross a slot while it is located between two poles.
Since the magnetic field strength is not uniform throughout the slot, there will also be differences in induced voltage over each strand in a stranded conductor, in accordance with Faradays law. Faradays law of induction is by Young and Freedman [8] expressed as:

$$\varepsilon = -\frac{d\Phi_B}{dt} \tag{3}$$

In stranded conductors, the difference in induced voltage between the strands lead to different currents. If the strands are all soldered together at the ends of the slots, this difference in current will start flowing between the strands, only producing losses since they do not contribute to any work. These currents can become quite high due to low resistivity in the stator conductors.

2.2 Stator winding configurations

The design of the stator windings can be done in numerous ways. Sequenz [6] treats a wide range of principles and designs, including much of the development from as early as 1891 and until 1950, and much of this theory is still relevant today. The treated material in the following is based on his work, but also influenced by the programs Di1277 and Di1359. The manuals for Di1277 [4] and Di1359 [3], as well as winding diagrams produced by Di1277 provide most of the limits for what information that is relevant for Gp2det.
The winding design is dependent on factors like the size of the machine, which purpose the machine is made for, and the costs of constructing and maintaining it. Important terms in regard of windings is for instance whether it is a bar- or coil-winding, whether or not transposition is applied, and for bar windings, what kind of arrangement it is configured with. This can be for instance lap- or wave winding. The windings may be arranged in one or two layers, and the machine may or may not have fractional slot windings. This and more will be enlarged upon in the follwing sections.

2.2.1 Definitions in relation to stator windings

As described by Sequenz [6], the conductors in a stator winding must be coupled in closed circuits, arranged as groups of coils or coil-like arrangements, constituting one or several branches. This can be done by use of bars or coils - each of these will be treated in the following - and the collective term ”stator winding” will be used for both of them. Further, the number of stator slots per pole per phase in a machine may be an integer number or a

fractional number. The terms that will be used in this regard are ”integer slot winding” or ”fractional slot winding”, respectively.
The two sides of a generator will be referred to as the ”non drive end”, NDE, and the ”drive end”, DE, after which side the turbine is mounted on. Further, the term ”circuit ring side” will be used to describe the side where the circuit ring of the machine is located. Most often, this is located on the NDE-side, but since that is not always the case, a specific term is defined.
A stator winding may consist of massive or stranded conductors. Several arrangements of strands in a stranded conductor is possible, as shown by the picture from National Electric Coil [5] in figure 2. The term ”stator conductor” will be used as a collective term for all strands or conductors within each layer of one slot. It should be noted that the term ”conductor” may be used in other regards as well.

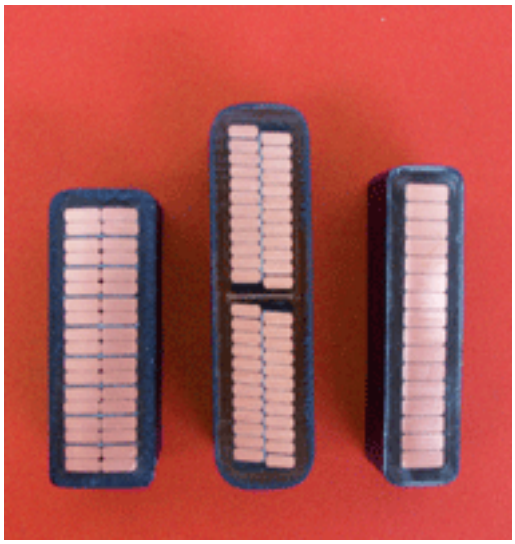


Figure 2: Various strand arrangments. Photo courtesy of National Electric Coil [5]

A bar winding may be arranged in different ways, but due to limitations in DI1277, only ”lap-winding” and ”wave-winding” will be mentioned. Winding diagrams with examples of these are shown in figure 3. All the vertical lines can be interpreted as bars, and all other lines can be viewed as connections between these bars. In case of a lap-winding, the bars are arranged in groups of several laps, somewhat resembling turns in a coil. In case of a wave-winding all the connections between the bars are done in the same direction, and the conductors do not cross each other to form groups, like in the lap-winding.

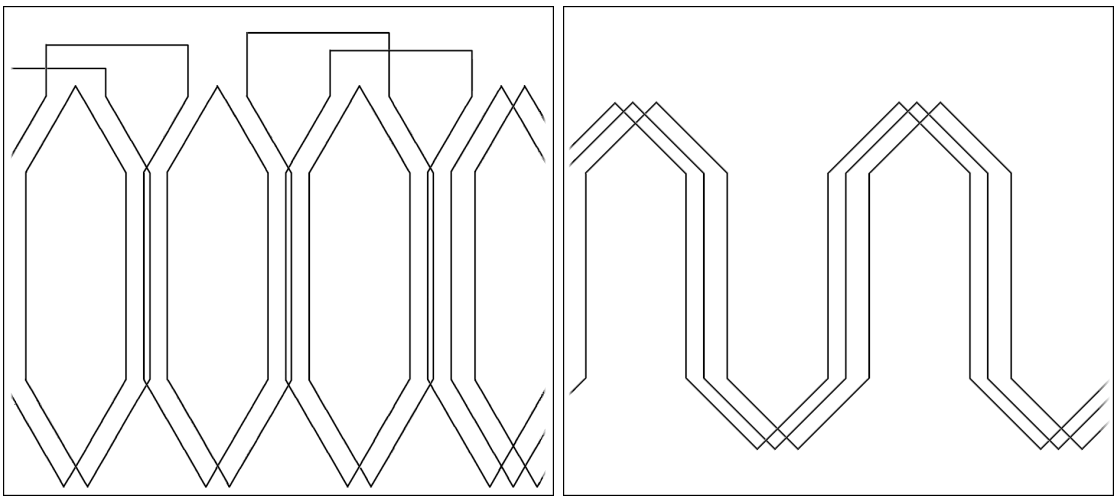


Figure 3: Examples of bar winding diagrams. Left: Lap-winging. Right: Wave-winding

2.2.2 Coil winding

Definition of terms

For the use in regard of electrical machines, a coil will here be defined as a conductor - or a group of such - wound in a continuous series of loops. One loop around the coil, from any given reference point and 360 degrees around the coil, will be defined as one turn. Each coil comprises two coil sides, where the coil side is defined as the part that lies within a slot. This is illustrated in figure 4.

Definition of coil winding

A coil winding will here be defined as a winding where several turns are used through one specific pair of slots before the conductors are connected either to the next slot-pair or to the terminal.
Various configurations of coil designs exist. The DI1359-manual [3] uses an explicitly defined arrangement of conductors within a coil. Two-layer windings are used, as shown in figure 4, where conductor 1, 3 and 5 are located in upper layer, and conductor 2, 4 and 6 are in lower layer. When viewed from the connection side, and with the stator slot opening upwards, the coil start is defined to be on the left side of the coil in the upper layer, and the coil end is defined to be on the right side in the lower layer. The connection point of the coil start is defined to be located in the turn farthest away from the slot opening, which is represented by the number 1 in the figure. When following this turn to the lower layer

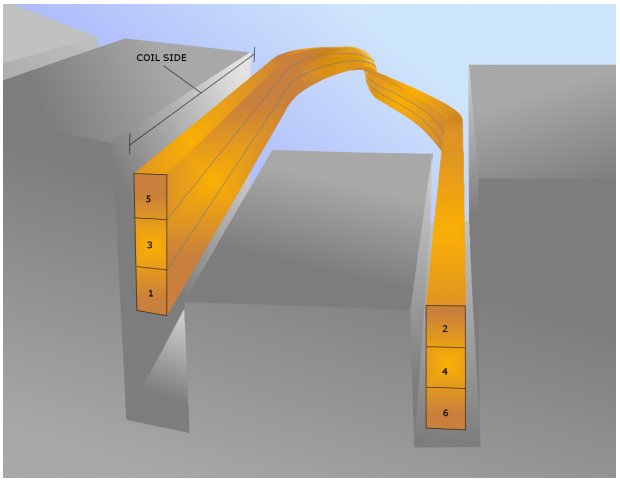


Figure 4: Figure of a 2-layer stator coil from the connection side. The numbers on each coil side show the path of the turns, starting at 1 and ending at 6.

in the connected slot, it continues in the top turn; in conductor 2. This pattern goes on until the coil end is reached, represented by the number 6. Thus is the order of the turns opposite in the upper layer of what it is in the lower layer. Such an arrangement assures that the coilsides close to the slot opening in one of the slots will be correspondingly far away in the connected slot. This is to lower the effect of magnetic field strength variation in the height direction throughout the slot. When the turns of a coil are stranded, the same altering of order applies to the strands. This will be enlarged upon below, in the section on strands and transposition.

The coils of a coil winding are arranged in groups. That means several neighbouring coils are connected together before a connection is made to the next group. Connections within a group will be referred to as ”normal connections”, and for connections between groups the term ”circuit connection” will be used. The winding diagram in figure 5 illustrates these. It can also be mentioned that the circuits in the figure looks like they are closed with themselves. This is because there are several turns within one coil, and does not mean that the conductors are short-circuited.

The number of groups per phase corresponds to the number of poles in the machine.

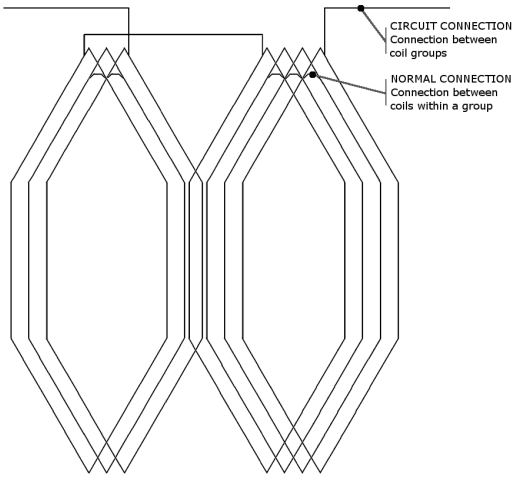


Figure 5: Coil winding diagram, illustrating circuit- and normal-connections

2.2.3 Bar winding

Definition of terms

One bar corresponds to what is defined as a stator conductor. That means, it differs from the coil winding in that a coil has several turns per stator conductor, whereas the bar does not.

Definition of bar winding

A bar winding will here be defined as a winding where a conductor - or a group of such - are going through one specific layer of one specific slot only once. That means, at both ends of each slot, the conductors are either connected to slots and layers where they have not already been, or to the terminal. The conductors in the upper- and lower-layer of a bar winding do per definition not have to be of the same size in height.
In case of a lap winding, bars are arranged in groups, somewhat similar to the coil windings. In this case the terms ”normal connection” and ”circuit connections” will also be used for bar windings.

2.2.4 Stranded conductors and transposition

Stranded stator conductors

Stranded stator conductors are often used in large electrical machines to reduce eddy current losses and circulating current losses, as mentioned in chapter 2.1.1. As shown in figure 2, the number of strands, the arrangement of them and the shape may vary from machine to machine. The strands are normally sized to minimize the skin effect in each of them, but the size is also a question of how practical they will be to build and how much of the cross sectional area that goes away to insulation.

Transposition of bar windings

All the strands of a bar are most often shorted at each end of the slots. If the entire bar was subjected the same magnetic field, the currents would very much be equally big in each strand. However, due to the variation of magnetic field strength throughout the slot, the induction in the strands closest to the slot opening will be bigger than the induction in the strands farther away, in accordance with chapter 2.1.2. In order to even out this effect, Roebel-transposition can be used. Roebel-transposition of a bar means that the position of each strand is altered throughout the stator conductor such that the strand is equally much located in all positions in the bar. Typically, in a bar with two columns of strands, a strand can start at the lower right position at one end of the slot, go to the top right position at the middle of the slot, then swap over to the left side and go down to the bottom left position at the other end of the slot. An illustration of this is shown by the CAD-model from Voith Siemens Hydro [9] in figure 6. By arranging all the strands like this, the variations in field strength in the height direction of the bar will be distributed equally over all the strands.
When considering the cross sectional area of the slot, Roebel-transposition increases the copper-less percentage. This is due to the bending of the strands at the top and bottom of the bar. Such bending leads to some space between the strands, and thus less copper per used volume.

Transposition of coil windings

For a coil winding with stranded conductors, the strands are usually not shorted within the coil, and therefore not shorted at the end of each coil side like the bar windings are. However, they may very well be shorted at the ends - the connection points - of each coil,

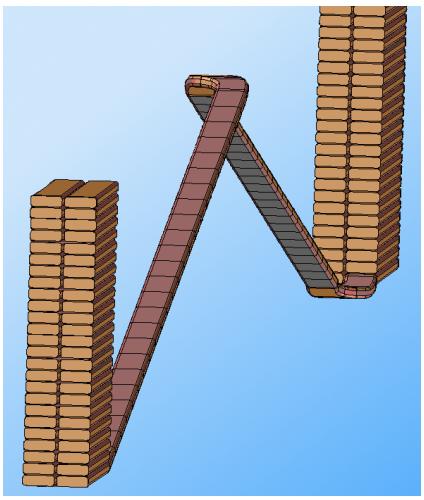


Figure 6: Path of a strand in a Roebel bar. (VSH R&D [9])

and are thus subjected to the same problem of circulating currents as the bar windings. But since the strands are insulated from each other throughout the entire coil, the order of the strands can be altered between the slots, such that the variation in magnetic field strength over the strands to some extent is evened out. Usually, Roebel-transposition is not applied to coil windings, due to a more complicated production process.
Coil windings can be transposed at several positions. Usually this is transposition within a coil, transposition in the connection between coils within a group, and transposition in the connection between coil groups. When transposition is applied within a coil, it is usually done outside the slots, between the coilsides. The position of every strand is then altered, and they are still electrically insulated from each other. When transposition is applied between coils within a group, or between groups, all strands can be connected uniquely, like within a single coil, but it is also possible to solder groups of a few strands together before connecting them. This will be referred to as ”blockwise connection”, and the number of strands within each block may vary from machine to machine.
The transposition itself - what kind of position altering that is applied - may vary from one machine to another. Seyler [7] has used one specific type of transposition in his work, where the strands are altered to the converse order in both height- and width-direction of the coil side. The Di1359-manual [3] however, specifies that conversed order is used only in the height direction for the Di1359-program. The untransposed order of the strands within a coil is shown in figure 7, and the transposition type used by Seyler is shown in figure 8.

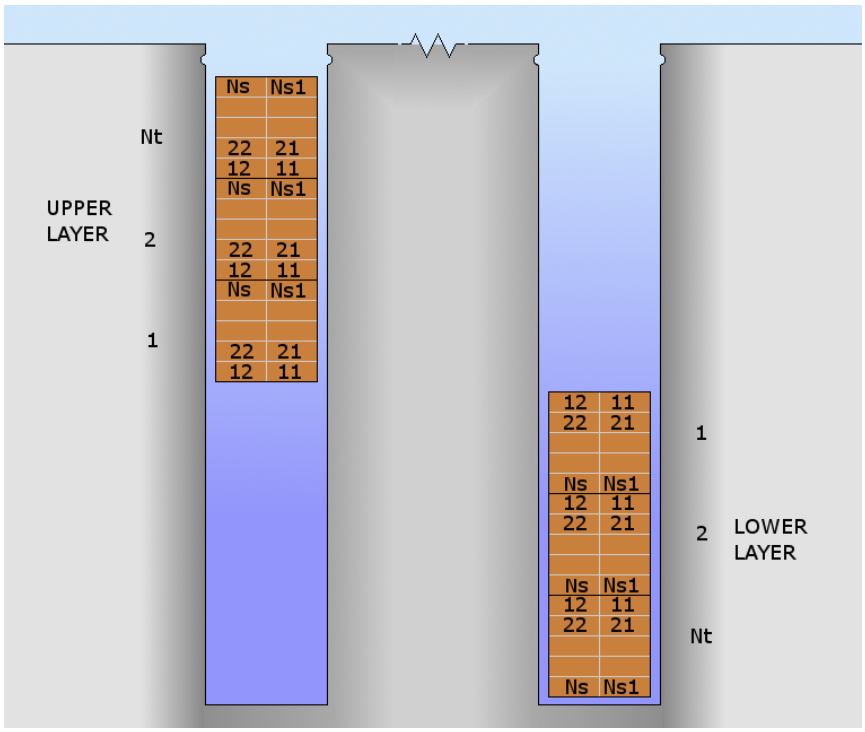


Figure 7: Untransposed strand arrangement, as in DI1359. Legend: Nt = Number of turns in coil. Ns = Number of strands in height. Ns1 = Strands in column 1. Numbering: "12" = Row 1, column 2.

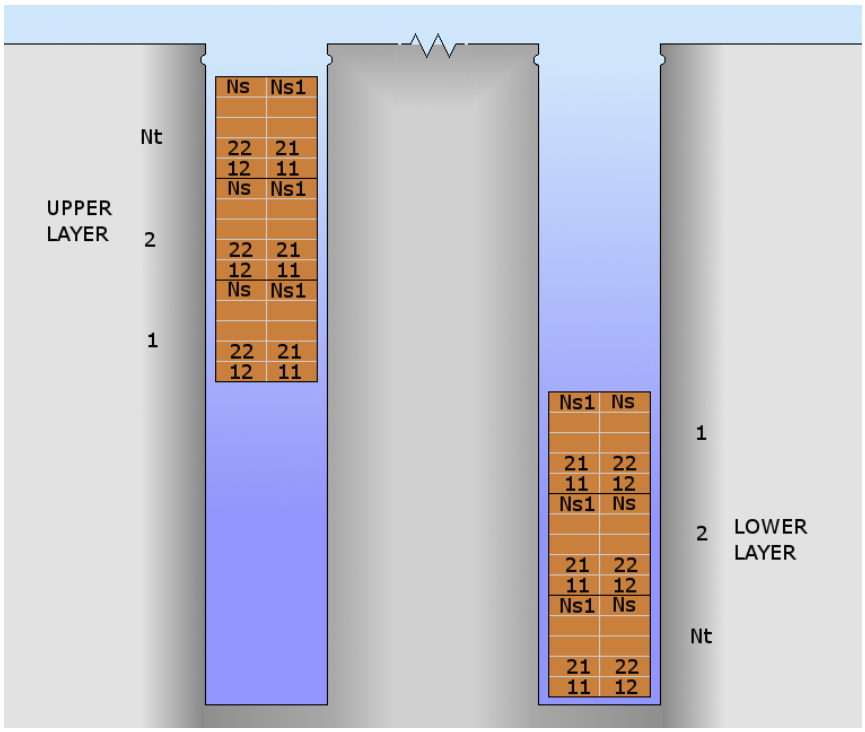


Figure 8: Transposed strand arrangement after Seyler. Legend: Nt = Number of turns in coil. Ns = Number of strands in height. Ns1 = Strands in column 1. Numbering: ”12” = Row 1, column 2.

2.3 Modelling in ANSYS

ANSYS uses amongst others element types and real number constants to specify and define the conditions for models that are to be simulated. An introduction to a few relevant element types and real number constants will be given.

2.3.1 Element types

ANSYS operates with a great variety of elements, each with specific properties, and thus certain abilities and limitations. Choice of which kind of element to use therefore depends on what it is desirable to simulate.
Generally, many settings for the elements can be adjusted by use of key-options. That means, when an element type is assigned with the `ET`-command, and thereby also the key-options are set, certain element-dependent attributes will be assigned to this type.

`ET,"number","elementname","keyoption1","keyoption2",...`

The issued `ET`-command above shows a typical assignation of element type. A unique number is assigned to this element type, the kind of element to use is chosen by name, and the key-options are assigned. The key options are what really defines the type, since several types of elements may be assigned for the same kind of element. The input value for each key-option is an integer number, and each number defines specific settings. In the following sections, a presentation will be given of relevant settings for relevant elements.

PLANE53

As given in the ANSYS-documentation [1], the PLANE53 is a 2D element for simulation of planar and axial symmetric magnetic fields. The element has 8 nodes and 4 degrees of freedom, DOF, per node. The degrees of freedom are given in the list below.

- AZ: Z-component of the magnetic vector potential
- VOLT: Time-integrated electric scalar potential
- CURR: Electric current
- EMF: Electromotive force

The element is meant for magnetostatic analyses, analyses of eddy currents, voltage forced electric fields and electromagnetic-circuit coupled fields. For eddy currents, AC time harmonic and transient analyses are available. This applies to voltage forced fields and electromagnetic-circuit coupled fields too, and in addition static analyses can be carried out for these.
Plane53 has magnetic material properties, and hereunder capabilites of handling non-linear B-H curves or permanent magnet demagnetization curves. In case of using such a curve, it must be assigned to the element specifically.
For this element, key-option 1 can be used to set the most relevant attributes for simulation of stator conductor losses. The different choices for key-option 1 are given in the list below. Each item of the list starts with the key-option input value, followed by an explanation.

- 0: AZ degree of freedom: Static domain, induced eddy current domain
- 1: VOLT and AZ degrees of freedom: Current-fed massive conductor
- 2: AZ and CURR degrees of freedom: Voltage-fed stranded coil
- 3: AZ, CURR and EMF degrees of freedom: Circuit-coupled stranded coil
- 4: AZ, CURR and EMF degrees of freedom: Circuit-coupled massive conductor

Circu124

In the Ansys-documentation [1], the Circu124-element is defined as a general circuit element, applicable to circuit simulation. It supports inteface with electromagnetic finite elements, and can be used for simulation of coupled electromagnetic-circuit field interaction. That is to say, it can be used together with Plane53 for such purposes. The element has up to 6 nodes, and up to three degrees of freedom, and it can be used for static, harmonic and transient analyses. The degrees of freedom are listed below.

- VOLT (voltage)
- CURR (current)
- EMF (potential drop)

It can be noted that the element has three of the same degrees of freedom as the Plane53-element, for certain settings. When coupling these elements, they can be arranged to share

the degrees of freedom that they have in common.
The nodes within the element are sorted into the categories ”active” and ”passive”. Active nodes are those in direct connection or inteference with a modelled electrical ciruit, and the passive ones are those used internally and not connected to the circuit. Passive nodes can be for instance nodes that are used to couple the circuit element with electromagnetic finite elements.
Key-option 1 is used to define element circuit components, sources, and coupled sources. For some of these settings, it is also required to adjust the real number constants, which is treated in the next section. A few of the available settings for key-option 1 are listed below. Each list item starts with the input value of the key-option, followed by a short description.

- 0: Resistor
- 1: Inductor
- 2: Capacitor
- 6: 2D Massive conductor voltage source

2.3.2 Real number constants

Real number constants are used in relation with element types to define properties for a simulation model. The constants are arranged in sets for each element type, defined by a unique set-number. Some key-option values for some elements require the real constants to be defined, and some not. This depends on whether or not additional information is needed, on top of what is defined by each key-option setting for each element. The relevant real constants values for the PLANE53- and the CIRUC124-elemenst will be treated below.

Real constants for PLANE53

For PLANE53, a set of real constant values must be specified if key-option 1 is equal 2,3 or 4. Table 1 shows the constants of that set, and which of them that must be assigned for the different key-option choices. Required real constants if key-option 2 for PLANE53 is assigned, will not be treated. Table 1 shows how different types of analysis requires different types of information to be assigned. If a PLANE53-element is defined to be a circuit-coupled massive conductor, that is key-option 1 = 4, then the CARE-, LENG- and DIRZ-real constants must be specified.

Nr	Name	Description
1	CARE	Cross sectional area. Required if KEYOPT(1)=2,3,4
2	TURN	Total number of turns. Required if KEYOPT(1)=2,3
3	LENG	Conductor length in Z-direction. Required if KEYOPT(1)=2,3,4
4	DIRZ	Z-direction of current. 1 or -1. Required if KEYOPT(1)=2,3,4
5	FILL	Fill factor for the coil. Required if KEYOPT(1)=2,3

Table 1: Real constant set for PLANE53 if key-option 1 is set to 2, 3 or 4

Real constants for CIRCU124

The real constant set for the CIRCU124-element is managed slightly different from what is the case with PLANE53. Each real constant for the PLANE53-element is defined as the same parameter, only with a change in magnitude, for all key-options. For CIRCU124 the real constants change both magnitude and meaning with the change of key-option. For instance real constant number 1 defines resistance if the CIRUC124-element is a resistor. But when the CIRCU124-element is an inductor, real constant number 1 defines the inductance. An overview of the required real constants for relevant key-options is given in table 2.

Circuit option	KEYOPT(1)	Real constants
Resistor (R)	0	R1 = Resistance
Inductor (L)	1	R1 = Inductance R2 = Initial inductor current
Capacitor (C)	2	R1 = Capacitance R2 = Initial capacitor voltage
2D Massive Conductor Voltage Source	6	R1 = Symmetry factor

Table 2: Real constants for CIRCU124 if key-option 1 is set to 0, 1, 2 or 6

3 Stator winding information sources

The GP2DET collects much of its information from a group of softwares called the POWERM-package. Information is imported from these programs: DI1102, DI1204 and DI1277. A short introduction to GP2DET, how it works, and how the bundle of softwares work together can be found in appendix A.

3.1 Data in DI1277

The DI1277-program from the POWERM-package is a software with extensive information about stator windings. The program can amongst others produce winding diagrams and perform different calculations, such as voltages between the stator conductors, symmetry of the phases, the armature field strength, etc. Information contained in the program follows specific standards, and important issues in this regard will be discussed below.

3.1.1 Definitions and standards

Supported machines and program features

According to the DI1277-manual [4], the program supports machines with three or six phases. Two-layer windings, as well as both lap- and wave-windings are supported. The program can produce winding diagrams for the stator, and also for the rotor as long as it is a three phase winding.

Terminology and definitions

In order to distinguish between the two sides of a generator - the drive end (DE) and the non drive end (NDE) - several expressions are available. In DI1277 the signs + and - are used. However, these are not explicitly defined; the user is free to choose which side he wants to name + and -. It is recommended though, when watching a winding diagram with the slot numbers in ascending order from the left to the right, to name the top side of the machine as +side, and the bottom as -side.
Standard step lengths between the normal-connected conductors of the stator windings are used in the DI1277. These are called the winding pitches Y1 and Y2. Y1 is defined as the step on the +side from an upper layer conductor to the connected lower layer conductor. Y2 is defined as the step on the -side from a lower layer conductor to the connected upper layer

conductor. For a coil winding, both the pitches have the same step length, but opposite signs. For a bar- and lap-winding the pitches have unequal step lengths and opposite signs.

3.1.2 Winding information

Arrays in Di1277

Three arrays in Di1277 contain much of the stator winding coupling information. `PLUS` is an array that contains all the conductor couplings on the +side. `MINUS` correspondingly contains all the couplings on the -side. `SYMF` contains the branch, phase and direction of each conductor. Special information on the terminal connections is not explicitly given in these arrays.
In the three arrays, each element consist of one integer. Still, much information is stored in each number, which can be retrieved by correct interpretation. The number of elements in the arrays is equal the number of stator conductors in the machine. That means, each array element holds information on the stator conductor in one specific layer of one specific slot. An overview of the information is showed in table 3.

Array name	Retrieved information for each stator conductor
`PLUS`	Slot number of connected stator conductor Layer number of connected stator conductor
`MINUS`	Slot number of connected stator conductor Layer number of connected stator conductor
`SYMF`	Branch of currently chosen stator conductor Phase of currently chosen stator conductor Direction of currently chosen stator conductor

Table 3: Arrays in Di1277, holding winding information

Difference between pitch and connection information

For normal connections in a bar winding, calculating the number of slots between two connected conductors based on `PLUS` or `MINUS` will give a number of slots that conforms with the winding pitch. For coil windings however, this is correct on the non circuit ring side, but not on the circuit ring side. On the circuit ring side, the Di1277-arrays give the connection to the next coil, which is different from the winding pitch.

4 Automated detailing of stator windings

4.1 Conditions for the modelling

The currently released version of GP2DET models all stator conductors as massive, and impresses a current density when analysing load situations. The used current density is uniform throughout the stator conductors, which is a simplified model of a stranded conductor; one assume the current density to not vary. However, the circulation currents discussed in chapter 2.1.2 can not be studied by such a model. Also, the effect that transposition has on the circulating currents, as well as the eddy current losses in each strand, can not be studied.
This chapter will outline which measures that were made, and what strategy that was chosen, to provide extra simulation options.

4.1.1 Detailing alternatives

In order to expand the possibilities of analysis, it was decided to model stator conductors in their full detail, rather than as a massive block. This means the conductors would have to be modelled with all their strands in correct geometry, which will be referred to as detailed stator conductor modelling hereafter.
To achieve detailed stator conductor modelling in a machine, it was decided to rework the geometry of a model after it is produced by GP2DET. This allows detailed modelling of only a certain number of stator conductors, whereas the rest will stay modelled as massive conductors. Dependent on which effects it is desirable to study, and how the machine is built, different alternatives for detailed modelling are desireable. For instance, analyses of transposition between two coil groups in a coil winding requires detailed modelling of all the stator conductors in both the affected coil groups, as well as coupling between the strands. But for example analysis of eddy current losses in an upper layer stator bar would only require detailed modelling of one stator conductor. Due to this variation in need of detailing, it was decided to support a certain series of interesting detailing alternatives. The list of alternatives can be viewed in appendix B.

4.1.2 Elements and real constants

Choice of elements

The purpose of the detailed stator conductor modelling is to provide the possibility of analysing circulating currents and eddy current losses. The FE-elements of the strands in the stator conductors must therfore be massive conductor elements, rather than stranded. Further, for modelling of coil windings it is necessary to couple elements in circuits, for instance when building a coil group, and therefore the FE-elements must also support circuit coupling. The PLANE53-element with key-option 1 = 4 supports all of this, and is therefore chosen.
For the electrical coupling of strands, both circuit elements for the representation of a strand, as well as a circuit elements for the electrical coupling between the strands, is needed. The circuit elements for the strands must be able to represent a massive conductor, and support coupling with the PLANE53-element. Therefore the Circu124-element is chosen with key-option 1 = 6. The circuit element for electrical coupling between each strand can be a Circu124-element with key-option 1 equal either 0, 1 or 2, hereby representing a resistor, inductor or capacitor, respectively. For now, a representation as resistor is used.

Setting of real constants

Using key-option 1 = 4 for the PLANE53-element requires definition a few real constants. For each strand using this element, the real constants for cross sectional area, conductor length in Z-direction, and direction of the current, either -1 or 1, must be defined. The DI1204-parameters specify that cross sectional area may vary between the upper and lower layer of the stator. Also, the direction of the current may vary from one stator conductor to another. The conductor length will be equal for all strands. In order to handle these variations, four sets of real constants were defined. Each set is assigned an unique number, and this number is stored in the parameters listed below.

- `rsn_e_st_strand_pos_u`
- `rsn_e_st_strand_pos_l`
- `rsn_e_st_strand_neg_u`
- `rsn_e_st_strand_neg_l`

In the syntax for the real set numbers, "pos" and "neg" shows the direction of the current, and "u" and "l" shows the layer.
The CIRCU124-element is used both as conductor-element and resistor-element. This requires setting of real constants for symmetry factor and resistance, respectively.
In case of the conductor-modelling, the entire circuit will be modelled, and no symmetry scaling is necessary. The real constant for symmetry, R1, will therefore be set to 1, and the number of the set to which this constant belongs will be stored in the parameter given below.

`rsn_e_st_strand_circ`

In case of the resistor-modelling, the resistance for stator conductors will be applied to the real constant R1, and the number of the set to which this constant belongs will be stored in the parameter given below.

`rsn_e_st_circ_con`

4.1.3 Structure of the model revision

The revision of existing machine models - the detailing of conductors - had to be automated in ANSYS in accordance with how the GP2DET-tool is built. That means, everything had to be fully automated, without the need of involving the user during execution. The ANSYS Parametric Design Language (APDL) was used to write scripts for this purpose.
This section will treat relevant limitations encountered, possibilities and strategies used, in regard of the APDL-scripting.

Strategical influence from ANSYS

In some cases ANSYS limits what is allowed to do and not in APDL. These limits make it necessary to treat certain problems in certain ways, and therefore enforce - to a limited extent - a certain program structure.
The GP2DET-model that is revised is already meshed, and in ANSYS it is not allowed to modify solid model entities that are meshed; that is, areas, lines or keypoints. Therefore, the mesh must be cleared from any entities before they can be modified. However, this does not only affect modification of each entity itself, but also entities that are used, but not changed, in creation of other entities. For example, if there is an empty hole in the middle of a group of areas, the lines that constitutes the borders of the hole can not be used to generate an area if they are meshed. This affects for example the creation of the slot air

areas, since they are deleted and re-created when the stator conductors are modified. The mesh must then be cleared from the slot borders before the slot air can be re-created, and thus sequences are added to the program run.

Module-based macro structure

The vast number of ways to configure a stator winding calls for some precautions to be taken in the stator conductor detailing procedure. Theoretically it would be possible to write one single macro for the entire procedure. However, this would dramatically limit the flexibility, and the possibility of making changes or add features at a later point would practically be prohibited. Therefore it was chosen to divide the procedure in several parts, preferrably by uncoupling sequences from each other, and write them in dedicated macros. Inspired by object-oriented programming, it was chosen to pursue a module-like buildup of the procedure, such that single modules can be edited, added or removed at a later point, without affecting other modules. An example of this implementation is the macro that inserts a detailed stator conductor. This macro has amongst others the responsibility of placing the stator conductor at the right position, to differ between coil- or bar-windings and so on. But for the actual building of the stator conductor, it calls a sub-macro, dependant on whether or not the stator conductor is Roebel-transposed. Therefore the Roebel-buildup procedure can be modified if desired, independant of other macros. Or additional ways of building a stator conductor can be added, regardless of the existing procedures. The complete list of macros can be found in appendix C. Most of these are mentioned in chapter 4.2 and 4.3, where an explanation is given of what they do and how they work together.

4.1.4 Handling and organization of data

Each model from the GP2DET contains a large amount of information and data, for example geometrical magnitudes, material properties, rated machine values and so on. While some information is practical to store in parameters, other may be more practical to store in arrays. In order to keep the information organized to a certain level, the GP2DET uses a standardized list of abbreviations. These have as far as possible been used for the detailed modelling of the stator conductors. The list can be viewed in appendix D.

Information organized in parameters

Many machine measurements and magnitudes are exported from Excel to ANSYS by use of the `vsvaluescad`-macro. For instance information from DI1204 on strand sizes, slot sizes and positions are extensively used in the detailed stator conductor modelling. However, a few additional parameters were needed to carry out some of the detailing alternatives. These were included in Excel, and exported with the other parameters in `vsvaluescad`. Table 4 shows an overview of the parameters and gives a short explanation.

Parameter name	Description
`n_wdg_detailing_alternative`	Number of the detailing alternative to carry out
`n_slot_replacement_start`	Start-slot for stator the conductor replacement
`n_pos_st_coil_transp`	Position of transposition within each coil
`n_pos_st_coilgr_transp`	Position of transposition between coils within a coil group
`n_pos_st_branch_transp`	Position of transposition between coil groups within each branch
`n_strands_per_block_w`	Number of strands per block in width; coil- and coil group-connection
`n_strands_per_block_h`	Number of strands per block in height; coil- and coil group-connection
`n_phase_to_replace`	Number of phase to replace, if replacing entire phase
`n_parallell_circuit`	Number of parallell circuit to replace, if replacing entire branch

Table 4: Additional parameters included in Excel

Information organized in arrays

Winding information from DI1277 is used frequently. Storage of this information in arrays is a practical way of organizing it and providing easy access. The names of the used arrays and a short explanation is given in table 5. All the contained information is stored in the format of integer numbers.
In order to keep record of which stator conductors that are replaced, an array with boolean

Array name	Content of array
ar_phase_slot_u	Phase of the UL-conductor
ar_direction_slot_u	Direction of the UL-conductor
ar_n_par_branch_u	Parallell branch this UL-conductor belongs to
ar_connectedslot_DE_u	Number of the connected slot on DE-side
ar_connectedlayer_DE_u	Number of the connected layer on DE-side
ar_connectedslot_NDE_u	Number of the connected slot on NDE-side
ar_connectedlayer_NDE_u	Number of the connected layer on NDE-side
ar_phase_slot_l	Phase of the LL-conductor
ar_direction_slot_l	Direction of the LL-conductor
ar_n_par_branch_u	Parallell branch this LL-conductor belongs to
ar_connectedslot_DE_l	Number of the connected slot on DE-side
ar_connectedlayer_DE_l	Number of the connected layer on DE-side
ar_connectedslot_NDE_l	Number of the connected slot on NDE-side
ar_connectedlayer_NDE_l	Number of the connected layer on NDE-side

Table 5: Winding information arrays

values was created. The name is given below.

ar_b_conductorreplaced

The array has two columns - one for the upper layer and one for the lower layer - and as many rows as there are slots in the FE-model. When created, all the values in the array are set to zero. Whenever a conductor is being replaced, the value at the corresponding position in the array is set to one.

For the electrical circuit coupling of coils and coil groups, a lot of information had to be treated, such as connections within coils, start- and end-position of coil groups, some specific node numbers for each strand, and so on. A few arrays were used to organize this information. The arrays are listed in table 6, and briefly explained below.

The two first arrays in table 6 were created to organize coil group information. Each group has six columns and as many rows as the number of replaced coil groups. The content and use of these are further explained in chapter 4.3.1 about gathering of coil group information. It can be noticed that these arrays are only created and used if one or more coil groups are replaced. Also, if all three phases are replaced, these two arrays are created for each phase, and the phase number is attached at the end of the name.

For the electrical coupling with circuit elements, a large number of nodes need to be

Array name	Content of array
ar_n_coilgroup_info	Various coil group information, numbers
ar_c_coilgroup_info	Various coil group information, characters
ar_nd_strand_coupling_[a]_[b]	Node numbers of CP- and CE-nodes for each strand
ar_nd_coil_coupling_[a]_[b]	Node numbers of CP- and CE-nodes at the coil ends
ar_nd_coilgr_coupling_[a]_[b]	Node numbers of CP- and CE-nodes at the coil group ends

Table 6: Arrays used for the coupling of coils and coil groups

accessed. The three last arrays in table 6 are used to store the numbers of those nodes. These arrays are created for every stator conductor where they are needed, and therefore the slot- and layer-numberes are stored in the array name. This storage is shown by the syntax at the end - [a]_[b] - where "[a]" is the slot number and "[b]" is the layer number. The node numbers in these arrays are the numbers of the reference nodes for sets of coupled degrees of freedom (CP), and node numbers for circuit elements (CE). This is done for couplings between strands within coils, coupling between coils, and coupling between coil groups, respectively. The size of the arrays may vary with the number of strands in each stator conductor, as well as type of coupling that is applied. It can be noticed that the strand-array is only created if one or more coils are replaced, the coil-array is only created if more than one coil is replaced, and the coilgr-array is only created if more than one coil groups is replaced. The content and the use of these arrays are further explained in chapter 4.3.3

4.2 Automation of the solid modelling

4.2.1 Conductor replacement sequences

The modification of the stator conductor geometry in a GP2DET-model is mainly done by 8 macros. Due to the module-based buildup, not all of them are used for every detailing alternative or every machine, and the order of how they are being used may vary. This chapter will outline how they work together and interact with each other. The flowchart in figure 9 visualises those sequences that will be explained in the following sections. A more

detailed explanation of special issues and important details for some of the macros will be given in the next chapter.
The names of the 8 macros are given in the list below. The macros operate solely on the parameters exported from Excel.

- `vsremoveslotair.mac`
- `vsinsertconductor.mac`
- `vsinsertconductor_agen1.mac`
- `vsinsertconductor_agen2.mac`
- `vsinsertconductor_loop1.mac`
- `vsinsertconductor_loop2.mac`
- `vscreatestrand.mac`
- `vscreateslotair.mac`

Removal of slot air

The macro `vsremoveslotair` takes a slot number as input and calculates all needed operation coordinates by use of parameters from Di1204. The global cylindrical coordinate system is used, since many machine coordinates are given by angle and raius. The macro selects and creates components for these entities: The slot air border lines, the area and the border lines for both the stator conductors in the slot. The entities are selected by position, using the `ASEL`-command. Finally, the mesh is removed from the slot air, and this area is deleted.

Deletion of the stator conductor

The macro `vsinsertconductor` manages the entire conductor replacement process. It takes a slot- and a layer number as input, and this macro too calculates all needed operation coordinates from the Di1204-parameters. It selects the chosen stator conductor by the component created in `vsremoveslotair.mac`, clears the mesh from all selected conductor areas, and deltes them. Also, every replaced conductor is registered in the array `ar_b_conductorreplaced`. If this array does not exist, it is created.

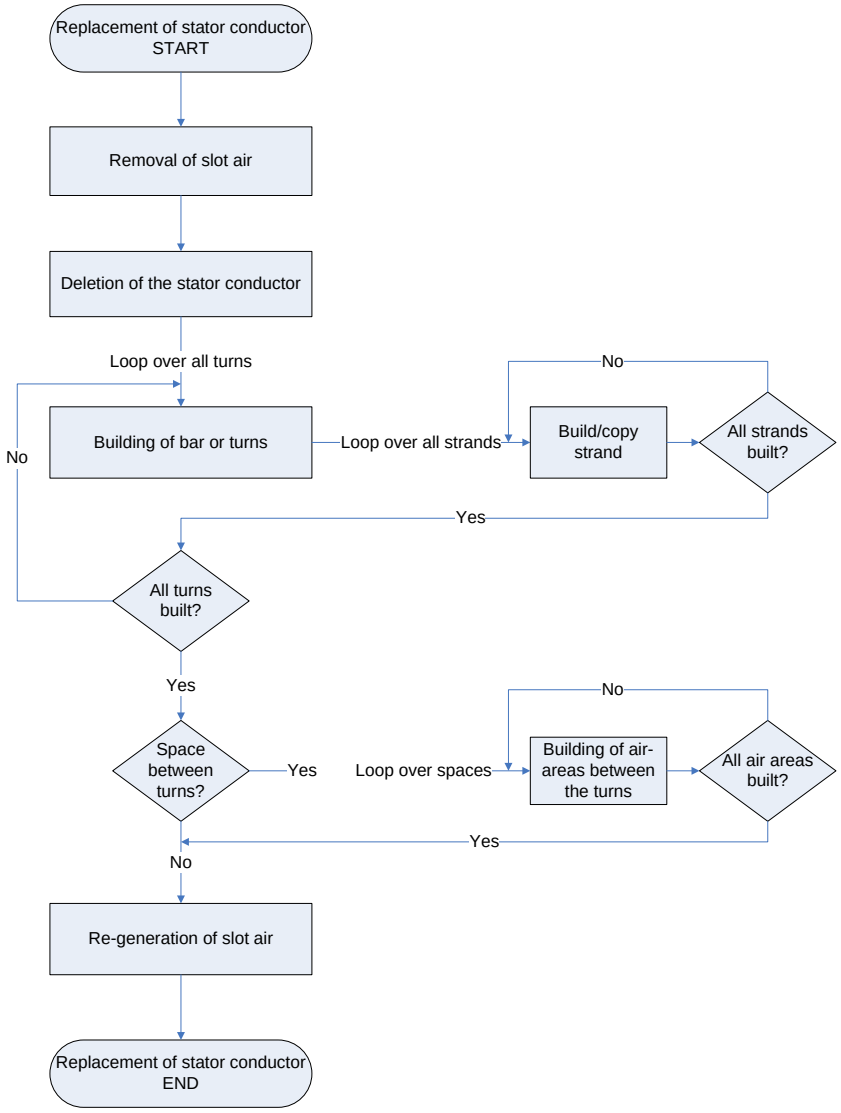


Figure 9: Flowchart showing the process of replacing a massive stator conductor with a detailed one

Building of bar or turns

This sequence is still managed within the `vsinsertconductor`-macro. A new local coordinate system is created, with the origin at the center position of the deleted conductor. The positive y-direction of this system is the same as the positive radial direction of the global cylindrical coordinate system; from the machine center and outwards. This system will be referred to as the stator conductor center coordinate system.
A loop is then entered, looping over all turns, building the entire stator conductor. In this loop a stator bar is regarded as a coil side with only one turn, and therefore the operation is compatibel with both coil- and bar-windings. Even though the macro loops over all turns, it does not produce the actual turns itself. This is because a turn will look different if it is Roebel-transposed compared to if it is not. Dedicated macros are written for each of the turn configurations, and these are named `vsinsertconductor` with the suffixes `_agen1, _agen2, _loop1` and `_loop2`. Two versions of turns are currently supported, as shown in figure 10. The turns can be built with two columns of strands beside each other, which is quite usual for coil windings. Or the turn can be built with one top- and one bottom strand, which is used for Roebel-transposed bars. Only stator conductors where all the strands have the same size is supported at the moment.

Build/copy strand

Each of the macros that build the turns uses a double loop over all strand rows- and columns to build the entire conductor. Loops are used to make the naming procedure easy. The difference between the `_agen`- and the `_loop`-macros, is that the first ones use area copying and merging of nodes during the building, whereas the latter ones build all the strands from scratch and glues the areas together. However, the `_loop`-macros are considered to be obsolete code, and only the `_agen`-macros are used, since they consume a substantially lower amount of time. The difference between the ”1”- and the ”2”-versions of the macros, is that ”1” builds turns with all the strands beside each other, and the ”2” builds a turn with Roebel-transposition. The ”1”-macros support stator conductors with any number of strands in height and widht, whereas the ”2”-macros only support stator conductors with three or more strands in height, and two or more strands in width.
All the macros make use of a start position coordinate for the first strand. This position is calculated in the parent macro - `vsinsertconductor.mac` - because each turn starts at a different position. The sequential order of the strand creation is basically the same for both the turn types; the first strand is created at the bottom left of the turn, referred to the

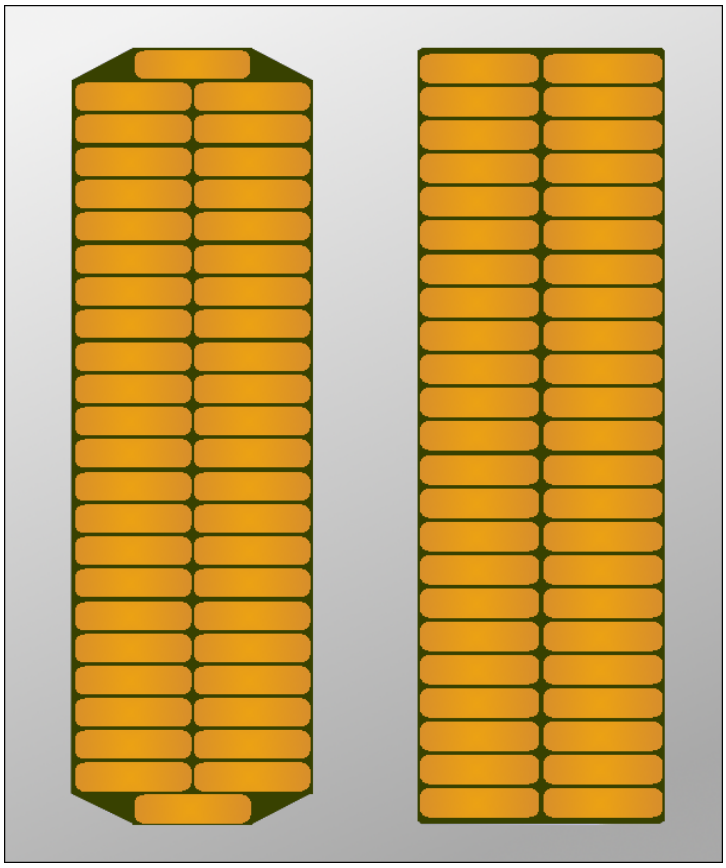


Figure 10: Available strand arrangements

stator conductor center coordinate system. Thereafter all the strands in the same column are created in positive y-direction, before the next column is created from the bottom to the top. However, for a Roebel-bar, the top- and bottom strands are created lastly. Also, the insulation areas surrounding the copper in these strands are modified, because of a meshing problem that occured when displacing the strands in horizontal direction. This will be enlarged upon below, in the section about special issues.
All the macros that build the turns use a sub-macro for the actual creation of a strand; the `vscreatestrand`-macro. A further presentation of this macro will be given in the section below about special issues.

Building of air-areas between the turns

Once the loop for building of turns in the `vscreatestrand`-macro has finished, it is tested whether or not there is a gap between the turns. If so, a loop over all gaps is run, and lines between the turns are created, based on the keypoints at those turn corners that border to the gap. An air area is created from the lines. Finally, all created areas - strands, insulation and air - are selected, and a stator conductor component is assigned to them. A component is also made for the outer border lines of the stator conductor.

Re-generation of slot air

After the `vsinsertconductor`-macro has completed the stator conductor generation and component assignation, the slot air is re-generated by use of the `vscreateslotair`-macro. This macro selects the slot air border lines by the component assigned in `vsremoveslotair`. It removes the mesh from the bordering areas and creates a new slot air area. Then the border lines from the stator conductors are selected by use of their respective components, the mesh is removed from the bordering areas, and the border lines are used to cut areas out of the slot air. These areas, which cover the same areas as the stator conductors, are deleted. At the end of the macro, all unmeshed areas are remeshed, except for the slot air itself. The slot air meshing is treated at the end of chapter B.

4.2.2 Naming of new areas

The main purpose of the area naming is to provide the possibility to select necessary areas in the stator slots when connecting the stator circuits. Area components are therefore assigned to the new areas as soon as they are created. The following section will treat the

logic behind the naming and explain the naming syntax.

Component groups

The principles for the naming are based on the coil windings, since they demand more specific information than the bar windings. The difference between coil windings and bar windings in this regard, is that coil windings have several turns per coil side, which is not the case the for bar windings. Still, the same naming convension is used for both of them. Therefore the bar windings are described as to have one turn per stator conductor, even though the turn and the stator conductor of a bar is the same thing, as explained in chapter 2.2.3.
The created components were categorized in three types of area groups. These groups are listed below.

- Strand components, for all the areas in each strand
- Turn components, for all the strands in each turn in one slot
- Conductor components, for all the turns in one stator conductor

The syntax for each type of component, and the corresponding illustrations, follows in next sections.
All the area components of each group are uniquely numbered. The numbering reflects their location in the machine, so that it will be easy to select them as soon as their position is known.

Components for the strands

The example below shows the naming syntax used for the strands, and the numbering is explained. Figure 11 shows the order of the coil side numbering as well as the layers, and figure 12 shows the order of the strand rows and columns.

```
cm_a_strand_[a]_[b]_[c]_[d]_[e]
```

The legend for the strand naming is:

a Number of the stator slot where the strand is located

b Number of layer. 1 = Upper Layer (UL), 2 = Lower Layer (LL)

c Number of turn. Always equal one for bar windings

d Number of strand row

e Number of strand column

The example below shows what the naming of such a strand component can look like. In accordance with the syntax explanation, the location of this strand is in column 2, row 8 of turn 3 in the upper layer of slot 14.

`cm_a_strand_14_1_3_8_2` (4)

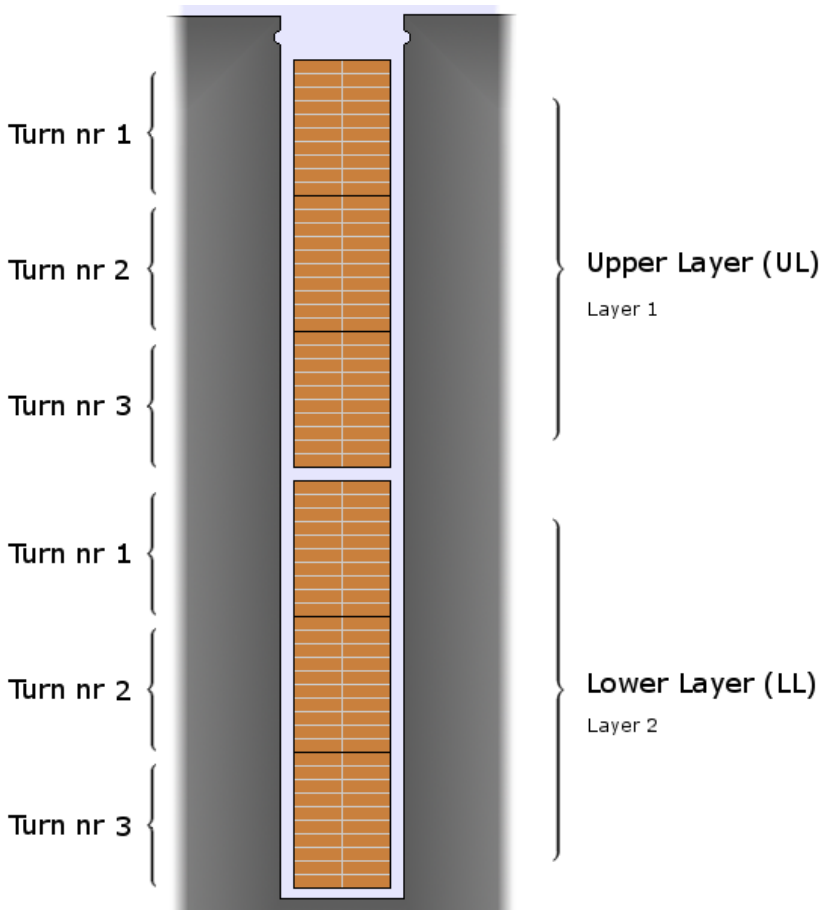


Figure 11: Shows the numbering of the layers and the turns

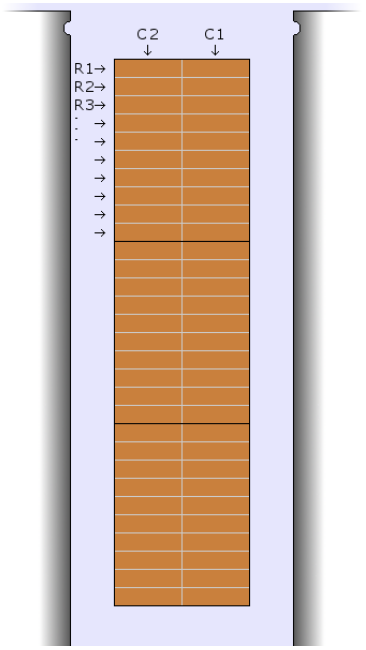


Figure 12: Numbering of strand rows and columns in one conductor

Components for the turns

The naming of the turns follows the same logic as the naming of the strands. This means, slot number, layer number and turn number is used to identify each component uniquely. The syntax is shown in example below.

```
cm_a_turn_[a]_[b]_[c]
```

The corresponding legend is the same as for strands, only without row and column number:

a Number of the stator slot where the strand is located

b Number of layer. 1 = Upper Layer (UL), 2 = Lower Layer (LL)

c Number of turn.

Components for the stator conductors

The stator conductors are also named after the same logic as the previously treated area components. That gives the syntax of the example below.

`cm_a_conductor_[a]_[b]`

Direction of the numbering

When studying figure 11 and 12 it can be noticed that the positive direction of the column-numbering is from the left to the right; the opposite of the reading direction. The positive direction of the row- and turn numbering is also different from what is used in the DI1359-program, which is explained in chapter 2.2.4. The reason for this inconsistence is that the components were named and numbered in the same order as they were created, as mentioned in chapter 4.2.1, under "Building of bar- or turns". Since it was most useful to carry out the area creation in the global cylindrical coordinate system, the view of the slots got turned upside down, compared to the convension that was used in DI1359. This is important to be aware of when the circuit connections shall be made in ANSYS.

4.2.3 Replacement issues and details

Strand creation in `vscreatestrand.mac`

A dedicated macro was created for the building of each strand, in case the demand for various strand shapes should occur. As far as the stator conductor detailing has been developed, this has not been an issue, but it is assumed that support for different shapes will be needed at a later point.
The macro builds up a strand by several smaller areas, and controls the meshing through assignation of line divisions for the different regions of the strand. The line division is strictly controlled in order to achieve sensible element sizes, wich will be discussed in the following.
This macro builds a detailed strand from scratch, based on the input values listed below.

- `n_strand`
- `x_pos_strand`
- `y_pos_strand`
- `w_strand_cu`
- `h_strand_cu`

- `w_strand_ins`
- `h_strand_ins`
- `b_rounded_strand_corners`
- `r_strand_corners`
- `b_create_mesh`

The first parameter, the number of the strand, is used in creation of a local cartesian coordinate system, in which the macro operates. Thus must this number be greater than 10, in accordance with the ANSYS-regulations. The x- and the y-positions are in cartesian coordinates, relative to the stator conductor center coordinate system. The coordinates point to the center position of the strand. `w_strand_cu` is the width of the strand copper, `h_strand_cu` is the height of the strand copper, `w_strand_ins` is the width, and `h_strand_ins` is the height, of the strand insulation. `b_rounded_corners` is a boolean value that must be set to 1 if the corners of the strand shall be rounded. Otherwise they will be modelled as squared. `r_strand_corners` is the radius of the corner arcs, in case of rounded corners. b_create_mesh tells whether or not the strand areas shall be meshed by this macro.
The macro builds the strands by generating keypoints, drawing lines between these and finally creating areas between the lines. All the line numbers are stored in temporary parameters, such that they easily can be selected, and line division can be applied to them. A map of all keypoints, lines and areas can be viewed in appendix E. In order to get the finest mesh near the strand edges, spacing ratios were used both in vertical and horizontal direction. Additionally, a coarsening factor was implemented, so that the mesh can be adjusted by the user. This provides the possibility to choose very accurate calculations with a very fine mesh, at the cost of computational time. Correspondingly, the mesh can be made coarser to make the computational time shorter, at the cost of accuracy.
The `vscreatestrand`-macro assigns copper material properties to the copper areas of a strand, and models the insulation areas as air. Figure 13 shows three versions of strand modelled in ANSYS with `vscreatestrand`. The top strand shows the copper areas in in pink and the insulation in red. The middle- and the bottom picture shows a strand meshed with coarsening factor equal 2 and 6, respectively. The coarsening factor for the mesh is in practice the number of line divisions applied to those lines that originate at the corners of the center rectangle area in the strand, and go outwards. A coarsening factor lower than

2 is not allowed. For the same lines, the spacing ratio is adjusted such that the divisions appear more and more often towards the outer edges. For the long horizontal lines crossing the center y-axis, the spacing ratio is set such that the divisions appear more and more often in direction of either end.

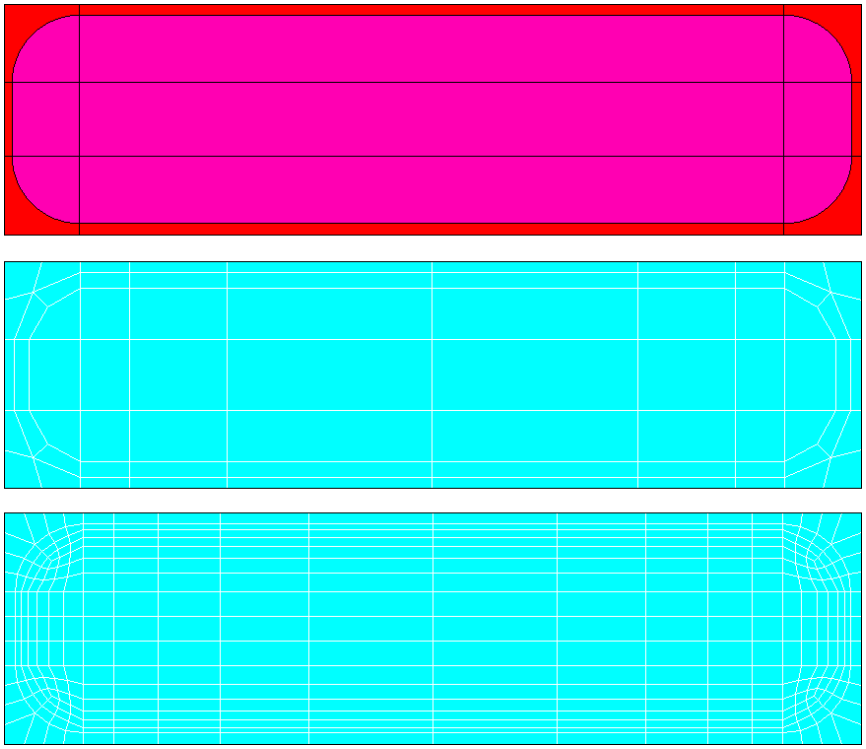


Figure 13: ANSYS-models of single strands. Top: Air- and copper areas and lines. Middle: Strand meshed with coarsening factor 2. Bottom: Strand meshed with coarsening factor 6.

All the areas are meshed with mapped mesh, as far as possible, and quadrilateral shaped elements. The choice of element type depends on whether an upper layer or lower layer conductor is being replaced, as well as the direction of the current, in accordance with the description in chapter 4.1.2.
This strict and uniform meshing procedure for each strands assures an uniform mesh throughout the entire stator conductor, providing identic simulation premises over all areas.

Modification of Roebel-bar insulation areas

During creation of Roebel-transposed bars, with two strand columns and one single top- and bottom strand, problems were encountered during meshing of the top- and bottom in-

sulation areas. These appeard to be bordering to other areas that were differently arranged, such that merging or gluing turned out failing, and thus preventing successful meshing. A procedure for reworking the insulation areas was therefore implemented. This procedure selects the insulation areas of both the single top- or bottom strand and the bordering strands, deletes them and re-creates one coherent area, as shown in figure 14. This way the conflict between the many small areas is worked around.
The actual reworking procedure turned out somewhat intricate, in order to be forward compatible with strands of various sizes. First the insulation areas are chosen by location, using the `ASEL`-command. Thereafter a re-selection is done by material number, to assure that no copper is included in the selection. A component is then assigned to the lines contained in these areas, before the mesh is cleared, and the areas are deleted. The keypoints in the remaining lines are then selected, and a re-selection of only the external keypoints is carried out. At this point, only the corner keypoints for the new coherent area are selected. These are used to generate new lines for the encapsulation of the new area. Once all the lines exist, line divisions are applied and the new area is generated and meshed. This procedure is programmed to support stator conductors with any number of strands in width.

Dependencies between the macros

Certain sequences in the replacement procedure have, as mentioned, been written in own macros to provide an overall program flexibility. Still, quite a few of the macros depend on parameters from other macros. This is because the APDL-language is a simple scripting language and not specifically designed for object-oriented programming.
For example the macro `vsremoveslotair` creates a line-component for the slot air border lines. Also, area-components for the stator conductors are made in `vsremoveslotair` and `vsinsertconductor` respectively, while `vscreateslotair` is the macro that uses all these. Therefore it is recommended to check the prerequisities for each macro before running them in a stand-alone manner, or applying changes or replacements to them.

4.2.4 Management of detailing alternatives

A macro was written to control, and manage the entire stator conductor replacement process for the different detailing alternatives. This macro, named `vsdetailedwdg`, calls the macros `vsremoveslotair`, `vsinsertconductor` and `vscreateslotair` in the required sequential order that corresponds to each detailing alternative. Also, it calls a few additional

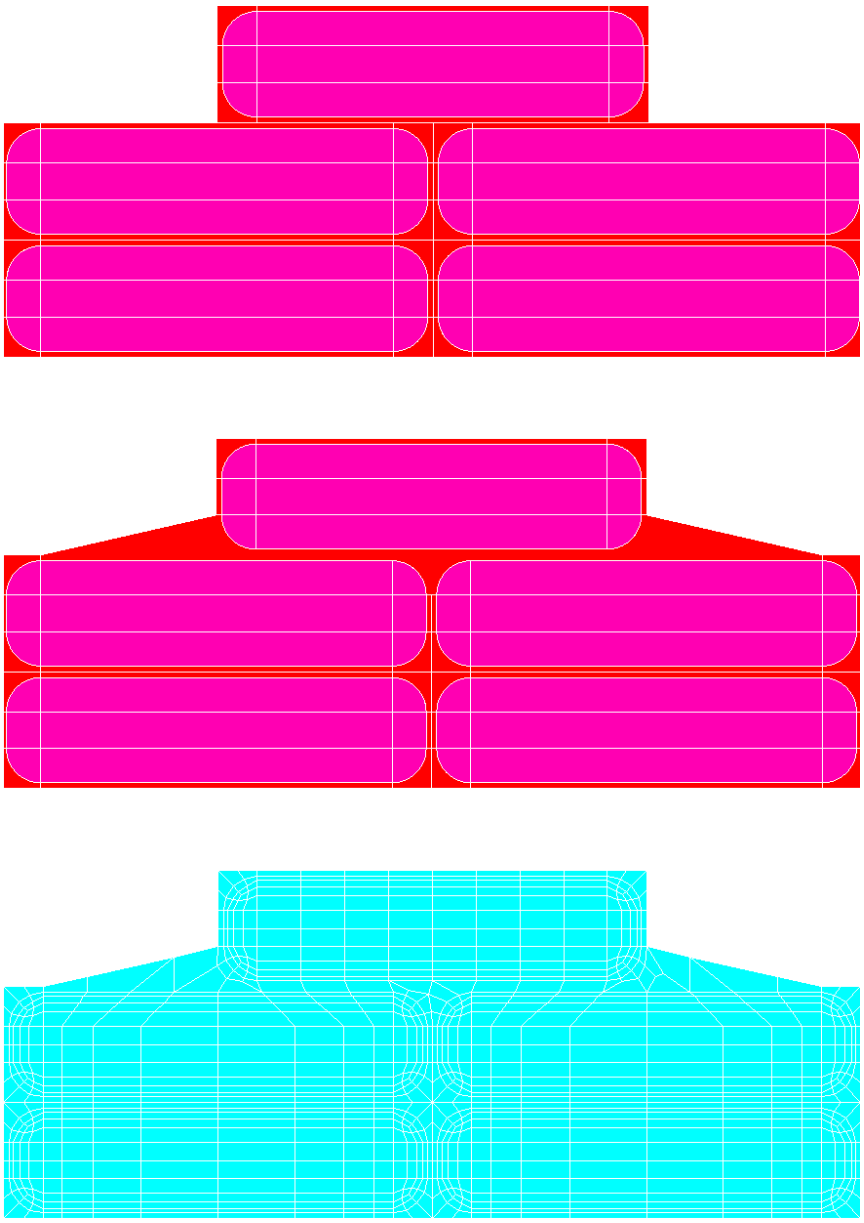


Figure 14: Rework of Roebel bar air insulation between top strand and the rest. Top: Untreated Roebel bar. Middle: Reworked Roebel bar. Bottom: Mesh after rework.

macros for information gathering. The flowchart in figure 15 shows the execution, which is further explained in the following sections.
An important basic principle for the `vsdetailedwdg`-macro is that it works in the ascending slot number direction. A start slot number is chosen in Excel by the user, and the stator conductor replacement, in accordance with each respective detailing alternative, is carried out in the positive slot number direction. That means, if for instance one single coil is to be replaced, the start slot number will define in which slot the lowest numbered side of the coil is located, regardless of layer. This is important to be aware of in order to choose the correct start slot number, for instance in case several conductors are desired to be replaced.
There is an exception to this rule for the three last detailing alternatives. Since these are replacing a parallell branch, one phase, and all three phases, respectively, there is no need for a start slot to be defined. Therefore, these alternatives do not make use of the start slot number. The complete list of available detailing alternatives can be viewed in appendix B.

Determination of the winding pitches

The winding pitches defined in DI1277 are used in `vsdetailedwdg` in the stator conductor replacement procedures for selection of which slots to treat, as well as choice of layer. However, only the absolute value of the mean pitch is exported from Excel, which leaves the pitches Y1 and Y2 undetermined. A macro named `vsdeterminepitch` was created to determine them, by use of the winding information arrays from DI1277. The side at which the circuit ring is located is not explicitly defined in DI1277, and therefore also this is determined in the macro.
The basic principle of the macro is to search for the ends of a coil- or bar-group in the middle of the FE-model, and determine at which of the sides - NDE or DE - the circuit connections are located. This side is then defined as the circuit ring side. As soon as this is known, any given connection on the non circuit ring side can be checked without the possibility of hitting a circuit connection, and therefore the connection will explicitly define the pitch on this side. Further, in DI1277 the pitches Y1 and Y2 are defined as to belong to the NDE- and the DE-side, respectively. Thus, if the NDE-side is the non circuit ring side, the pitch Y1 is the one that has been defined, or if the DE-side is the non circuit ring side, the pitch Y2 has been defined. Once this has been determined, the pitch on the circuit ring side is calculated from the mean pitch value from Excel and that pitch that has been determined.

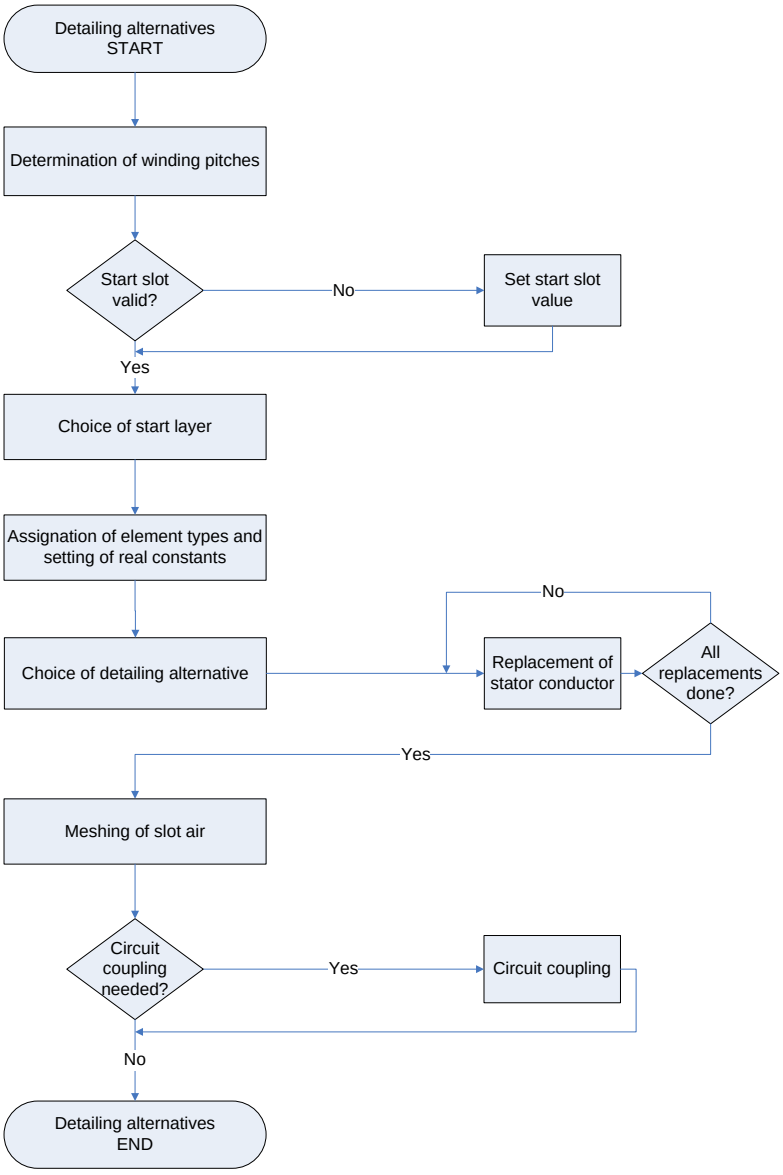


Figure 15: Flowchart showing the process of carrying out the detailing alternatives

This procedure will define the winding pitches reagardless of how the machine connections are arranged, but the circuit ring side might not be correctly determined if the machine is arranged with circuit connections on both sides.

Validation of start slot number

The parameter `n_slot_replacement_start` holds the number of the slot where the conductor detailing will be started. Since this parameter is set in Excel by the user, a validation process was written, in case the user has mistyped the number, or just made an illegal choice. The allowed range of slot numbers are from 1 and up to the FE-model maximum number if not the entire machine is modelled. If the entire machine is modelled, the allowed range is from slot number 1 and up to the highest slot number minus two times the biggest winding pitch. This is to avoid problems with connection of coil groups, which is discussed in chapter 4.3.1 about gathering of coil group information.
An if-statement is used for the validation. If the start slot is not specified - the value is equal zero, that is - the parameter is set equal 1. Further in the same statement, if the start slot value is negative, it is altered to positive. If this positive value exceeds the allowed range, the slot number is also here set equal 1. Finally in the statement, it is tested whether the value exceeded the allowed upper limit already at the beginning of the test. If so, the slot number is set equal the FE-model maximum if not the entire machine is modelled. However, if the entire machine is modelled, the slot number is set equal the machine maximum slot number minus two times the biggest winding pitch.

Choice of layer

Since the `vsdetailedwdg`-macro operates in the positive slot numbering order, it must be determined in which layer the replacement shall start. This choice of layer is only needed for replacement alternatives where connected stator conductors are to be replaced - that is coils, coil groups or connected bars.
By checking the sign of the winding pitches, the layer is explicitly determined, regardless of which side the circuit ring is located on. This is a consequence of how the pitches are defined. An if-statement is used to test whether the Y1-pitch is positive or negative. Since Y1 is defined as the connection from an UL-conductor to an LL-conductor, a positive value must result in a choice of UL as the starting position. Correspondingly, a negative value of Y1 must result in a choice of LL as starting position.

Assignation of element types and setting of real constants

The `vsdetailedwdg`-macro is the head macro for the entire winding detailing process, and it directly or indirectly makes all the calls of those macros that create elements in the model. It was therefore chosen to assign the needed element types and set the real number constants here.

Choice of detailing alternative

The parameter `n_wdg_detailing_alternative` is set in Excel by the user and defines which detailing alternative that should be carried out. The parameter can only have a value from 1 to 17, as a result of being chosen from a drop-down menu containing only these alternative. A validation is therefore not needed.
An if-statement in `vsdetailedwdg` checks the value of the parameter, and executes a tailored stator conductor replacement procedure for the given alternative. This can be a simple sequence of running `vsremoveslotair`, `vsinsertconductor` and `vscreateslotair` after each other for the simplest detailing alternatives, or it can span over several nested loops for the more advanced detailing alternatives. However, the basic puropse is always to replace stator conductors, and therefore that is the description used in the flowchart of figure 15.
Alternative 1 to 7 is basically carried out by use of a simple sequence, running the three mentioned macros in various orders, and at some points small loops are used.
More intricate procedures apply to the alternatives 8 to 14, where mainly coils are treated. Variations of nested loops occur, but in alternative 13 and 14 there is also use of a helping macro called `vsbuildcoilgrarrays`. This macro gathers information about coil groups and stores it in the arrays `ar_n_coilgroup_info` and `ar_c_coilgroup_info`. The information that detailing alternative 13 and 14 uses from these arrays, is the start positions of the groups and the number of coils within them. The macro and the arrays are enlarged upon in chapter 4.3.1 under the explanation of the coil group information arrays.
Alternative 15 to 17 are as mentioned the only alternatives where the start slot is not used. But then again the replacement procedure here is somewhat different from the rest. These alternatives run through all stator conductors in the model and checks each of them if they must be replaced.

Meshing of slot air

The slot air is not being meshed in the `vscreateslotair`-macro like the areas bordering to the slot. Meshing in that macro would cause the air area to be cleared and remeshed more than once if two neighbouring slots are treated. This is because the slot air areas are bordering to each other, and must therefore be cleared when the borderlines are being used to generate new areas. To save time on meshing, which is significant if quadrilateral elements are used, meshing of the air is carried out after all the stator conductors have been replaced. All the slot air areas are then selected and meshed by using the macro `vsmeshslotair`, which takes a parameter for element shape as input. At the time being, triangular shaped elements are used, due to lower time consumption and more reliable execution.

Coupling of electrical circuits

Before exiting the `vsdetailedwdg`-macro, it is checked whether or not circuit element couplings must be built for the replaced conductors. If coilgroups have been replaced, and the macro `vsbuildcoilgrarrays` has not yet been run, this macro is called. Subsequently, a macro named `vscoilwdgconnect` is run for the actual coupling procedure. This macro, and the entire circuit coupling procedure is treated in chapter 4.3.

4.3 Circuit coupling of coils

Since the strands in a coil winding are not shorted at the end of each coil side, like they are in a bar winding, a procedure was needed that could create electrical connections between the strands in each stator slot of the coil winding. In accordance with the description in chapter 2.2.4, this internal connection within each coil had to be done for every strand, and support both transposed and untransposed connections. Coupling between the coils within a coil group was also needed, as well as coupling between the groups. Blockwise connection was required to be supported for the two latter alternatives.
The following sections in chapter 4.3.1 describes how information about the coilgroups in the machine was retrieved. The sections thereafter, in chapter 4.3.2, outlines the circuit coupling procedure, and how the coil group information was applied. Important details and special issues for the procedure is treated in chapter 4.3.3.

4.3.1 Gathering of coil group information

Quite a bit of information is needed to carry out the circuit coupling. Two macros were written for the information search, and one for the organisation of the information. The name of the macros are listed below, and the following sections explain their purpose and how they work together.

- `vsgetcoilgrinfo`
- `vsfindneighcoilgrouops`
- `vsbuildcoilgrarrays`

Coil group search

The macro `vsgetcoilgrinfo.mac` was created to collect information about any randomly chosen coil group. The macro takes a slot number and a layer character as input - with valid characters being "`u`" and "`l`" - and collects information about the entire coil group that this stator conductor belongs to. The macro uses the input values as start position in a searching loop, and stores the phase and direction of the conductor at this location. The loop then runs through the neighbouring stator conductors in the same layer until conductors with a different phase and direction have been found. Positions are stored for the last conductors that have the same phase and direction as the starting conductor. After the loop has finished, these two positions are used to determine which is the start- and end-conductors of the coil group. Several tests are then carried out to determine various information about the group. Temporary parameters are used to store the information, and if the macro is run again, the parameters will be overwritten. It can be mentioned that the macro supports search across the FE-model start- and end-slot. That means, the macro will be able to start a search in the last available slot of the model and continue in slot number one, or converse.
The procedure for gathering of information is dependent on how the circuit ring side is defined, and support only machines with circuit connections on one side.

Parameters created by the coil group search

The parameters stored by the coil group search are given in table 7. As the parameter description states, slots are stored as numbers, but the layers are stored as characters. This is because the slot numbers often are used to select elements from the Di1277-arrays, while

the layer often is used to choose the correct array. Since the names for the layer-dependent arrays are distinguished by the characters "u" and "l", the same characters are stored.
The suffix of most of the parameters is either "high" or "low". This always reflects whether the parameter belongs to the highest- or lowest-numbered stator conductor of the group. Further, the strings c_coilgr_start and c_coilgr_end define if the start and the end of the coil group is on the higher- or lower-numbered side. By combining these two strings with the rest of the parameters, several combinations of parameter selections are availiable. This provides some selection freedom that is necessary in those macros that use the information. The start of the group is in this regard defined as that conductor in the group which has a connection in the negative slot number direction. The group end is then correspondingly defined as the conductor with a connection in the positive slot number direction.

Parameter name	Unit	Description
n_curr_slot_high	[#]	Highest nubmered slot of the group
n_curr_slot_low	[#]	Lowest nubered slot of the group
c_layer_high	[-]	Character, layer on the high-side; "u" or "l"
c_layer_low	[-]	Character, layer on the low-side; "u" or "l"
b_coilgr_exceeds_high	[boolean]	Out-of-bounds parameter for high-side
b_coilgr_exceeds_low	[boolean]	Out-of-bounds parameter for low-side
n_coils_within_group	[#]	Number of coils in the group
c_coilgr_start	[-]	String; group start position; "high" or "low"
c_coilgr_end	[-]	String; group end position; "high" or "low"
c_coilgr_conn_high	[-]	String; connection type; "terminal" or "normal"
c_coilgr_conn_low	[-]	String; connection type; "terminal" or "normal"

Table 7: Parameters stored by the coil group search

Search after neighbouring coil group

The information arrays from DI1277 provides extensive information about the connections in the stator, and these arrays can often be used to find the neighbouring coil groups. However, if either detailing alternative 16 or 17 is chosen, conductor replacement may be necessary across terminal connections in the machine. When encountering a coil group with connection to terminal, looking up this connection in the DI1277-arrays would not provide the neighbouring coil group location, but rather the other end of the same branch. Therefore the macro vsfindneighcoilgroups was written to explicitly determine the neighbours of a coil group, regardless of the connection type.
The macro uses several parameters from vsgetcoilgrinfo.mac, amongst others the slot

numbers and the layer parameters for the highest- and lowest-numbered conductors in a group. The procedure checks the phase and direction of these two conductors against the other conductors in the same slots, and returns the slot number in case of a match. In case the phase and direction do not match, the macro searches in positive and negative direction for the highest- and lowest-numbered slots, respectively, until a match is found. It should be noted that although the macro always returns slot numbers from the neighbouring coil groups, it might not necessarily be returning the end slots of this group. Therefore, if the group ends are needed, `vsgetcoilgrinfo.mac` should subsequently be used to validate the returned information.

Coil group information array

Two arrays containing information about all affected coil groups are created when one of the winding alternatives from 13 to 16 are chosen. If alternative 17 is chosen, two arrays are created for each phase. One of the arrays, `ar_n_coilgroup_info`, stores amongst others various positions or slot numbers, and the other array, `ar_c_coilgroup_info`, stores amongst others layer information as characters. Each of them have 6 columns, and the contents are described in table 8 and 9 The information in these arrays is crucial for the later coupling procedure of the coils. The macro `vsbuildcoilgrarrays.mac` was written to build the arrays.
The information that is stored in the arrays is gathered by use of the `vsgetcoilgrinfo`-marco. The number of rows in the arrays is decided by which detailing alternative that is used. For alternative 13 and 14 information is stored for for 1 and 2 coil groups respectively, whereas alternative 15 to 17 requires all the groups in the FE-model. This is adjusted by the `vsbuildcoilgrarrays.mac`-macro, and in addition, a validation test is carried out. By validation it is here meant a test of whether or not the coil groups are within bounds. For detailing alternative 13 and 14, groups containing conductors that are out of the FE-model boundaries are not allowed. For alternative 14, two groups with a terminal connection between them is also not allowed. If any of these cases are encountered, the macro searches the neighbouring groups until valid values are found. Detailing alternative 15 to 17, however, allows all coil groups that have one or more conductors within the FE-model boundaries. This is because coil groups at the end of the model might both contain conductors that are within and out of bounds. If these end-groups were to be rejected, information about the end conductors within the model would be left out.

Col	Content of column
1	Start slot number for the coil group
2	End slot number for the coil group
3	Slot number of the connection to the start slot
4	Slot number of the connection to the end slot
5	Number of coils within the group
6	Slot number of the lowest numbered group slot

Table 8: Information stored in the array "`ar_n_coilgroup_info`"

Col	Content of column
1	Layer for the start conductor of the group; "`u`" or "`l`"
2	Layer for the end conductor of the group; "`u`" or "`l`"
3	Layer for the connected conductor to the start slot; "`u`" or "`l`"
4	Layer for the connected conductor to the end slot; "`u`" or "`l`"
5	Connection type at start side; "`terminal`" or "`normal`"
6	Connection type at end side; "`terminal`" or "`normal`"

Table 9: Information stored in the character array `ar_c_coilgroup_info`

4.3.2 Automation of the circuit coupling

The automated procedure for circuit coupling is, like the solid modelling, written in several macros to preserve flexibility. In this section a short introduction to how the macros work together will be given, and a flowchart is included for the visualisation. Important issues and special treatements are discussed more thoroughly in next section.
Seven macros were written for the circuit coupling procedure, as listed below. The macros use some parameters from DI1204 for the placement of nodes, and information from the array `ar_b_conductorreplaced` for the selection of conductors to couple. Winding information in addition to this, from the arrays `ar_n_coilgroup_info` and `ar_c_coilgroup_info`, is also used.

- `vscoilwdgconnect.mac`
- `vscreatecoilcp.mac`
- `vscreatecoilce.mac`
- `vscreatecoilgrcp.mac`
- `vscreatecoilgrce.mac`

- `vscreatebranchcp.mac`
- `vscreatebranchce.mac`

The macro `vscoilwdgconnect` is the head macro of the coil coupling. It contains the logic for which treatment that shall be carried out - coupling within a coil, coupling between coils within a group, and coupling between groups, respectively - and calls all other needed macros based on this logic. A flowchart for the execution can be viewed in figure 16

Creation of circuit elements for each strand

The `vscoilwdgconnect`-macro uses a loop to run through the `ar_b_conductorreplaced`-array and check which stator conductors that must be treated. For every replaced stator conductor it runs the `vscreatecoilcp`-macro, which carries out the circuit element creation for all strands. This macro is only briefly described here; a more thorough description is included in chapter 4.3.3.
The `vscreatecoilcp`-macro loops over all strands in a stator conductor and creates circuit elements for all of them. These elements consist of three nodes each; one for the DE-side, one for the NDE-side, and one node from the FE-domain of the strand. The elements are placed outside the stator model in such an order that coupling between them will be easy to carry out, and a certain lucidity is preserved. Factors like layer and transposition decides the position of each element. At the time being, it is possible to choose between one transposition or none at all per coil. The node number of all the treated nodes are stored in a dedicated node number-array for each treated stator conductor.

Coupling of circuit elements within each coil

After the circuit element creation for all the replaced stator conductors have been completed, the macro `vscreatecoilce` is run for the coupling of these elements. This macro includes a loop over all treated coils, and does therefore not have to be run for each replaced conductor.
The macro selects the nodes to couple by use of the node numbers from the arrays created in `vscreatecoilcp`. For the coupling of each pair of nodes, on either the DE- or NDE-side, a CIRCU124-element with key-option 1 = 1 is created between them. That means a resistor is used. This is done for all the internal couplings in each coil, but the coil ends - the connection points of the coils - are always excluded. It can also be noted that all

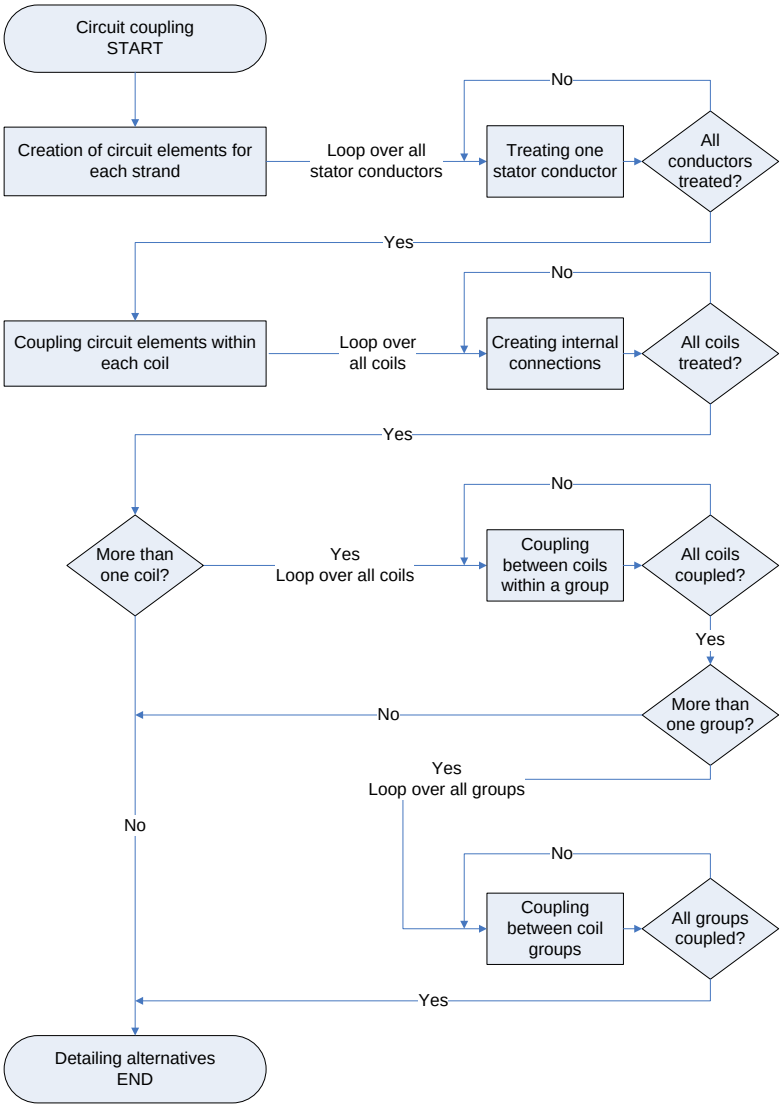


Figure 16: Flowchart showing the procedure of how the circuit coupling in coil windings is carried out

connections are uniquely done from strand to strand; there is no use of short ciruiting of several strands.

Coupling between coils within a group

If coil coupling internally in one or more coil groups is needed, `vscoilwdgconnect` calls the macro `vscreatecoilgrcp` and `vscreatecoilgrce`. This is done inside a loop over all replaced groups. If this is not needed, the execution proceeds to the end of the `vscoilwdgconnect`-macro.
The `vscreatecoilgrcp`-macro has much in common with `vscreatecoilcp`, but there are also a few differences. Since the coupling between coils can be done blockwise, the macro supports short circuiting of a specific amount of strands per block. This is achieved by selecting those circuit element nodes at the coil ends that belong to each block, and couple the degrees of freedom for these. The macro creates a new node and couples it with the rest of the nodes in the block; a procedure that is somewhat similar to the circuit element creation for each strand. This node is placed outside all previously created nodes, in order to make the geographical arrangement more lucid. This placement makes it easy to differ between internal coil connections and connections between coils.
The node numbers of the reference nodes for the coupled sets, as well as the new created nodes, are stored in arrays, similar to how it is in `vscreatecoilcp`. It can also be mentioned that the macro handles transposition between coils within a group, if such a transposition is applied. One transposition per group is currently supported.
The `vscreatecoilgrce`-macro is called for the actual connection of the new generated nodes. A loop over all blocks in one connection is run, generating resistor-elements, somewhat similar to the internal coupling within a coil.
In order to select the correct coil ends - those ends with connections within a group - the array `ar_n_coilgroup_info` is used. This array provides the start- and end location of the treated coil groups, as well as well as the number of contained coils.

Coupling between coil groups

After all the couplings between coils within each group have been made, it is checked whether or not more than one coil group have been replaced. If that is the case, the `vscoilwdgconnect`-macro enters a loop over all replaced groups and creates connections between them. The procedure is very much similar to coupling between coils within a group, and the same features are supported. The only difference is that the created nodes

are placed even farther outside the stator - outside all previously created nodes - and that the macros `vscreatebranchcp` and `vscreatebranchce` are used.

4.3.3 Coupling issues and details

Creation of circuit elements in `vscreatecoilcp.mac`

The macro takes a slot- and layer number as input, and for that stator conductor defined by these values, it creates circuit elements (CE) for all contained strands. Also, the macro creates a connection between each CE and those elements in the FE-domain corresponding to that strand. Various information is used during this procedure, and some useful factors in this regard will be summarized before the actual procedure is explained.
The array `ar_nd_strand_coupling_[a]_[b]` is used for the storage of some node numbers related to each strand in each stator conductor. In the suffix `_[a]_[b]`, the slot- and layer numbers respectively, are stored. Five sets of node numbers are stored, and for each set, node numbers for each strand is saved. Therefore the array has as many rows as the number of strand rows times turns in a stator conductor, and five times as many columns as there are strand columns in a stator conductor. The information contained in the array is explained in the follwing, each after where it is used.
The macro places the created CEs outside the stator model, arranged after how they will be coupled. The CE-nodes are created in the global cylindrical coordinate system, and therefore the angle and radial position is calulated for each one of them. For this purpose, some standard values for distance are created to simplify the calculation. These are shown in table 10. Every CE-node will be created by using the angle from one of the two first parameters in the table, and calculating the radial position from the four last parameters. The macro runs a triple loop over all turns, strand-rows and strand-columns, to treat all the

Parameter name	Content of column
`an_slot_NDE`	Angular position for the NDE-nodes of this slot
`an_slot_DE`	Angular position for the DE-nodes of this slot
`xr_st_air_o_end`	Radial coordinate; outer end of the stator model
`d_circ_elem`	Standard radial displacement between CEs
`h_CE_layer`	Height of all CEs in one layer
`rat_d_NDE__DE`	Displacement ratio between NDE- and DE-nodes

Table 10: Position and distance parameters used for placing of circuit elements

strands in the stator conductor. Each strand is selected by its respecitve area component,

and therefore the sequential order of the CE-creation is the same as when the strands were created. This order is the same for both upper- and lower layer, but in DI1359, conductors are connected in a converse-ordered arrangement from upper- to lower layer. This issue is handeled by parting the CE creation procedure from the CE coupling procedure, and running them in each their macros; the `vscreatecoilcp`- and the `vscreatecoilce`-macros, respectively. The CE creation procedure, and how it treats the problem, is explained in the following.
Inside the triple loop, all nodes within the strand are selected and coupled together in the `VOLT, EMF` and `CURR` degrees of freedom by use of the `CP`-command. The lowest numbered node in each strand is by ANSYS defined as reference node for these coupled sets, and is therefore stored in the node number array. These nodes will be referred to as the CP-nodes. Further down in the loop, new nodes are created for the CEs of the strand; one for the NDE-side and one for the DE-side. When creating a CE from these, also the CP-node of the respective strand is included, so that the CEs are connected with the FE-domain. The element type is set to CIRCU124 with key-option 1 = 6, which is a massive 2D-conductor with connection to the FE-domain. Direction is assigned to each CE by defining the DE- and NDE-node as either start- or end-point, and the direction is based on the DI1277-information arrays. The node numbers for all the created CE-nodes are stored twice in the node number array; one time in the same order as the sequential treatement, and one time in the geographical placement order. The sequential treatement order is, as mentioned, equal for all strands. However, the geographical row-order is altered to the converse for the lower layer. That means, the nodes belonging to the lower layer strands closest to the rotor, are geographically placed farthest away. This way the `vscreatecoilce`-macro can connect all CE-nodes in the geographical order, and the connection between upper- and lower layer becomes equal to what it is in DI1359. However, this is for an untransposed connection. For transposed connections, the geographical order in lower layer is further altered; the order in column-direction is altered to the converse, but the order in row-direction is kept equal to the sequential.

Transposition

For transposition of the coil windings, it was chosen to alter the order to the converse in both row- and column-direction for the strand coupling. This was considered to be a more complete transposition than only altering the order in column-direction.
This version is used at all the three locations where transposition can be applied.

5 Results and discussion

The achieved results, and a discussion of these, will be outlined in the following. Primarily, issues in regard of the solid modelling automation and circuit coupling of GP2DET-models will be treated. Since only a small amount of actual electromagnetic calculations and analysis have been performed, the available material from such, at the time being, is too small to give any broader discussion.
This chapter gives a summary of important features that were implemented and a look at some limitations. Suggestions are made for improvement of existing macros, as well as features to add.

5.1 Solid modelling

5.1.1 Solid modelling results

Machine compatibility

The macros for stator conductor replacement, discussed in chapter 4.2.1, were tested for three different machine models created in GP2DET. The test models included two machines with bar windings and one with coil windings. Additionally there were other important differences, as the short abstract of machine information in table 11 indicates.

Parameters	Machine 1	Machine 2	Machine 3
Number of slots	522	144	150
Number of poles	84	14	10
Number of strands, height	23	8	38
Number of strands, width	2	2	2
Height of strand [mm]	2.22	2.28	1.42
Width of strand [mm]	8.2	8.55	9.30
Number of turns	1	3	1

Table 11: Abstract of data for test machines

Detailing alternative number 2 was successfully applied to all the test models. The conductor replacement was carried out automatically without bugs or interruptions. Additionally, all detailing alternatives applicable for bar windings were successfully carried out for machine number 1, and and all alternatives applicable for coil windings were successfully carried out for machine number 2. Two examples of reworked GP2DET-models are shown

in figure 17. The upper figure shows a bar winding where two bars in the same slot have been replaced, and the lower figure shows a coil winding where the upper layer coil side in one slot has been replaced.

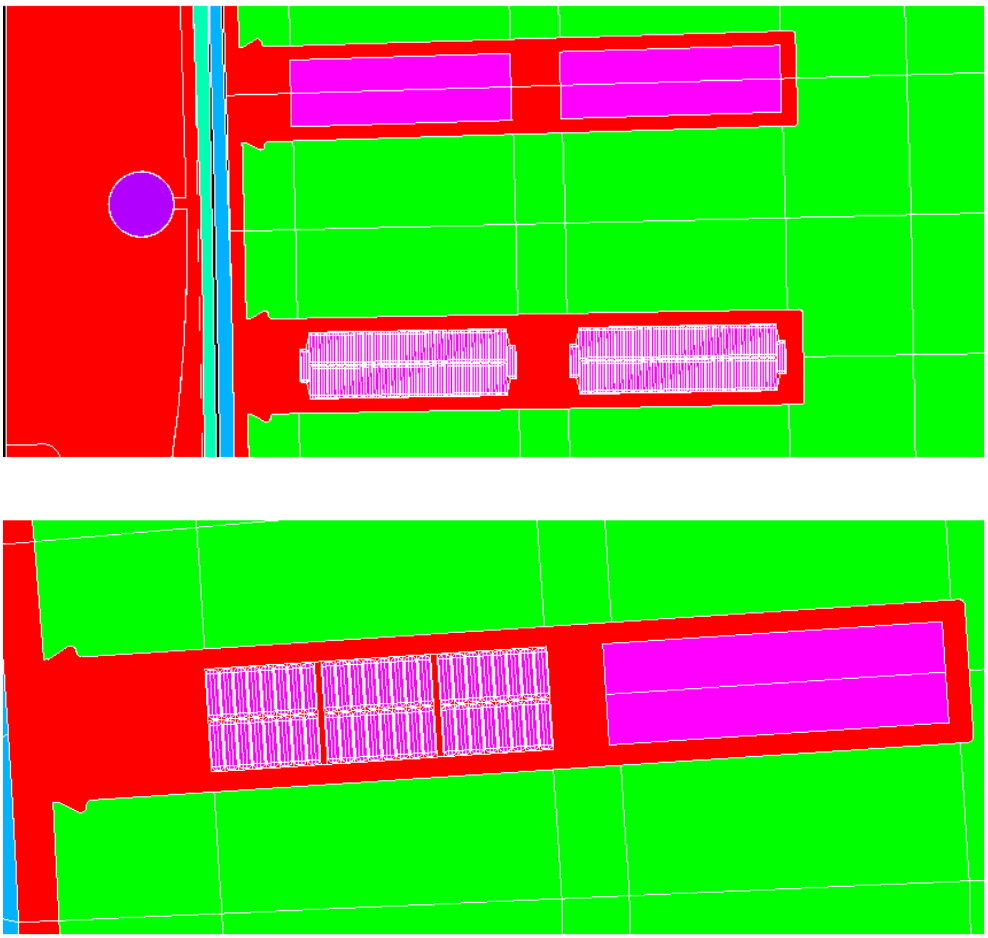


Figure 17: Examples of replaced conductors. Top: Both bars in one slot have been replaced. Bottom: Upper Layer coil side have been replaced.

Time consumption

The time required to complete a model revision strongly depends on which detailing alternative that is chosen. For instance, execution of detailing alternative 2 on machine number 2 requires approximately 2.5 minutes, whereas alternative 14 requires approximately 1 hour and 6 minutes. Because of the large requirements for some alternatives, a few measure-

ments were made in order to determine where the time is consumed. The measurements were made for execution of detailing alternative 2 in machine number 2, and the results can be be viewed in table 12. The times will vary slightly from machine to machine, due to variation in number and arrangement of strands. The time measurements for detail-

Process	Time consumed [s]	Part of total [%]
Total time	144	100
Building one stator conductor	134	93
Building of one strand	41	28
Creation of slot air area	5	4
Meshing of slot air area	0.5	0.3

Table 12: Time consumption for replacement of one stator conductor in machine 1

ing alternative 2 shows how the creation of the stator conductor consumes nearly all the processing time; more than 90 %. Inside this sequence, the creation of one strand is run three times for this machine, which amounts to a total of 123 seconds. That is 92 % of the stator conductor creation time. For other machines the creation time for one strand has been measured down to approximately 25 seconds, while other sequences stayed more or less the same. Still, the strand creation consumes a significant part of the total.

Meshing

Different coarsening factors for the meshing of the strands were tested on machine number 1 and 2. Various detailing alternatives were used for this testing. The best meshing results were achieved while using an even number from 2 to 10 for the mesh coarsening factor; 2 being the coarsest mesh, and 10 the finest. Factors higher than 10 resulted in ANSYS run-time errors.

5.1.2 Solid modelling discussion

Machine compatibility

All the tested machines were configured with either bar- and lap-windings or coil windings. Bar- and wave-windings have not been specifically treated nor tested during the development, and it can not be concluded that the software would work faultlessly for this kind of arrangement. In some cases, the `vsdetailedwdg`-macro uses the winding pitches to choose conductors to replace, and the pitch-determination procedure in `vsdeterminepitch` does

not support wave-windings. It is expected that the pitch determination test can be modified to support this.
The `vsdeterminepitch`-macro also determines the circuit ring side. This is currently done by testing at which side the circuit connections are located. However, this is not an explicit determination in all cases, since the circiut connections may appear on both sides of some machines. Therefore the procedure currently do not supports machines with circuit connections on both sides.
Stranded stator conductors with all the strands of equal size is the currently supported arrangement. Such stator conductors can be built with or without Roebel transposition.

Time consumption

Various improvements have been applied to the solid modelling procedure during the development. The most important upgrade in regard of time consumption was the upgrade from using the `vsinsertconductor-loop`-macros to using the `agen`-macros. The `loop`-macros were measured to use more than 3 minutes for replacement of one stator conductor in machine 1, whereas the `agen` uses 27 seconds. Most of this time difference originates from the actual area creation process in the `loop`-macros, but also the meshing is important. Since the `agen`-macros copy areas instead of building them from scratch, and in practice mesh only one strand and copy the rest, the amount of time saved becomes substantial. Even though the treatment of one strand is relatively quickly done, the time consumption for this process is of utter importance for the overall process. The number of strands that must be built for each slot is so big, that the strand creation by far constitutes the biggest part of the total time consumption.
While the area generation for strands is carried out many times per slot, the slot air area is generated only once. However, the difference in time consumption for different meshing processes is substantial. If the slot air is meshed with quadrilateral shaped elements, 46 seconds is needed to complete the process. However, for triangular shaped elements the required time is 1 second. Also, meshing with quadrilateral shaped elements was not always possible; Ansys did not manage to complete such a meshing.

Meshing

The meshing of the strands is strictly controlled in the `vscreatestrand`-macro. As long as the coarsening factor is kept within the recommended values from 2 to 10, the meshing is carried out faultlessly for all known cases. A meshing factor of 1 resulted in badly shaped

elements at the strand corner edges, and is therefore not allowed. A meshing factor above 10 is in some cases possible, but results in a finer mesh than what is expected to be useful in practice.
Since very few electromagnetic calculations have been performed until now, it has not been determined which coarsening factors that may be useful for different calculations. It is unknown how much the accuracy of the calculations will vary with the coarsening factor, but it is expected that the mesh can be adjusted to a fine enough degree for any useful purposes.

Possible weaknesses

At an early point of the macro-development it was several times experienced problems with entity-selection by position in ANSYS. For instance selection of the border lines of the slot air appeared to be problematic. Either too many or to few lines got included in the selection, varying from machine to machine. This was solved by first selecting the slot air area and subsequently select those lines included. Manipulation of this selection turned out to be the most reliable way of retrieving the slot air borders. Other cases also occured, and thus has the selection of different entities more and more frequently been done by material-, element type- or component-selection instead. However, some selection by position is impossible to avoid. Such selection still occurs in the code, but where it appears, it has proven itself to withstand testing. It is still considered, though, to be one of the weaker points in the code, due to the many experienced problems in the early development.
Meshing of slot air areas was at some points not possible when quadrilateral shaped elements were used. No problems were encountered with triangular shaped elements.

5.1.3 Suggested modifications

Area-copy procedure for entire stator conductors

The current version of the detailing procedure builds every stator conductor by looping over all strands. This makes it easy to name each new strand uniquely and assign correct element types. However, when many stator conductors are to be replaced, a lot of time is required for the strand-building. If a procedure was built for copying of entire stator conductors, this would be likely to reduce the time consumption substantially. Light testing on machine number 2 shows that copying of an entire stator conductor requires approximately 0.5

seconds, whereas building the same conductor with the `agen`-macros takes 134 seconds. If many conductors are to be replaced, this would make a big difference on the total amount. Even though a little more than 0.5 seconds per conductor is expected, as result of an implementation procedure, the improvement is expected to be big.
Copying of an entire conductor would require a tailored procedure of naming the new strands. This is expected to be feasable by creating a naming loop. Further, it would be needed to make a register of where stator conductors with the correct element types can be found. For each of the four supported element types, the first stator conductor would have to be generated in the already implemented way, but any stator conductors built after this could be copied.

Additional strand configurations

A procedure for building of stator conductors with other strand arrangements than the already supported ones is recommended. In particular, stator conductors containing strands of different sizes should be supported, as well as nested arrangements, as illustrated in figure 18. This could be done by modification of the existing `loop`-macros, since these already build each strand from scratch. It would be possible to let the user configure the stator conductor buildup in Excel by defining copper size and center point for each strand. The `vscreatestrand`-macro would probably have to be modified, since the air areas of the different strands could interfer with each other. However, this could be solved by not creating air areas in this macro at all, but rather generate air areas for the entire stator conductor after building all the strands.
This procedure would probably be slower than the `agen`-procedure, but if combined with area copying of entire stator conductors, the difference is expected to be very little for larger detailing alternatives.

5.2 Circuit coupling

5.2.1 Circuit coupling results

Machine compatibility

The circuit coupling-macros described in chapter 4.3 were tested on machine number 2 from table 11. Detailing alternative 4, 13 and 14 were carried out for the machine model, and the circuit creation was executed successfully. Figure 19 shows an example of a machine where detailing alternatvie 14 has been applied - two coil groups have been replaced. The circuit

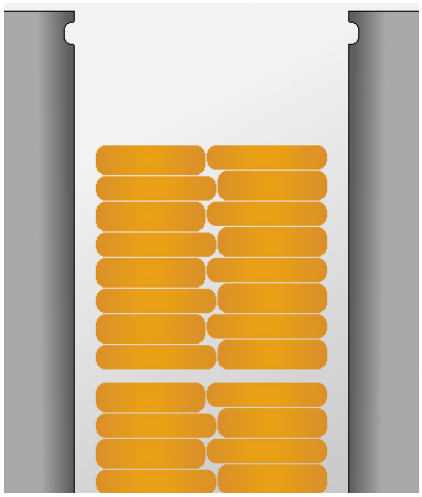


Figure 18: Example of stator conductor buildup with different strand sizes

elements are placed outside the stator, and the distance to the edge of the model varies from element to element. The upper layer elements are closest to the stator. Farther out comes the lower layer elements, followed by the coupling elements for connections between coils within a group. Farthest out are the coupling elements for connections between coil groups.
Transposition was applied within a coil, between coils within a group, and between coil groups. The two latter connections were carried out blockwise. As mentioned, one type of transposition is supported; altering of strand order to the converse in both row- and column-direction.
The searching macro `vsgetcoilgrinfo` is used for all alternatives larger than 12, and therefore these are only supported for machines with circuit connections on one side.

5.2.2 Circuit coupling discussion

Machine compatibility

Only one machine have been tested until now, and even though nothing indicates incompatibility with other machines that follow the same standards, the circuit coupling should still be tested on more machines.
Detailing alternative 13 and 14 do currently not support circuit connections across slot number 1, even though alternative 15 to 17 do. It has been determined how such connections should be done, but the implementation is not yet finished.

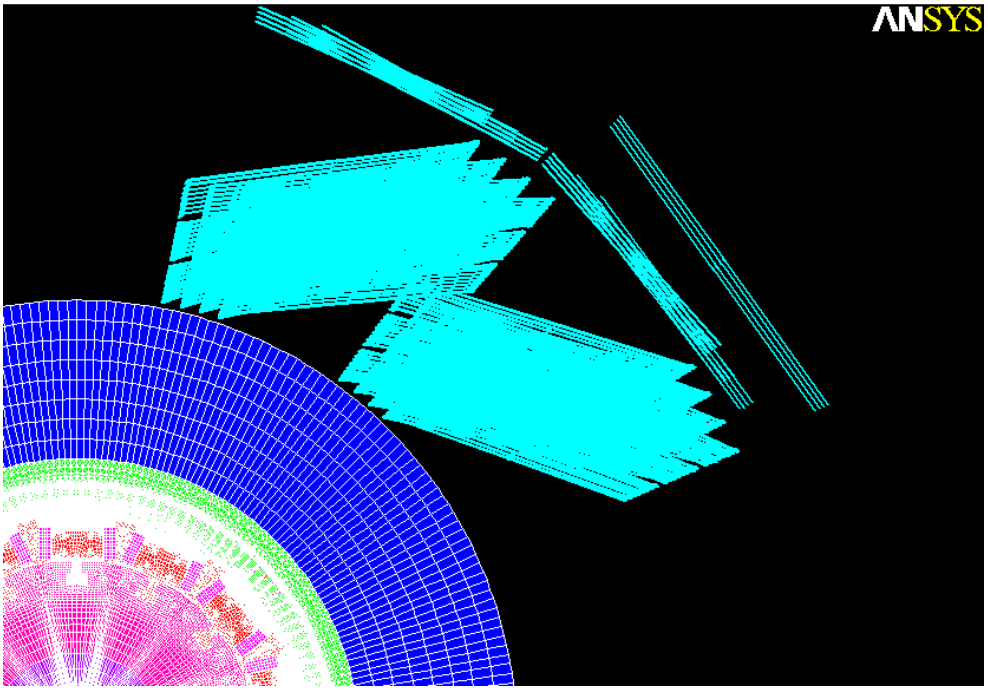


Figure 19: Figure of a GP2DET-model where two coil groups have been replaced, and the circuit elements for all the coils have been created and connected outside the stator.

5.2.3 Suggested modifications

The handling of transposition within a coil is at the moment built inside the macro `vscreatecoilcp`. Correspondingly is the transposition handling between coils within a group and between coil groups built inside `vscreatecoilgrcp` and `vscreatebranchcp`, respectively. The transposition sequence should be decoupled from these macros and written in an own sub-macro, such that the transposition procedure can be accessed without interfering with other parts of the code. This way the same flexibility would be obtained as for strand configurations in the solid modelling, and transposition options could easily be modified, added or removed.

Even though all couplings inside coils, between coils and between coil groups are being carried out automatically, there is still no automated procedure for closing the circuits. That is, the ends of a detailed winding must still be coupled. This can be connection to undetailed parts of the model, if only a small segment of the machine has been detailed, or in case the entire stator winding has been replaced, terminal connections should be applied. This has not yet been automated.

It is necessary to adjust macros in the current release version of GP2DET so that they can

handle detailed windings. For instance, when impressed current is applied to the GP2DET-models with massive stator conductors, current density is used. This would not make sense for detailed conductors, and must therefore be modified.
The coupling procedure should be extended to support connections across slot number 1 for detailing alternative 13 and 14.

5.3 Other issues

5.3.1 Challenges with APDL

General issues

The ANSYS scripting language APDL is a powerful tool for automation of processes in ANSYS. It has a vast number of commands and features, and is supported by good documentation. However, the large number of commands may also be a source to confusion, and it takes some time to aquire a good overview. Further, ANSYS has its own way putting bounds on processes and procedures. As mentioned, an area can not be modified before the mesh has been removed, which is only one example of which limitations one may encounter. Due to this, many of the macros for the winding detailing have been reworked numerous times, and many procedure improvements have been carried out throughout the development. In particular, this applies to the automation of the solid modelling, but also to the circuit coupling to some extent.

Lack of object orientation

APDL is a low level scripting language, and does not support programming techical advanced features, like object orientation, or private variables. Due to this, and the splitting of the winding detailing sequences into numerous macros, it has been necessary to take special precautions in regard of variables. Since all parameters are global - accessible for any part of the program at any time - it was needed to apply unique naming to them. If different variables were to be used in different macros, but with the same name, the risk of overwriting one or more of them is overhanging during interaction between these macros.

5.3.2 Current status and further development

At this point the procedure for detailed modelling of windings in a GP2DET-model is quite developed, and should support a big number of machines. The core of the procedure, and

the basic framework of model revision is present. Even though some additional features may be desired and recommended, these do not imply major changes of the current structure, as far as known.
Further development should focus on actual simulation, and implementation of different types of analyses. Substantial work remain in regard of calculation options, and this should be given priority in the following.

6 Conclusion

6.1 Conclusions

This chapter summarizes the most important achievements for the wining detailing procedure. These, and some suggestions for the further work, are listed in the following sections.

Solid modelling conclusions

- The procedure for automatic reworking of an existing GP2DET-model executed faultlessly for three tested machines. 17 detailing alternatives are supported.

- The procedure replaces stator conductors that are modelled as one massive block, with new, stranded stator conductors. Where current density earlier has been impressed on massive stator conductor blocks, the foundation has now been laid for simulation of how the stator current distributes itself over all the strands, and thus analyses of losses from circulating currents and eddy currents.

- Two arrangements of strands are supported; with or without Roebel transposition. All the strands are of the same size for both of them.

- Creation of new detailed stator conductors is time demanding when many conductors shall be replaced. The building of a strand from scratch consumes approximately 90 % of the time needed for creation of one stator conductor for some machines.

- Bar- and coil windings are supported. Amongst bar windings the modelling is limited to lap-windings. The procedure does not support machines with circuit connections on both sides.

Circuit coupling conclusions

- Circuit elements are automatically created for each strand. These elements are coupled to the strands in the FE-domain through common nodes. Coupling of the circuit elements are automatically carried out. Transposition can be applied up to three places: At one position within a coil, at one position between coils in a coil group, and at one position between coil groups within a branch. The two latter ones can be done blockwise.

- One type of transposition is supported; coupling of strands in the converse order both in row- and column direction within a stator conductor.

6.2 Recommandations

Solid modelling

Recommendations for the solid modelling:

- Implementation of a copying-procedure for entire stator conductors is recommended.

- Support for additional strand arrangement alternatives should be built in.

Circuit coupling

Recommendations for the circuit coupling

- A procedure for coupling the detailed windings to the rest of the stator conductors, or to the machine terminals, should be implemented. Macros in the existing release version of GP2DET should be modified such that this procedure executes automatically.

- It should be considered whether additional transposition alternatives are necessary, and whether or not different types of transposition is needed at differnet locations. If so, the transposition procedure should be decoupled from the circuit element creation sequence, and be written in dedicated macros.

- It should be evaluated whether or not coupling across slot number 1 is needed for detailing alternative 13 and 14.

6.3 Direction for further development

The main focus has been model revision, and making the GP2DET compatible with several types of machines. The focus in the following development should be directed on actual simulation and analyses of machines, like implementation of different calculation options.

References

[1] Ansys software documentation, *ANSYS 11 Documentation*. Ansys Inc, Canonsburg, Release 11, 2007. 2.3.1, 2.3.1

[2] Ludwig Bergmann und Clemens Schaefer, *Lehrbuch der Experimentalphysik Band 2, Elektromagnetismus*. de Guyter, Berlin, New York, 8. Auflage, 1999. 2.1.1

[3] Generator Design Guidelines, Di1359*: Stray Losses in Coil Windings of Three-Phase Machines*. Voith Siemens Hydro Power Generation, Heidenheim, GDG 0721-12, 2006. 2.2, 2.2.2, 2.2.4

[4] Generator Design Guidelines, Di1277*: Winding Examination Program*. Voith Siemens Hydro Power Generation, Heidenheim, GDG 0721-07, 2006. 2.2, 3.1.1

[5] National Electric Coil, www.national-electric-coil.com 800 King Avenue Columbus, Ohio 43212 (614) 488-1151 (document), 2.2.1, 2

[6] Heinrich Sequenz, *Die Wicklungen Elektrischer Maschinen, Erster Band*. Springer Verlag, Wien, 1950. 2.2, 2.2.1

[7] Jan Werner Seyler, *Finite element analysis of stator winding losses in large rotating electrical machines*. McMaster University, Hamilton, Ontario, 2004. (document), 1, 2.2.4

[8] Hugh D. Young and Roger A. Freedman, *University Physic, with modern physics*. Pearson, Addison Wesley, San Fransisco, 11th Edition, 2004. 2.1.2

[9] Voith Siemens Hydro, Department of Research and Development. Htg, GPM, Heidenheim, November, 2006 (document), 2.2.4, 6

Appendix

A Introduction to the GP2DET-tool

The General Purpose 2D Electromagnetical Tool (GP2DET) is a tool developed for simulation and calculation of electrical parameters in large hydro power machines. The GP2DET is based on the Finite Element (FE) method, and uses several different software programs for the different parts of the tool. The GP2DET is under development by Voith Siemens Hydro in Heidenheim.
It is practical to categorize the different parts of the GP2DET after the four main softwares. These will be discussed briefly in the following sections.

The POWERM-package

Voith Siemens Hydro posesses a package of software programs for analytical simulation and calculation of electrical machines in various physical diciplines. This package, called the POWERM-package, was developed within the company, and holds a great amount of information about various machines. A centralized database is used to store this information, and many machines that Voith Siemens Hydro have built, or planned to build, are included. A few of the POWERM-programs are therefore used as library, or information source, for the GP2DET. At the time being, the programs DI1102, DI1204 and DI1277 are used. From DI1102, a few parameters with geometrical magnitudes, as well as a few physical, are retrieved. The DI1204-program provides a great amount of geometrical magnitudes, as well as electrical values and some temperature parameters. Information normally contained in a winding diagrams for a stator winding is retrieved from the DI1277-program, such as coupling of the conductors.
The POWERM-programs are bound by certain standards and limitations, for instance in regard of which machine types they support. Thus, the GP2DET-tool inherits some of these, and can also not simulate any thinkable machine type.

Microsoft Excel

Microsoft Excel is the user interface of the GP2DET. All the sequences of a simulation project - from the gathering of model information, to the building of the geometry, to the meshing, to the initiation of calculations, to the postprocessing - are managed from here. This is the only program the user will be operating directly during normal use. An

important role for Microsoft Excel is to manage the many parameters for each project. Many of these are imported from Di1102 and Di1204 as recommended values, and must be manually verified before they are applied. Or if it is desireable to change for example the geomtery of the model, it has to be done here.

Solid Edge

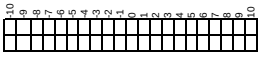
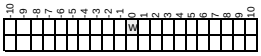
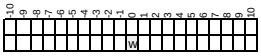
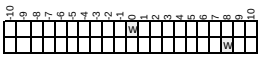
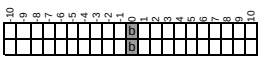
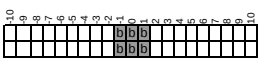
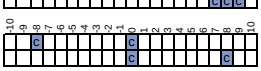
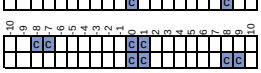
Solid Edge is used to create Cad-geometry for the Gp2det-models. Based on the many parameters in Excel, Solid Edge automatically generates a 2D Cad-model for the rotor and stator when executed from Excel. Jpeg-images of the result are generated and can be viewed in Excel before the model is exported to Ansys. This gives the opportunity to study the model and look for faults, prior to the meshing. Also, it is a quicker way of generating the models than if it should have been done in Ansys.

Ansys

Ansys is a FE-analysis software which can operate in several physic diciplines and provide multiple simulation choises. The Gp2det uses Ansys for the electromagnetic FE-analysis of the simulation projects.
Meshing of an Ansys-model can be executed from Excel. When doing this, all the machine parameters - material properties, rated machine values, geometrical magnitudes, and so on - are exported from Excel. Thereafter, these parameters and a Cad-model from Solid Edge are imported to Ansys, and the model is then meshed. After the meshing has been completed, calculations in Ansys can be initiated from Excel. Once the calculations are finished, the results can be read into Excel for evaluation.
To automize processes in Ansys there is a dedicated scripting language included; Ansys Parametric Design Language (APDL). This language is used to create macros that automate basically everything that is desireable to do in Ansys.

B List of detailing alternatives

b stator bar
c stator coil

ndex	description		winding scheme
1	No detailed conductor	Slot index Upper layer Lower layer	
2	One upper layer conductor	Slot index Upper layer Lower layer	
3	One lower layer conductor	Slot index Upper layer Lower layer	
4	One coil or connected bars	Slot index Upper layer Lower layer	
5	One slot	Slot index Upper layer Lower layer	
6	Two neighboring slots	Slot index Upper layer Lower layer	
7	Three neighboring slots	Slot index Upper layer Lower layer	
8	Two neighboring coils	Slot index Upper layer Lower layer	
9	Three neigboring coils	Slot index Upper layer Lower layer	
10	One slot + connected bars/coil sides	Slot index Upper layer Lower layer	
11	Two slots + connected bars/coil sides	Slot index Upper layer Lower layer	
12	Three slots + connected bars/coil side	Slot index Upper layer Lower layer	
13	One coil group	Slot index Upper layer Lower layer	
14	Two adjacent coil groups	Slot index Upper layer Lower layer	
15	One parallel circuit	Slot index Upper layer Lower layer	
16	One phase	Slot index Upper layer Lower layer	
17	Three phases	Slot index Upper layer Lower layer	

C List of macros

Created macros:

- `vsdetailedwdg`
- `vsdeterminepitch`
- `vscreatestrand`
- `vsremoveslotair`
- `vsinsertconductor`
- `vsinsertconductor_agen1`
- `vsinsertconductor_agen2`
- `vsinsertconductor_loop1`
- `vsinsertconductor_loop2`
- `vscreateslotair`
- `vsmeshslotair`
- `vsbuildcoilgrarrays`
- `vsgetcoilgrinfo`
- `vsfindneighcoilgroups`
- `vscreatecoilcp`
- `vscreatecoilce`
- `vscreatecoilgrcp`
- `vscreatecoilgrce`
- `vscreatebranchcp`
- `vscreatebranchce`
- `vscoilwdgconnect`

Existing macros that were modified:

- `vsstatorcircuitread`
- `vsnumbering`

Files read for information gathering:

- `DI1277FE.OUT`
- `Terminalconnections.txt`

D List of abbreviations in GP2DET

Legend of abbreviations for parameters:
`a_` : area number
`an_` : angle in degrees
`anr_` : angle in radians
`anm_` : mechanic angle
`ane_` : electric angle
`area_`: area (german: Flächeninhalt)
`b_` : boolean / flag
`c_` : code
`cos_` : cosine
`csa_` : cross sectional area
`d_` : distance. For distance between different components '__' separates them. E.g.: `d_a__b`
`d2dt_`: second derivative of subsequent quantity with respect to time
`ddt_` : derivative of subsequent quantity with respect to time
`dia_` : diameter
`gcd_` : greatest common divisor
`h_` : height
`i_` : current
`ind_` : inductance
`ii_` : index (starting with 1)
`j_` : current density
`jj_` : index (starting with 0)
`I1_` : stator current
`J1_` : stator current density
`kp_` : keypoint number
`l_` : line number
`len_` : length
`lcs_` : local coordinate system
`mom_` : Moment
`n_` : number of ...
`nd_` : node number
`pow_` : active power
`powa_`: apparent power

powr_: reactive power
r_ : radius
rat_ : ratio
res_ : resistance
ro_ : rotor
t_ : time
th_ : thickness
TC_ : temperature in degrees Centigrade
TK_ : temperature difference in Kelvin
v_ : volume number
w_ : width
x_ : x coordinate
xr_ : radial coordinate
y_ : y coordinate
z_ : z coordinate

Legend of abbreviations for ANSYS data structures:
ar_ : array
cm_ : components, cm_a_ indicates a component of areas, cm_e_ for elements, and so on
rsn_: real set number
mpn_: material number
etn_: element type number
s_ : character string
tb_ : table
p_ : path (convenient to be that small because the maximum size of the name is 8 chars.)

Extensions used at end of name or inside the name:
_i : inner
_c : center
_o : outer
_l : lower
_u : upper
_ro : rotor
_st : stator

`_e` : electrical
`_h` : hydraulic
`_m` : mechanical
`_abc` : for all three phases A, B, and C
`_abs` : absolute value (power values are typically given as absolut values)
`_act` : actual value
`_cart` : cartesian (coordinate system)
`_circ` : circuit
`_con` : connection
`_cond` : electrical conductivity
`_csa` : cross sectional area
`_cylin` : cylindrical (coordinate system)
`_inclin` : inclination
`_infl` : influence
`_ini` : initial value
`_LCs(i)` : element i of array of steady state load cases
`_LCt(i)` : element i of array of transient load cases
`_max` : maximal, maximum, amplitude
`_min` : minimal, minimum
`_pc` : value for one parallel circuit
`_pk` : peak value, amplitude (for sine: sqrt(2) x rms-value)
`_period` : periodic, periodicity
`_pos` : positioning, position
`_ref` : reference value (E.g. electric resistance at 20°C, wich is not the operating temperature)
`_rms` : root-mean-square value, effective value (for sine: 1/sqrt(2) x peak value)
`_rtd` : rated value
`_segm` : segment
`_wdg` : winding

`CAD` : Computer Aided Design, used to distinguish between imported CAD geometry and later on used FE geometry
`DE` : drive end side, bottom side of generator
`FE` : Finite Element, used to distinguish between imported CAD geometry and later on

used FE geometry
NDE : non drive end side, upper side of generator

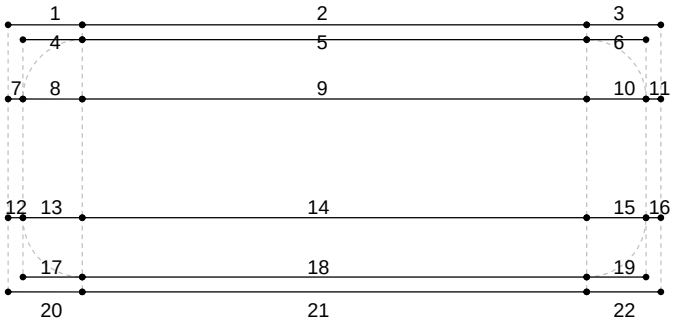
Legend for abbreviations in comments:
2D : two-dimensional
CS : coordinate system
BC : boundary conditions
CP : coupling
CE : constraint equation
DE : drive end side, bottom side of generator
DOF : Degree(s) of freedom
GPM : Generator Physics Modeling (project and project group in VSH R&D Basic Development, started in 2002)
NDE : non drive end side, upper side of generator

E Map of keypoints, lines and areas in a strand

Keypoints

For keypoints and lines, parameters are used to temporarily store the item numbers.
Keypoint naming syntax: **kp_strand_#** Example, keypoint 18: kp_strand_18

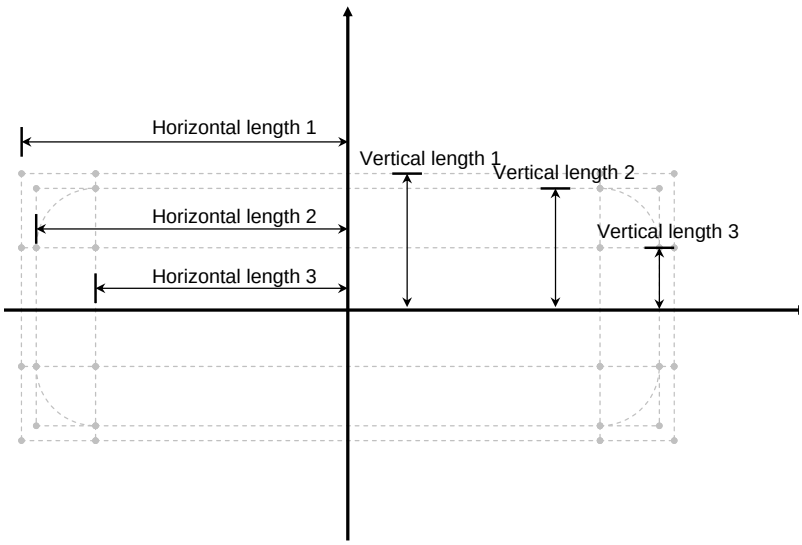
Horizontal lines



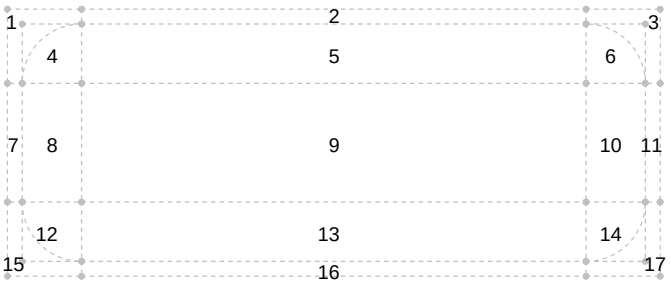
Vertical lines

Line naming syntax: **L_strand_#** Example, line 37: l_strand_37

Custom lengths for buildup



Area numbering



Area naming
Components are assigned to each area, instead of parameters.
Naming syntax: **cm_a_strand_part_#** Example, area 13: cm_a_strand_part_13