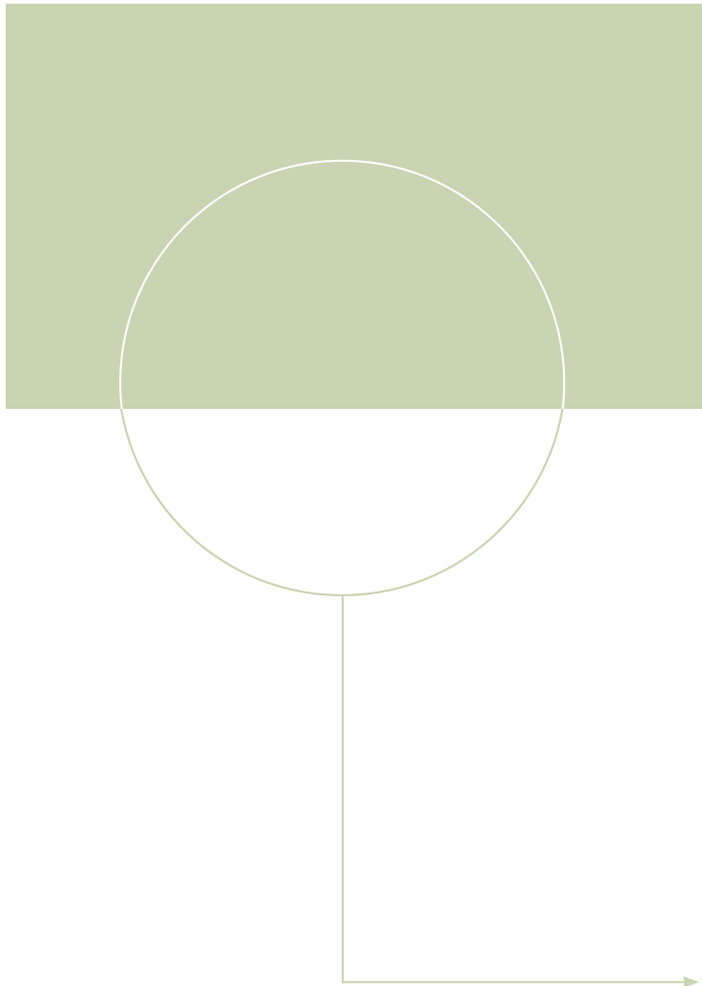


Master's thesis

2019

Magnus Borgersen

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Magnus Borgersen

Custom Design and Analysis of Modular High Voltage DC Generator

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Custom Design and Analysis of Modular High Voltage DC Generator

Magnus Borgersen

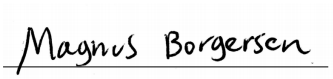
Master of Science in Mechanical Engineering
Submission date: June 2019
Supervisor: Terje Rølvåg
Co-supervisor: Pål Keim Olsen

Norwegian University of Science and Technology
Department of Mechanical and Industrial Engineering

Preface

This thesis concludes a five years master's degree in mechanical engineering, with a specialization in Product development and Materials engineering. The thesis has been carried out throughout the spring of 2019, and is written on behalf of the Department of Mechanical and Industrial Engineering at the Norwegian University of Science and Technology (NTNU).

First, I would like to thank my supervisor, Professor Terje Rølvåg for his help defining an interesting master's thesis, and all the guidance and feedback throughout the semester. I would also like to thank Pål Keim Olsen at the Department of Electrical Power Engineering for his valuable help and commitment during this project. Furthermore, I would like to thank Hilde Nordvik and Marta Karoline Husebø for the cooperation and their input during the project.



Magnus Borgersen,
Trondheim, June 2019

Abstract

This thesis is written as part of the first phase for the ModHVDC generator project, which is planned to constitute an EC funded project with stakeholders from several countries and institutions. The ModHVDC generator technology aims to match the future power grids that will operate with direct current (DC) for long-distance energy transportation. The work is a contribution towards an iteration-based design process of the generator core, by creating a simple Engineer-to-Order system within Siemens NX.

Two generator concepts have been chosen, based on the requirements from the ModHVDC project proposal and a study of traditional electrical machines. Furthermore, the selected concepts have been parametrically modelled and presented in a tailor-made user menu, based on Product Template Studio within Siemens NX. Automatic generation of manufacturing drawings has been implemented to the product template, by creating parametrically controlled drawing templates for each generator component. The drawings can be exported directly to PDF from the product template menu, using an executable program that is written based on the NX Open API. Modal analysis templates have also been created for both concepts, making it possible to run modal analyses on the stator fastening mechanism without having to manually pre-process the FE-model. To deal with the automatic pre-processing, a NX Open script has been written for each concept, which applies the desired materials and mesh parameters based on simple user input in the template menu. Finally, the system has been validated by testing several different generator configurations. The methods used in this solution will be presented in the thesis.

Sammendrag

Denne masteroppgaven er skrevet som en del av den første fasen til ModHVDC-generator prosjektet, som er planlagt å utgjøre et EU-finansiert prosjekt med deltakere fra flere ulike land og institusjoner. Målet med ModHVDC teknologien er å matche det fremtidige strømnettet som skal baseres på likestrøm for langdistanse energitransport. Arbeidet som ligger til grunn for denne oppgaven er et bidrag til en iterasjonsbasert designprosess av generatorkjernen, der et enkelt Engineer-to-Order-system har blitt utviklet i Siemens NX.

To generatorkonsept har blitt valgt, basert på kravene som foreligger i prosjektforslaget til ModHVDC og en studie av tradisjonelle elektriske maskiner. De to konseptene har deretter blitt parametrisk modellert og presentert i en skreddersydd brukermeny, basert på verktøyet ”Product Template Studio” i Siemens NX. En løsning for automatisk genering av produksjonsunderlag har også blitt implementert, basert på parametriske tegnings-maler som har blitt laget for hver generatorkomponent. Tegningene kan eksporteres direkte fra brukermenyen til PDF-format ved hjelp av et program som er skrevet basert på NX sitt API, NX Open. Videre har også maler for automatiske modalanalyser blitt laget og implementert for begge konsepter. Disse gjør det mulig å utføre modalanalyser på innfestningsmekanismen til statoren, uten behov for manuell pre-prosessering av modellen. For å håndtere den automatiske pre-prosesseringsprosessen har et NX Open-script blitt skrevet for hvert av de to konseptene, som påsetter materialer og mesh-egenskaper for de ulike komponentene, basert på enkel input i brukermenyen.
Til slutt har systemet blitt validert ved å teste en rekke ulike generatorkonfigurasjoner.
Metodene som er brukt i denne løsningen presenteres i oppgaven.

Table of Contents

List of Tables

List of Figures

Abbreviations

API	=	Application Programming Interface
CAD	=	Computer-Aided Design
CAE	=	Computer-Aided Engineering
CAM	=	Computer-Aided Manufacturing
CAPEX	=	Capital Expenditure
FEA	=	Finite Element Analysis
FEM	=	Finite Element Method
HVDC	=	High Voltage Direct Current
IEL	=	Department of Electric Power Engineering
MTP	=	Department of Mechanical and Industrial Engineering
PM	=	Permanent Magnet
PTS	=	Product Template Studio
VB	=	Visual Basic

Chapter 1

Introduction

1.1 Background

This thesis is a continuation of my project work on parametric design of the ModHVDC generator. The topic for this thesis was worked out in cooperation with my supervisor Terje Rølvåg (MTP, NTNU) and co-supervisor Pål Keim Olsen (IEL, NTNU).

For the proposed future electric system with super grids transporting the electricity between EU countries, the electricity is stored and transformed at HVDC [1]. For the power grid it is proposed that a new type of compact, high voltage electric machine called the Modular HVDC generator should be used. The ModHVDC project proposal contains a detailed plan for a potential EC project, which will be a collaboration between three universities and six companies, all with their special field of competence.

Since the official ModHVDC project has not been started yet, this thesis is a part of a small pre-project which is a collaboration between the Department of Electric Power Engineering (IEL) and the Department of Mechanical and Industrial Engineering (MTP) at NTNU. In total there are three master students working on the ModHVDC concept, one from IEL and two from MTP. The main focus for this pre-project is thermal, mechanical and electrostatic design, as well as development of an electrical insulation system.

1.2 Objectives

The following objectives has been developed in cooperation with professor Terje Rølvåg and will mainly be the focus during this thesis.

	Main Objectives
O1	Identify the best generator concept for offshore windmills based on requirements from the ModHVDC project proposal.
O2	Make a PTS for custom made generator design based on outputs from task 1.
O3	Embed (if possible) automatic coupled thermal – structural FEA in the PTS.
O4	Implement (if possible) automatic generation of manufacturing drawings in the PTS.

Table 1.1: Main objectives

1.3 Limitations

Several challenges and problems have been met during this project, which has brought some limitations to the outcome.
The field of electrical machines was a completely new subject for both the students representing the MTP faculty. Spending time, studying the electromagnetic principles and generator design aspects was therefore necessary at project start. Since there was only one student with proper knowledge on the field, getting successful results and load cases from electromagnetic analyses proved to be difficult.

In addition, the fact that NX does not have any electromagnetic solvers makes it challenging to predict the impacts of design changes, with regards to the force distribution and magnitudes.

1.4 Vision

The goal for this thesis is to contribute towards a simple and easy to use Engineer to Order system for the ModHVDC generator concept, based on Product Template Studio (PTS) within Siemens NX. The system should be able to generate a detailed 3D model of the generator concept based on dimensional input from the users, perform automatic modal analyses of the stator and fastening mechanism, and automatically generate production drawings, which can be exported to PDF from the user menu created in PTS.

Since the IEL faculty does not use Siemens NX, the system should be applicable for people without proper NX experience.

1.5 Methodology

Due to the limited knowledge on electric machines, a study of previous work and the core theory concerning PM machines with concentrated windings was considered important, in order to pick a suitable generator concept. Especially, literature on vibrational load cases

for the desired machine architecture were of particular interest during this study.
Furthermore, the NX Open API and the programming language Visual Basic was a completely new field at project start. A study of the NX Open guide [2] i addition to a lot of testing in Visual Studio was necessary to get started with the coding. All the code developed during this thesis has been done through numerous iterations of planning, testing and evaluation of the code, until a satisfactory result has been achieved.
The analysis part of Product Template Studio was also an unknown field at project start, which had to be explored by a trial-and-error approach, due to the lack of available documentation.

1.6 Disposition

After the introduction, **Chapter 2** will present the theoretical background of this project, describing the concepts, software and technology used throughout this thesis.

Chapter 3 presents the superior workflow and software implementation of the ModHVDC project during the spring 2019.

Chapter 4 will present the development process of the two ModHVDC generator concepts that has been chosen for this thesis.

Following the introduction of the ModHVDC generator concepts, **Chapter 5** describes the modelling process of the concepts and how they are implemented in PTS.

Chapter 6 describes the approach for automatic generation of manufacturing drawings and how they are exported directly from PTS.

In **Chapter 7**, the finite element analysis setup is introduced. That includes a description of how the relevant bodies are linked into separate analysis part files, how the finite element models are defined, and finally, how the finite element analyses are set up, using Nastran SOL 103.

Chapter 8 presents the product template analysis setup, the challenges using analysis templates for flexible models, and how these challenges have been solved.

Chapter 9 contains a validation of the system based on PTS. The system will be evaluated by testing the parametric model, the generation of drawings and analysis templates for several different generator configurations.

In **Chapter 10**, all the work, results and conclusions from the system development will be discussed.

At the end, **Chapter 11** contains a conclusion for this thesis and my thoughts on how the work presented in this thesis can be further improved and developed.

Chapter 2

Theory

2.1 Electrical machines

This section will briefly present the core theory for Electrical Machines with concentrated windings and the ModHVDC technology. Basic theory describing how electricity is generated with Magnets and Conductors will not be provided.

2.1.1 Generator Terminology

In this thesis, a generator layout is taken from a low speed PM Generator prototype, with concentrated windings, designed by Øystein Krøvel [3]. The generator consists of a stator with 60 single-layer concentrated windings, and a rotor with 116 poles [4]. Figure 2.1 shows a cross section of the prototype layout with its main components. The color coded slots A, B and C, represents the three coil phases, where the windings in each phase are connected in series. Figure 2.2 shows the magnet and coil configuration in more detail. As shown, the magnets are configured with alternating polarity. The stator core is made of laminated iron plates.

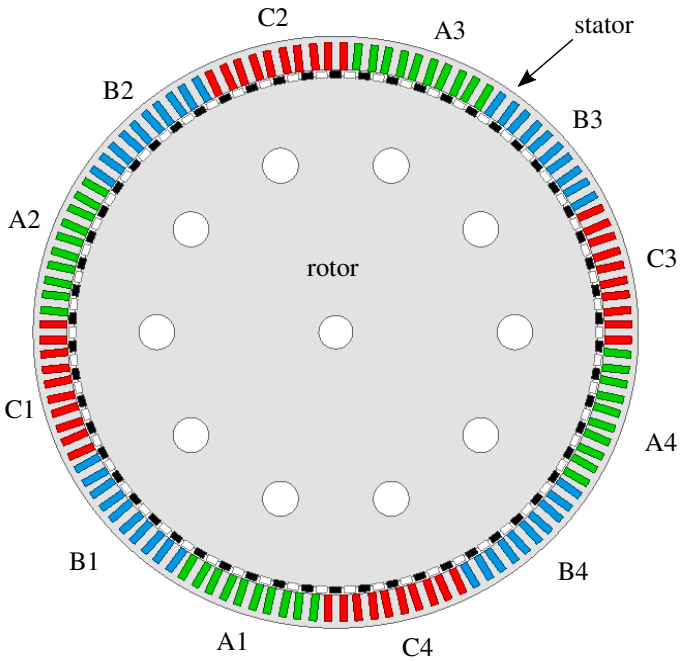


Figure 2.1: Cross section of the generator showing the major components

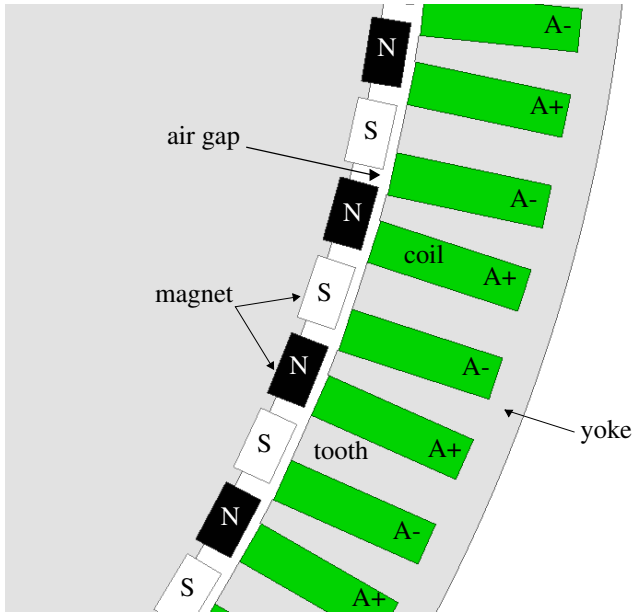


Figure 2.2: Detailed cross section showing the configuration of magnets and coils

2.1.2 The ModHVDC Technology

ModHVDC's main concept is to combine the functions of a generator and a high voltage transformer in one unit by stacking the converter voltages instead of the generator coil group voltages.

For traditional electric machines, a generator consists of a stator (stationary component) and a rotor (rotating component). What makes the ModHVDC concept unique is that the stator core is divided into modules (MG1-MG4 on Figure 2.1, taken from [5]), where each module is connected to the DC potential levels of the AC-DC converters (1M-4M on Figure 2.1). The modularization and galvanic connection separates the AC and DC voltage stresses inside the machine, which allows for optimal distribution of voltage stresses. Better distribution of voltage stresses means that less electric insulation is needed. This will increase the power per weight ratio compared to state-of-the-art conventional machines at the same voltage rating.

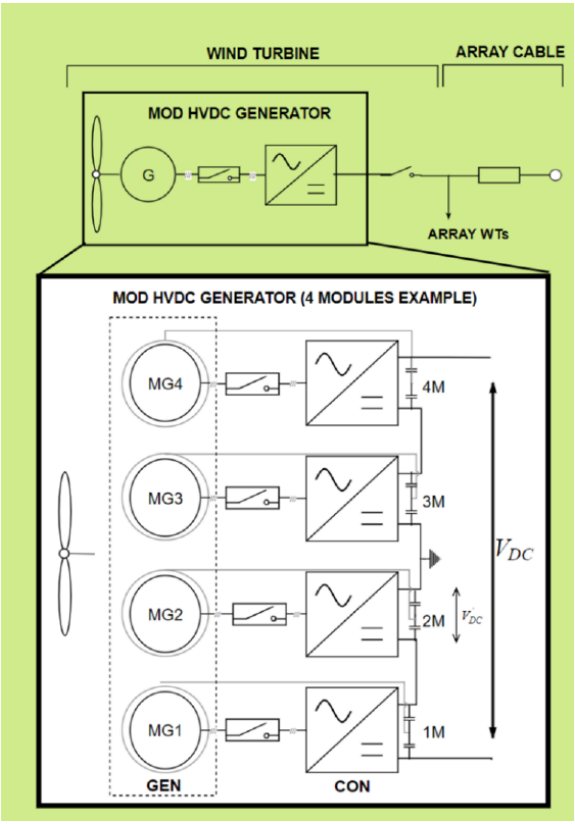


Figure 2.3: Simplified electric conversion system concept of ModHVDC
Source: [5]

Figure 2.4 illustrates the proposed ModHVDC stator design, where the stator is divided

into 4 segments, with three winding phases on each segment. Each segment is then connected to a converter (1M-4M), as described in the section above.

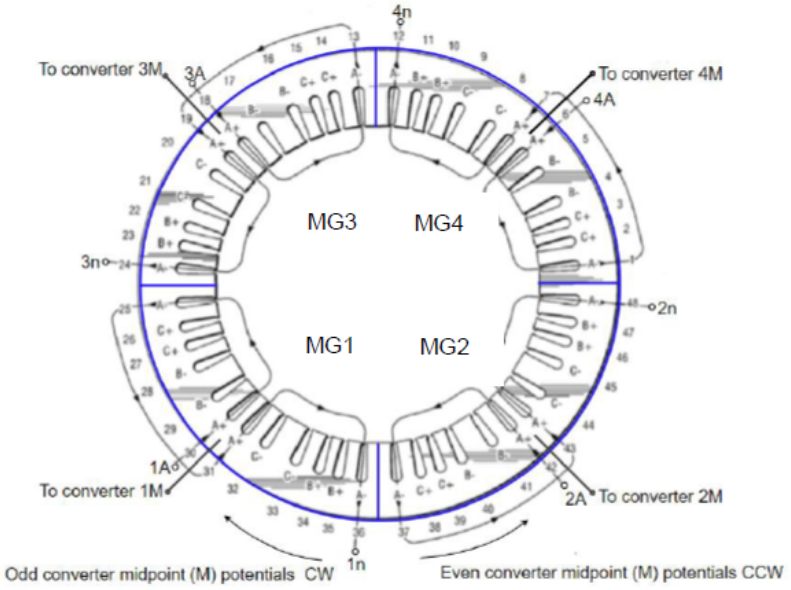


Figure 2.4: Conventional stator divided into 4 segments
Source: [5]

Another direct benefit of the ModHVDC technology is that today's use of low voltage/high current to high voltage/low current transformers will be eliminated, as these stages now will be performed in the wind turbine's nacelle. In the conventional systems, these transformer steps introduce efficiency losses up to 2%. At a system scale, the construction, management and maintenance of the transformers introduces a large capital investment when planning new energy plants. Eliminating these stages will therefore reduce the CAPEX [1] for ModHVDC power plants.

At the time of writing, hydro power and wind power are the two main renewable energy sources. The ModHVDC technology will target wind power due to capacity growth limitations in hydropower. ModHVDC technology is believed to open up for new onshore wind sites with fewer turbines distanced from residential areas. The technology is believed to enhance social acceptance of onshore wind power because of its reduced environmental footprint and reduced electricity cost.

The main advantages of the ModHVDC technology can be summarized to be:

- Theoretical calculations indicate a **75-90% reduction of equivalent slot insulation thickness** in the stator compared to state-of-the-art conventional high voltage machines.

[1] CAPEX: Capital expenditure. Defined as the money a company spends to buy, maintain or improve its fixed assets, such as buildings, vehicles, equipment and land [6].

- Same power per weight as state-of-the-art machines, at 4-10 times the voltage output.
- Modularity of the ModHVDC stator increases manufacturability and ease of assembly.
- Decrease in LCOE for onshore EU wind by implementing ModHVDC technology, making onshore wind energy competitive with fossil fuel sources, as well as other renewable energy sources.

2.1.3 Load Cases in PM Machines with Concentrated Windings

The PM generators with concentrated windings can experience a vibration level that is considerably high compared to traditional machines [4]. These vibrations does mainly occur due to the presence of low harmonic orders in the radial magnetic force distribution, acting on the stator teeth. Especially, the low modes of vibration are caused by this rotating magnetic force distribution. In addition, the dimensional characteristics of machines with a high number of poles must be considered with respect to vibration. The high number of poles makes the stator diameter large, its yoke thin and its length relatively short. This usually causes low-speed PM generators to have a moderate mechanical stiffness, and critical magnetic vibration levels.

It is the difference in slot and pole number that creates the rotating force waves that can cause unsymmetrical forces and excite vibrations in the structure. [3]. The force distribution form is decided by the pole and slot number, while the amplitude and harmonic content in the force waves are decided by the magnet width, air gap, slot width, etc.

Figure 2.5 shows the force wave distribution for machines with 6 different pole and slot combinations. The forces are found from no load flux densities, by integrating the force density on each tooth. In order to be comparable, the forces are normalized.

Radial Forces

Maxwell stress tensor is a well used method to calculate the magnetic forces in different types of electric machines. According to Maxwell stress tensor, the radial and tangential components of the force density in the air gap of the electrical machine can be expressed as [4, 7]

$$f_r = \frac{1}{2\mu_o}(B_r^2 - B_t^2) \tag{2.1}$$

$$f_t = \frac{1}{2\mu_o}(B_r B_t) \tag{2.2}$$

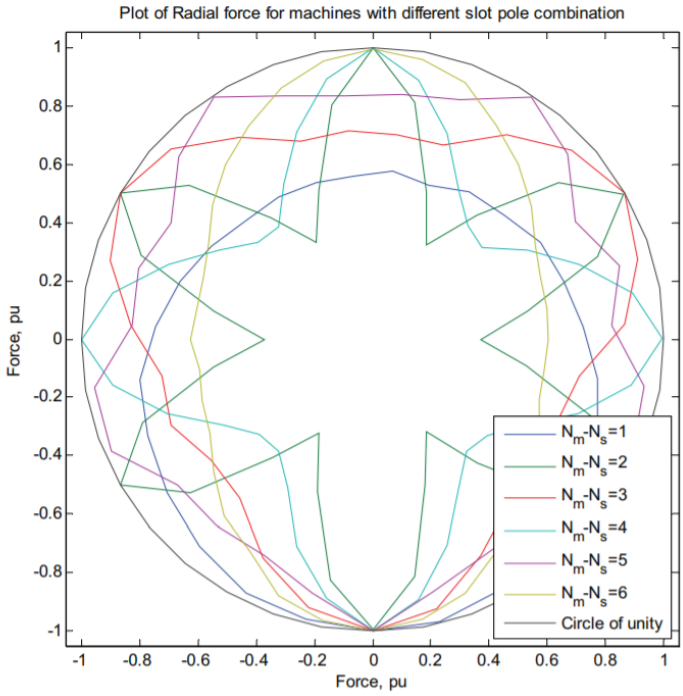


Figure 2.5: Fundamental radial forces for different slot pole combination
Source: [3]

where f_r and f_t are the radial and tangential components of the force density. B_r and B_t are the radial and tangential components of the magnetic flux density. By using Equations 2.1 and 2.2, the total force acting on the stator tooth is found to be [4]

$$F_{tooth} = L_s \int_{tooth\_line} f_r dl, \tag{2.3}$$

where L_s is the stator stack length. The forces acting on the stator teeth are then calculated by using the line integral over each tooth.

Figure 2.6 shows the typical waveforms that occurs when the force acting on each tooth is plotted for the whole stator. Comparing the force distribution curve in figure 2.6 with the curve for the $N_m - N_s = 4$ configuration in figure 2.5, we can see that the same sinusoidal force distribution is present [4].

The radial forces are causing deformations in the stator. The amplitude of the deformations is inversely proportional to m^4, where m is the mode number. This implies that the lowest vibration modes is the most critical. Also, if the lowest modes of vibration is caused by magnetic forces, the natural frequencies can be in the range of the rotor speed, which may increase the risk of resonance [4].

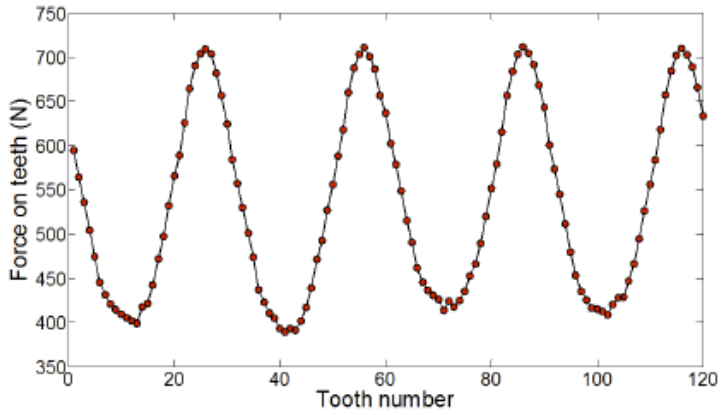


Figure 2.6: Simulated force distribution on the teeth for a 120-slot/116-pole generator
Source: [4]

2.1.4 Mechanical Design Fundamentals in Electrical Machines

Mechanical design is an important factor when an electrical machine is developed. In general, it is three main concerns that should be investigated in the mechanical design analysis [8]:

1. Mechanical structures
2. Field of stress and material strength
3. Modal analysis for vibration and noise

The two first points are usually the least critical and can be readily satisfied through empirical design. However, the third point requires great attention, as already introduced in chapter 2.1.3. The modal analysis is used to calculate the generators resonance frequency in operation, and it is important to make sure that the operating frequency is outside the critical resonance frequency area.

By comparing the eigenfrequencies from the modal analysis with the electromagnetic frequency, the mechanical design can be optimized in order to ensure that resonance frequency will be avoided.

For the ModHVDC project, the modal analyses will be of extra interest, due to the fact that a solid stator is divided into segments. Dividing a solid structure into segments will affect the geometry and stiffness of the overall structure, thus also its eigenvalues and mode shapes.

2.2 Engineer-to-Order

Engineer to order (ETO) is a manufacturing process where the products are designed, engineered and produced after an order has been placed by a customer. The products are engineered to meet required specifications from the customer. To ensure that the given specifications are met, representatives from the customer company usually engage with the manufacturing team trough the whole development process [9].

For companies using an ETO approach, it is important to have a well structured platform for information exchange between all stakeholders involved with the product. Product information and specifications are often moving constantly back and fourth between the ETO company and the customers. If the information exchange is poorly managed, the risk of bad communication, confusion and misassumptions between the stakeholders will increase significantly.

2.3 Siemens NX

Siemens NX is an advanced and powerful software solution, providing tools for solving CAD (Computer Aided Design), CAM (Computer Aided Manufacturing) and CAE (Computer Aided Engineering) problems [10].

2.3.1 Parametric Modelling

Parametric is a term used to describe a dimension's ability to change the shape of model geometry as soon as the dimension value is modified. [11]

The main key in parametric modelling is to obtain a robust model, where all the different modelling features acts properly relative to each other, when a parameter has been changed. By introducing systematic naming for all the changeable feature parameters, they can easily be recognized and updated/changed in the expressions menu. Parametric models are often controlled by mathematical or logic rules/ relations, which can also be stored as expressions and used as input in different modelling features. In addition to well defined names and rules, the use of "geometric constraints" in sketches and "assembly constraints" in assemblies, is trivial when parametric models are being made.

Creating a fully parametric model is usually a bit more time demanding compared to a non-robust model, but the parametric models ability to adapt for sudden design changes and reusability can save a lot of time in the long run. Due to its reusability and adaptability, parametric models can for example bring a great amount of value to Lean Product Development (LPD) projects, which has been a popular development approach in recent years.

2.3.2 Product Template Studio

Product Template Studio (PTS) is an integrated tool in NX, where a parametric model can be presented with a graphical user interface (GUI). PTS is a powerful tool that enables user friendly and reusable templates for parametric models and simulations, which enhance the efficiency in product development processes.

Modelling PTS

Parametric models and assemblies tend to end up quite complex, containing a lot of different, rules, naming and other metadata. Since the design methodology between engineers varies, it can be hard for other people than the original designer to configure a parametric model, due to its high level of complexity. Using PTS to create a user-friendly product template, makes the model editable, even for an inexperienced CAD user.

Simulation PTS

Product Template Studio does also have the possibility to create templates for simulation modules in NX. After a simulation has been set up and run, the .sim file will appear as a related CAE part in PTS author. A user-friendly GUI can then be wrapped around the simulation set up. A simulation PTS makes it possible to perform FE-analyses immediately after a geometry change has been done in the Modelling PTS, without having to manually change application and re-mesh the geometry. The author can decide if the users should have the ability to change the material, mesh and load properties from the template or not, making it possible to create a fully automated and rule-based template that can be used by people without proper experience with Finite Element Analysis (FEA).

2.3.3 NX Open

NX Open is a collection of several APIs that enables NX users to create custom applications through an open architecture, using the most popular programming languages C/C++, Java, Visual Basic, Python and C# [12]. The custom applications can for example be used to automate complex and repetitive tasks, export model information to external files, integrate third party applications with NX and to customize the NX user interface.

The NX Open scripts can be run from the Journal functionality in NX. The software will automatically detect which programming language that is being used and run the code. The Journal function can also record commands performed by the user and translate it to NX Open code for all the supported languages. Recording a Journal can be a good way to get started with a NX Open script. However, the recorded code usually contains a lot of unnecessary lines that can be deleted without affecting the program, thus making the code more complex and difficult to interpret. Before getting started with a journal file, it can be smart to check out NXJournaling.com [13]. The web page contains a few guides on

how to get started with NX journaling, in addition to a great forum where a lot of example programs can be found.

Visual Basic API

The Visual Basic (VB) API is best documented of all the APIs in NX Open. Siemens has published a ”Getting Started With NX Open”[2] guide for the VB API, which is highly recommended to read before getting started with NX Open. For people using other programming languages, it can still be smart to check out the guide in order to get used to the different NX programming commands.

2.4 Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) that can be used to edit, debug or build code and then publish an application [14]. In addition to the standard editor and debugger, Visual Studio includes compilers, code completions tools, graphic designers, and many other features to ease the software development process.

It is recommended to use Visual Studio 2015 when creating a VB script for NX Open [2]. The NX system folders contains VB templates that can be integrated in Visual Studio, making it possible to run the code in debug mode and to create executable programs (.exe) that can be run outside an ongoing NX session. Creating an executable program makes it possible to run automatic processes in batch mode without using the NX GUI. The VB templates contains a class library for NX Open, which makes coding in Visual Studio preferable, as the program can help you by suggesting alternatives, completing words, correct mistakes, show documentation, etc.

2.5 Manufacturing Drawings

In mechanical engineering, technical production drawings are key in order to communicate component details to all the people involved in a development and production process. It is highly important that the drawings contain enough details and at the same time are easily interpretable, in order to avoid costly and time-consuming misunderstandings. This section will present the ”best practice” rules for creating production drawings. The content and figures in the following chapter is mainly retrieved from the books ”Tegning og dokumentasjon” (Bergeland, Hansen & Herø) [15] and ”Machine Drawing” (Narayana, Kannaiah & Reddy) [16].

2.5.1 Sheet Layout

The sheet layout is the first thing that must be considered, when creating a manufacturing drawing. This section will briefly explain how the sheet layout should be set up.

Sheet Sizes

There are five main sizes that are used for manufacturing sheets. These are:

Type	Dimensions (mm)
A0	841 x 1189
A1	594 x 841
A2	420 x 594
A3	297 x 420
A4	210 x 297

Table 2.1: Sheet dimensions

In this thesis, all sheets will have the A3 size format.

Title Block

The title block should be placed in the lower right-hand corner of the sheet and have a maximum length of 170 mm. It usually contains the following information:

- Title of the drawing
- Scale
- Symbol, denoting the projection method
- Company name
- Initials of the staff drawn, checked and approved.

Borders

Borders should be provided on all drawing sheets. For A2, A3 and A4 sheets, the distance from the sheet edges to the borders that marks the drawing space should be 10 mm. The title block should lie within these borders. Figure D shows a drawing of the generator rotor, encircled by the sheet borders.

2.5.2 Scaling

Scale is the ratio of the dimension of an element as represented in the drawing, relative to the real dimension of the element itself. If possible, it is desirable to project a component in a full-size drawing, to represent its true shapes and sizes. If full scale is not practicable, the largest possible scale should be used. For small objects, it is often desired to to use enlarging scales. The standard scales used in technical drawings are presented in Table 2.2. The scale is dependent on the sheet and component dimensions.

Category	**Standard scales**		
Enlarged Scales	50:1	20:1	10:1
	5:1	2:1	
Full Size			1:1
Reduced Size	1:2	1:5	1:10
	1:20	1:50	1:100
	1:200	1:500	1: 1000

Table 2.2: Standard scales

If all views in a drawing are made with the same scale, the scale value should be placed in the title block. However, if it is necessary to use more than one scale, the main scale should be shown in the title block and the other scales should be placed near the view.

2.5.3 Dimensions

General Principles

A drawing must include information regarding the size descriptions of a component. This information is provided through the distances between surfaces, location of holes, cornering radii etc. Dimensioning can be challenging, especially for more complex components containing a lot of details. The book ”Machine Drawings” [16] contains a 8-point list, describing the general rules for dimensioning:

1. As far as possible, dimensions should be placed outside the view.
2. Dimensions should be taken from visible outlines rather than from hidden lines.
3. Dimensioning to a centre line should be avoided except when the center line passes through the center of a hole.
4. Each feature should be dimensioned only once on a drawing.
5. Dimensions should be placed on the view or section that relates most clearly to corresponding feature.
6. Each drawing should use the same unit for all dimensions, but without showing the unit symbol.

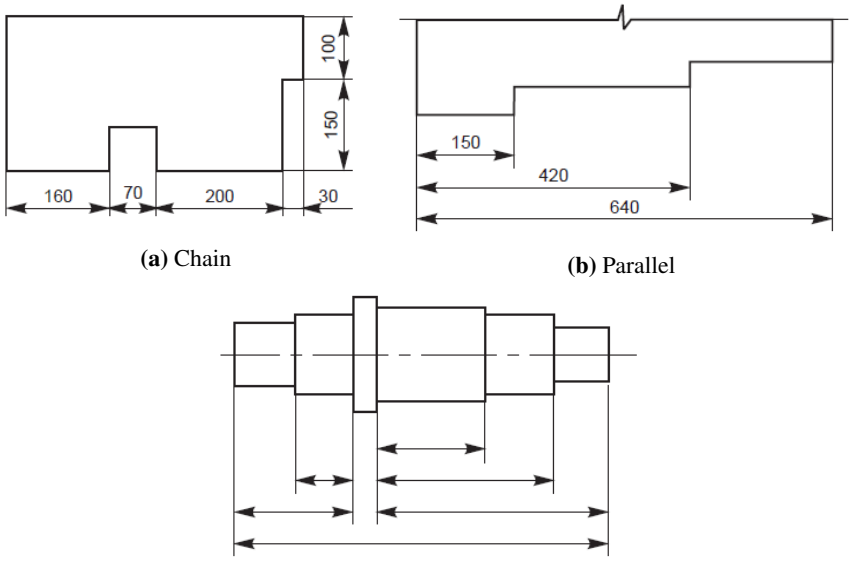
7. No more dimensions than absolutely necessary should be provided on a drawing.
8. No features of a part should be defined more than once.

The dimensions should be presented in characters of sufficient size to ensure that they are fully readable. They should also be placed in such a way that they are not crossed or separated by any other line in the drawing.

Distance Methods

In this thesis, the dimensions are projected according to the *aligned system*. The aligned system requires that the dimensions should be placed parallel to their dimension lines, preferably above, near the middle and clear off the dimension line, as shown in Figure 2.7a and 2.7b.

There are different ways of arraigning the dimensions on a component. Figure 2.7 shows the three most common arrangements. *Chain Dimensioning* (Fig.2.7a) is only used where the possible accumulation of tolerances does not endanger the functional requirement of the part. In *Parallel Dimensioning* (Fig.2.7b), several dimension lines, parallel to one another are present. This method is mainly used where the dimensions included have a common datum feature. A combination of the two methods above is called *Combined Dimensioning* (Fig.2.7c).



(a) Chain

(b) Parallel

(c) Combined

Figure 2.7: Arrangement of dimensions
Source: [16]

Diameters and Radii

There are different ways of dimensioning cylindrical objects and holes, but the value should always be preceded by ⌀. The general rule is that they should be dimensioned in the most appropriate way. It is most common that small holes are marked with an arrow as shown in the left circles in Figure 2.8. For larger holes, projection lines (Upper right circle in Fig.2.8) and arrows pointing directly towards the circle contour is used.

For the radius dimensions, an arrow and the symbol R will always be present (Fig.2.8). If the size of the radius can be derived from other dimensions, it may only be indicated by a radius arrow and the symbol R.

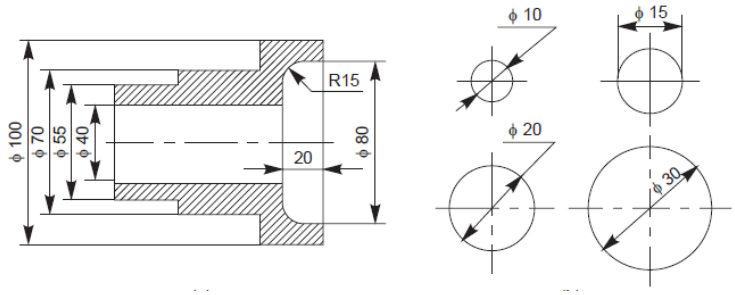


Figure 2.8: Dimensioning of diameters and radii
Source: [16]

Equal Dimensions

For symmetrical objects with several equal dimensions, it is not necessary to specify each of the equal dimensions. Figure 2.9 shows such a symmetric component with six equal holes and corner radii. The number of equal dimensions can be presented before the dimension type and size, as shown in the figure.

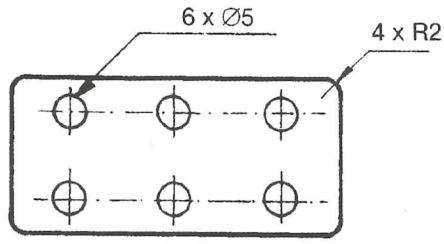


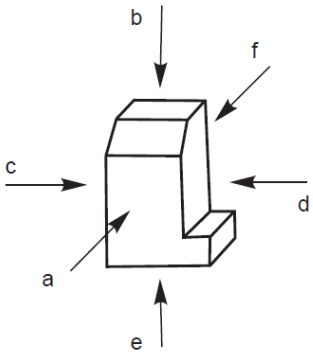
Figure 2.9: Symmetrical component with several equal dimensions
Source: [15]

2.5.4 Projection Methods

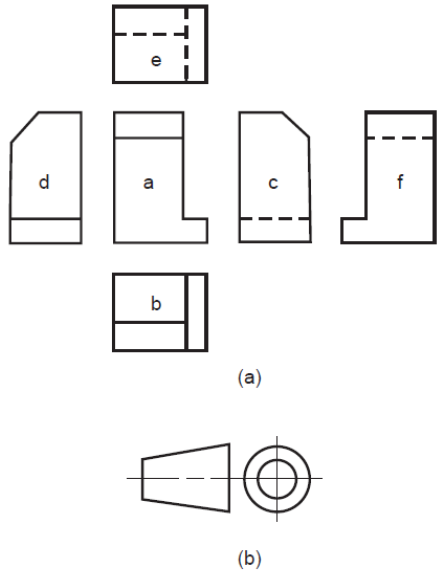
A projection is known as a representation of a three-dimensional object in a two dimensional plane. The projections of an object should convey all three dimensions, along with other details on a sheet of paper. There are several projection methods being used worldwide, but the *First Angle* method is one of the most preferred by different standards, and will be presented and used in this thesis.

First Angle Projection

In first angle projection, the object is imagined to be positioned in the first quadrant. Figure 2.10b shows the overview of how a component is projected in first angle. The three most commonly used views in manufacturing drawings are ”view from the front” (a), ”view from above” (e) and ”view from the left” (c). A general rule is that a drawing should not contain more views than absolutely necessary. However, extra views can sometimes be used in order to provide a better explanation of the object. The figure marked as (b) in Figure 2.10b is the symbol implying that the first angle projection method has been used. This symbol should be found within the title block.



(a) Designation of the views



(b) Relative positions of six views in first angle projection

Figure 2.10: First angle projection
Source: [16]

When positioning the objects, it is important that the object is imagined to be positioned such that its principal surfaces are parallel to the planes of projection. The reason for this

is that a line or face on an object only will show its true size when it is parallel to the plane of projection.

Sectional Views

Standard orthographic views do only present the external features of the objects. However, some objects have complicated interior details that cannot be truly represented in such a standard orthographic view. This may be solved by presenting a sectional view of the component. Figure 2.11 shows circular component containing several holes. To provide a more detailed description of these holes, the component is cut in half (Fig.2.11a). In the section view, the oblique lines represents solid material that is cut in half. The cutting plane is represented by its trace in Figure 2.11c and the direction of sight in the sectional view is shown by the arrows marked x.

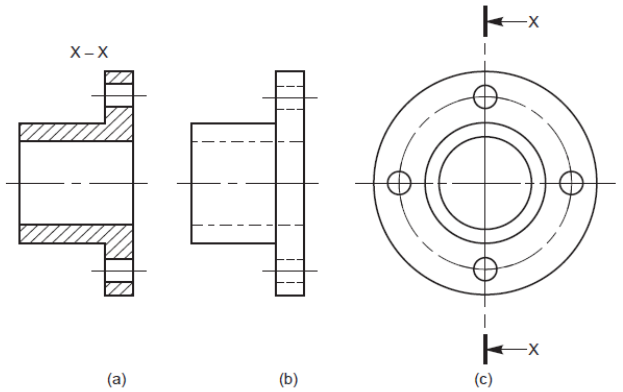


Figure 2.11: Sectional view
Source: [16]

2.6 Finite Element Analysis

The Siemens NX CAE application contains a lot of different finite element tools for solving most structural problems, with both linear and nonlinear solvers. The solver used in this thesis is *Nastran SOL 103 - Real Eigenvalues*, for modal analysis. The solver will be introduced below.

2.6.1 Pre-Processing

Preparing a component or assembly for analysis purposes is completely necessary and can often turn out to be a demanding job. The process of deleting unnecessary features,

selecting the right element types, dimensions and other mesh-features has a vital influence on the analysis results as well as the calculational time. The pre-processing tools used in this thesis will be introduced below.

3D-Elements

Three-dimensional elements, also known as ”solid elements”, are generally used in FEM models of solid structures, where it is important to obtain the models three-dimensional geometry. A great example of such a geometry is an engine block, which has a natural three-dimensional shape. The NX Nastran element library contains four different types solid elements, which is separated by their shapes, number of faces and connected nodes [17]. In Table 2.3, the different elements types and their characteristics are presented.

Type	Shape	Faces	Nodes
CTETRA	Tetrahedral	4	4-10
CPENTA	Wedge	5	6-15
CPYRAM	Pyramid	5	5-13
CHEXA	Brick	6	8-20

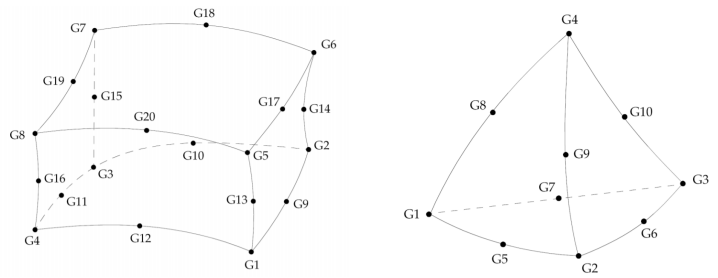
Table 2.3: Solid element characteristics

CHEXA is the recommended element type for general use, but the analysis accuracy degrades when the element shape is skewed [17]. Figure 2.12a shows the shape and element connection for the CHEXA element. For geometry with a certain complexity (Sharp edges, Fillets, etc.), other element types should be considered, as the CHEXA shape will be skewed in the complex areas. For such complex geometry, CTETRA elements is a preferred choice. Due to its tetrahedral shape (Figure 2.12b), the CTETRA element has a better ability to retain its original shape in the complex areas. However, it must be stressed that 4-noded CTETRA elements has to be used carefully. These elements are overly stiff for structural applications and should only be considered for thermal applications [17]. For structural applications, the 10-nodes configuration should be used.

Mesh Mating

Mesh Mating is a great tool for achieving correct interaction between elements from different meshes. There are three different types of mesh mating conditions, all applied at pairs of faces [18]. These are:

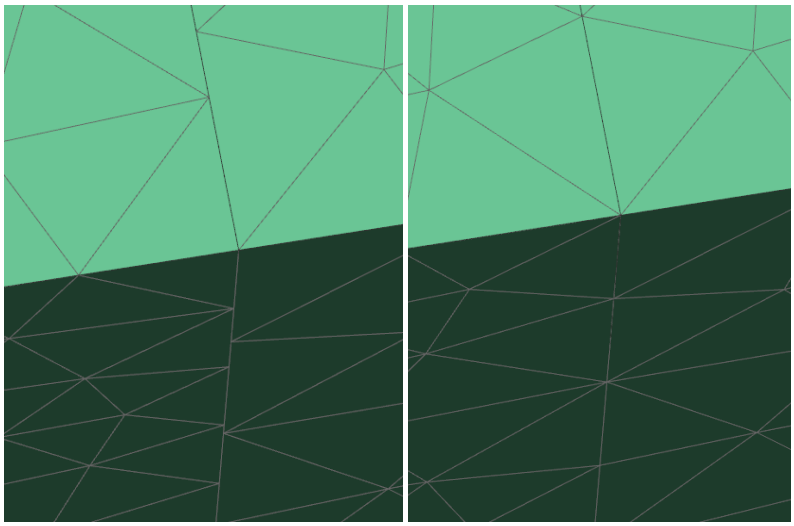
- Glue Coincident
- Glue Non-Coincident
- Free Coincident



(a) CHEXA **(b)** CTETRA

Figure 2.12: Shape and element connection
Source: [17]

The *Glue Coincident* condition creates identical face pairs with shared nodes, by projecting one face onto another and split the faces, to make the touching faces identical (Figure 2.13). *Glue Non-Coincident* condition creates a discontinuous mesh between faces with displacement constraints. *Free Coincident* condition aligns the meshes of two faces with identical node locations, without connecting the aligned nodes.



(a) Before Mesh Mating **(b)** After Mesh Mating

Figure 2.13: *Glue Coincident Condition* assigned at face pairs

Mesh mating can either be done automatically or manually. The *Automatic Creation* feature will automatically find face objects among several selected bodies, while the *Manual Creating* feature requires that the user manually selects two touching faces.

2.6.2 SOL 103 - Real Eigenvalues

The simplest way to perform a modal analysis of a structure, is by determining its natural frequencies and mode shapes in SOL 103. The mode shapes describe the structural dynamic behaviour and indicates how the structure behaves when subjected to dynamic loading. Structural dampening is neglected in SOL 103, which can be done in most cases [19]. The natural frequencies and mode shapes are influenced by the structural properties and boundary conditions. If the boundary conditions change, the natural frequencies and mode shapes will both change. However, if the structural properties are being changed, the natural frequency will change, but the mode shape might necessarily not.

Chapter 3

Project Workflow

As mentioned in the introduction, three master students were working on the ModHVDC project during the spring of 2019, one representing the IEL faculty and the other two representing the MTP faculty. The student representing IEL was studying how segmentation of the stator would affect the electromagnetic behaviour in the generator, while the other student representing the MTP faculty, was working on a digital twin framework for the ModHVDC generator in ANSYS Twin Builder.

Since the electromagnetic behaviour is directly depending on the mechanical design, the project group agreed on having meetings at least every other week, in order to keep track of the overall project progress, exchange knowledge and discuss suggested solutions and analysis results. A suggested workflow model for the project was also developed, as the two other students would use the 3D model created in this thesis.

3.1 Workflow Chart

As already introduced, PTS in Siemens NX contains the configurable 3D model which were used by all the project members. However, due to the lack of electromagnetic solvers in NX, another analysis software had to be used for the electromagnetic analyses. Since a digital twin should be developed within ANSYS, it was beneficial to use ANSYS Maxwell as the desired tool for performing the electromagnetic analyses as well. Another important reason for choosing ANSYS, is that it supports CAD integration with Siemens NX, meaning that a parametric 3D model can be linked to ANSYS workbench and be updated as soon as the CAD model has been changed/updated in Siemens NX.

Figure 3.1 shows the suggested workflow chart for the project. The geometrical changes are done within Siemens NX, but since the model is linked to ANSYS Workbench it will update directly. Inside ANSYS Workbench, thermal and mechanical solvers can be coupled directly to the electromagnetic analysis, thus giving the ability to obtain more accurate results compared to analytical calculations made by hand. The analysis results in

combination with analytical models should then be used for design optimisation and implementation of load cases in Siemens NX. However, it is important to note that the link between Siemens NX and ANSYS Workbench is a one-way communication, meaning that design optimisation and load implementation has to be done manually in Siemens NX.

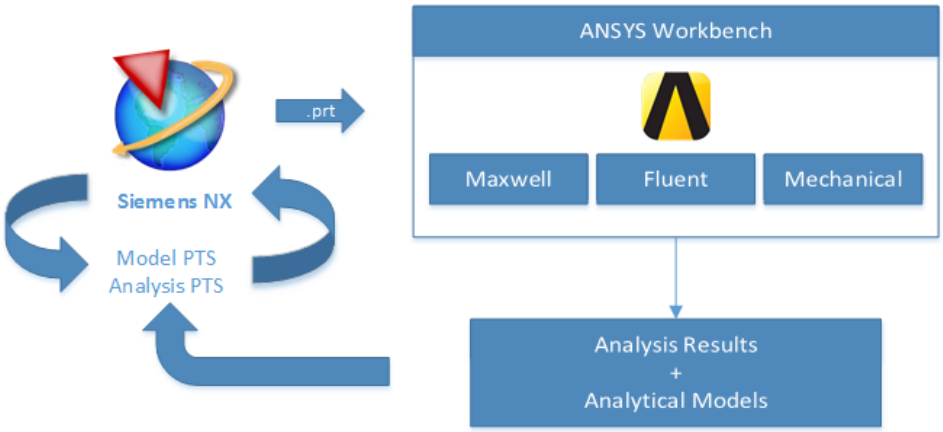


Figure 3.1: Suggested workflow for the ModHVDC project.

The seamless CAD integration between NX and ANSYS Workbench enhance the ability to run multiple analyses without having to waste unnecessary amounts time on preprocessing between each analysis. Performing this analysis-optimisation loop multiple times can lead to an iteration-based development process, that converges towards an optimal generator design.
Further details about the integration of NX within ANSYS Workbench can be found in Appendix B.

Chapter 4

Concept Description

Introducing a new generator concept requires a lot of resources in terms of expertise in different fields and funding. As already mentioned, the official ModHVDC project is planned to be a global collaboration between several different stakeholders with their special fields of expertise. In terms of design, the project proposal includes three main fields of interest. These are:

- Magnetic and Thermal Design
- Electrostatic Design
- Mechanical Design

The three fields are heavily depending on each other. Therefore, a close cooperation between the stakeholders is completely necessary in order to achieve an optimal and robust generator design.

In this pre-project however, only one mechanical student is working on the generator design. As a result, this thesis is mainly focused on the mechanical design of the generator core. The following chapter will introduce the process of choosing a desired concept for the parametric model to be presented with PTS.

4.1 Requirements for the Segment Fastening Methods

As described in 2.1.2, one of the main features regarding mechanical design of the ModHVDC generator is the modularization of the stator. Segmentation decomposes electrical machines into independent segments or modules. This leads to benefits like increased manufacturability, improved fault tolerance and reduced downtime [20]. The ModHVDC stator segments has to be physically separated and operate independent of each other. Due to this, specific requirements for the support structure of the stator segments have been developed.

The fastening of the segmented stator must be strong enough to withstand radial, tangential and torsional loads on the segment. Without segmentation, the stator and rotor resist the generated forces with the help of internal stresses, and the forces are therefore not transferred directly to the structure [21]. Segmentation will thus make the support structure of particular importance.

Insertion/Extraction of Segments

The segments are to be inserted and removed in the axial direction. Insertion and removal of segments in the radial direction is also possible but introduces more challenges. The process of insertion and extraction of the segments should introduce minimal friction and wear on the insulation, as this could lead to breaking of the electrical insulation. Interchangeability of the segments is important as it should be possible to extract segments in the field and perform maintenance and repairs on-site.

Segment Insulation

The segments has to be electrically insulated from the rest of the structure, as well as from each other. Insulating the segments from the stator could be achieved with an insulating material between the segments and the fastener of the stator, or the fastener itself could be made of an insulating material. To ensure that the segments are insulated from each other, an air gap between 2-8mm should be introduced.

Flexibility

One important incentive for introducing segmentation is to increase the flexibility of the generator. This means that it should be possible to extract single segments in the field for performing maintenance and repair.

Each segment needs to be able to operate independently. In this way, a critical error in the generator can be isolated to a single segment and not the whole generator. The generator should also be able to operate at reduced capacity with one or more failed segments.

Geometry of Fasteners

A minimum radius of 5mm on all edges and corners is required. Sharp edges and corners have to be avoided in the segment geometry for two reasons: to introduce less stress concentrations on the insulating material and to reduce the concentration of magnetic fields which occur at such sharp edges and corners.

Insulating Material

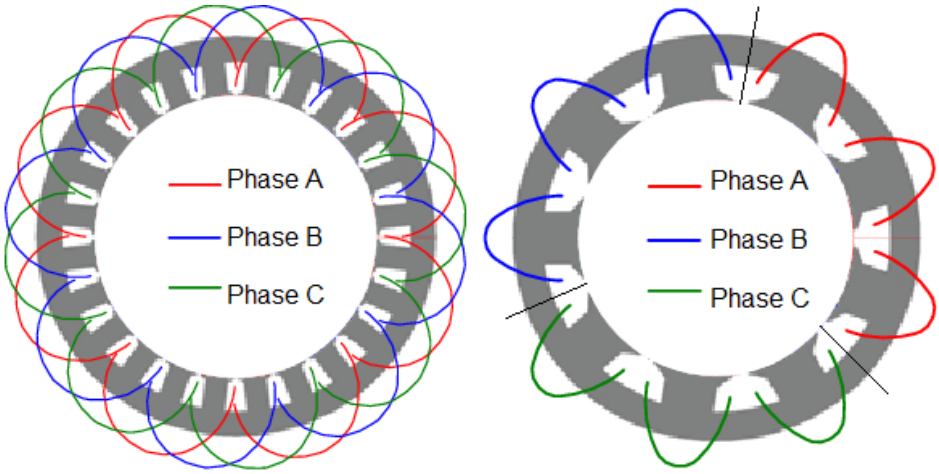
The material chosen for the insulation must be able to withstand high temperatures (up to 70°C) in the segments during operation.

When producing the insulating material, its tolerances must be as precise as the surfaces in the fastening method.

4.2 Concept Development

4.2.1 Winding Configuration

Since the stator segments must be electrically insulated from each other, it was decided at project start that a concentrated winding configuration should be used. As shown in Figure 4.1a, using a distributed winding configuration on a segmented stator will result in electrical contact between the segments, as the windings for each phase are coupled together. However, Figure 4.1b shows that a stator with a concentrated configuration can be split between two phases, and the segments will be electrically insulated from each other.



(a) Distributed **(b)** Concentrated

Figure 4.1: Winding configurations

In recent years, it has been done a lot a research on the PM Generator prototype with concentrated windings, built by Øystein Krøvel at NTNU [3]. The prototype was mainly built for research purposes, regarding windmills. All this research has resulted in a great amount of available data and knowledge at the IEL institute, which can be used in the ModHVDC project. At an early stage of this project, it was decided that the student representing the

IEL institute should investigate how the performance of this prototype would be affected by dividing the stator into 4 segments. This led to the decision of using the prototype as the basic generator layout for the ModHVDC pre-project. Figure 4.2 shows the winding configuration from the prototype. Splitting the stator into four segments, will result in 4 electrically insulated segments, containing three phases, as desired in the ModHVDC project proposal [5].

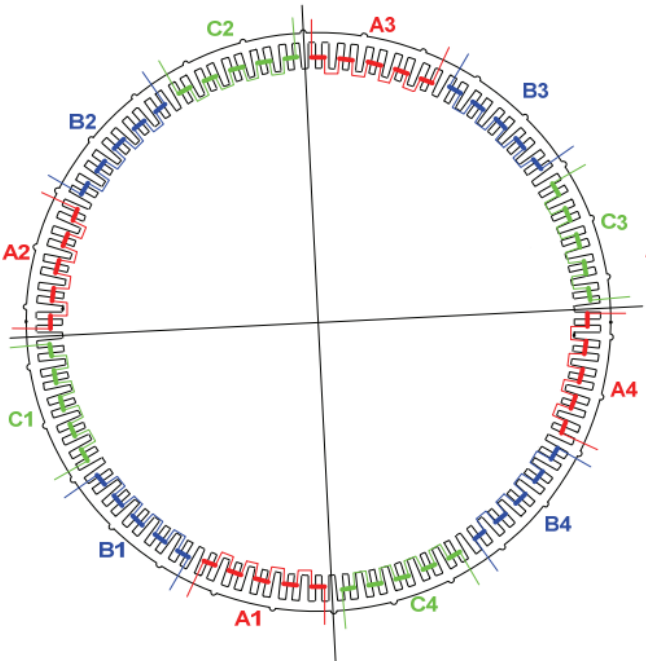


Figure 4.2: Stator and winding layout from the prototype built by Øystein Krøvel
Source: [3]

4.2.2 Stator Fastening Concept

As already mentioned, the main challenge regarding mechanical design for the ModHVDC generator is to determine a fastening concept for the segments. Based on the requirement list above (Section 4.1), it was decided that the first concept iteration should consist fasteners made of an insulating material. Using fasteners from an insulating material eliminates the risk of having electrical contact between the stator and chassis due to wear in the insulation layer, which again eliminates the risk of big performance losses in the generator. By using insulating fasteners, it can also be possible to have an air gap between the stator and the chassis, which can be used for cooling. However, obtaining a material and a structural geometry that fulfils all the requirements, and are strong enough to resist the loads and temperatures is a great challenge.

Geometry

Due to the limited amount of time in this pre-project, it was agreed on exploring the fastening principles in Figure 4.3 for the ModHVDC generator. These are presented as possible fastening concepts for modular generators in the paper "Modularity in Wind Turbine Generator Systems - Opportunities and Challenges" [20], where Figure 4.3b and 4.3c are based on typical fastening concepts used on electrical machines to date.

Figure 4.3a shows a typical i-profile geometry. However, it is not specified any further how the profile actually is intended to be mounted on the stator segment. The most commonly used concept is the dovetail principle shown in 4.3b. These dovetails can stand out from the stator as shown, or they can go inwards, which is also common.

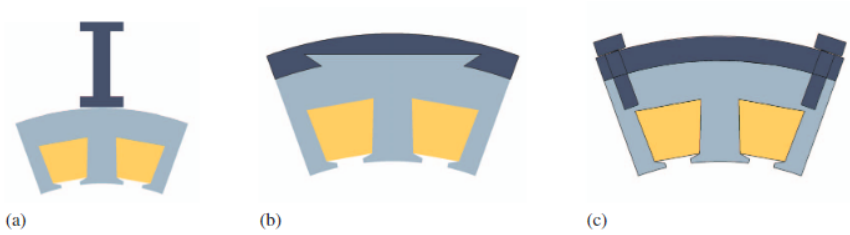


Figure 4.3: Fastening principles
Source: [20]

After seeking some expertise on insulating composite materials from Professor Nils Petter Vedvik at the MTP institute, it was suggested to use i-profiles in a composite material, which is commonly used in structural applications. The i-profile geometry can achieve a higher strength than the dovetail geometry and is cheaper to use, as they are produced as standard profiles which can be cut in desirable lengths. The specified cornering radius can be obtained by either grinding a standard profile or casting a custom profile. The typical i-profile geometry is shown in figure 4.3a.

It was also suggested to use Nylon 66 filled with 50% fibreglass reinforcement for the profiles [22]. In this thesis, the chosen fibre reinforcement consists of chopped fibres. A matrix reinforced with chopped fibres can be assumed isotropic material behaviour. Using oriented fibres will result in higher strengths in certain directions, but will not be considered in this thesis.

	Nylon 66, 50% Glass Fibre Filled
Young's modulus [GPa]	15.5
Density [kg/mˆ3]	1570
Poisson's ratio	0.384

Table 4.1: Material properties for Nylon 66, 50% fibreglass reinforced

4.2.3 Concept 1

Figure 4.4 shows the first suggested fastening concept for the ModHVDC generator. The stator is fastened with composite i-profiles which is glued into tracks that are cut out from the stator and chassis. The chassis part does not contain accurate details about how the chassis should look like. It is only present to show how the fastening i-profiles should be mounted into the stator and a chassis. As discussed above, the air gap between the stator and chassis permits the use of an air-cooling system, which is far less complex compared to a liquid cooling system. Since the concept will be presented in the parametric model, the air gap, number of profiles and profile dimensions can be changed in the configurator presented with PTS.

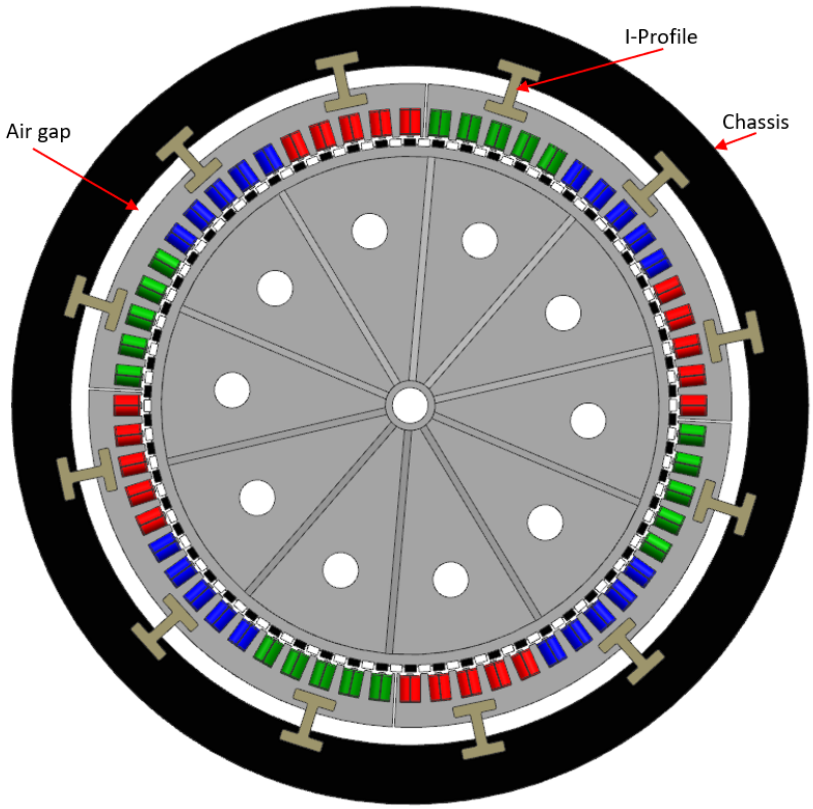


Figure 4.4: Suggested concept 1

4.2.4 Concept 2

The second proposed fastening concept is shown i figure 4.5. This concept is also using i-profiles between the stator and chassis. However, the air gap is replaced with a thin insulating layer. Having a contacting layer between the stator and chassis enables a greater structure stiffness, as the profiles can be entered with pre-tension in the radial direction. In addition, the insulation should be glued onto both the stator and chassis.

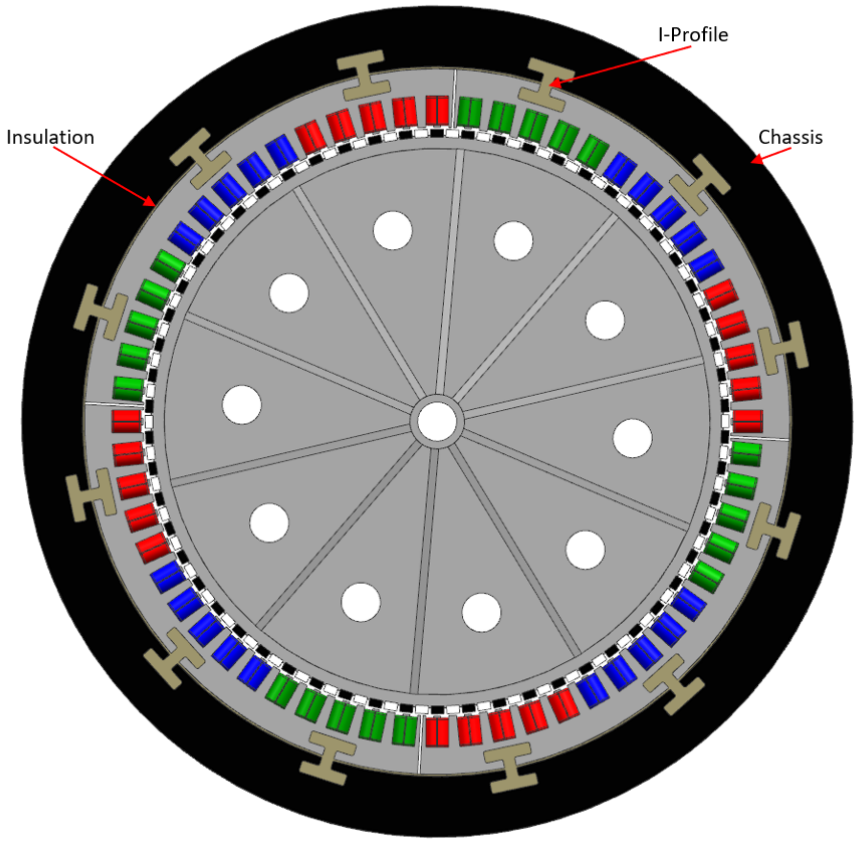


Figure 4.5: Suggested concept 2

The structural stiffness that is gained using fastening concept 2 has to be evaluated up against the cooling benefits from concept 1, by performing several analyses on both concepts. The design space for the concepts is still quite open and can be further developed, based on the analysis results from ANSYS and NX.

Chapter 5

Product Template Model

This chapter introduces the parametric 3D model for the ModHVDC generator concept and its representation in Product Template Studio. To give a better understanding on how the model works, a brief explanation of the assembly and modelling process will be given.

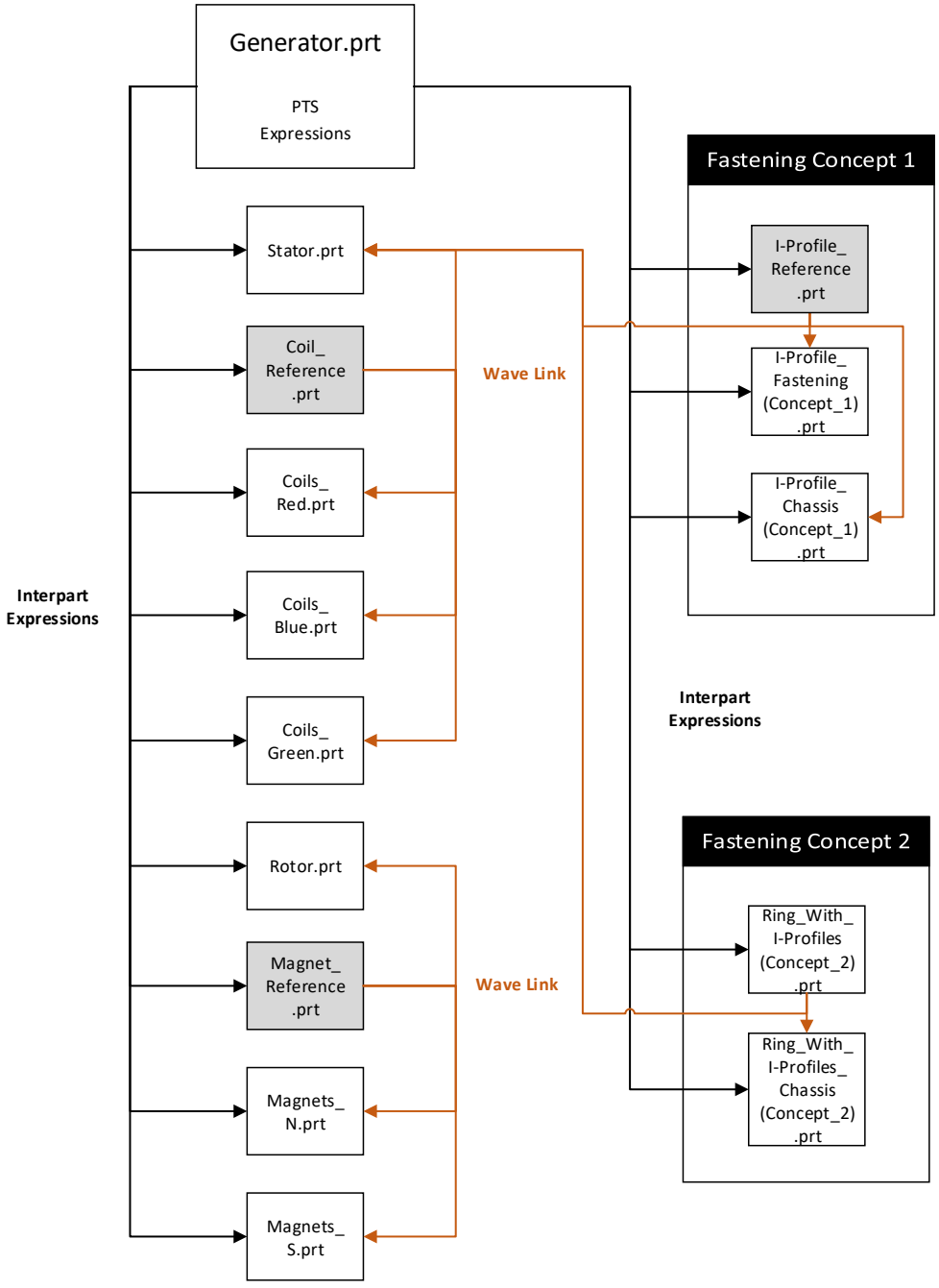
5.1 The Parametric Generator model

The original generator prototype from Krøvel [3] was parametrized and presented in PTS during the project thesis. Since then, the assembly structure has been slightly changed and the fastening concepts have been implemented. In order to give a full overview of the assembly structure after the updates has been made, a short comparison between the new and old assembly charts will be given. The new model features will also be properly introduced.

An extraction of the modelling process from the project thesis can be found in Appendix A. The extract provides an introduction of the assembly structure for the previous PTS version (Fig.A.1), an introduction to parametric modelling in NX and a full description on how the Generator prototype was parametrically modelled. It is highly recommended to read through the extract before reading the rest of this chapter.

5.1.1 Assembly Structure

A final assembly structure for the ModHVDC concept is shown Figure 5.1. The top assembly file ”Generator.prt” contains all the expressions which controls the whole assembly. Interpart expressions are represented by the black arrows on the chart, while the orange arrows indicate how solid bodies are linked into other components with the Wave Link command. Components that are visible in the assembly are marked white, while the hidden reference components are marked in grey.


Generator.prt
PTS
Expressions
Fastening Concept 1
Stator.prt
I-Profile_
Reference
.prt
Coil_
Reference
.prt
Wave Link
I-Profile_
Fastening
(Concept_1)
.prt
Coils_
Red.prt
I-Profile_
Chassis
(Concept_1)
.prt
Interpart
Expressions
Coils_
Blue.prt
Interpart
Expressions
Coils_
Green.prt
Fastening Concept 2
Rotor.prt
Ring_With_
I-Profiles
(Concept_2)
.prt
Magnet_
Reference
.prt
Wave Link
Ring_With_
I-Profiles_
Chassis
(Concept_2)
.prt
Magnets_
N.prt
Magnets_
S.prt

Updates in the Assembly Structure

Comparing the final assembly chart (Fig.5.1) with the assembly chart from the Appendix (Fig.A.1), shows that a few changes have been made in the assembly structure. Apart from the two fastening concepts that has been added, two new reference parts has appeared. As mentioned above, these reference parts are not visible in the model itself but contains geometry that is linked into other parts. The reference parts are implemented in order to keep better control of the original part geometry and for the manufacturing drawings. Take the magnets for example. As described in appendix A.1.2 the main body was originally modelled in magnet_N and patterned within the same part file. When creating a manufacturing drawing, all the patterned magnets appeared on the sheet. Parametrizing a single magnet in a reference part, however, allows us to create a manufacturing drawing, containing only one magnet. Since the magnets and coils are used as tool bodies for subtraction in other components as well, it is beneficial to have a main reference part, making it easier to keep track of the body source in case of broken Wave Links.

Another change that has been done to the assembly, is that the stator insulation has been removed. Since the two new concepts are based on fasteners made of an insulating material, the stator insulation layer is no longer necessary.

5.1.2 Implementing the Fastening Concepts

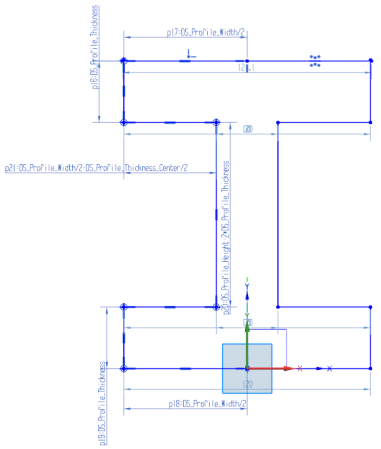
Concept 1

As described in Chapter 4, fastening concept 1 is based on i-profiles that attach the segments to the chassis, with an air gap in between.

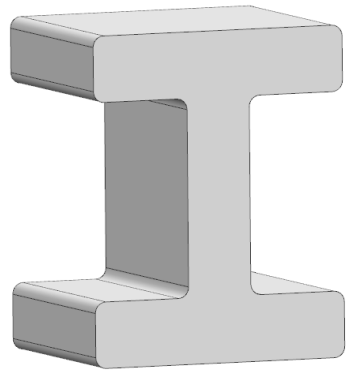
This concept consists of three parts (Fig.5.1), *I-Profile_Reference.prt*, *I-Profile_Fastening(-Concept_1).prt* and *I-Profile_Chassis(Concept_1).prt*.

I-Profile

Since the i-profile body will be patterned and used as tool body for subtraction in the stator and chassis, a reference body is used. As for the other components, the profile outline was first sketch and extruded. In addition, edge blends were added as the profile should have a minimum radius of 5mm in all sharp corners (Ch.4.1). Figure 5.2 shows the modelling process of the i-profile. The reference part does also contain a reference point that is to be used for the pattern feature command and assembly constraints (Fig.5.3a). Figure 5.3b shows the profile pattern made in *I-Profile_Fastening(Concept_1).prt*, where the profile body is Wave Linked from the reference body into the fastening concept part and patterned. The profile bodies are then linked into the stator part file and used as tool a body for subtraction of the profile tracks.

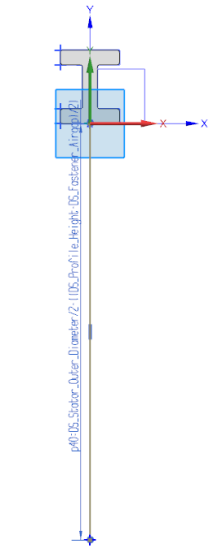


(a) Sketch of the I-profile outline

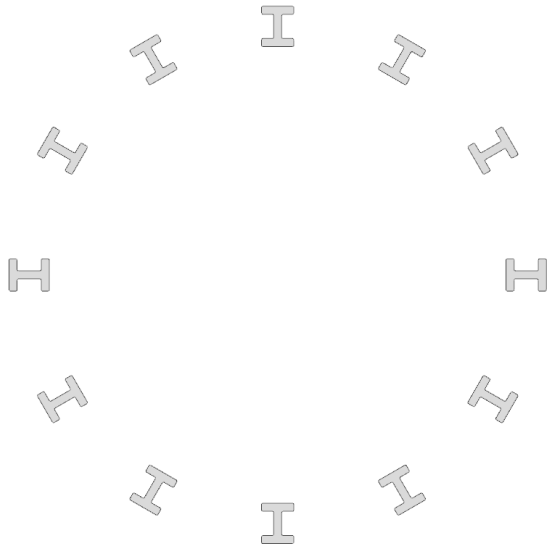


(b) I-profile

Figure 5.2: I-profile modelling process



(a) Parametrized reference point for pattern feature and assembly constraints



(b) I-profile

Figure 5.3: I-profile pattern

Chassis Ring

The chassis ring is modelled after the exact same principle as the stator. A simple ring has been extruded from two parametric circles. The profiles are then used as subtraction tools to create the tracks in the ring. The ring is parametrically affected by the outer stator diameter, the profile air gap and the profile height. As discussed in chapter 4, the chassis ring is only a simple representation of a fixed chassis without any further details.

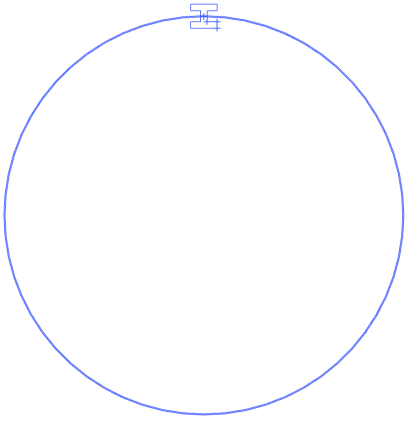
Figure 4.4 shows the final generator assembly for fastening concept 1. The figure shows how the i-profiles are fitted into the stator and chassis ring by using the profile geometry as tool body for subtraction of the profile tracks.

Concept 2

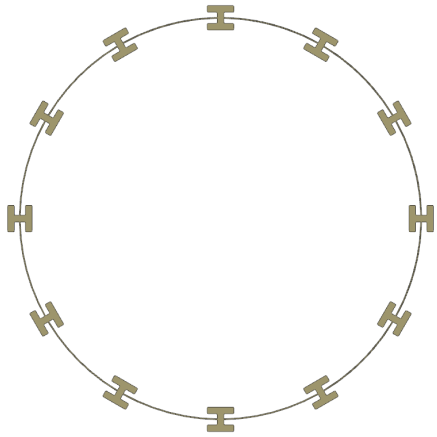
As for the first fastening concept, the second concept was introduced in Chapter 4. This concept is also based on i-profiles, but with a layer of insulation between the stator and chassis.

Insulation Ring with I-profiles

First, the outline of the i-profile and the insulation ring were sketched (Fig.5.4a). The outlines were then extruded into two separate bodies, before the i-profile body were patterned around the insulation body. After being patterned, all the profiles were united together with the insulation ring, creating one single body, as shown in Figure 5.4b. In the same way as for the other concept, the solid body is Wave Linked into the stator part file and used as tool body for subtraction of the profile tracks.



(a) Sketched outline of one i-profile and the insulation layer



(b) Insulation ring with i-profiles

Figure 5.4: Insulation ring with i-profiles, modelling process

The chassis ring is made in the exact same way as for the first concept, except for having a smaller inner diameter as there is no air gap in concept 2.

Both concepts are constrained by the same principles as for the other components, by using points and datums hidden in the reference sets when adding the chassis rings and i-profiles. To make sure that the profiles are symmetrically applied to the stator, an angle constraint has been added, using one of the stator split planes and the local i-profile datums as reference. The angle is controlled by an expression that recalculates the preferred angle when the number of profiles changes.

Figure 4.5 shows the final generator assembly for fastening concept 2.

5.2 Updates for the PM Generator Configurator

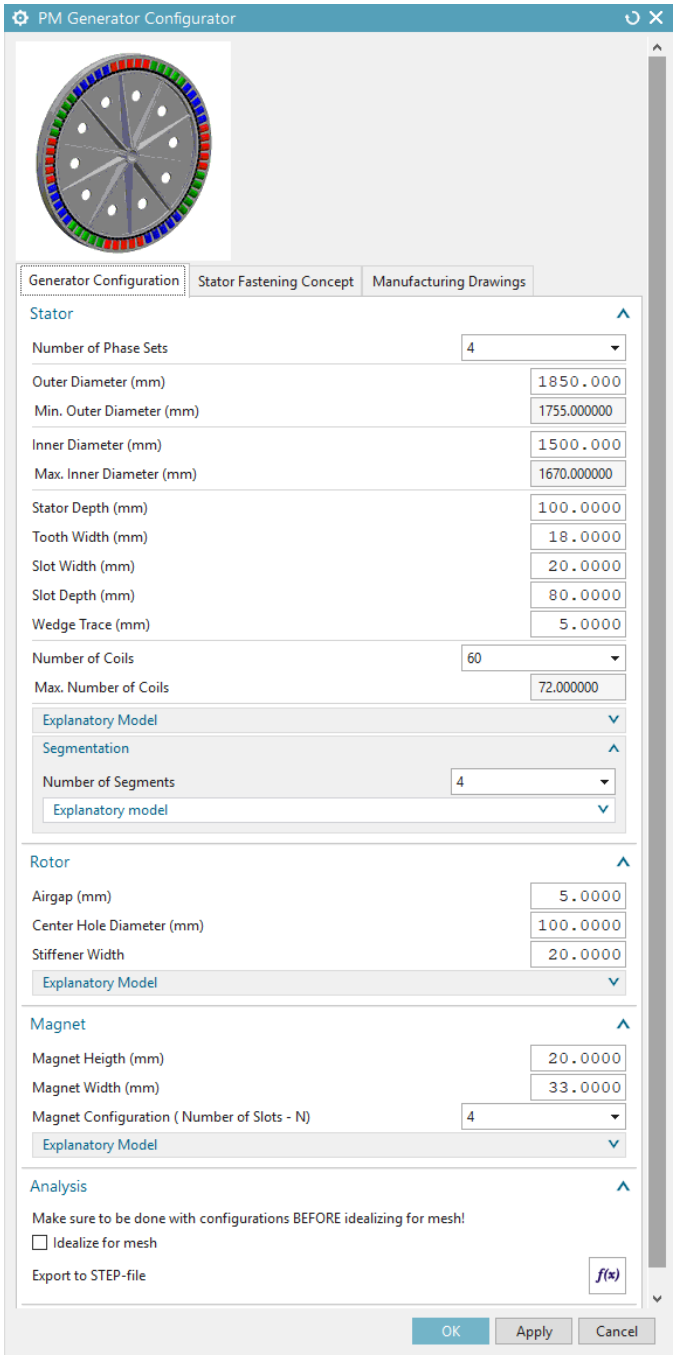
The final model representation from PTS is shown in Figure 5.5. In comparison to the previous version (Fig.A.19), a few small updates have been done to the main configuration window. In addition, two new menu tabs has appeared.

In the main configurator, the insulation features have been removed, due to the new insulating fasteners. Another update has been done to the magnet configuration. Electrical PM machines with concentrated windings does usually have a magnet configuration that equals the total number of coil slots minus 1 to 6 magnets [3]. Therefore, the total number of magnets in the generator is determined by the number of pole slots - N (1 to 6) magnets, based on the user input.

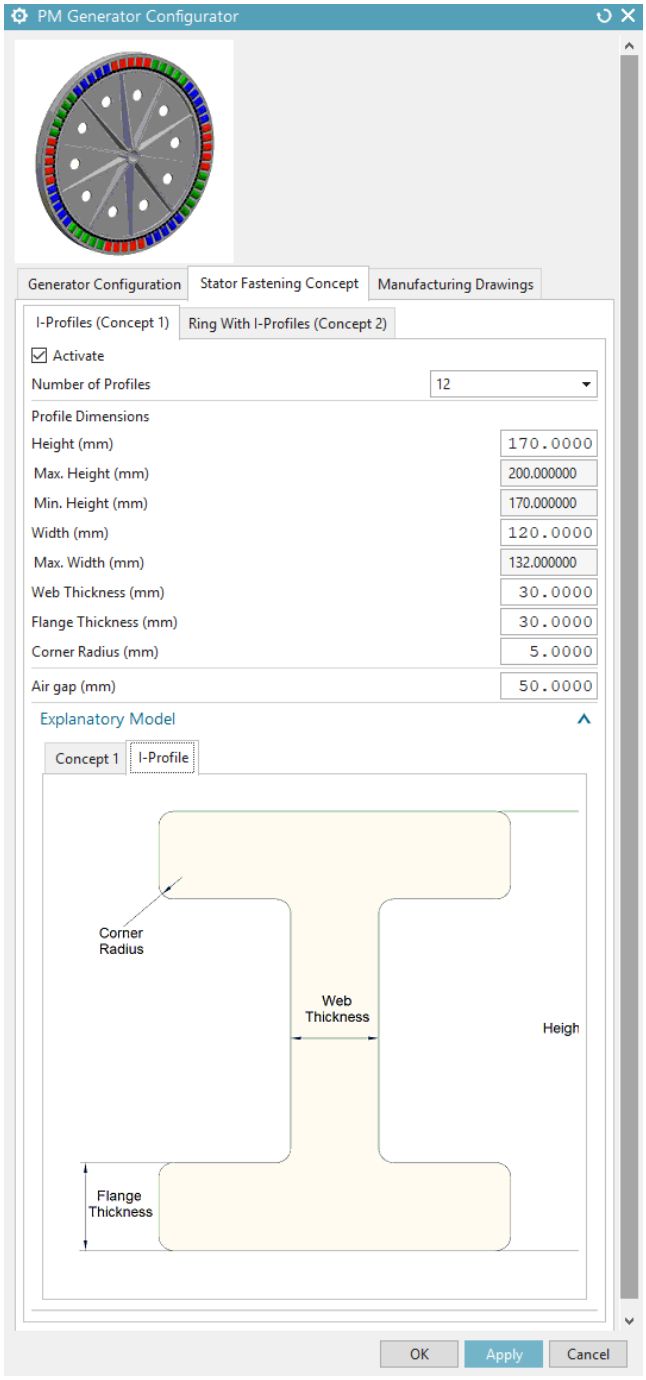
Figure 5.5b shows the configuration tab for fastening concept 1. All the dimensions that affects the profile geometry can be changed within this tab, in addition to the number of fastening profiles. The available number of fasteners will change based on the number of stator segments, in order to ensure a minimum of two fasteners per segment. As described in the project thesis (Appendix A), the check functions regarding max. and min. expression input will not update until the apply button is clicked. This becomes a problem when major changes are being done to the input values. Since this model should have the ability to be configured as a small prototype as well a full-scale windmill generator, these major input changes will appear. Instead of using the check functions, max and min displays has been added to the profile height and width. These displays provide information about the possible max. and min. values for the outer profile dimensions and will always be up to date. By reducing the model flexibility, these problems could be avoided. However, at this early project stage, the flexibility is considered important in order to collect analysis data from a lot of different configurations.

All feature tabs contains explanatory models which can be found in Appendix E.

The manufacturing drawings tab will be introduced in the next chapter.



(a) Main configurator



(b) Concept configurator

Figure 5.5: PM Generator Configurator

Chapter 6

Automatic Generation of Manufacturing Drawings

As described in chapter 2.5, manufacturing drawings are the main communication tool between engineers and manufacturers worldwide. Due to the cost of errors, it is extremely important that the drawings are precise, easy to understand and that they follow a given set of rules/standards. Creating these drawings can be time demanding and requires a lot of experience, especially for more complex components. In the same way as for 3D models, template-based manufacturing drawings can bring a great amount of value to engineering and manufacturing companies in terms of more effective use of resources and a reduction human errors.

The main goal in this chapter is to explore whether manufacturing drawings can be generated directly from the ”PM Generator Configurator” presented in PTS, without the users needing to enter the drafting application. By simply clicking ”Export to PDF” in the configurator, the drawings should be exported directly to PDF files.

This challenge has been divided into two separate problems.

1. Create drawing templates that lives up to the rules presented in chapter 2.5.
2. Find a way to export these drawings to PDF directly from PTS.

6.1 Creating the Drawing Templates

When creating a drawing inside a model that is presented with PTS, the drawing will appear under the ”Template Controlled Data” tab in PTS author. This means that the base views are controlled by the CAD model. When a model change has been done in the modelling application, the drawing views will automatically update when ”Update Drawings” is selected in the drafting application. Both the base views and dimensioning

will be updated. In addition, the scaling and note features can be controlled by expressions, thus providing support for automatically controlled templates.

An assembly drawing containing all components and a part list will be made. In addition, the following components will be presented in its own detailed drawings:

- Stator
- Rotor
- Magnet
- I-profile
- Insulation ring with i-profiles

The coils will not be presented in their own drawings, since they are made of winded copper wires and will not follow the exact dimensions presented in the CAD model. The coils are usually presented in spec sheets, containing information about the copper wire, number of winds per coil, the total number of coils per phase etc. Since this thesis is focusing on the mechanical design, the coil spec sheets will be excluded. The chassis components for the fastening concepts are also excluded, since they do not contain any important details about the generator. They are only present to give a visual sense of how the fastening profiles will be fitted to a chassis and for analysis purposes.

6.1.1 Sheet Layout

In this thesis, an A3 layout template for the NTNU - MTP faculty has been used (Fig.D.1). The template can be found at NXPortalen.com [23]. In order to make the template accessible within NX, the template file was first copied into the
"Siemens\NX 12.0\DRAFTING\templates" folder. A palette file was then created in the same folder, containing the specifications for the drawing template. The palette file can be seen in Appendix F.

Title Block

Some of the fields in the title block has a constant value/name, while others have to update with the model changes. The fields containing the part name, company name and projection method can be assigned with constant values. However, the fields containing the date, constructor initials, and scale should have the ability to change. This has been solved by controlling these values with expressions. Figure 6.1 shows an example of the title block from the rotor drawing.

Date
The date will automatically update every time the model is updated for external change, by being extracted from the following expression
$Date\_Full = StringUpper(dateTimeString("localTime?", True))$, which saves the time and date when the model is updated for external change.

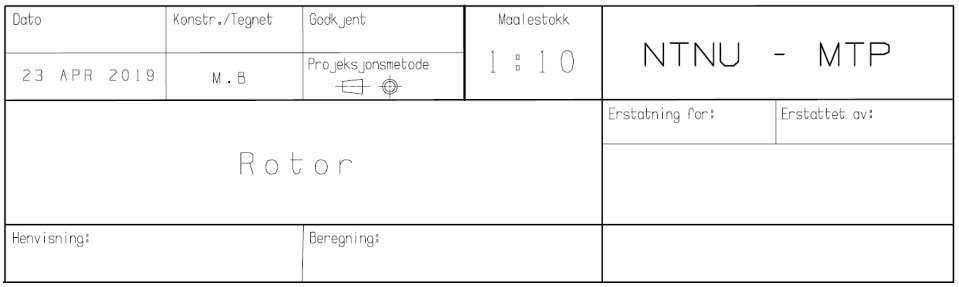


Figure 6.1: Title block

Scale
Since the 3D model can be defined as a small size prototype or a full scale windmill generator, the scaling has to flexible in order to get the best possible drawing views. This has been solved by defining expressions that changes the scaling resolution, based on the largest dimension in every component. The scale resolution for the stator drawing is for example controlled by the value of the outer diameter. Based on the sheet dimensions and the standard scale resolutions, introduced in chapter 2.5.2, different scaling intervals has been defined for the dimensions affecting the outer geometry of every component. The expression below controls the scaling resolution for the assembly, stator, rotor and insulation ring drawings.

```
Generator_Scale = if(DS_Stator_Outer_Diameter < 2000) 0.1
 ↪  else if(DS_Stator_Outer_Diameter < 4000 &
 ↪  DS_Stator_Outer_Diameter > 2001) 0.05 else (1/50)
```

Constructor Initials
The constructor initials are defined as a string expression, which can be modified by the user in the configurator menu (Fig.6.4) .

6.1.2 Projection

All the components have been projected, using the first angle projection method. The number of projections per drawing has been held to a minimum, in order to keep the drawings as neat as possible. As already discussed above, the projection scaling is determined by expressions.

6.1.3 Dimensioning

The dimensions has been assigned, using the principles presented in chapter 2.5.3. They can either be applied by choosing the lines from the 2D projection as reference objects or by using body faces. In order to make the dimensions more robust for changes, the faces have been chosen as reference.

It is the dimensioning part which is most critical when it comes to the automatic drawings. On some components, a high number of faces can be added or deleted, thus making it hard to avoid that the dimensions will lose their references and fail. Take the stator for example. It can be divided into eight bodies; the number of coil slots can vary from 48 to 192 and the fastening tracks will come and go. Since both the coil slots and fastening tracks are made, using a reference body which is patterned, the dimensions has to be placed in the tracks/slots made by the original reference body. Thanks to symmetry it is enough to dimension the ”original” tracks/slots and add the total number of features in front of the dimensions (2.5.3).

A useful tool when creating template-controlled drawings, is the ”Suppress Drafting Object” function. If a model feature can be added or removed, as the fasteners for example, it is necessary to suppress the dimensions at the same time as the modelling feature is suppressed. Otherwise, the dimensions will lose their references and fail.

6.1.4 Assembly Drawing

The assembly drawing shows the whole assemble from a top view. A parts list is created, using the automatic parts list function in NX (Figure 6.2). Since the total number of coils is divided into three different part files, the coils have been manually edited in the list. Instead of having three different coil columns separated by its CAD colors, only one common coil column is created. In the quantity row, the number of coils is determined by the *DS_Total_Number_of_Coils* expression. The exact same thing is also done with the magnets, as they are divided into two part files based on their polarity. Since the stator can be divided into segments, but is added in one part file, the stator quantity number is also determined by an expression. The same goes for the fasteners as they are added in one part file, containing multiple bodies.

PC NO	PART NAME	QTY
6	I-PROFILE_FASTENING_FIXED_RING	1
5	I-PROFILE_FASTENING	12
4	COIL	60
3	MAGNET	116
2	ROTOR	1
1	STATOR	4

Figure 6.2: Parts list

6.2 Sheet Export from PTS to PDF

Initially, it was a hope that the drawing sheets could be exported directly from PTS by creating a NX Open script that could be run from the configurator and perform the exporting process within the existing NX session. However, in order to successfully export drawing sheets, NX has to enter the drafting application, which cannot be entered while PTS is open. So, running the export script directly from the model application workes fine, while running it from PTS leads to an error message, saying that the drafting application cannot be entered. The first attempt for solving this problem was to rewrite the script, so that the program could close PTS, export the sheets and then reopen PTS after. Unfortunately, the approach did not work as PTS cannot not be closed from a NX open command. The next and final approach was to perform the exporting process in a new NX session, executed in batch mode.

6.2.1 Batch Mode Export

Running a NX session in batch mode requires that a NX Open script has been built as an executable (.exe) file in Visual Studio, which can be done by using the NX Open library for visual studio. The .exe file can then be executed directly from the command prompt like any other executable. However, since this is a managed application, the NX.NET library files needs to be copied into the same directory as the executable file. These files can be found in the %UGII_ROOT_DIR%\managed directory [24].

The export script that was made for the previous approach was a great starting point for the executable program. In order to successfully run it in batch mode, functions that finds the file path for the generator.prt-file and the PDF output folder was also implemented. The program was developed by combining journal recording, retrieving a few tips from `NXJournaling.com` [25] and a lot of testing and development in Visual Studio.

Figure 6.4 shows a graphical representation of how the PDF export process is done. The two main boxes with white on black headers represents the two scripts necessary for executing a batch mode session from inside PTS.

cmd_PDF_call.vb

The script named cmd_PDF_call.vb is called from PTS and is responsible for starting the executable file by entering its file path in command prompt. All the boxes coloured in grey represents functions within the program code. The grey box within the cmd_PDF_call.vb script represents a function that searches for the PDF_Exporter file in all folders stored at the desktop and return its file path to the main program. The full code can be found in Appendix G.1.

PDF_Exporter.exe

The executable program PDF_Exporter.exe is represented in the second main box in figure 6.3. This is the core program that executes the PDF export in batch mode. Also here, a function is first called to return the file path for the file to be opened in the main program. In this case the generator.prt file. To keep control of the output location for the PDF sheets, another function is called, searching for and returning the file path for a folder named ”Machine Drawings”, in all folders at the desktop. This folder has already been added inside the folder containing all the CAD-, FEM- and SIM-files. After getting the file paths, the program starts a new session, loads the part file and set it to workpart. Since the assembly could have been saved within another application than the modelling or drafting applications, the program always switch to the drafting application after the part file has been loaded.

After entering the drafting application, a subroutine is called. The subroutine exports each sheet to PDF, taking the file name, work part and output file path as input. As shown, the subroutine retrieves the suppression status for the fastening concepts, in order to control which sheets that should be exported or not. When the component suppression statuses are retrieved, the function starts looping through the relevant sheets, assigning the output names, update each sheet, define the PDF setup and export each sheet to the specified folder. When all sheets have been successfully exported, the subroutine returns to the main program. The program then closes the open parts before it terminates. The full code for the executable can be found in Appendix G.2.

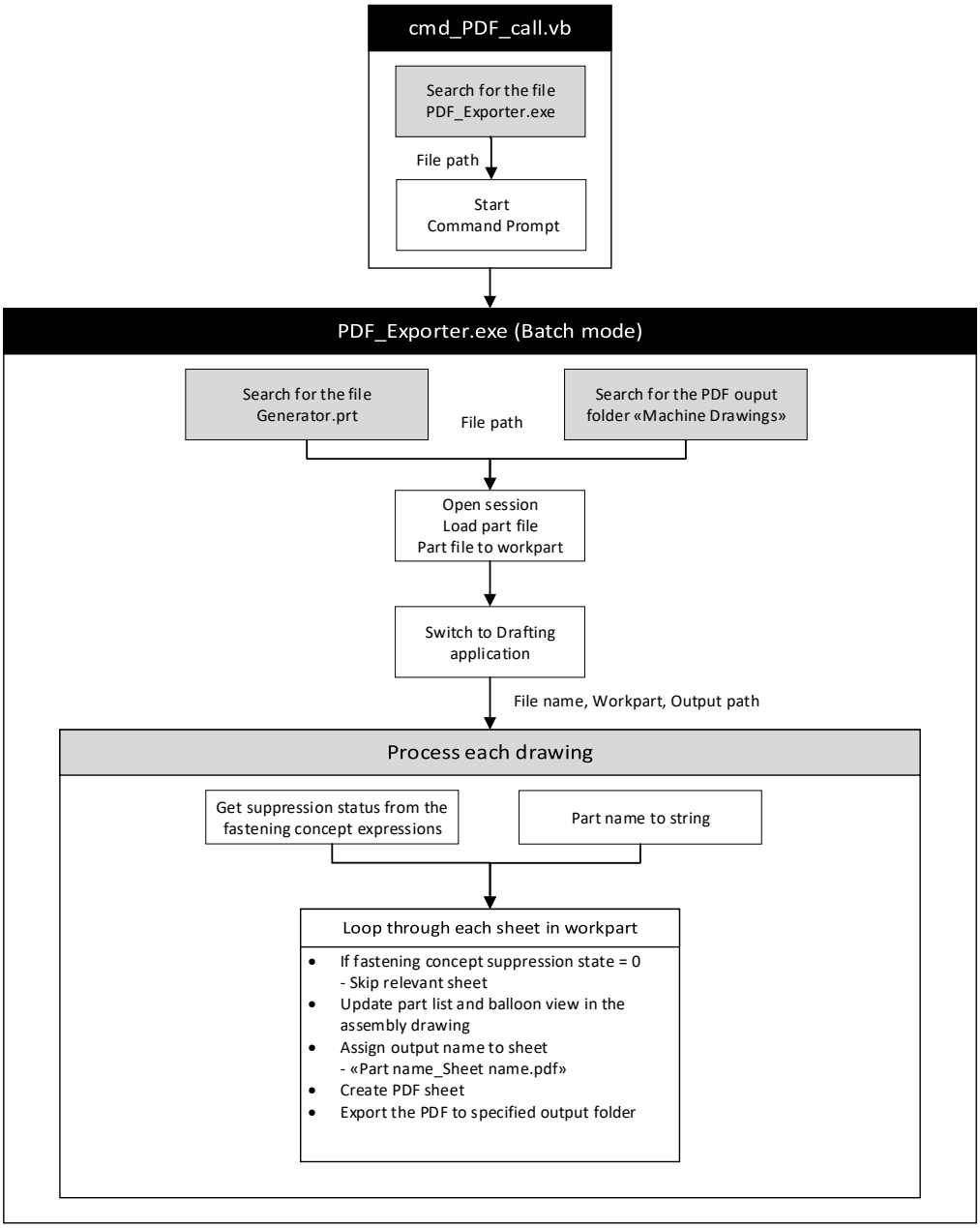


Figure 6.3: PDF exporter overview

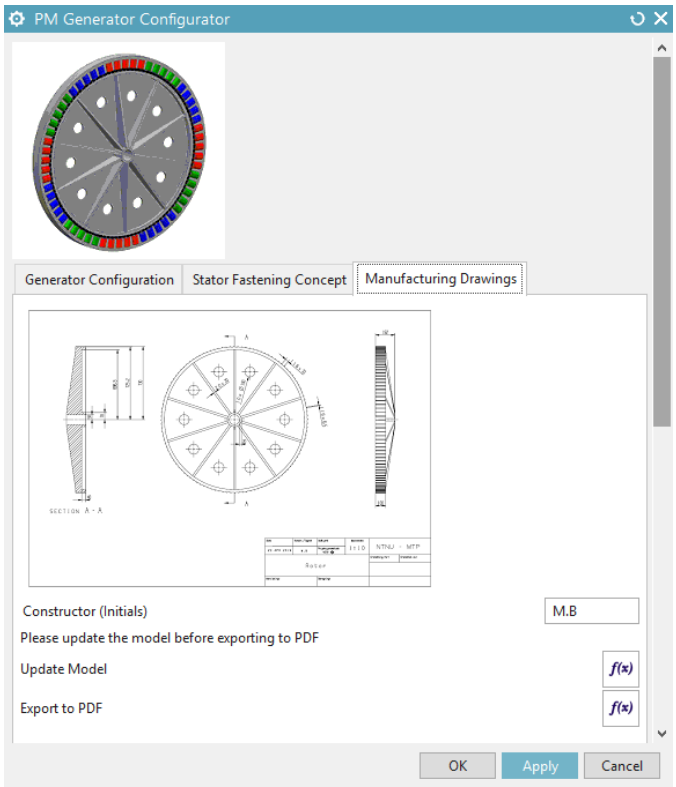


Figure 6.4: Drawing exporter in PTS

Chapter 7

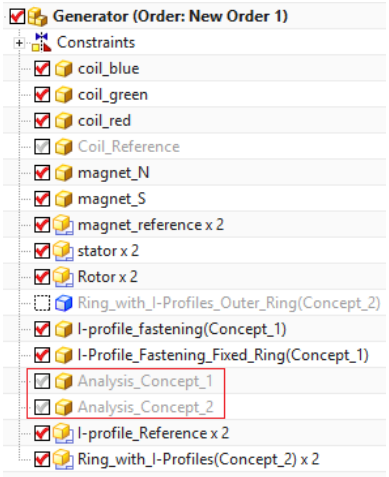
Finite Element Analysis

As presented in chapter 2.1.4, modal analysis for vibration an noise is the most critical when a new electrical machine design is developed. Therefore, a robust modal analysis setup for both fastening concepts has got the highest priority during this thesis. Since these analyses should be presented in PTS, it is important to obtain a fully robust setup that does not fail when the model is being heavily reconfigured from the Generator Configurator. This chapter will introduce the model and analysis setup, while the implementation in PTS will be introduced in the next chapter.

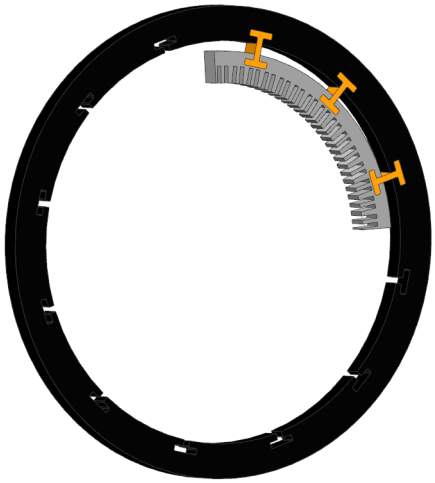
7.1 Model Setup

For the structural analysis, it is mainly the stator and fastening concepts that is of particular interest. To avoid unnecessary complexity in the analysis setup, all the components that does not affect the stator structure are excluded. This is done by creating two new part files, one for each fastening concept. The relevant modelling features are then wave linked into each part file, making sure that the structure is always up to date. Figure 7.1a shows the two part files in the generator assembly tree. Both parts are assigned with empty reference sets, meaning that they are not visible within the generator assembly. Since the analyses will be presented in PTS, the analysis part files must be included in the generator assembly, to make the analyses accessible from PTS.

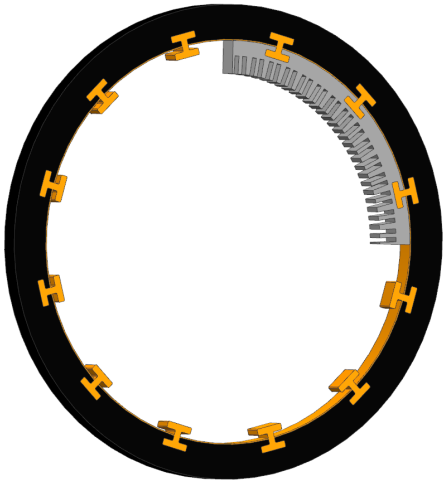
Figure 7.1b and 7.1c shows the geometry that is Wave Linked into the two analysis part files. Due to symmetry, only one stator segment is linked into the part files. Including only one segment reduces the analysis complexity and solver running time significantly. In addition, having several bodies that are added and removed from the mesh does lead to challenges when the analysis should be presented in PTS. These challenges will be introduced in more detail later in the thesis.



(a) Analysis part files in the assembly tree



(b) Concept 1



(c) Concept 2

Figure 7.1: Analysis model setup

7.2 Finite Element Model

7.2.1 Meshing

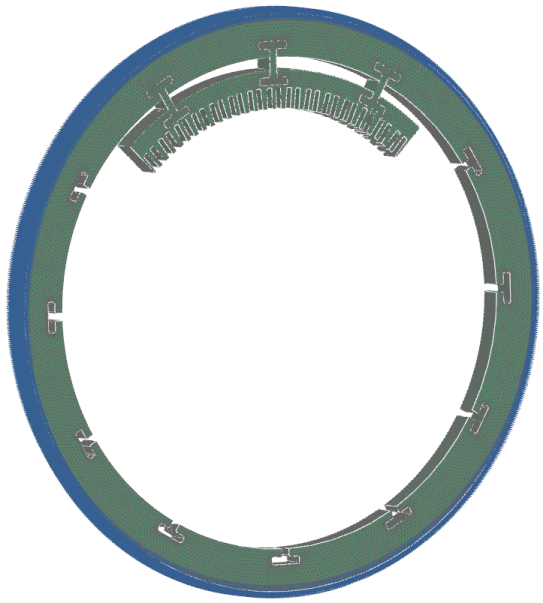
In order to achieve accurate simulation results, it is essential that the mesh is assigned properly, with appropriate element types and sizes. An overly fine mesh will result in a dramatically increased solver running time, without affecting the result accuracy. On the other hand, a too rough mesh will lead to inaccurate results. Since the model can be scaled from a small prototype to a full-size windmill generator, an appropriate mesh size has to be evaluated and reassigned by the user in PTS. Due to the great model flexibility, it is important to use an element type that can be suitable for complex geometry. The CTETRA element is commonly used in more complex geometries and fits the generator components quite well. Some of the components could be accurately swept meshed with CHEXA elements, but due to the high model flexibility, it is more robust to use the same element type on all components. The models for both concepts have been assigned with CTETRA elements on all components.

7.2.2 Mesh Mating

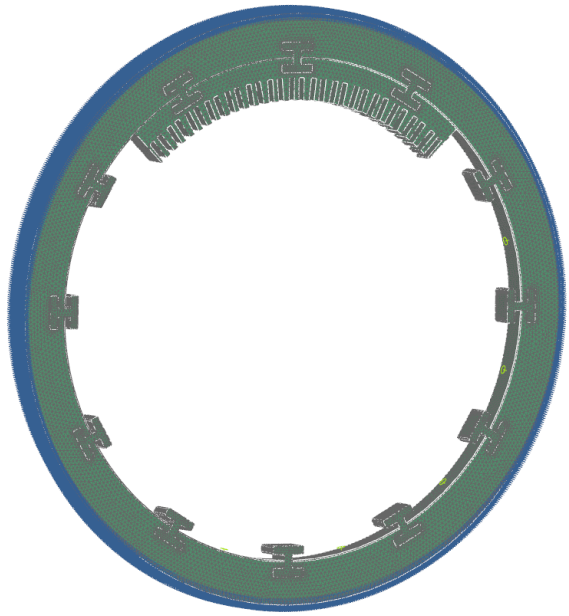
To ensure a best possible accuracy for the results, the tool Mesh Mating - Glue Coincident Condition is used on all surfaces in contact, in both models. As already described in chapter 2.6.1, the condition creates identical face pairs with shared nodes. Due to the model flexibility, the contacting faces has to be selected automatically by the function. After careful inspection of the mesh, no mesh distortions or face irregularities has been found when using the automatic function.

7.2.3 Materials

The 50% fiberglass reinforced Nylon 66 fastener material were created in the materials library. Materials assignment will be discussed further in the next chapter.



(a) Concept 1



(b) Concept 2

Figure 7.2: Finite Element Analysis setup

7.3 Finite Element Analysis setup

7.3.1 SOL 103 - Real Eigenvalues

As described in chapter 2.6.2, SOL 103 - Real Eigenvalues is commonly used when the natural frequencies of a structure are to be determined. The following section describes how the solver i set up, before the analysis can be run.

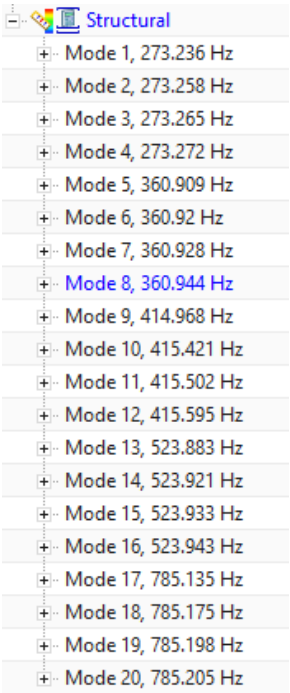
Boundary Conditions

The boundary conditions for the two concepts does only include a fixed constraint. Since the outer rings in both fastening concepts represents a simplification of a fixed chassis for the stator, a fixed constraint has been assigned on the outer surface of the rings. The fixed constraints are shown as the blue surfaces in figure 7.1. If the outer rings had not been fixed, the lowest modes would have been determined by the outer ring. However, since the fastening profiles will be mounted into a fixed chassis, the resulting modes will be more accurate when the outer ring is fully constrained.

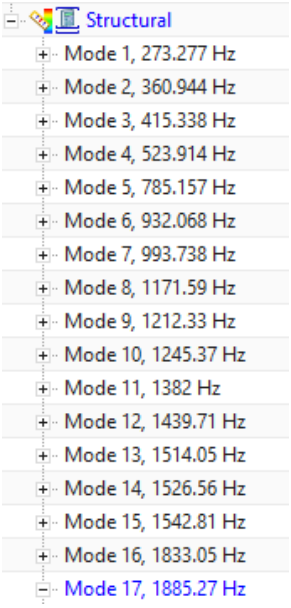
Since the typical operating temperature in the stator lies around 60-70°C, the temperature can affect the material behaviour, thus also the natural frequencies for the structure. A constant temperature load was assigned, but unfortunately, a few tests showed that SOL 103 does not take temperature loads into account, even though it can be applied in the solver menu.

Symmetry

Excluding geometry due to symmetry can lead to fatal errors in a modal analysis. Several modal analyses have been subjected to a model containing all stator segments vs another containing only one of the original four segments. A careful comparison of the results from both ANSYS and NX, showed that the model containing all four segments experienced four of each mode, acting at one segment at the time. Figure 7.3 shows the results of such a comparison. The result to the left (7.3a) is from the model including all four segments, while right-hand side (7.3b) shows the result from the model containing only one segment. If a free free modal analysis (no constraints) had been performed, the lowest modes would have occurred in the outer ring rather than in the stator segments. Including all segments would not have resulted in four of each mode, and removing segments due to symmetry would have influenced the results.



(a) All segments included



(b) One segment included

Figure 7.3: Modal analysis symmetry comparison

Chapter 8

Product Template Analysis

Since the parametric model supports a high number of different configurations, an automatic analysis setup can bring great value for the ModHVDC project as the model can be effectively fine-tuned against undesirable eigenfrequencies, before importing and preprocessing the model in ANSYS Maxwell.

As the two generator concepts have different geometry and number of bodies, they have to be presented in two different analysis templates, one created in each analysis component file. In order to create an analysis template in PTS, a sim-file must be created in advance. The .sim file will then be recognised by PTS and can be loaded into PTS author.

8.1 Product Template Configuration

When the SIM-file is accessed within PTS author, the same drag-and-drop approach that is used for the CAD models, is also used for the analysis templates. As shown i figure 8.1, all the expressions, CAE Bodies, CAE meshes, CAE Loads and CAE Solutions related to the SIM-file can be accessed in the PTS explorer menu, to the left. These controls enable tailor made systems that either have a very limited support for user inputs, or support a high degree of user configuration. In order to get a better understanding on how the modal analysis templates are created, the different CAE controls are introduced below:

- **CAE Bodies** - Controls the material of the solid bodies within the model.
- **CAE Meshes** - Controls the details for each mesh within the model.
- **CAE Loads** - Controls the applied loads, constraints and simulation objects.
- **CAE Solution** - Controls the solution solvers and post views.

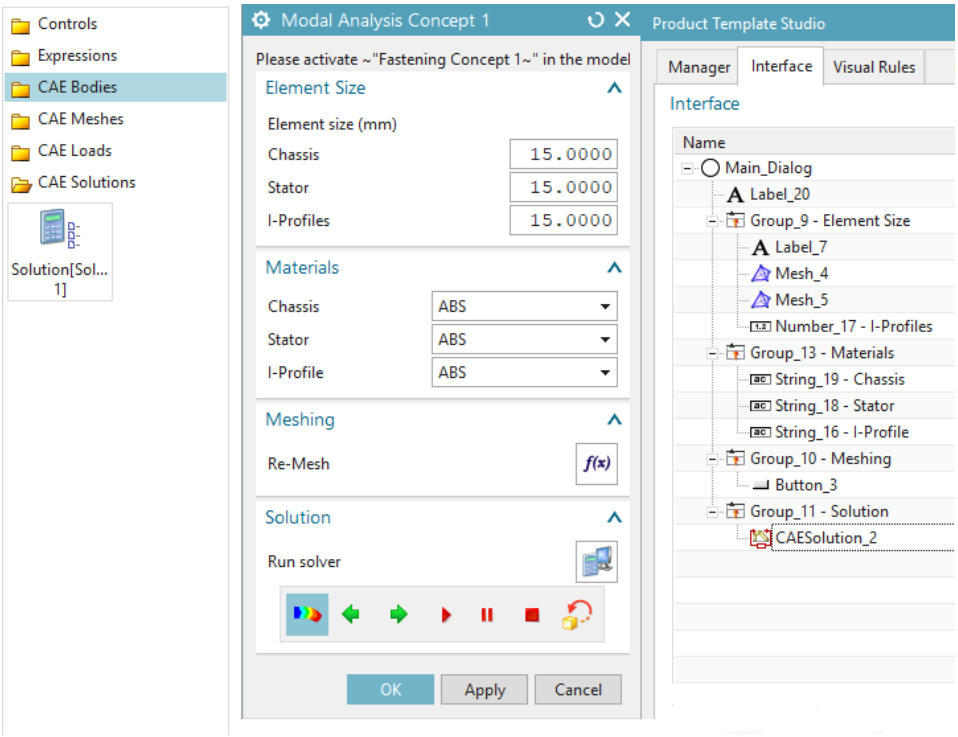


Figure 8.1: PTS author for analysis templates

Post View Templates

In order to get a visual representation of the simulation presented in PTS, a post view template must be created. This is done by saving a post view as a template in the *Post Processing Navigator*, within the simulation application. For modal analyses, one template has to be created per mode, which inn this thesis will be the five lowest modes.

8.1.1 Difficulties with Analysis Templates for Flexible Models

Unfortunately, creating analysis templates for flexible models in PTS is not straightforward. The geometrical flexibility in the generator model leads to problems when the analysis template is to be created.

Variability in the Number of Bodies

Since the analysis template is based on a SIM-file, the template does only enable the users to customise an already pre-defined FE-model with a fixed number of bodies. This becomes a great problem for generator concept 1, as the number of fastener bodies varies,

based on the user input. If the number of fasteners increases, the new CAE bodies has to be assigned a mesh before the solver can be run. New meshes have to be assigned in the FEM-file, not the SIM-file, which the template is based on. The mesh mating conditions does also need to be re-assigned as new bodies appears, which cannot be done from the SIM-file.

Major Geometry Change

When a solid body is experiencing major geometrical changes, a new problem arises, when going from the CAD to FEM-file. Within the FEM-file, the polygon body will be marked for update. This means that the old polygon body will be deleted and recreated with the geometrical updates. When a polygon body is deleted and recreated, it will appear as a new body within the FEM-file. Since the materials are assigned to the polygon bodies, the materials will be removed when the polygon body is updated. As described in section 8.1, the polygon bodies are selected in the PTS explorer for material control. These body references will be deleted when the bodies are updated, which means that the materials cannot be reassigned from the analysis template.

When the polygon bodies are mesh mated, using Clue Coincident condition, all polygon bodies will be fully updated each time a body has been subjected to geometrical updates. However, it is actually a good thing that all the bodies are updated instead of just the one that has been modified.

8.1.2 Dealing with Flexible Models in Analysis PTS

Even though the problems described above indicates that the generator model is too flexible for being presented in an analysis template, a NX Open script can be used to deal with these problems. Figure 8.3 shows a simple explanation of a NX Open script that is written specifically to deal with the problems related to generator concept 1. The button named Re-Mesh in figure 8.1 executes the script from PTS. After materials and element sizes has been selected, the script must be executed before the solution can be solved.

When the script is executed from PTS, the SIM-file will automatically be loaded into the session as the analysis template is based on the SIM-file. Since material, mesh, and mesh mate conditions must be assigned in the FEM-part, the script switches from SIM to FEM as active workpart. Expression values regarding component materials, number of segments and element size for the fastener mesh are then retrieved. The number of fasteners expression is used for defining the length of a list, containing the fastener bodies that must be meshed or re-meshed. Next, the finite element model, mesh builder, material builder, mesh mate builder, desired element dimension unit, and materials are defined.

In order to apply the different properties to the CAE-bodies, the script has to loop through all the bodies and apply the desired properties. Getting access to the CAE bodies in NX Open is done by looping through all object tags in the current session, as shown in the

code below. The tags containing polygon bodies are found by searching for tag objects with a journal identifier including the string ”CAE_Body”.

```
 ' Looping through all objects, looking for bodies

Do
    thisTag = ufs.Obj.CycleAll(workFemPart.Tag, thisTag)
    If thisTag <> NXOpen.Tag.Null Then
        thisObject = NXOpen.Utilities.NXObjectManager.Get(thisTag)

        If thisObject.JournalIdentifier.contains("CAE_Body") Then

    ' Defining cAEBody1 as CAE body object

                    Dim cAEBody1 As NXOpen.CAE.CAEBody = CType(
                        workFemPart.FindObject(thisObject.
                        JournalIdentifier), NXOpen.CAE.CAEBody)
```

Unfortunately, the only information that is accessible from the tags within the FEM-session is the polygon body names (e.g. ”CAE_Body(1), ”CAEBody(2)” etc.), CAE feature type and Tag id. No feature information from the modelling application can be found in these tags, thus making it difficult to recognize the different bodies in a NX Open script. Since the polygon bodies updates each time the cad file is reconfigured, the tags updates as well. However, since the bodies are always updated in the same order, based on how the they are sorted in the cad file, it is known which polygon body that belongs to each feature.

Figure 8.2 shows the body order in *Analysis_Concept_1.prt*. The order tells us that the chassis body (outer ring) always will appear first in the loop, the stator as body number two and the rest will be the fastener bodies.

Figure 8.2: Order of bodies in *Analysis_Concept_prt*

The first polygon body will be assigned the chassis material, the second body will be assigned the stator material, while the remaining bodies will be assigned the fastener materials. The fastener bodies will also be re-meshed before Glue Coincident mesh mating conditions are applied to all contacting surfaces in the model.

After assigning the polygon body properties and updating the mesh mating conditions, the script updates the FE-model before switching back to the SIM-file.

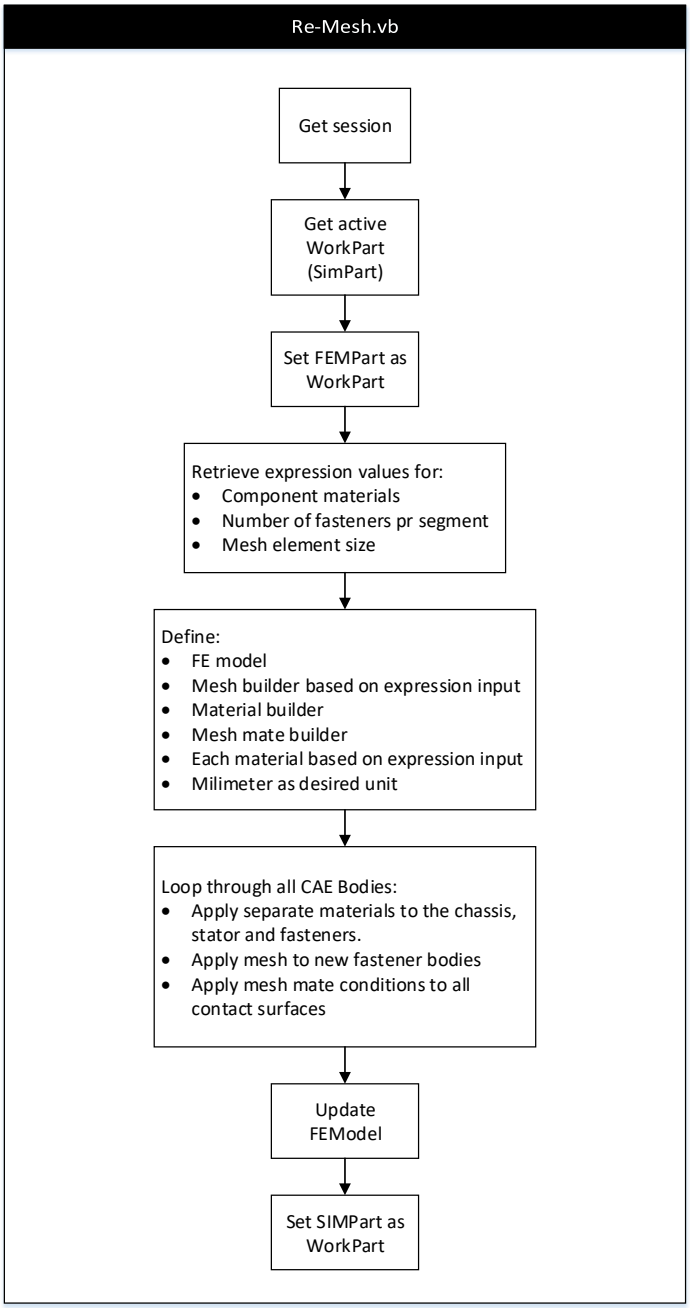


Figure 8.3: NX Open script for reassigning materials, mesh and mesh mate conditions

When the Re-Mesh script has been successfully executed, the solution can finally be solved without having any material or mesh issues.

The same problem with body updates occurs in the FEM-file for fastening concept 2. However, since the number of bodies does not change within the model, it is only necessary to reassign materials and mesh mating conditions when the model geometry is updated. The NX Open script written for the analysis concept 2 is almost identical to the script described in figure 8.3, just a bit simpler as no new polygon bodies has to be meshed.

The full code for the re-mesh scripts can be found in Appendix H.

8.1.3 Configuration of the Modal Analysis Templates

The configuration of the modal analysis template for generator concept 1 is shown in figure 8.1. Since the modal analysis setup is quite simple, the boundary conditions are not added as configurable parameters in the template. However, the parameters that can be changed are the mesh element size and materials.

As already discussed, the materials cannot be assigned in the regular way by adding the CAE Bodies from PTS Explorer. The interface menu to the right in figure 8.1, shows that string expressions has been added in the material group. These expressions are presented as material lists within PTS, containing all the materials from the NX materials library. The user cannot enter any other material names than the ones available within the list. The script retrieves the string values from the expressions and assigns the desired materials to the chosen components, as described in the previous section.

The fastener mesh is either not selected from the PTS Explorer, as the number of fastener bodies that is to be meshed varies. A number expression takes the element size for the fastener mesh as input. This value is also retrieved by the script and used as input value for the element size in the fastener mesh.

Since one analysis template is created for each of the two concepts, it is important that the corresponding fastening concept is activated in the generator configurator, when an analysis template is opened. If fastening concept 1 is activated in the CAD model, the analysis will fail when trying to run a modal analysis for concept 2. To make sure that this wont happen, the visibility for all the groups within the template dialogs is controlled by the suppression status for the two concepts. Figure 8.4 shows the message that appears if the user tries to open the wrong analysis template. All the groups shown in the template in figure 8.1 is hidden, and the user is told to switch fastening concept in the model PTS.

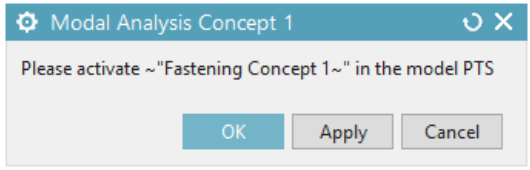


Figure 8.4: Modal analysis template, warning message

Chapter 9

System Validation

In this chapter, a validation of the product template created in this thesis will be given. That includes a short summary of the generator configurator with its key specifications and limitations, and a full test of the drawing exporter and analysis templates with different model configurations. Any errors arising during the evaluation process will be described in this chapter.

A short video, demonstrating the system can be found here:
`https://youtu.be/FWNSsEgVMO4`

9.1 The PM Generator Configurator

The layout for the model configurator was already introduced in figure 5.5. As shown, the configurator has quite a lot of parameters that can be changed by the user. To get a better overview of the configurator capabilities, some of its key specifications are presented below:

- **Stator outer diameter range:** 1000 - 8000mm
- **Stator depth range:** 100 - 1000mm
- **Stator segment configuration:** 1 - 8 segments
- **Phase set configuration:** 4 or 8 phase sets
- **Coil configuration:** 12 - 96 coils
- **Magnet configuration:** 18 - 190 magnets
- **Integrated fastening concepts:** 2
- **Fastening profile configuration:** 6 - 24 profiles

In addition to these key specifications, all the smaller dimensions regarding stator tooth dimensions, air gaps, magnet dimensions etc. can be specified in the configurator menu.

As discussed in the project thesis (Appendix A.1.5) and chapter 5.2, check functions that does not allow the users to enter illogical values are applied to most of the input parameters. However, since these check functions does not update until the ”apply” button is clicked, it can create problems for the user when the model is heavily modified. The problem has been solved by using read only displays that tells the user what max. and min. values that can be entered for the stator diameters and fastener profile height and width, as shown in figure 5.5a and 5.5b. This solution enables the user to enter all the desired specifications before clicking apply and wait for the geometry to update. However, the user has to ensure that the entered values are within the stated max. and min. values.

The PM generator configurator has been continuously tested and developed through the whole semester, and has been used by the two other students without experiencing any major faults.

9.2 Generation of Manufacturing Drawings

Parts of the automatic drawing generation approach has proved to be working as expected, however, several problems have been met during various testing without a proper solution being found due to the limited amount of time during this thesis. The problems are mainly related to how the dimensions and base views behaves when the model is subjected to changes. Exporting the drawings directly from PTS works perfectly fine. A set of example sheets can be found in Appendix D, for fastening concept 1.

9.2.1 Sheet Problems

Dimensions

When the model is subjected to small geometrical changes, where the dimensions are changing but the number of faces remains unchanged, the approach works as expected. All the dimensions updates as they should, and the drawings looks fine. However, when the complex components (stator and rotor) are subjected to changes where the number of faces is changing, some of the dimensions will lose their references and fail. The dimensions that does not fail due to lost references, does usually get misplaced and the sheets becomes messy.

Figure 9.1 shows a part of the rotor sheet after the model has been reconfigured. As shown, all the dimensions in the section view has lost their references. Every time the rotor diameter changes, these section dimensions will fail due to lost references. As a result, the section dimensions has to be manually reattached each time the stator diameter changes. The right-hand view shows how the dimensions gets quite badly placed, compared to figure D.2.

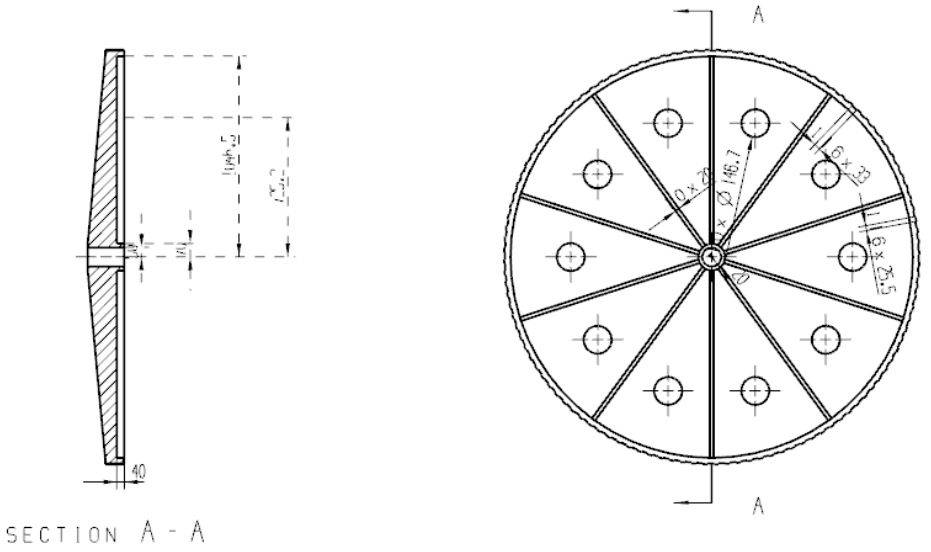


Figure 9.1: Rotor sheet after geometrical changes has been done to the model

Scaling

Another problem occurs when the model is automatically scaled. Since the base view positions cannot be constrained within the sheet, the views get misplaced when they are re-scaled. In some cases, the views can end up being placed outside the sheet and has to be manually pulled back onto the sheet.

9.3 Automatic Analysis Templates

The FEA templates can easily be accessed from the Template Explorer in the modelling application, as shown in figure 9.2. When the templates are opened, the SIM-file opens, and the configured user menus will be available. All interaction with the simulation set-up has to be done via the customized user menu.

Figure 9.3 and 9.4 shows the FEA templates for both concepts. The first menu option is the element size. Each component can be assigned individual element sizes by entering a desired value for each of them.

The next option is the material selection, which enables the user to assign different materials from the local materials library to the objects.

Before the solver can be run, the mesh needs to be updated. In both FEA templates, the update mesh button runs the re-mesh scripts. For fastening concept 1, the script assigns the chosen materials, updates the mesh mating conditions and mesh new fastener bodies. The

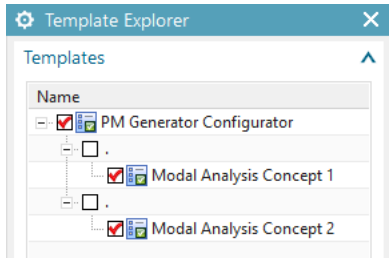


Figure 9.2: How to access the templates

script for fastening concept 2 assigns the chosen materials and updates the mesh mating conditions.

When the solver has successfully completed, the solution can be visualized as shown in figure 9.3. In this case, the five lowest modes can be shown within the template. By adding more post view templates, higher modes can be displayed.

Due to the great model scalability, the user has to carefully select a suitable mesh size to avoid unnecessary solver running time or inaccurate results. Varying the mesh size from 10 to 20 mm on the set-up shown below, has a huge impact on the solver running time, while the results almost remains the same.

Concept Comparison

The results in figure 9.3 and 9.4 are obtained with the user input as shown. In both cases, the profile dimensions are equal expect for the height, since the air gap in concept 1 is thicker than the insulation layer in concept 2. Both stator segments are fastened with three i-profiles each. As expected the lowest eigenfrequency for fastening concept 1 is significantly lower than for concept 2.

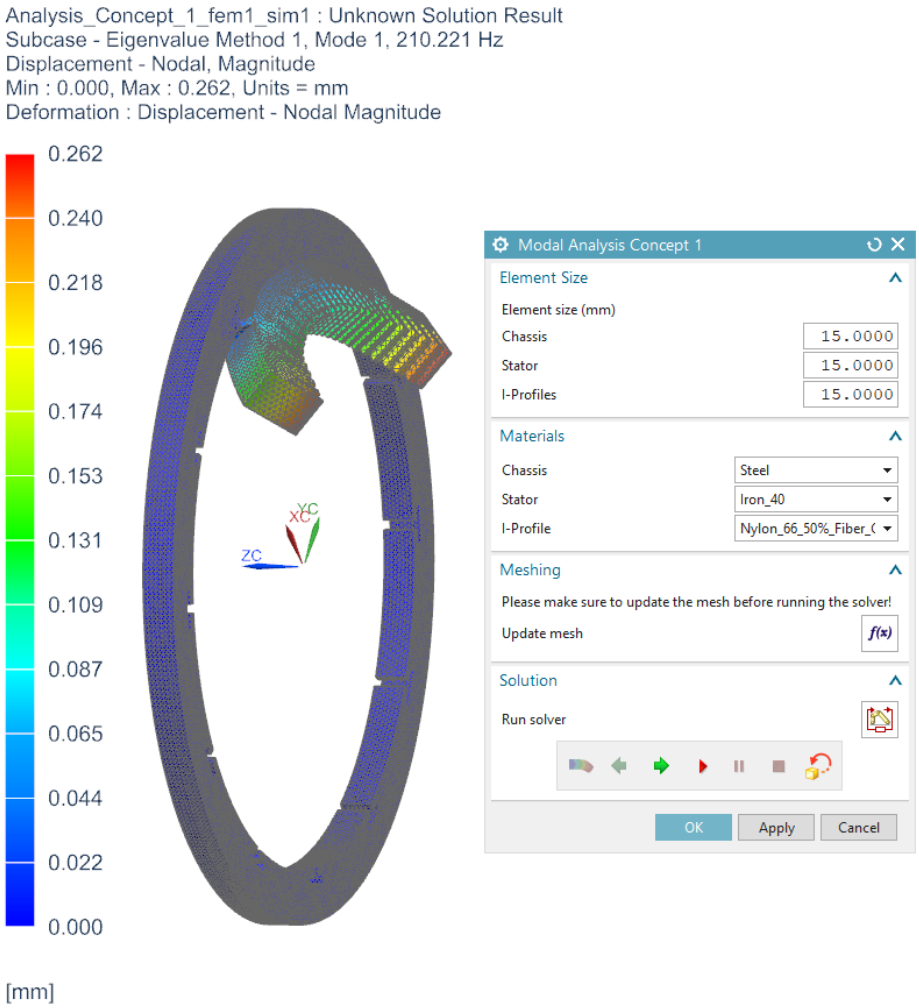


Figure 9.3: Analysis template fastening concept 1

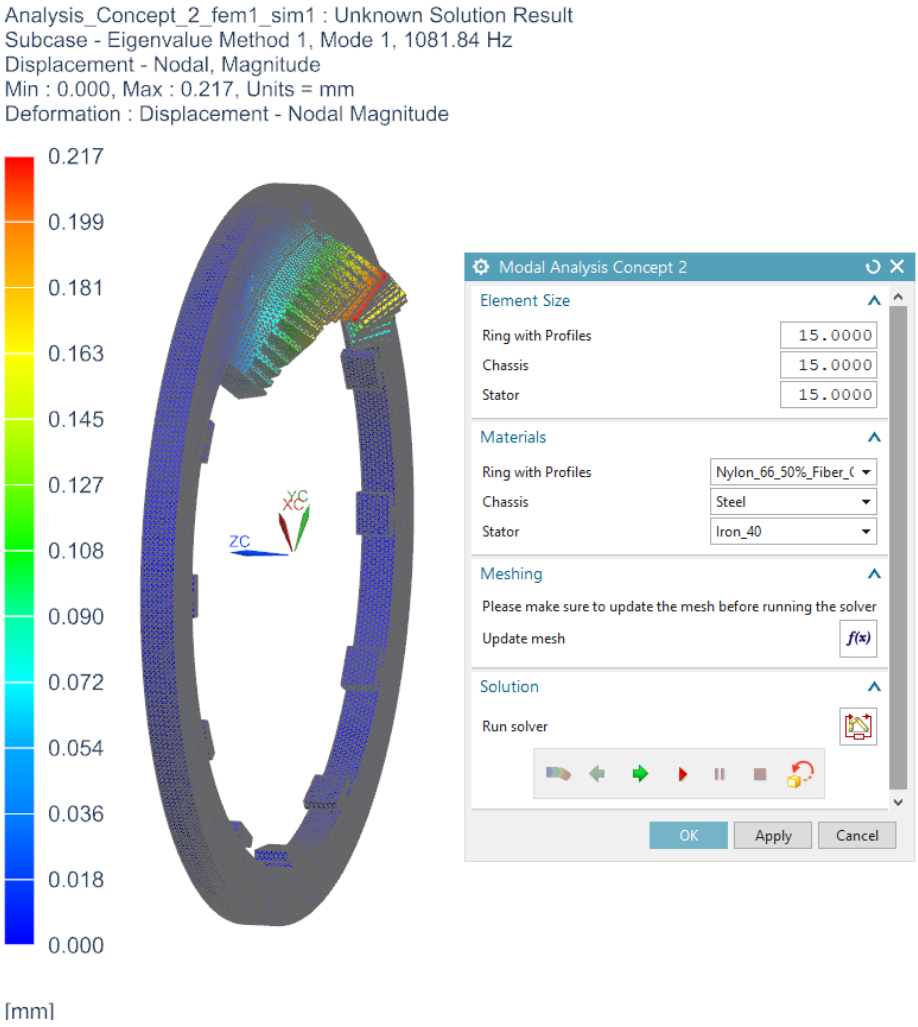


Figure 9.4: Analysis template fastening concept 2

Chapter 10

Discussion

This chapter will discuss the challenges, shortcomings, and possibilities for the PM generator configuration system, the software environment in Siemens NX and the ModHVDC project.

10.1 Siemens NX

10.1.1 Parametric Models

Parametric models are well supported in NX and provides great value in terms of knowledge-based model flexibility and component reusability. However, some challenges have been met during this thesis. Parametric assemblies that are controlled by numerous expressions, interpart expressions and wave links can be powerful in terms of flexibility and robustness, but can also be quite vulnerable. These models tend to be complex in terms of a high number of expressions and wave links, which can make it difficult to keep control of how the components are related to each other. Trying to fix errors that has been done at an early stage can be challenging and result in several broken wave links and model failures, that can take hours to fix. To avoid this problem, it is highly important to have some kind of system for the expression naming and grouping, and how the interpart expressions are distributed. For other people to take over the model and keep working on it, can be challenging if a proper system has not been implemented.

A challenge that was discussed during the project thesis, is that the pattern component tool can neither be selected as feature for another pattern command or be suppressed, which made it impossible to implement different number of phase sets, when using the pattern component tool. Even though this was solved by having all the coils per phase inside one part file, it lead to new challenges when the assembly drawing was created, since the automatically created part list had to be manually edited, and expressions had to be used in order to show the correct number of coils, magnets and fasteners.

As long as a proper modelling plan/system is used from the start, I will definitely recommend using Siemens NX for parametric modelling applications. It has shown to be a powerful tool with a lot of genius functionality that enables parametric models with a high degree of complexity and flexibility.

10.1.2 Product Template Studio

Product Template Studio is a powerful tool that makes parametric models a lot more user friendly, especially for people that does not have proper CAD experience. A problem that was introduced during the project thesis, is that the check functions does not update immediately after a value has been entered by the user. If the check functions could respond directly to input values, models with high flexibility could be more robust against failure due to illogical user input.

Another challenge that has been faced is related to how PTS access external files. For a file to be opened in PTS, the exact file path has to be entered. This becomes a problem if the model is opened in another computer, where the specified file path does not exist. For pictures (.bmp), this has can be solved by entering "Ug_find_file(<picture name>)" in the file path field. NX will then automatically search for the picture with the desired name at the local discs. However, the Ug_find_file approach is not supported for NX Open scripts, which means that the file path has to be re-entered each time the model is used on a new computer. In this case, it would be of great value if the Ug_find_file approach supported NX Open scripts, since the model is to be shared with several stakeholders.

10.1.3 Finite Element Analysis

A great advantage with Siemens NX is the selection of powerful solvers and the ability to seamlessly switch between the modelling and pre/post applications, which enables an iteration-based design approach. Due to the lack of electromagnetic solvers in NX, the only analysis that can be performed directly is a modal analysis. For the use of other structural solvers, a load case from each generator configuration has to be retrieved from ANSYS Maxwell. This makes it challenging to implement other structural analysis templates in NX, since the load cases are depending on the electromagnetic field. Trying to replicate the load cases from ANSYS in NX will also probably give poor accuracy compared to a coupled electromagnetic-structural analysis within ANSYS.

In reality, the eigenfrequency of a structure will be temperature dependent since the modulus of elasticity for materials does vary with temperature. Since Nastran SOL 103 does not take temperature loads into consideration, coupled thermal-modal analyses cannot be performed in NX. For the purpose of electrical machines, operating with relatively high temperatures in the stator, it would be of great interest to take temperature loads into consideration when running a modal analysis.

10.1.4 Product Template Analysis

In the same way as for CAD templates, Analysis templates brings great value in terms of wrapping a user-friendly GUI around an analysis setup. The templates enable the possibility to perform FE analyses, without having to spend unnecessary time on pre-processing. The most robust templates can even be used by people with limited experience on FEA. However, since the user does not have any direct interaction with the FE model, the tool is best suited for models that cannot be subjected to major geometrical changes. Since the old body references are deleted when the polygon models are updated within the FEM-file, the model has to be handled in a NX Open script, as in this thesis. Creating a script that successfully re-mesh, assign materials and update the mesh mating conditions for a simple modal analysis has shown to require a lot more work than setting up the analysis manually. In this case, the ModHVDC project is planning to consider several different design combinations, which will require a frequent use of the modal analysis templates, thus making it worth the time it takes to create the script.

Introducing more complex solvers and load cases on flexible models can be really challenging, since the loads can lose its reference faces each time the polygon bodies are updated. A test performed at the stator teeth showed no systematic naming of the tooth faces, when the number of teeth was changed in the model. Applying loads to each tooth in a script showed to be difficult as there were no clear pattern for how the faces were named.

10.1.5 Using the Visual Basic API

The NX Open API is a powerful tool that has been completely necessary in this thesis. It gives the programmer access to almost all functionality that NX offers, and it can be run in different ways, such as Journals and batch mode applications. The ”Getting started with NX Open” guide [2], the website `www.NXJournaling.com` and NX's ability record journals provides good help on how to get going. However, the guide is quite basic and are mainly focusing on modelling cases. More guidance and examples from FE applications would be of great value for people that want to work on automatic analysis setups in NX. It has also been discovered that the journal recording function does not generate code of some commands that exists in the API. Trying to record the ”update part list” command for assembly drawings will not generate any executable code. However, the parts list can actually be updated from a NX Open script, as shown in *PDF_Exporter.exe*. A further development of the journal recording function, making it able to record all possible commands that can be performed in a NX Open script would make it easier and more efficient to get known with the API and its possibilities.

10.1.6 Generation of Drawings

As presented in the previous chapter, the drawing generation approach did not work as expected when the model was subjected to geometrical changes. The main challenge is to

deal with the dimensions and scaling when the model is exposed to major changes. Using the ”suppress drafting object” function makes it easier to deal with features that comes and goes on a part. In this project however, the model is too flexible to make the approach successful.
It can also be discussed whether or not it is necessary to create detailed manufacturing drawings of the model at this early stage in the development process. Implementing the desired approach at a later stage, when the design space has been narrowed down and the model is less flexible, can result in successful sheets that meets the required standards introduced in chapter 2.5.

10.1.7 Generalization

Most of the challenges that has been discussed above is caused by the great model flexibility. At this stage, it should be considered whether it is favourable to keep the model as flexible as it is. Focusing on the development and testing of a successful prototype for the ModHVDC technology can be more important at this stage compared to having the ability to perform analyses for both the prototype and a full-scale model, since the technology has not been tested yet.

Using NX native in a project that includes several stakeholders has proved to be a bit inconvenient. Since the model has been continuously developed, new revisions have been manually updated to Onedrive. The other students have then manually downloaded the new revisions and deleted the old ones on their computers. Using a PLM system like Teamcenter [26] would make the workflow more efficient, as new revisions would be automatically available for the other students. In addition, it provides access to the old revisions, in case of a sudden model failure. When the ModHVDC project expands, a proper PLM system will be even more important to keep control of the workflow and different roles within the project.

Instead of creating one template around a complete generator design, another option could be to create robust templates of each generator component that can be included in a digital component library. Some of these templates should be based on standardized and existing components, including all available metadata. Building such a library would be of great value for the IEL institute, as it enables quick modelling of different generator designs with a great model accuracy.

10.1.8 ModHVDC Generator Concept

Intentionally, it was a hope that the prototype design space should be further narrowed down during this spring, based on a series of FE analyses performed in ANSYS Maxwell. However, due to the lack of analysis results from the IEL institute, no feedback has been given on the two concepts. Due to the uncertainty around how the electromagnetic field and load cases will be affected by segmenting the stator, more resources should be devoted to the electromagnetic field of study, before the parametric model is further developed.

Chapter 11

Conclusion and Further Work

11.1 Conclusion

In this thesis, a simple ETO system has been created based on a parametric model of a PM generator, using Siemens NX and its respective API. The parametric model should be scalable from a small-size prototype to a full-size windmill generator. The system should (if possible) be able to automatically generate manufacturing drawings of the generator components and contain templates for automatic coupled thermal-structural analyses. Due to the lack of electromagnetic solvers in NX, only modal analysis templates were created. Unfortunately, a few tests showed that the modal analysis solver Nastran SOL 103 does not take thermal loads into account, thus neither support coupled thermal-structural analyses.
A system validation was performed, using several different generator configurations. The validation uncovered problems with the manufacturing drawings after the model had been subjected to major changes. The modal analysis templates however, worked as expected. Even though the drawing generation approach did not work as expected for a highly flexible model, it showed a great potential for more standardised and less flexible parametric models.

The creation of this system has shown that Product Template Studio is a powerful tool with a great potential as part of an ETO-system. Using template-based models and analyses enables people/customers without proper experience with Siemens NX to configure and analyse a product without having to spend time learning how to use the software, which brings great value in terms of time that can be spent on other important tasks.

11.2 Further Work

Since the ModHVDC project is in an early development phase, several aspects are considered as a natural extension of the work presented in this thesis. The aspects are listed below:

- The design space for the prototype should be narrowed down, based on feedback and analysis results from the IEL institute.
- After narrowing down the design space, the generator chassis with all its remaining components should be included in the 3D model.
- Implement structural analysis templates based on the load cases obtained from ANSYS Maxwell.
- Explore other, more robust approaches for automatic generation of manufacturing drawings.
- Test the possibilities to replace the base views onto the sheets in a NX Open script, after the views has been automatically scaled.
- Explore the possibility to retrieve analysis results directly from ANSYS into Siemens NX.
- Study the material characteristics with directional oriented fibres, for the fastener profiles.
- Since the ModHVDC project proposal includes stakeholders from several countries and institutions, a PLM system should be set up, providing a platform for data sharing and project control between the different stakeholders. The system should support integration of both NX and ANSYS, like e.g. Teamcenter from Siemens.

Bibliography

[1] Transcontinental and global power grids | JRC Smart Electricity Systems and Interoperability. https://ses.jrc.ec.europa.eu/transcontinental-and-global-power-grids, (Accessed April 21, 2019).

[2] Getting Started with NX Open. https://docs.plm.automation.siemens.com/data_services/resources/nx/12/nx_api/common/en_US/graphics/fileLibrary/nx/nxopen/nxopen_getting_started_v12.pdf, (Accessed February 10, 2019).

[3] Øystein Krøvel. *Design of large permanent magnetized synchronous electric machines : low speed, high torque machines, generator for direct driven wind turbine, motor for rim driven thruster*, volume 2011:25 of *Doktoravhandlinger ved NTNU (online)*. Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering, Department of Electric Power Engineering, Trondheim, 2011.

[4] M. Valavi, A. Nysveen, R. Nilssen, R. D. Lorenz, and T. Rølvåg. Influence of Pole and Slot Combinations on Magnetic Forces and Vibration in Low-Speed PM Wind Generators. *IEEE Transactions on Magnetics*, 50(5):1–11, May 2014.

[5] Pål Keim Olsen. ModHVDC Research Application. Technical report, NTNU et. al., October 2018.

[6] Wikipedia. Capital expenditure. https://en.wikipedia.org/w/index.php?title=Capital_expenditure&oldid=866634765, 2018 (Accessed December 1, 2018).

[7] Z. Q. Zhu, Z. P. Xia, L. J. Wu, and G. W. Jewell. Analytical Modeling and Finite-Element Computation of Radial Vibration Force in Fractional-Slot Permanent-Magnet Brushless Machines. *IEEE Transactions on Industry Applications*, 46(5), September 2010.

[8] Gang Lei, Jianguo Zhu, and Youguang Guo. *Multidisciplinary Design Optimization Methods for Electrical Machines and Drive Systems*. Power Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[9] Zigu. Engineer-to-Order Definition | Operations & Supply Chain Dictionary. https://www.mbaskool.com/business-concepts/operations-logistics-supply-chain-terms/15193-engineer-to-order.html, (Accessed April 8, 2019).

[10] NX. https://www.plm.automation.siemens.com/global/en/products/nx/, (Accessed April 9, 2019).

[11] Parametric Modelling, Process, Advantages and Parametric Modelling Tools. https://www.designtechsys.com/articles/parametric-modelling, (Accessed April 9, 2019).

[12] Siemens Documentation: Programming Tools. https://docs.plm.automation.siemens.com/tdoc/nx/10/nx_api/#uid:index, (Accessed April 6, 2019).

[13] NX Journaling. http://nxjournaling.com/.

[14] Gewarren. Overview of Visual Studio. https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide, (Accessed May 5, 2019).

[15] Håvard Bergeland, Johnny Hansen, and Eyolf Herø. *Tegning og dokumentasjon*. TIP-SERIEN. Gyldendal Norsk Forlag AS, 1 edition, 2006.

[16] K.L. Narayana, P. Kannaiah, and K. Ventkata Reddy. *Machine Drawing*. New Age International Publishers, third edition, 2006.

[17] Siemens. Element Library Reference. https://docs.plm.automation.siemens.com/data_services/resources/nxnastran/10/help/en_US/tdocExt/pdf/element.pdf, (Accessed April 30, 2019).

[18] I.A.T.K. P. Goncharov. *Engineering Analysis with NX Advanced Simulation*. Lulu.com, 2014.

[19] Siemens. Basic Dynamic Analysis User's Guide. https://docs.plm.automation.siemens.com/data_services/resources/nxnastran/10/help/en_US/tdocExt/pdf/element.pdf, (Accessed April 30, 2019).

[20] U. Shipurkar, H. Polinder, and J. A. Ferreira. Modularity in wind türbine generator systems — Opportunities and challenges. In *2016 18th European Conference on Power Electronics and Applications (EPE'16 ECCE Europe)*, pages 1–10, September 2016.

[21] E. Spooner and A. Williamson. Modular, permanent-magnet wind-turbine generators. In *IAS '96. Conference Record of the 1996 IEEE Industry Applications Conference Thirty-First IAS Annual Meeting*, volume 1, pages 497–502 vol.1, October 1996.

[22] Overview of materials for Nylon 66, 50% Glass Fiber Filled.

http://matweb.com/search/DataSheet.aspx?MatGUID=12fd937e0c754d25babfc6268703b67a, (Accessed April 05, 2019).

[23] Modellering 1 – Leksjon 11 | NX-Portalen. http://nxportalen.com/blog/2014/08/07/modellering-1-leksjon-11/, (Accessed February 05, 2019).

[24] Siemens Documentation: NX Open for .NET. https://docs.plm.automation.siemens.com/tdoc/nx/11/nx_api#uid:xid1162445:index_nxopen_prog_guide:id1142076:id1253956:genid_for_net_22_1442, (Accessed February 25, 2019).

[25] Exporting Drawings to .pdf Files Using NX Journal | NX Journaling. http://nxjournaling.com/content/exporting-drawings-pdf-files-using-nx-journal, (Accessed February 05, 2019).

[26] Teamcenter. https://www.plm.automation.siemens.com/global/en/products/teamcenter/, (Accessed May 30, 2019).

[27] NX Journaling. Regard - how to change below code for step file export. https://nxjournaling.com/content/regard-how-change-below-code-step-file-export, 2018 (Accessed December 10, 2018).

Appendix

A

Extract from Project Assignment

This chapter is an extraction from the project assignment, describing the modelling process of the generator.

A.1 Parametrizing the Generator with PTS

In order to achieve a lean design and analysis process, a parametrized 3D model has been developed. This CAD model makes it possible to perform an iteration-based optimisation process of the generator design, or to perform analyses and evaluations of several design approaches in parallel.

The parametric model is presented with Product Template Studio (PTS) in Siemens NX. PTS is a tool for creating robust models that can be easily reused, even by people with little CAD experience. By simply choosing the desired specifications from the PTS menu a complex 3D model can be generated. The PTS user interface used in the ModHVDC project is called the configurator.

A great benefit with reusable parametric models is that they can contain important design rules, while having the possibility to change certain parameters. Taking a closer look at the generator configurator, it contains the possibility to change almost all important parameters and generate the model within seconds, while the input parameters are defined by rules and dependencies, so that the user cannot type in values that does not make sense.

The parametric model is based on the generator designed by Ø.Krøvel [3] and has subsequently expanded its features.

A.1.1 Parametrizing the Generator Components

Since the ModHVDC project aims for a lean design process, the 3D-model has to be as flexible as possible in terms of design changes, in order to bring value in the design and simulation process. Thus making it extremely important to have a robust work flow in the modelling process and to properly know the different tools within NX.

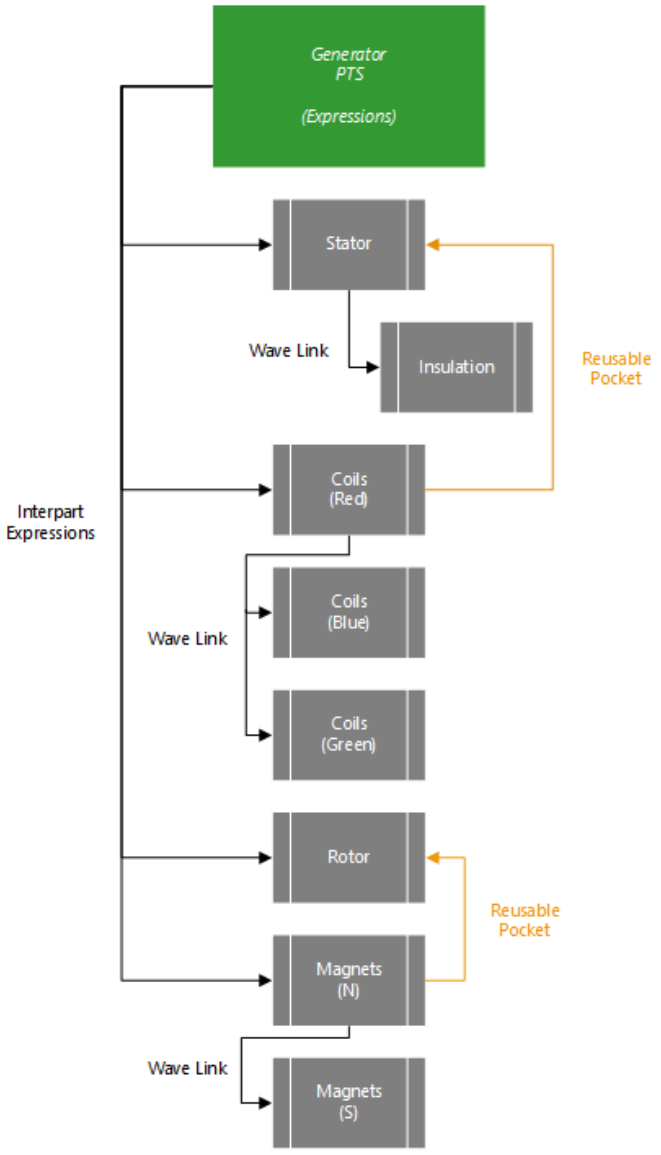


Figure A.1: Assembly chart with an overview of the information flow

The parametrization process has been done in several systematic steps to eventually obtain a fully robust model. First, the different parameters have to be implemented as *Expressions* in NX, in order to recognise them in the modelling process and for allowing automatic model updates from inputs in the PTS menu. After defining the necessary user expressions, the different models have to be created, using input parameters defined in the expressions.

A robust plan for how the components should be modelled and interact with each other was then developed. Since the assembly contains a lot of components, it is important that they behave correctly relative to each others. Figure A.1 shows the initial plan for the information flow between the components. At this stage, it is rather important to make a good plan on how the different components should be parametrized. The rotor, for example, should always update when the dimensions of the stator, magnets or air gap changes. Instead of using a direct input for the rotor, its dimensions are controlled by the inner diameter and depth of the stator as well as the air gap and magnet height.

Expressions

Defining expressions is considered the most important part of a parametrization process in NX. The rules that govern the entire design are defined as expressions and can be used to implement limitations and constraints, making it possible to create a totally rule-based PTS model. Through expressions defined as option variables, boolean variables, lists and if/else-statements, different features and components can be controlled in complex assemblies.

As shown in Figure A.1, the top assembly takes the user input from the PTS menu and stores it as expression variables. The expressions will then be used as parameters in all the sub-components and assembly constraints. The expressions can be found in the Appendix, and only the most important ones will be discussed.

Logic Statements

The expressions menu in NX supports the use of *if* and *else* statements, which is highly useful for a PTS. In combination with other integrated features in NX, the use of logics enables implementation of design rules or the possibility to change between different sub-solutions for a component or assembly to state a few examples.

In the generator PTS, *if* and *else* statements are used for selecting different lists, values and suppressing different features, feature groups and components based on input from the user. The example below shows an expression that gives feedback to the user on how many coils that fits the stator, based on the expression *Max_Number_of_Coils_Calc*, that compares the coil width with the circumference of the stator.

```
Max_Number_of_Coils_Limit=if(Max_Number_of_Coils_Calc < 24)
 ↪  12 else  if(Max_Number_of_Coils_Calc < 36) 24 else
 ↪  if(Max_Number_of_Coils_Calc < 48) 36 else if
 ↪  (Max_Number_of_Coils_Calc < 60) 48 else
 ↪  if(Max_Number_of_Coils_Calc < 72) 60 else
 ↪  if(Max_Number_of_Coils_Calc < 84) 72 else
 ↪  if(Max_Number_of_Coils_Calc < 96) 84 else 96
```

Interpart Expressions

For all sub-components to update every time an input variable changes from the PTS menu, Interpart Expressions needs to be linked to the sub-components. Figure A.1 shows how the stator part-file get its variables from the top assembly file (*Generator.prt)* via interpart expressions. To ensure complete robustness, all features used in a component should be taking values from interpart expressions rather than constant values defined within the part. By simply pressing ”Create/Edit Multiple Interpart Expressions”, the user can choose a desired part and link the needed expressions into your work part. The interpart expressions will then update when the original expression changes its value.

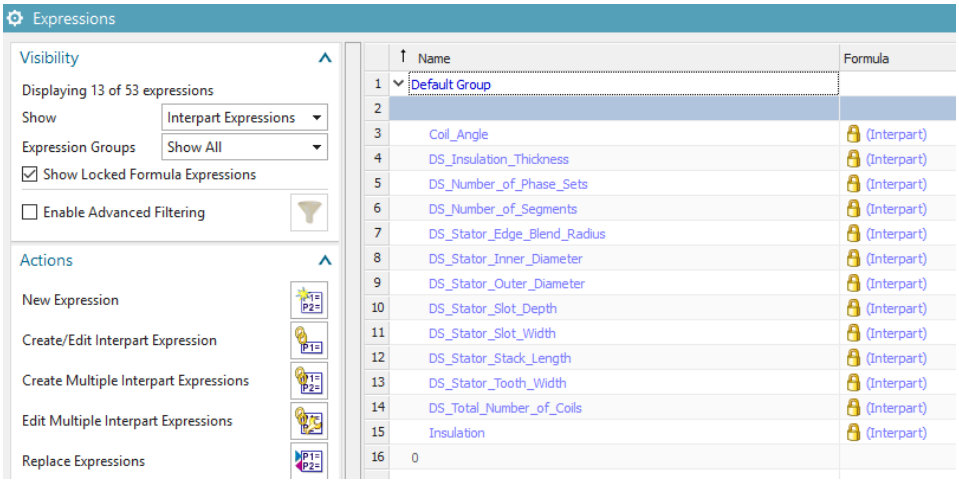


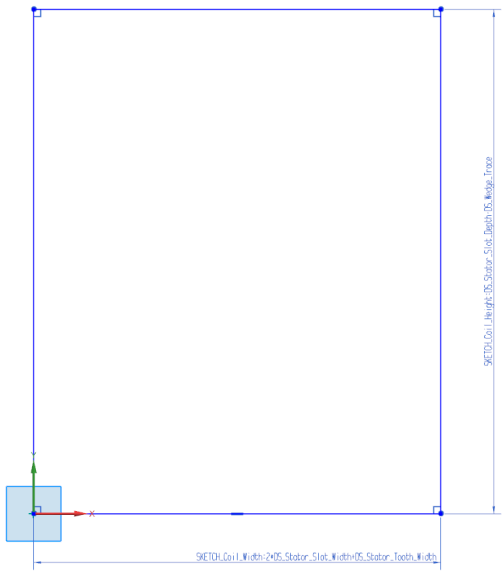
Figure A.2: Interpart Expressions linked from *Generator.prt* to *Stator.prt*

A.1.2 Modeling the Generator Components

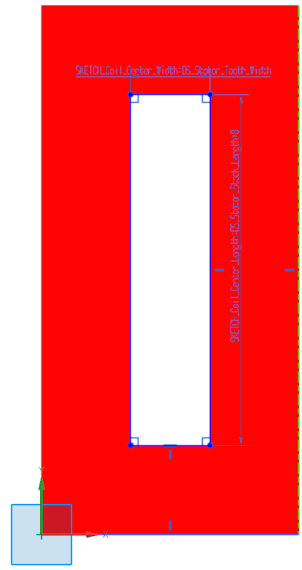
Coils

The modelling process started with the coils. First, the outline of the coils was sketched and extruded (Figure A.3). An outline of a stator tooth was then sketched on the centre of the extruded body, and subtracted, making a hole for the stator teeth (Figure A.3b). To

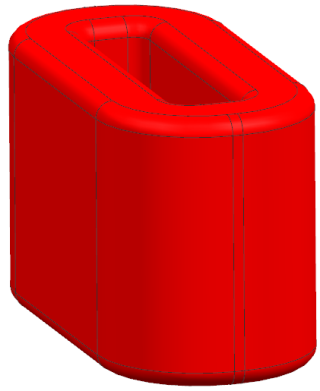
ensure that the tooth hole always stays within the centre of the coil body, the centre lines of the sketch were constrained with the centre lines of the solid body. Edge blends were then added to all edges, making a realistic coil shape (Figure A.3c).



(a) Sketch of the coil outline



(b) Tooth hole subtracted from the coil body



(c) Complete coil

Figure A.3: Coil modelling process

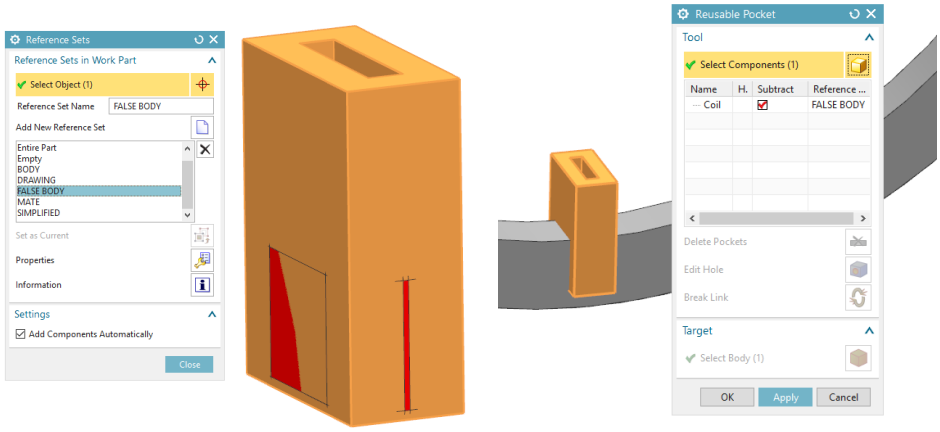
Since the coils are operating in three different phases, three different part-files are needed in order to distinguish them both visually and for analysis setup. The coil body are there-

fore Wave Linked into two other part-files *coil_blue.prt* and *coil_green.prt* and assigned new colours.

Stator

As for the coils, the outline for the stator was first sketched. The sketch was then extruded into a solid cylinder, making the stator (Figure A.6a).

Since the slot dimensions can be changed from the PTS menu, the coils need to update when the slots are being modified. To ensure that all the coils always stays within their slots, a coil is used for subtracting the slots into the stator. This is done by creating a ”False body” within the coil part-file, which is hidden in a reference set named ”FALSE BODY”. The false body is then used as a subtraction tool for the coil slots in the stator, by using the feature ”Reusable Pocket” as shown in Figure A.4.



(a) False body hidden in reference set **(b)** False body subtracting from stator

Figure A.4: False body subtraction in the stator

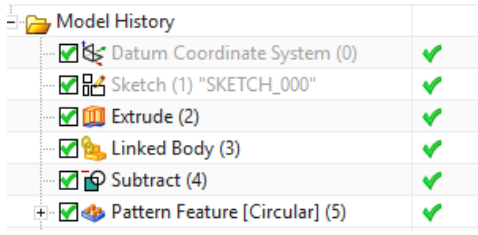
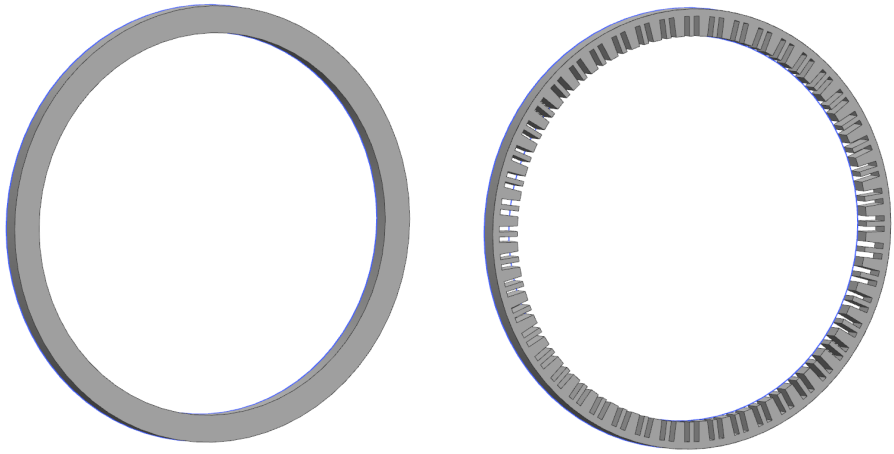


Figure A.5: Model history stator

The reusable pocket tool simply Wave Links the false body geometry into the stator part-file and uses it as a subtraction tool. After creating one coil slot in the stator, ”Pattern Feature” is used to copy the coil slots around the stator, taking the given number of coils as input (Figure A.5). Figure A.6b shows the stator after all the coil slots have been made.



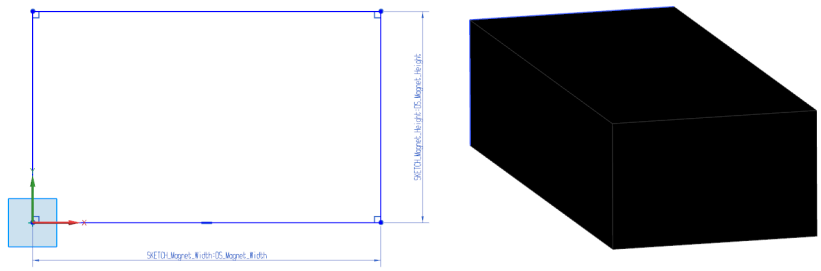
(a) Stator without coil slots **(b)** Stator with coil slots

Figure A.6: Stator before and after false body subtraction

Magnets

The magnets have a simple geometry, making them very simple to model. The outline is sketched before being extruded.

Sine the magnets are mounted with the south and north poles every other time, two different parts are needed for the same reason as the coils. The magnet body is Wave Linked into another part-file *Magnet_S.prt* and assigned another colour.



(a) Sketch of the magnet outline **(b)** Stator with coil slots

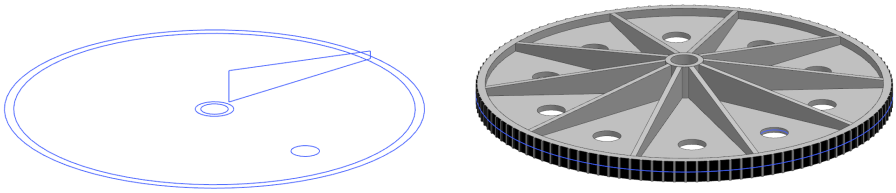
Figure A.7: Magnet modelling process

Rotor

Like the other components, the rotor is completely parametrized. Figure A.8a shows an overview of all the sketches making the rotor. These are all parametric and constrained, but NX cannot show the parameters on several sketches at the same time. The sketch parameters can be found in Appendix

After sketching, the tools extrude and pattern feature has been used to create the solid body of the rotor. As previously mentioned, the rotor diameter is controlled by the inner diameter of the stator, the magnet height and the air gap between the stator and the magnets. The centre hole diameter and stiffener thickness can be changed by the user from the configurator. The cut-out diameter is parametrized and changes with the rotor diameter.

To make the magnet slots, reusable pocket has again been used. This time without a false body hidden in the reference sets. In this case, the magnet body itself is used as the tool body. Figure A.8b shows the complete rotor.



(a) Overview of the rotor sketches **(b)** Complete rotor

Figure A.8: Rotor modeling process

A.1.3 Creating the Generator Assembly

The main challenge creating a generator configurator is to successfully obtain a fully robust assembly where all components are acting as expected, relative to each other. Assembly constraints are key for making components behave properly in relation with each other. In this case, all components can change their dimensions, and some can be added/removed (number of magnets and coils). As a result of this flexibility a lot of faces will be changed or deleted when something is being modified in the configurator, making the different constraint possibilities significantly smaller.

Adding the Coils

The simplest way of getting full control of the coils in the assembly, is to make the three different coil files containing all the number of coils that belong to the given phase. This means that only three coil part-files will be added to the *Generator.prt* assembly, containing several coils within each part. There are several reasons for doing it this way instead

of adding one component file per coil to the assembly, which will be discussed further in Section A.2.

Since the coils are added as one component per phase in the top assembly, the setup for coil distribution is done within the coil part-file. Figure A.9 shows how the coil part-files are prepared before being added to the assembly. First, two reference points are added in a sketch, as shown in Figure A.9a. The first point is placed on the x-axis and in the centre of the coil width, by parametrization. This point will be used as reference for an assembly constraint later. The second point is defining the centre point for the coil distribution, thus also for the generator. This point will be used as reference for the pattern feature tool (distributing the coils) and for an assembly constraint. As shown in Figure A.9a, the second point is also fully parametrized. The first step of the coil distribution process is to define the distribution of the phase sets. This is simply done by using the pattern feature tool with the following input:

- Selected feature: Coil body
- Specified vector: z-axis
- Specified point: Parametrized centre point
- Spacing: Count and span
- Count: Number of Phase Sets
- Span angle: 360°

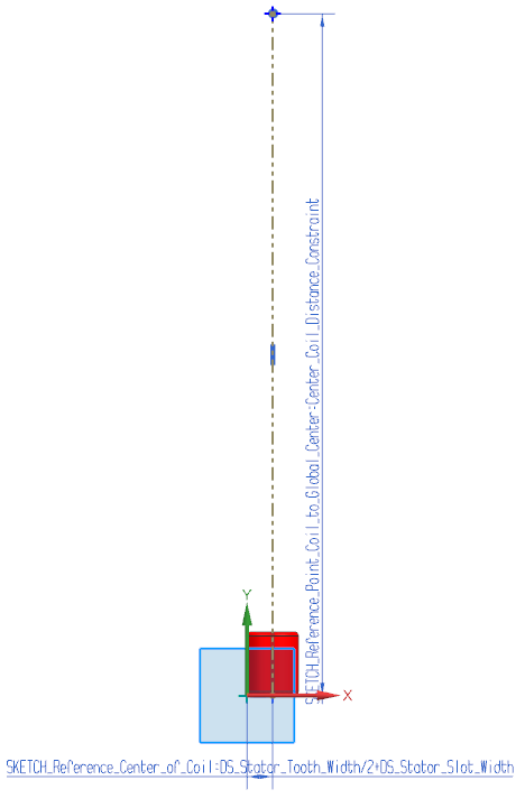
Figure A.9b shows the coil phase distribution.

The next step is to distribute the rest of the coils within each phase set. This is also done by using the pattern feature tool with the following input:

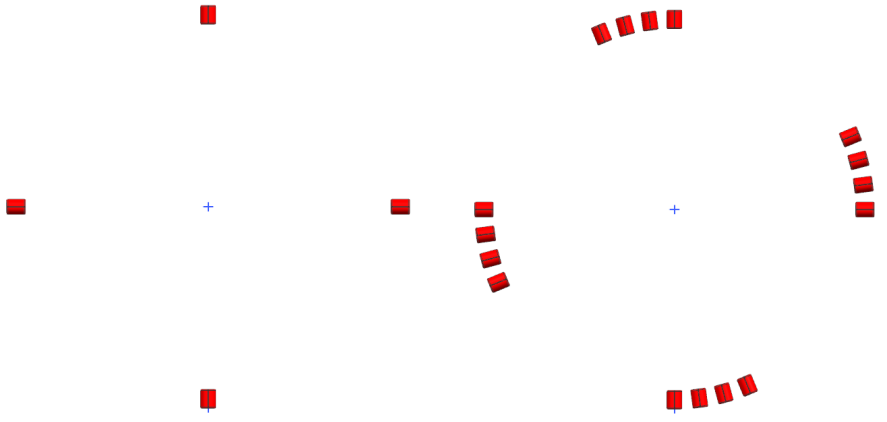
- Selected feature: Coil Body + previous Pattern Feature execution (Phase distribution)
- Specified vector: z-axis
- Specified point: Parametrized centre point
- Spacing: Count and pitch
- Count: Number of Coils per Phase
- Pitch angle: 360°/(Number of Coils)

The final distribution of all the red coils (Phase 1) are shown in Figure A.9c. In the final assembly, the two other phases (blue and green) will be added, making an even distribution of coils.

The exact same procedure is then executed for the two other coil part-files.



(a) Parametrized reference point for pattern feature and assembly constraints



(b) Phase set pattern

(c) Complete phase pattern

Figure A.9: Coil pattern

After organising the coil distribution within the part-files, the coils were added to the top assembly. At this point the reference points defined within the coil part-files are key to obtain a robust assembly adaptable for changes. In order to access these points, the reference sets are changed from ”MODEL” to ”Entire Part”. As shown in Figure A.10 the local points and datum coordinate systems are visible in the top assembly after changing the reference sets. Since the coil faces and edges change every time the model updates, it is not recommended to use them as reference objects in assembly constraints. Best practice is therefore to use the local reference points, which is parametrized and not affected by changes in faces and edges.

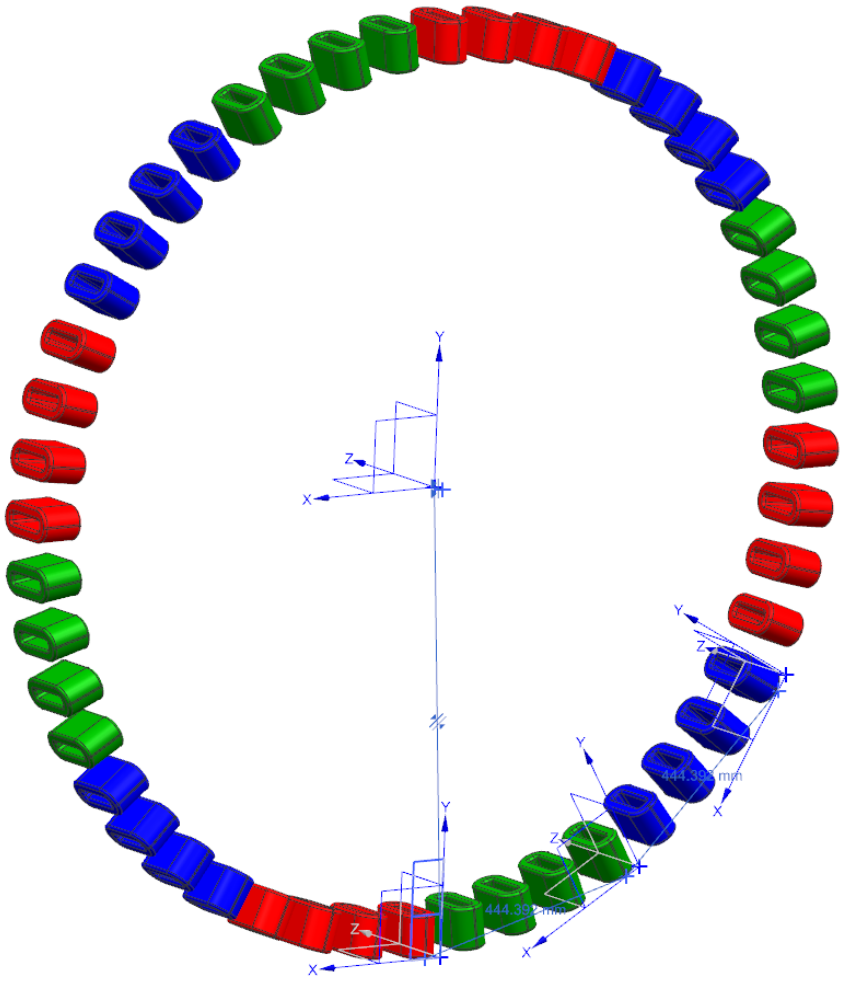


Figure A.10: Assembly Constraints between coils and global datum coordinate system

The first step in the process of adding assembly constraints was to align all the local centre points defined in the coil part-files with the z-axis in the global datum coordinate system. Figure A.10 shows the centre points which are aligned with the global z-axis. After aligning the centre points, distance constraints were defined between the three original coils (The coils having a local datum coordinate system in Figure A.10). These constraints were defined between the local reference points placed in the local x-axis within the coil part-files (See Figure A.11). The distance constraints are parametrized and ensures that the distance between all the coils are equal.

To ensure that the coils cannot rotate around the z-axis, a parallel constraint was added between local xz-plane for the red coils and the global xz-plane in the assembly (Figure A.10).

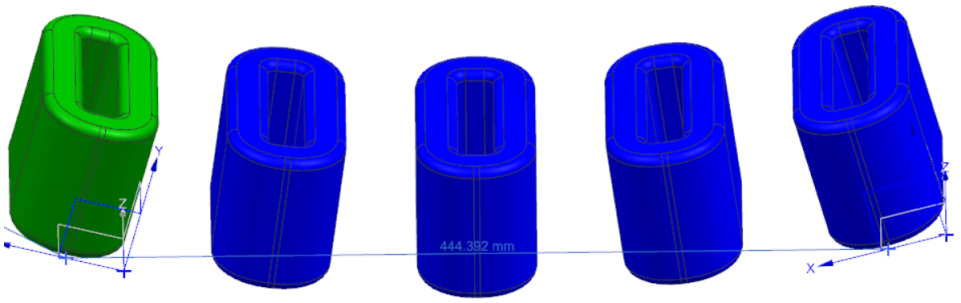


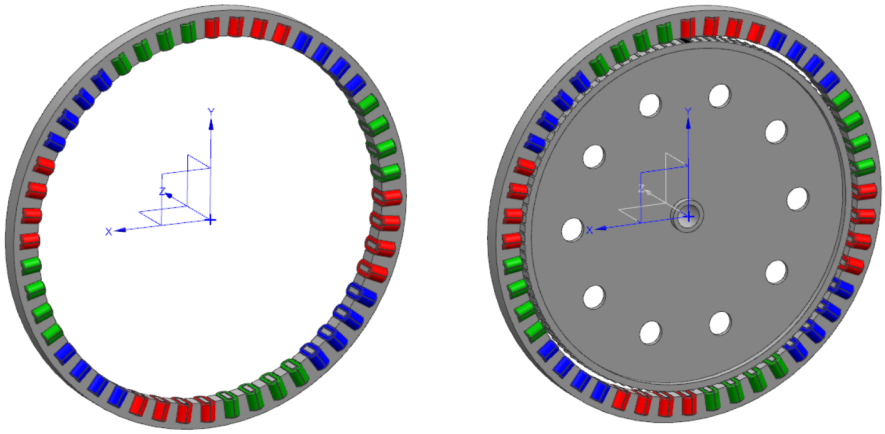
Figure A.11: Distance constraints between the coils

Adding the Stator and Rotor

The process of adding the stator and rotor was less complex than for the coils. Since the coils are used as the tool for making the stator slots, they will always fit perfectly within the slots.

A fixed constraint was first added, making sure that the stator is completely fixed. The coils were then centred between each side of the stator by using centre constraints.
Figure A.12a shows the stator interacting perfectly with the coils.

After adding the stator, the rotor was added using a aligned constraint referenced to the z-axis in the local datum coordinate system for the stator and the z-axis in the global coordinate system. A distance constraint was also set between the absolute centre point i the rotor and the global coordinate system (Figure A.12b). The distance was set to be half of the stack length.



(a) Stator completely added **(b)** Stator and rotor completely added

Figure A.12: Stator and rotor added to the assembly

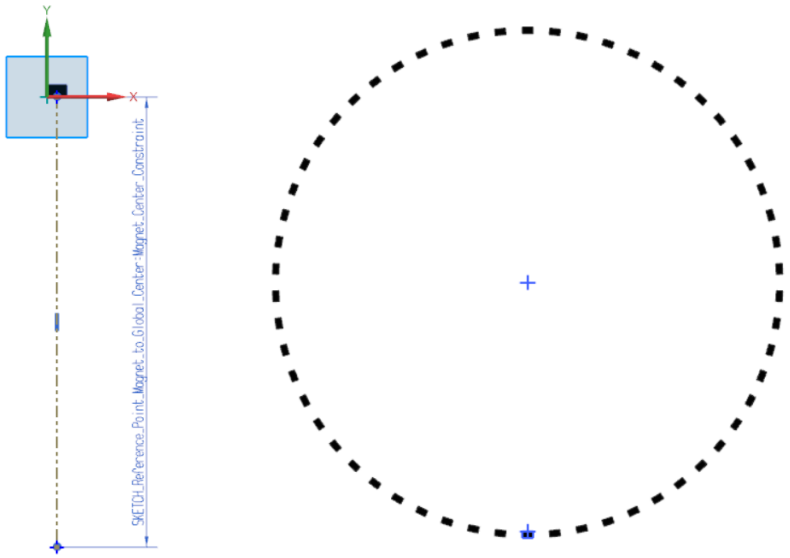
Adding the Magnets

The magnets are added in the same way as the coils, using one part-file in the assembly that contains all the magnets with a given polarity. As shown in Figure A.13a, two reference points are added in the sketch, parametrized in the exact same way as for the coils. Since there are only two different magnet parts, the distribution process is slightly simpler compared to the coils. Yet again the distribution process is done by using pattern feature, this time with the following input:

- Selected feature: Magnet body
- Specified vector: z-axis
- Specified point: Parametrized centre point
- Spacing: Count and span
- Count: (Number of Magnets)/2
- Pitch angle: 360°

Figure A.13b shows the distribution of the north pole magnets.

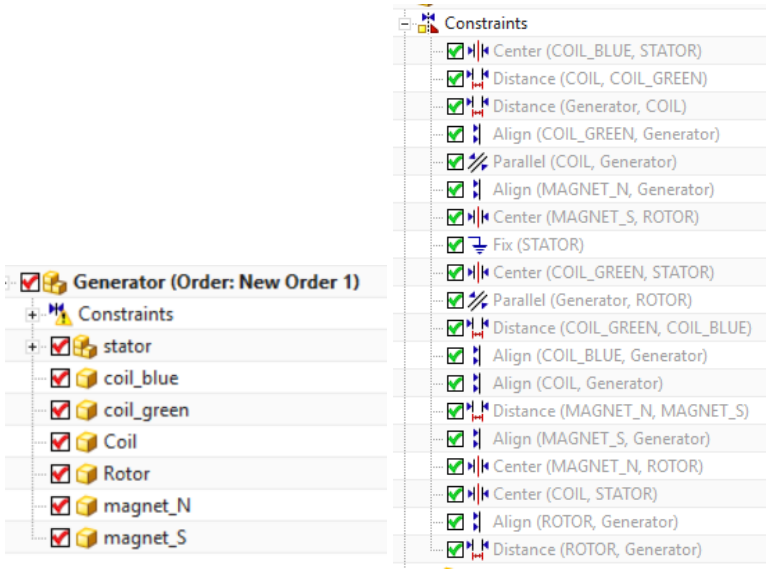
After organising the magnet distribution within the part-files, the magnets were added to the top assembly, using the same technique for defining assembly constraints as for the coils. The only difference is that the magnets is fastened to the rotor instead of the stator.



(a) Parametrized reference point for pattern feature and assembly constraints

(b) Complete pattern for the north pole magnets

Figure A.13: North pole magnet pattern



(a) Assembly hierarchy

(b) Assembly constraints

Figure A.14: Component hierarchy and assembly constraints

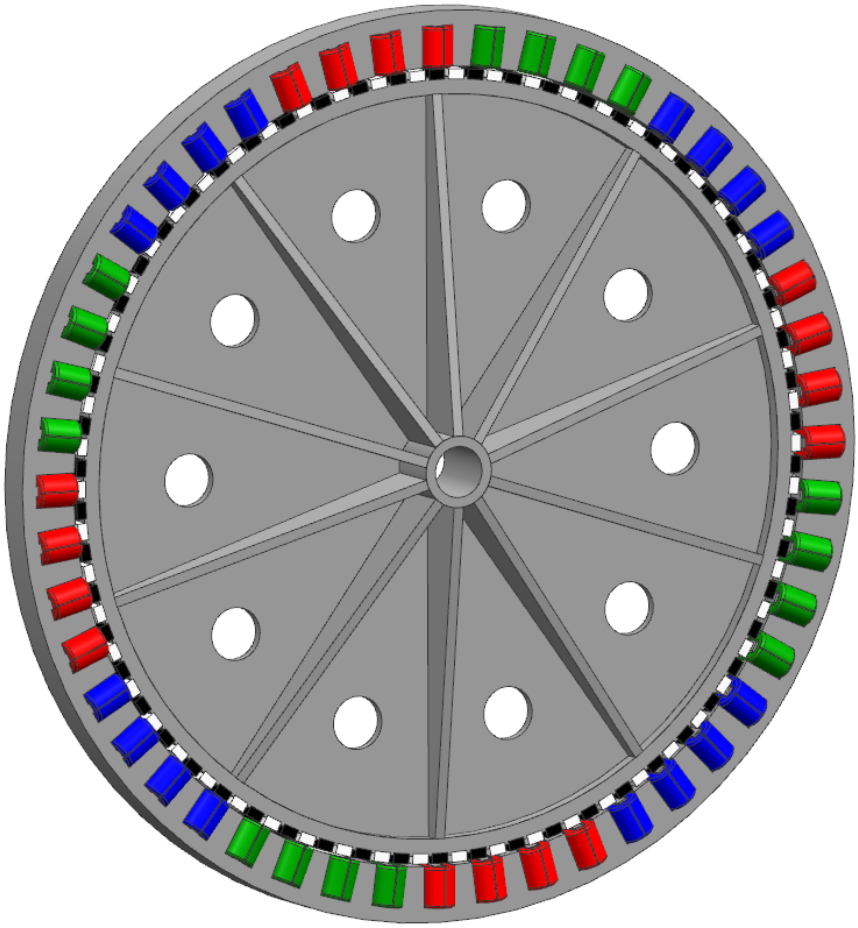


Figure A.15: Generator assembly

A.1.4 Segmentation and Insulation

One of the main purposes of the parametrized 3D-model is to perform different analyses with a segmented stator, for testing out different fastening and insulation methods. In this case, the user should be able to choose between 2, 4 or 8 segments, depending on the number of phase sets chosen for the generator. For analytical purposes, the user should also have the possibility to choose between having insulation on the segments or not, in addition to change the insulation thickness and adding a radius on the stator edges.

Segmentation of the Stator

For dividing the stator into segments, the split body command was used. A datum plane was first defined based on a parametrized angle, making sure that the stator is split exactly in between two phase sets. Another datum plane was then defined perpendicular to the first one, making it possible to get 4 segments. Yet another two datum planes were defined perpendicular to each other and with a $45°$ angle relative to the other two planes, thus making it possible to get 8 segments. The split body command was then executed with all the datum planes as splitting tools. Since the number of segments should be able to change based on input in the configurator, the different split body commands are suppressed by expressions.

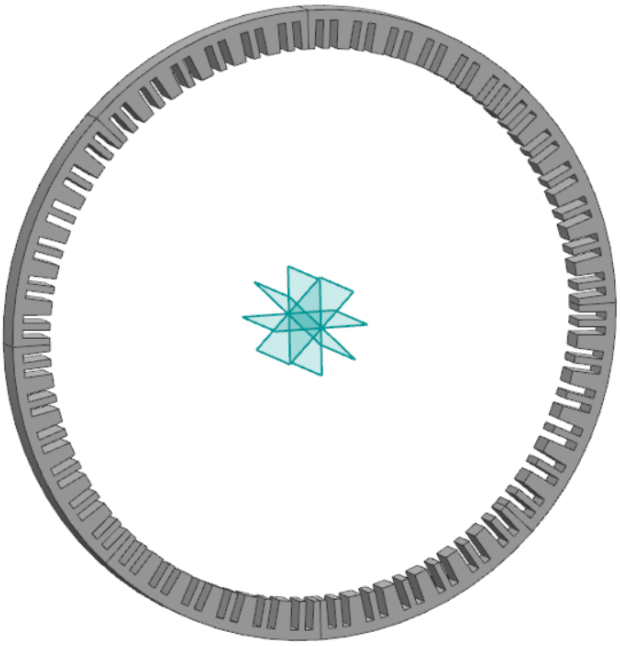


Figure A.16: Stator split into 8 segments

A.1.5 Product Template Studio

In Product Template Studio Author, the user interface of the generator configurator is created. The interface can be customised by using tabs and groups, making it possible to create sub-menus for more complex models or assemblies. Groups can be hidden or shown when opening the menu based on how the creator wants the menu to look. By adding expressions to the interface menu, they can either be added as an input value that can be changed or just as a read-only value. It is also possible to define check options which gives the user a warning if a wrong value has been entered. NX Journal scripts can be run by adding an ”action” button to the interface, which opens up for automation of more complex processes, like exporting files into other formats than `.prt` for example.

When the PTS Author is opened, a preview of the interface is shown to make the configuration process more efficient.

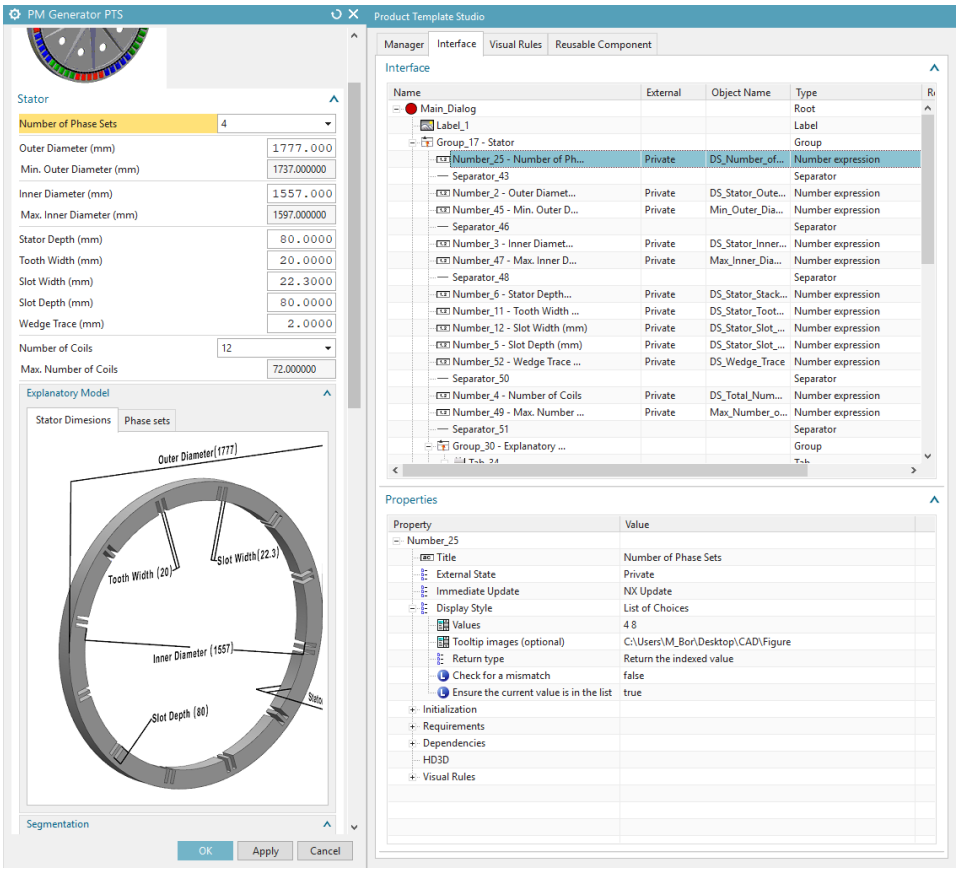


Figure A.17: Overview of the PTS Author environment

Creating an Interface

From the PTS Explorer (Figure A.18), controls and expressions can be added by right-clicking the desired element and clicking ”add”. The element will then appear in the interface in the preview window. Inside the interface, the different elements can be dragged and dropped as the user wants. When an expression is added to the interface, several options for presentation and input can be picked depending on the purpose of the expression. These options are: Keyin, List of Choices, Linear Dimension, Angular Dimension, Scale, Spin, Label, Checkbox, Readonly Text, Simple Keyin and List Expression. Title, display style and immediate update has to be defined for all options.

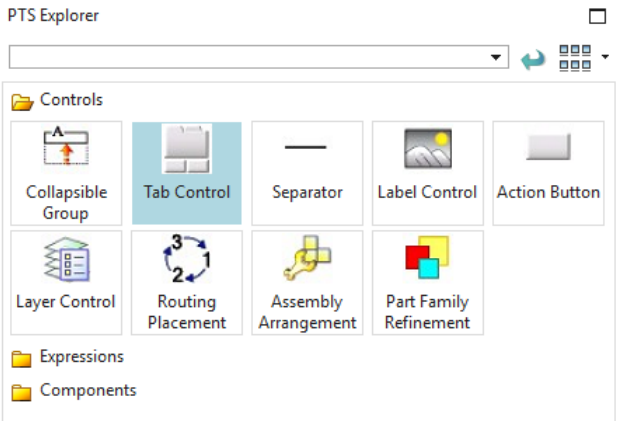


Figure A.18: Interface options in PTS Author

The Generator Configurator interface has been divided into four sub-menus: Stator, Rotor, Magnets and Analysis (Figure A.19).

In the stator menu, all the inputs that affect the stator can be changed. The number of coils option is also added under this menu, since a change in the number of coils are affecting the number of teeth in the stator. When adding a check option for the coils’ inner- and outer stator diameter, the check status will not update before the ”Apply” button is pressed. An alternative solution to the check option was to create an expression that calculates the minimum outer diameter and maximum inner diameter, and display the max/min values in a ”read only” display right below the input. Rule implementation was, via the check status or max/min values, successfully applied to the rest of the inputs, making sure that user cannot use any wrong input. Two explanatory models are added, making sure that the user understand all the inputs.

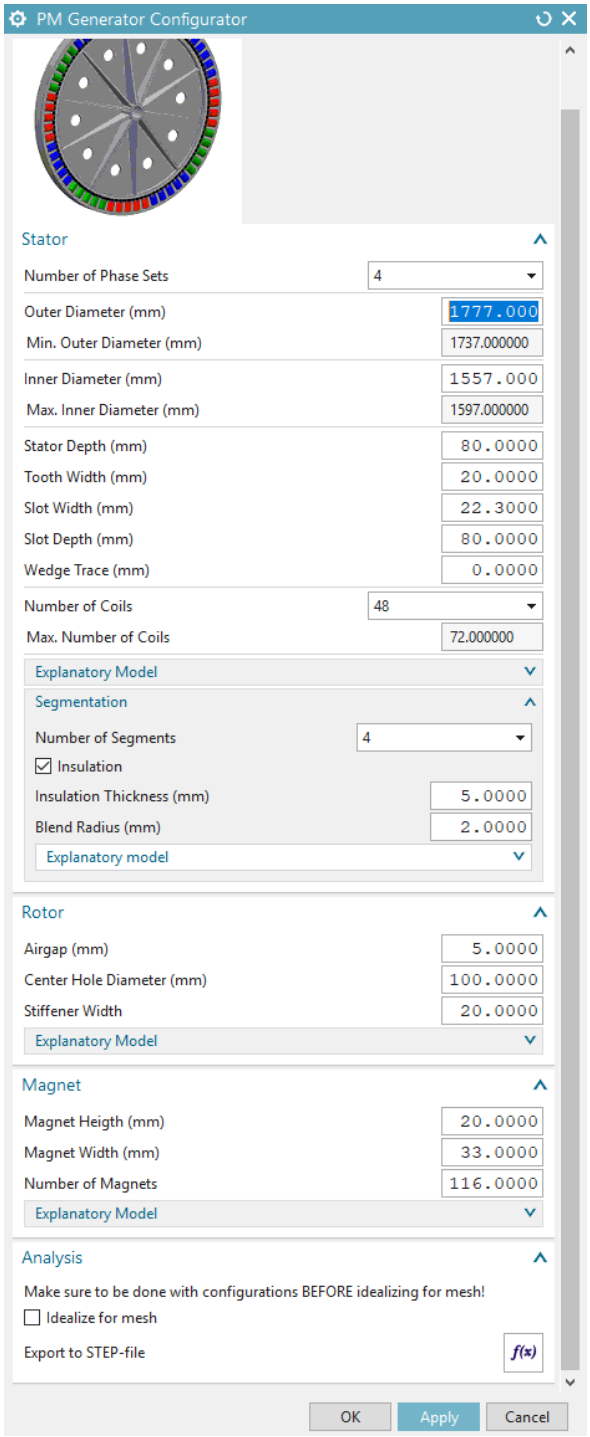


Figure A.19: Configurator interface

All the inputs in both the rotor and magnet menus have implemented rules or max/min values so that wrong values cannot be chosen. Explanatory models have also been added.

In the analysis menu, two buttons are added. The ”Idealize for mesh” button prepares the model for analysis by removing the edge blends and the false body from the coils. Although the false body is not visible within NX, it will be visible when the model is opened in other programs, that does not support reference sets. An export to STEP-file button has also been added. The button is running a Journal script that exports all solids to a .stp file and saves it in the same folder as the generator.prt file. The Journal script is taken from nxjournaling.com [27].

A.2 Coil Assembly in Siemens NX

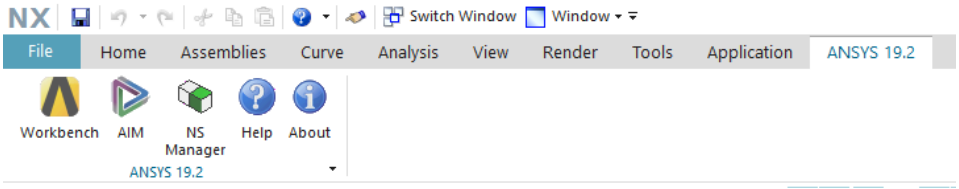
As presented in Chapter A.1.3, all the coils are added in three part-files divided into three different phases. It means that all the red coils are placed into one part-file, all the blue coils in another part-file and all the green coils in a third file.

The coils were originally added as one part-file per coil, by using pattern component. The process was successful with 4 phase-sets, but did not work when the possibility to choose between 4 and 8 phase sets was added. The main source of the problem was that the pattern component tool could neither be selected as feature for another pattern command or be suppressed. In addition, the assembly ended up with a total number of 106 constraints and 366 expressions when 96 coils were added, compared to the chosen solution that has a total number of 19 constraints and 76 expression regardless if the number of coils are 12 or 96. The increase in complexity that appears when adding one part-file per coil does also increase the risk of model failure, both inside NX and when exported to other formats.

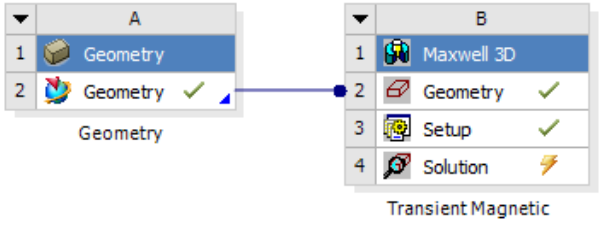
B

Integrating NX with ANSYS Workbench

The implementation between ANSYS CAD Configuration manager and NX is shown in Figure B.1. By simply clicking the ”Workbench” logo in NX (B.1a), a new Workbench session will be opened automatically, containing the 3D geometry. Whenever a change has been done to the model in NX, the geometry will update by clicking the ”Geometry” tab (B.1b) in Workbench.



(a) The ANSYS Workbench Geometry Interface for NX



(b) Importing the NX geometry in ANSYS Workbench, and using it in a transient magnetic analysis

Figure B.1: Integration of the parametric NX model into ANSYS Maxwell.

C

Model Expressions

C.1 Generator Assembly

```
The following expressions are controlled from user inputs
 ↪ in PTS
Expressions starting with DS can be accessed and updated
 ↪ within ANSYS Workbench

[mm]DS_Airgap_Length = 5.000000
[mm]DS_Fastener_Airgap = 50
[mm]DS_Fastener_Width = 40
[mm]DS_Fastening_Concept1_Suppression_State = 1
[mm]DS_Fastening_Concept2_Suppression_State = 0
[mm]DS_Insulation_Thickness = 5
[mm]DS_Magnet_Configuration = 4
[mm]DS_Magnet_Height = 20.000000
[mm]DS_Magnet_Width = 33.000000
DS_Number_of_Fasteners = 12
DS_Number_of_Phase_Sets = 4
[mm]DS_Number_of_Segments = 4
[mm]DS_Profile_Height = 170.000000
[mm]DS_Profile_Height_Concept2 = 110.000000
[mm]DS_Profile_Radius = 5.000000
[mm]DS_Profile_Thickness = 30
[mm]DS_Profile_Thickness_Center = 30
[mm]DS_Profile_Width = 120
[mm]DS_Rotor_Center_Diameter = 100.000000
[mm]DS_Rotor_Stiffener_Width = 20.000000
```

```
[mm]DS_Segment_Airgap = 3
[mm]DS_Stator_Edge_Blend_Radius = 0.000000
[mm]DS_Stator_Inner_Diameter = 1500.000000
[mm]DS_Stator_Outer_Diameter = 1800.000000
[mm]DS_Stator_Slot_Depth = 80.000000
[mm]DS_Stator_Slot_Width = 20.000000
[mm]DS_Stator_Stack_Length = 100.000000
[mm]DS_Stator_Tooth_Width = 18.000000
DS_Total_Number_of_Coils = 60
DS_Total_Number_of_Magnets = 2*DS_Total_Number_of_Coils -
 ↪  DS_Magnet_Configuration
[mm]DS_Wedge_Trace = 5.000000
[mm]Profile_Radius_Suppression = if(DS_Profile_Radius != 0)
 ↪  1 else 0
Idealize_For_Mesh = 0

The following expressions are rules, controlling the check
 ↪  functions and read only user feedback in PTS

[mm]Max_Inner_Diameter_Limit =
 ↪  if(DS_Fastening_Concept1_Suppression_State = 1 |
 ↪  DS_Fastening_Concept2_Suppression_State = 1)
 ↪  DS_Stator_Outer_Diameter-2*DS_Stator_Slot_Depth
 ↪  -110-DS_Profile_Thickness else
 ↪  DS_Stator_Outer_Diameter-2*DS_Stator_Slot_Depth-20
[mm]Max_Magnet_Width = (Pi*(DS_Stator_Inner_Diameter-2*
 ↪  DS_Airgap_Length-2*DS_Magnet_Height))/
 ↪  DS_Total_Number_of_Magnets
Max_Number_of_Coils_Calc =
 ↪  (Pi*DS_Stator_Inner_Diameter)/(2*DS_Stator_Slot_Width+
 ↪  DS_Stator_Tooth_Width)
[mm]Max_Number_of_Coils_Limit = if(Max_Number_of_Coils_Calc
 ↪  < 24) 12 else if(Max_Number_of_Coils_Calc < 36) 24 else
 ↪  if(Max_Number_of_Coils_Calc < 48) 36 else if
 ↪  (Max_Number_of_Coils_Calc < 60) 48 else
 ↪  if(Max_Number_of_Coils_Calc < 72) 60 else
 ↪  if(Max_Number_of_Coils_Calc < 84) 72 else
 ↪  if(Max_Number_of_Coils_Calc < 96) 84 else 96
Max_Number_of_Magnets_Limit =
 ↪  (DS_Stator_Inner_Diameter-DS_Airgap_Length-
 ↪  DS_Magnet_Height)*Pi/(DS_Magnet_Width)
[mm]Max_Profile_Height_Concept_1 =
 ↪  ((DS_Stator_Outer_Diameter+DS_Fastener_Airgap)/
 ↪  2-(DS_Stator_Inner_Diameter/2+DS_Stator_Slot_Depth+10))*2
```

```
[mm]Max_Profile_Height_Concept_2 =
 ↪  ((DS_Stator_Outer_Diameter+DS_Insulation_Thickness)/
 ↪  2-(DS_Stator_Inner_Diameter/2+DS_Stator_Slot_Depth+10))*2
[mm]Max_Profile_Width = round(DS_Stator_Outer_Diameter/14)
[mm]Max_Slot_Width_Limit =
 ↪  0.5*((DS_Stator_Inner_Diameter*Pi/
 ↪  DS_Total_Number_of_Coils)-DS_Stator_Tooth_Width)
[mm]Max_Tooth_Width_Limit = (DS_Stator_Inner_Diameter*Pi/
 ↪  DS_Total_Number_of_Coils)-2*DS_Stator_Slot_Width
[mm]Min_Outer_Diamater_Limit =
 ↪  if(DS_Fastening_Concept1_Suppression_State = 1 |
 ↪  DS_Fastening_Concept2_Suppression_State = 1)
 ↪  (DS_Stator_Inner_Diameter+2*DS_Stator_Slot_Depth+
 ↪  40+DS_Profile_Height_Concept2/2)+50+DS_Profile_Thickness
 ↪  else DS_Stator_Inner_Diameter+2*
 ↪  DS_Stator_Slot_Depth+40+DS_Profile_Height_Concept2/2
[mm]Min_Profile_Height_Concept_1 =
 ↪  DS_Fastener_Airgap+2*DS_Profile_Thickness+60
[mm]Min_Profile_Height_Concept_2 =
 ↪  DS_Insulation_Thickness+2*DS_Profile_Thickness+60
[mm]Slot_Depth_Limit =
 ↪  (DS_Stator_Outer_Diameter-DS_Stator_Inner_Diameter)/2 -
 ↪  20
Pi = 3.14159265358979323846264338327950
 ↪  28841971693993751058209749445923078164062 86
```

The following expressions are input lists for PTS

```
(List) Segment_Number_List = if(DS_Number_of_Phase_Sets ==
 ↪  8) 1,2,4,8 else 1,2,4
(List) Phase_Set_Number_List = 4,8
(List) Dovetails_Number_List = if(DS_Number_of_Segments =
 ↪  1) 1, 2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15
 ↪  ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,23 , 24 else
 ↪  if(DS_Number_of_Segments = 2)
 ↪  4,6,8,10,12,14,16,18,20,22,24 else
 ↪  if(DS_Number_of_Segments = 4) 8,12,16,20,24 else 16,24
(List) Magnet_Configuration_List = 2,4,6
(List) Coli_Number_List = if (DS_Number_of_Phase_Sets== 8)
 ↪  24,48,72,96 else 12,24,36,48,60,72,84,96
```

The following expressions are controlling the assembly constraints

```
Phase_Set_Angle_Correction=if(DS_Number_of_Segments == 8)
 ↪  1.5 else if(DS_Number_of_Segments==2) 2 else 1
```

```
[degrees]Fastener_Split_Plane_Angle = 45 /
 ↪  (DS_Number_of_Fasteners /
 ↪  ((DS_Number_of_Segments/mmFix) *
 ↪  Phase_Set_Angle_Correction))
[mm]Magnet_Center_Constraint = DS_Stator_Inner_Diameter /
 ↪  2-DS_Airgap_Length-DS_Magnet_Height
[mm]Magnet_to_Magnet_Constraint =
 ↪  (Pi*(DS_Stator_Inner_Diameter-2 *
 ↪  DS_Airgap_Length-2*DS_Magnet_Height)) /
 ↪  DS_Total_Number_of_Magnets
[mm]Inner_Radius = DS_Stator_Inner_Diameter/2
[degrees]CoilAngle_phase = 30
[degrees]Coil_to_Coil_Angle =
 ↪  CoilAngle_phase/Number_of_Coils_pr_Phase
[mm]Coil_to_Coil_Distance_Constraint =
 ↪  sin(15)*(Inner_Radius+DS_Stator_Slot_Depth)*2
[mm]Center_Coil_Distance_Constraint =
 ↪  DS_Stator_Inner_Diameter/2+DS_Stator_Slot_Depth
[degrees]CoilAngle_phase = 30

The following expressions are controlling the phase set
 ↪  pattern for the coils

Number_of_Coils_pr_Phase =
 ↪  DS_Total_Number_of_Coils/(DS_Number_of_Phase_Sets*3)
Phase_Angle =
 ↪  ((CircPat1_CircularPattern_pattern_Circular_Dir_count-1)/
 ↪  DS_Total_Number_of_Coils)*360
CircPat1_CircularPattern_pattern_Circular_Dir_count =
 ↪  Number_of_Coils_pr_Phase

The following expressions are controlling the manufacturing
 ↪  drawings

(String) Constructor = "M.B"
(String) Date = Day + " " + Month + " " + Year
Profile_Scale = if(DS_Stator_Stack_Length > 110 ||
 ↪  DS_Magnet_Height > 180 ) 0.5 else 1
(String) Profile_Scale_String =
 ↪  "1:"+stringValue(1/Profile_Scale)
Magnet_Scale = if(DS_Stator_Stack_Length < 261) 1 else
 ↪  if(DS_Stator_Stack_Length > 260 &
 ↪  DS_Stator_Stack_Length < 500) 2 else 5
(String) Magnet_Scale_String =
 ↪  stringValue(round(Magnet_Scale))+":1"
```

```
Generator_Scale = if(DS_Stator_Outer_Diameter < 2000) 0.1
 ↪ else if(DS_Stator_Outer_Diameter < 4000 &
 ↪ DS_Stator_Outer_Diameter > 2001) 0.05 else (1/50)
(String) Generator_Scale_String =
 ↪ "1:"+stringValue(round(1/Generator_Scale))
Number_of_Slots = 2*DS_Total_Number_of_Coils

(String) Year = subString(Date_Full,21,25)
(String) Month = subString(Date_Full,5,7)
(String) Day = subString(Date_Full, 9, 10)
(String) Date_Full =
 ↪ StringUpper(dateTimeString("localTime?", True))
```

C.2 Stator

```
The following expressions are interpart expressions, linked
 ↪ from Generator.prt

[degrees]Coil_Angle = "Generator"::Coil_to_Coil_Angle
[mm]DS_Fastening_Concept2_Suppression_State =
 ↪ "Generator"::DS_Fastening_Concept2_Suppression_State
DS_Number_of_Fasteners =
 ↪ "Generator"::DS_Number_of_Fasteners
DS_Number_of_Phase_Sets =
 ↪ "Generator"::DS_Number_of_Phase_Sets
[mm]DS_Number_of_Segments =
 ↪ "Generator"::DS_Number_of_Segments
[mm]DS_Segment_Airgap = "Generator"::DS_Segment_Airgap
[mm]DS_Stator_Edge_Blend_Radius =
 ↪ "Generator"::DS_Stator_Edge_Blend_Radius
[mm]DS_Stator_Inner_Diameter =
 ↪ "Generator"::DS_Stator_Inner_Diameter
[mm]DS_Stator_Outer_Diameter =
 ↪ "Generator"::DS_Stator_Outer_Diameter
[mm]DS_Stator_Slot_Depth =
 ↪ "Generator"::DS_Stator_Slot_Depth
[mm]DS_Stator_Slot_Width =
 ↪ "Generator"::DS_Stator_Slot_Width
[mm]DS_Stator_Stack_Length =
 ↪ "Generator"::DS_Stator_Stack_Length
```

```
[mm]DS_Stator_Tooth_Width =
 ↪  "Generator"::DS_Stator_Tooth_Width
DS_Total_Number_of_Coils =
 ↪  "Generator"::DS_Total_Number_of_Coils
[mm]Fastening_Concept1_Suppression_State =
 ↪  "Generator"::DS_Fastening_Concept1_Suppression_State

The following expressions are controlling the different
 ↪  modeling features within stator.prt

[mm]EXTRUDE_Stack_Length = DS_Stator_Stack_Length
[mm]OFFSET_Segment_Airgap_4Seg1 = DS_Segment_Airgap
[mm]OFFSET_Segment_Airgap_4Seg2 = DS_Segment_Airgap
[mm]OFFSET_Segment_Airgap_8Seg1 = DS_Segment_Airgap
[mm]OFFSET_Segment_Airgap_8Seg2 = DS_Segment_Airgap
PATTERN_Coil_Tracks = DS_Total_Number_of_Coils
[degrees]PATTERN_False_Body = 360
PATTERN_Fasteners = DS_Number_of_Fasteners
[degrees]PATTERN_Fasteners_Angle = 360
PATTERN_Number_of_Fasteners = DS_Number_of_Fasteners
[degrees]PLANE_1_Angle = if(DS_Number_of_Phase_Sets == 8)
 ↪  Coil_Angle/4 else Coil_Angle/2
[degrees]PLANE_2_Angle = 90
[degrees]PLANE_3_Angle = 90
[degrees]PLANE_4_Angle = 45
[mm]SKETCH_Stator_Inner_Diameter = DS_Stator_Inner_Diameter
[mm]SKETCH_Stator_Outer_Diameter = DS_Stator_Outer_Diameter
SUPPRESS_8_Segments = if(DS_Number_of_Segments = 8) 1 else
 ↪  0
SUPPRESS_Dovetail_Tracks_Concept1 =
 ↪  Fastening_Concept1_Suppression_State
SUPPRESS_Dovetail_Tracks_Concept2 =
 ↪  DS_Fastening_Concept2_Suppression_State
SUPPRESS_Fastener_Track_Geater_Then_1Seg =
 ↪  if(DS_Number_of_Segments > 1) 1 else 0
SUPPRESS_Fastener_Track_Less_Then_4Seg =
 ↪  if(DS_Number_of_Segments < 4) 1 else 0
SUPPRESS_Fastener_Track_Subtract_1Seg =
 ↪  if(DS_Number_of_Segments = 1) 1 else 0
SUPPRESS_Fastener_Track_Subtract_2Seg =
 ↪  if(DS_Number_of_Segments = 2) 1 else 0
SUPPRESS_Fastener_Track_Subtract_4Seg =
 ↪  if(DS_Number_of_Segments = 4)1 else 0
SUPPRESS_Segment_Offset = if(DS_Number_of_Segments = 1) 0
 ↪  else 1
```

```
SUPPRESS_Split1 = if(DS_Number_of_Segments == 1) 0 else 1
SUPPRESS_Split2 = if(DS_Number_of_Segments == 1) 0 else
 ↪  if(DS_Number_of_Segments == 2) 0 else 1
```

C.3 Rotor

```
The following expressions are interpart expressions, linked
 ↪  from Generator.prt

[mm]DS_Airgap_Length = "Generator"::DS_Airgap_Length
[mm]DS_Magnet_Height = "Generator"::DS_Magnet_Height
[mm]DS_Magnet_Width = "Generator"::DS_Magnet_Width
[mm]DS_Rotor_Center_Diameter =
 ↪  "Generator"::DS_Rotor_Center_Diameter
[mm]DS_Rotor_Stiffener_Width =
 ↪  "Generator"::DS_Rotor_Stiffener_Width
[mm]DS_Stator_Inner_Diameter =
 ↪  "Generator"::DS_Stator_Inner_Diameter
[mm]DS_Stator_Stack_Length =
 ↪  "Generator"::DS_Stator_Stack_Length
DS_Total_Number_of_Magnets =
 ↪  "Generator"::DS_Total_Number_of_Magnets

The following expressions are controlling the different
 ↪  modeling features within Rotor.prt

[mm]EXTRUDE_Cut_Out_Negative_Direction = -10
[mm]EXTRUDE_Cut_Out_Positive_Direction = 10
[mm]EXTRUDE_Rotor_CenteR_Hole_Negative_Direction =
 ↪  -DS_Stator_Stack_Length/2
[mm]EXTRUDE_Rotor_Center_Hole_Positive_Direction =
 ↪  DS_Stator_Stack_Length/2+92
[mm]EXTRUDE_Rotor_Depth_Negative_Direction =
 ↪  -DS_Stator_Stack_Length/2
[mm]EXTRUDE_Rotor_Depth_Positive_Direction =
 ↪  DS_Stator_Stack_Length/2
[mm]EXTRUDE_Stiffener_Width_Negative_Direction =
 ↪  -DS_Rotor_Stiffener_Width/2
[mm]EXTRUDE_Stiffener_Width_Positive_Direction =
 ↪  DS_Rotor_Stiffener_Width/2
[degrees]PATTERN_Cut_Out_Degrees = 360
```

```
PATTERN_Magnet_Tracks = DS_Total_Number_of_Magnets
[degrees]PATTERN_Magnet_Tracks_Degrees = 360
PATTERN_Number_Of_Cut_Outs = 10
PATTERN_Number_of_Stiffeners = 10
[degrees]PATTERN_Stiffener_Degrees = 360
[mm]SKETCH_Center_Inner_Diameter = DS_Rotor_Center_Diameter
[mm]SKETCH_Center_Thickness = DS_Rotor_Stiffener_Width
[mm]SKETCH_Cut_Out_Diamater = DS_Stator_Inner_Diameter/15
[mm]SKETCH_Magnet_Width = DS_Magnet_Width
[mm]SKETCH_Reference_Center_to_Cut_Out =
 ↪  DS_Stator_Inner_Diameter/3
[mm]SKETCH_Reference_Center_to_Magnet_Track =
 ↪  DS_Stator_Inner_Diameter/2-
 ↪  DS_Magnet_Height-DS_Airgap_Length
[mm]SKETCH_Rotor_Inner_Diameter =
 ↪  DS_Stator_Inner_Diameter-2*
 ↪  DS_Magnet_Height-2*DS_Airgap_Length-57
[mm]SKETCH_Rotor_Outer_Diameter =
 ↪  DS_Stator_Inner_Diameter-2*
 ↪  DS_Airgap_Length-2*DS_Magnet_Height+10
[mm]SKETCH_Siffener_Height_Inner =
 ↪  DS_Stator_Stack_Length/2+82
[mm]SKETCH_Stiffener_Diagonal =
 ↪  DS_Rotor_Center_Diameter/2+17
[mm]SKETCH_Stiffener_Height_Outer =
 ↪  DS_Stator_Stack_Length/2-10
[mm]SKETCH_Stiffener_Width =
 ↪  DS_Stator_Inner_Diameter/2-DS_Airgap_Length-
 ↪  DS_Magnet_Height-DS_Rotor_Center_Diameter/2-45
```

C.4 Coil_Reference

```
The following expressions are interpart expressions, linked
 ↪  from Generator.prt

[mm]Center_Coil_Distance_Constraint =
 ↪  "Generator"::Center_Coil_Distance_Constraint
[degrees]Coil_Angle = "Generator"::Coil_to_Coil_Angle
DS_Number_of_Phase_Sets =
 ↪  "Generator"::DS_Number_of_Phase_Sets
[mm]DS_Stator_Slot_Depth =
 ↪  "Generator"::DS_Stator_Slot_Depth
```

```
[mm]DS_Stator_Slot_Width =
 ↪  "Generator"::DS_Stator_Slot_Width
[mm]DS_Stator_Stack_Length =
 ↪  "Generator"::DS_Stator_Stack_Length
[mm]DS_Stator_Tooth_Width =
 ↪  "Generator"::DS_Stator_Tooth_Width
DS_Total_Number_of_Coils =
 ↪  "Generator"::DS_Total_Number_of_Coils
[mm]DS_Wedge_Trace = "Generator"::DS_Wedge_Trace
Idealize_for_Mesh = "Generator"::Idealize_For_Mesh
Number_of_Coils_pr_Phase =
 ↪  "Generator"::Number_of_Coils_pr_Phase

The following expressions are controlling the different
 ↪  modeling features within Coil_Reference.prt

[mm]EDGE_BLEND_Ends =
 ↪  (DS_Stator_Tooth_Width+2*DS_Stator_Slot_Width)/2.1
[mm]EDGE_BLEND_Inner = DS_Stator_Tooth_Width/4
[mm]EDGE_BLEND_Outer = DS_Stator_Tooth_Width/4
[mm]EXTRUDE_Coil_Depth =
 ↪  DS_Stator_Stack_Length+2*DS_Stator_Slot_Width+8
[mm]EXTRUDE_Falsebody_Depth =
 ↪  DS_Stator_Stack_Length+2*DS_Stator_Slot_Width+8
[mm]SKETCH_Coil_Center_Length = DS_Stator_Stack_Length+8
[mm]SKETCH_Coil_Center_Width = DS_Stator_Tooth_Width
[mm]SKETCH_Coil_Height =
 ↪  DS_Stator_Slot_Depth-DS_Wedge_Trace
[mm]SKETCH_Coil_Width =
 ↪  2*DS_Stator_Slot_Width+DS_Stator_Tooth_Width
[mm]SKETCH_Falsebody_Height = DS_Stator_Slot_Depth+100
[mm]SKETCH_Falsebody_Width =
 ↪  2*DS_Stator_Slot_Width+DS_Stator_Tooth_Width
[mm]SKETCH_Reference_Center_of_Coil =
 ↪  DS_Stator_Tooth_Width/2+DS_Stator_Slot_Width
[mm]SKETCH_Reference_Point_Center_of_Coil =
 ↪  DS_Stator_Slot_Width+DS_Stator_Tooth_Width/2
[mm]SKETCH_Reference_Point_Coil_to_Global_Center =
 ↪  Center_Coil_Distance_Constraint
SUPPRESS_Blends = sqrt((Idealize_for_Mesh-1)^2)
SUPPRESS_FalseBody = sqrt((Idealize_for_Mesh-1)^2)
```

C.5 Coils (Red, Green, Blue)

```
The following expressions are interpart expressions, linked
 ↪ from Generator.prt

[mm]Center_Coil_Distance_Constraint =
 ↪ "Generator"::Center_Coil_Distance_Constraint
DS_Number_of_Phase_Sets =
 ↪ "Generator"::DS_Number_of_Phase_Sets
[mm]DS_Stator_Slot_Depth =
 ↪ "Generator"::DS_Stator_Slot_Depth
[mm]DS_Stator_Slot_Width =
 ↪ "Generator"::DS_Stator_Slot_Width
[mm]DS_Stator_Tooth_Width =
 ↪ "Generator"::DS_Stator_Tooth_Width
DS_Total_Number_of_Coils =
 ↪ "Generator"::DS_Total_Number_of_Coils
Number_of_Coils_pr_Phase =
 ↪ "Generator"::Number_of_Coils_pr_Phase

The following expressions are controlling the different
 ↪ modeling features within Coil_<color>.prt

[degrees]PATTERN_Angle_Between_Coils =
 ↪ 360/DS_Total_Number_of_Coils
PATTERN_Number_of_Coils_pr_Phase = Number_of_Coils_pr_Phase
PATTERN_Phase_Sets = DS_Number_of_Phase_Sets
[degrees]PATTERN_Phase_Sets_Degrees = 360
[mm]SKETCH_Reference_Center_of_Coil =
 ↪ DS_Stator_Slot_Width+DS_Stator_Tooth_Width/2
[mm]SKETCH_Reference_Point_Center_of_Coil =
 ↪ DS_Stator_Slot_Width+DS_Stator_Tooth_Width/2
[mm]SKETCH_Reference_Point_Coil_to_Global_Center =
 ↪ Center_Coil_Distance_Constraint
```

C.6 Magnet_Reference

```
The following expressions are interpart expressions, linked
 ↪ from Generator.prt
```

```
[mm]DS_Magnet_Height = "Generator"::DS_Magnet_Height
[mm]DS_Magnet_Width = "Generator"::DS_Magnet_Width
[mm]DS_Stator_Stack_Length =
 ↪  "Generator"::DS_Stator_Stack_Length
DS_Total_Number_of_Magnets =
 ↪  "Generator"::DS_Total_Number_of_Magnets
[mm]EXTRUDE_Magnet_Depth = DS_Stator_Stack_Length
[mm]Magnet_Center_Constraint =
 ↪  "Generator"::Magnet_Center_Constraint

The following expressions are controlling the different
 ↪  modeling features within magnet_reference.prt

[mm]SKETCH_Magnet_Height = DS_Magnet_Height
[mm]SKETCH_Magnet_Width = DS_Magnet_Width
[mm]SKETCH_Reference_Point_Magnet_to_Global_Center =
 ↪  Magnet_Center_Constraint
```

C.7 Magnets (N, S)

```
The following expressions are interpart expressions, linked
 ↪  from Generator.prt

[mm]DS_Magnet_Height = "Generator"::DS_Magnet_Height
[mm]DS_Magnet_Width = "Generator"::DS_Magnet_Width
[mm]DS_Stator_Stack_Length =
 ↪  "Generator"::DS_Stator_Stack_Length
DS_Total_Number_of_Magnets =
 ↪  "Generator"::DS_Total_Number_of_Magnets
[mm]Magnet_Center_Constraint =
 ↪  "Generator"::Magnet_Center_Constraint

The following expressions are controlling the different
 ↪  modeling features within Magnet_<N,S>.prt

[mm]EXTRUDE_Magnet_Depth = DS_Stator_Stack_Length
[degrees]PATTERN_N_Magnets_Degrees = 360
PATTERN_Number_of_N_Magnets = DS_Total_Number_of_Magnets/2
[mm]SKETCH_Magnet_Height = DS_Magnet_Height
[mm]SKETCH_Magnet_Width = DS_Magnet_Width
```

```
[mm]SKETCH_Reference_Point_Magnet_to_Global_Center =
 ↪  Magnet_Center_Constraint
```

C.8 I-Profile_Reference

```
The following expressions are interpart expressions, linked
 ↪  from Generator.prt

[mm]DS_Fastener_Airgap = "Generator"::DS_Fastener_Airgap
[mm]DS_Profile_Height = "Generator"::DS_Profile_Height
[mm]DS_Profile_Radius = "Generator"::DS_Profile_Radius
[mm]DS_Profile_Thickness =
 ↪  "Generator"::DS_Profile_Thickness
[mm]DS_Profile_Thickness_Center =
 ↪  "Generator"::DS_Profile_Thickness_Center
[mm]DS_Profile_Width = "Generator"::DS_Profile_Width
[mm]DS_Stator_Inner_Diameter =
 ↪  "Generator"::DS_Stator_Inner_Diameter
[mm]DS_Stator_Outer_Diameter =
 ↪  "Generator"::DS_Stator_Outer_Diameter
[mm]DS_Stator_Slot_Depth =
 ↪  "Generator"::DS_Stator_Slot_Depth
[mm]DS_Stator_Stack_Length =
 ↪  "Generator"::DS_Stator_Stack_Length
[mm]Profile_Radius_Suppression =
 ↪  "Generator"::Profile_Radius_Suppression

The following expressions are controlling the different
 ↪  modeling features within I-Profile_Reference.prt

[mm]EDGE_BLEND_Radius = DS_Profile_Radius
[mm]EXTRUDE_Profile_Depth = DS_Stator_Stack_Length
[mm]SKETCH_Flange1 = DS_Profile_Thickness
[mm]SKETCH_Flange2 = DS_Profile_Thickness
[mm]SKETCH_Profile_Bottom_to_Center_Reference =
 ↪  DS_Stator_Outer_Diameter/2-
 ↪  ((DS_Profile_Height-DS_Fastener_Airgap)/2)
[mm]SKETCH_Web1 = DS_Profile_Width/2
[mm]SKETCH_Web2 = DS_Profile_Width/2
[mm]SKETCH_Web_Height =
 ↪  DS_Profile_Height-2*DS_Profile_Thickness
```

```
[mm]SKETCH_Web_To_Flang_Dist =
 ↪ DS_Profile_Width/2-DS_Profile_Thickness_Center/2
SUPPRESS_Edge_Blends = Profile_Radius_Suppression
```

C.9 I-Profile_Outer_Ring

```
The following expressions are interpart expressions, linked
 ↪ from Generator.prt

[mm]DS_Fastener_Airgap = "Generator"::DS_Fastener_Airgap
DS_Number_of_Fasteners =
 ↪ "Generator"::DS_Number_of_Fasteners
[mm]DS_Profile_Height = "Generator"::DS_Profile_Height
[mm]DS_Profile_Thickness =
 ↪ "Generator"::DS_Profile_Thickness
[mm]DS_Stator_Outer_Diameter =
 ↪ "Generator"::DS_Stator_Outer_Diameter
[mm]DS_Stator_Stack_Length =
 ↪ "Generator"::DS_Stator_Stack_Length

The following expressions are controlling the different
 ↪ modeling features within
 ↪ I-Profile_Fixed_Ring(Concept_1).prt

[mm]EXTRUDE_Ring_Depth = DS_Stator_Stack_Length
[degrees]PATTERN_Fastener_Angle = 360
PATTERN_Fastener_Tracks = DS_Number_of_Fasteners
[mm]SKETCH_Inner_Diameter = DS_Stator_Outer_Diameter+
 ↪ 2*DS_Fastener_Airgap
[mm]SKETCH_Outer_Diameter = DS_Stator_Outer_Diameter+
 ↪ 2*DS_Fastener_Airgap+DS_Profile_Thickness+300
```

C.10 I-Profile_Fasteners

```
The following expressions are interpart expressions, linked
 ↪ from Generator.prt
```

```
DS_Number_of_Fasteners =
 ↪  "Generator"::DS_Number_of_Fasteners
[mm]DS_Stator_Inner_Diameter =
 ↪  "Generator"::DS_Stator_Inner_Diameter
[mm]DS_Stator_Slot_Depth =
 ↪  "Generator"::DS_Stator_Slot_Depth

The following expressions are controlling the different
 ↪  modeling features within
 ↪  I-profile_fastening(Concept_1).prt

PATTERN_Number_of_Profiles = DS_Number_of_Fasteners
[degrees]PATTERN_Profiles_Angle = 360
```

C.11 Ring_With_I-Profiles

```
The following expressions are interpart expressions, linked
 ↪  from Generator.prt

[mm]DS_Fastener_Width = "Generator"::DS_Fastener_Width
[mm]DS_Insulation_Thickness =
 ↪  "Generator"::DS_Insulation_Thickness
DS_Number_of_Fasteners =
 ↪  "Generator"::DS_Number_of_Fasteners
[mm]DS_Profile_Height_Concept2 =
 ↪  "Generator"::DS_Profile_Height_Concept2
[mm]DS_Profile_Radius = "Generator"::DS_Profile_Radius
[mm]DS_Profile_Thickness =
 ↪  "Generator"::DS_Profile_Thickness
[mm]DS_Profile_Thickness_Center =
 ↪  "Generator"::DS_Profile_Thickness_Center
[mm]DS_Profile_Width = "Generator"::DS_Profile_Width
[mm]DS_Stator_Edge_Blend_Radius =
 ↪  "Generator"::DS_Stator_Edge_Blend_Radius
[mm]DS_Stator_Outer_Diameter =
 ↪  "Generator"::DS_Stator_Outer_Diameter
[mm]DS_Stator_Stack_Length =
 ↪  "Generator"::DS_Stator_Stack_Length
[mm]Profile_Radius_Suppression =
 ↪  "Generator"::Profile_Radius_Suppression
```

```
The following expressions are controlling the different
 ↪ modeling features within
 ↪ Ring_with_I-Profiles(Concept_2).prt

[mm]EDGE_BLEND_Radius = DS_Profile_Radius
[mm]EXTRUDE_Profile_Depth = DS_Stator_Stack_Length
[mm]EXTRUDE_Ring_Depth = DS_Stator_Stack_Length
[degrees]PATTERN_Angle = 360
PATTERN_Number_of_Fasteners = DS_Number_of_Fasteners
PATTERN_Profiles = DS_Number_of_Fasteners
[mm]SKETCH_Flange_Thickness = DS_Profile_Thickness
[mm]SKETCH_Insulation_Thickness = DS_Insulation_Thickness/2
[mm]SKETCH_Ring_Inner_Diameter = DS_Stator_Outer_Diameter
[mm]SKETCH_Ring_Outer_Diameter =
 ↪ DS_Stator_Outer_Diameter+2*DS_Insulation_Thickness
[mm]SKETCH_Web_Height =
 ↪ DS_Profile_Height_Concept2/2-DS_Profile_Thickness
[mm]SKETCH_Web_Thickness = DS_Profile_Width/2
[mm]SKETCH_Width_Web_Thickenss_Dist =
 ↪ DS_Profile_Width/2-DS_Profile_Thickness_Center/2
SUPPRESS_Radius = Profile_Radius_Suppression
```

C.12 Ring_With_I-Profiles_Outer_Ring

```
The following expressions are interpart expressions, linked
 ↪ from Generator.prt

[mm]DS_Insulation_Thickness =
 ↪ "Generator"::DS_Insulation_Thickness
[mm]DS_Profile_Height_Concept2 =
 ↪ "Generator"::DS_Profile_Height_Concept2
[mm]DS_Stator_Outer_Diameter =
 ↪ "Generator"::DS_Stator_Outer_Diameter
[mm]DS_Stator_Stack_Length =
 ↪ "Generator"::DS_Stator_Stack_Length

The following expressions are controlling the different
 ↪ modeling features within
 ↪ Ring_with_I-Profiles_Outer_Ring(Concept_2).prt
```

```
[mm]SKETCH_Inner_Diameter =
 ↪  DS_Stator_Outer_Diameter+2*DS_Insulation_Thickness
[mm]SKETCH_Outer_Diameter =
 ↪  DS_Stator_Outer_Diameter+2*DS_Profile_Height_Concept2+100
```

D

Manufacturing Drawings

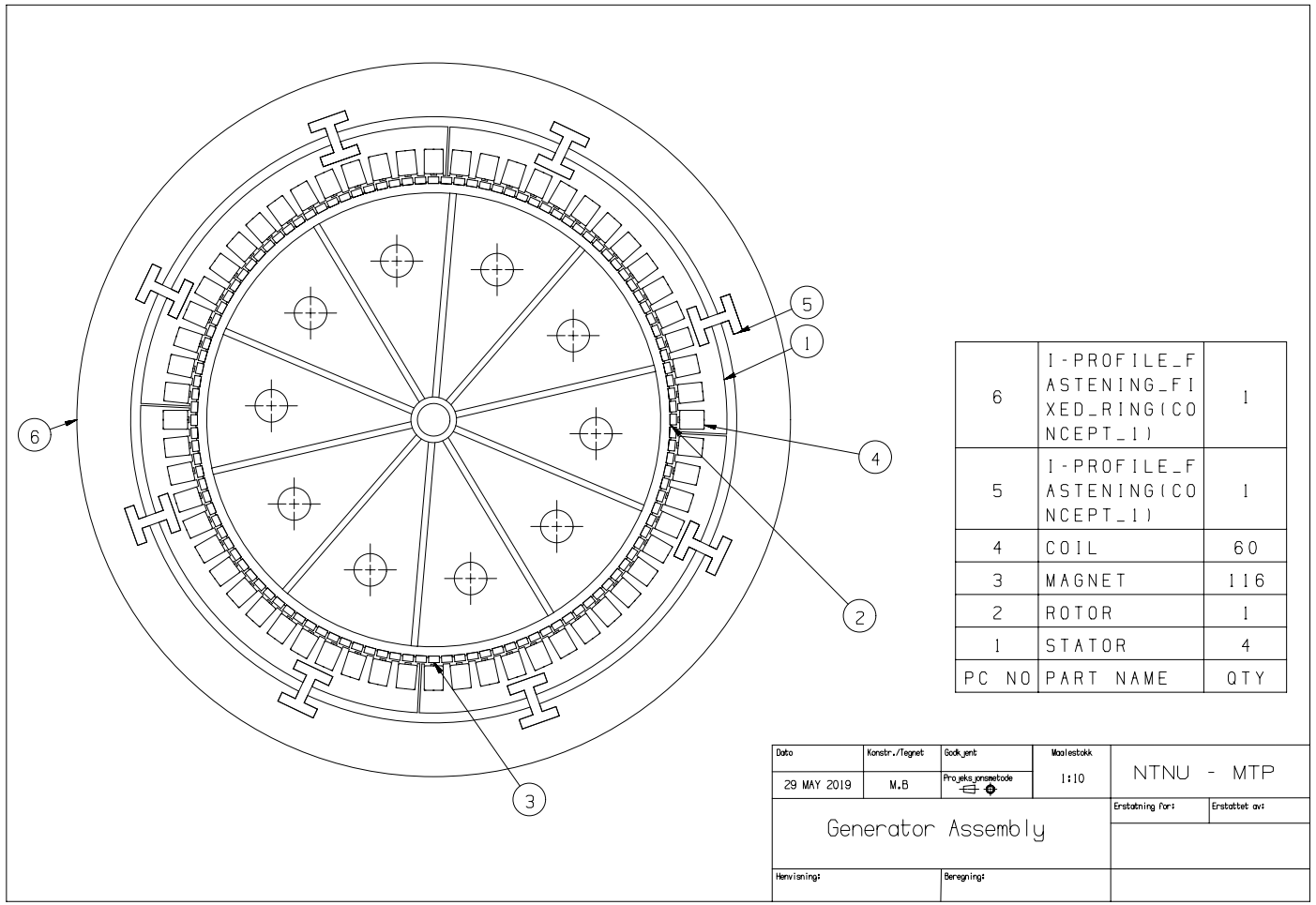


Figure D.1: Assembly

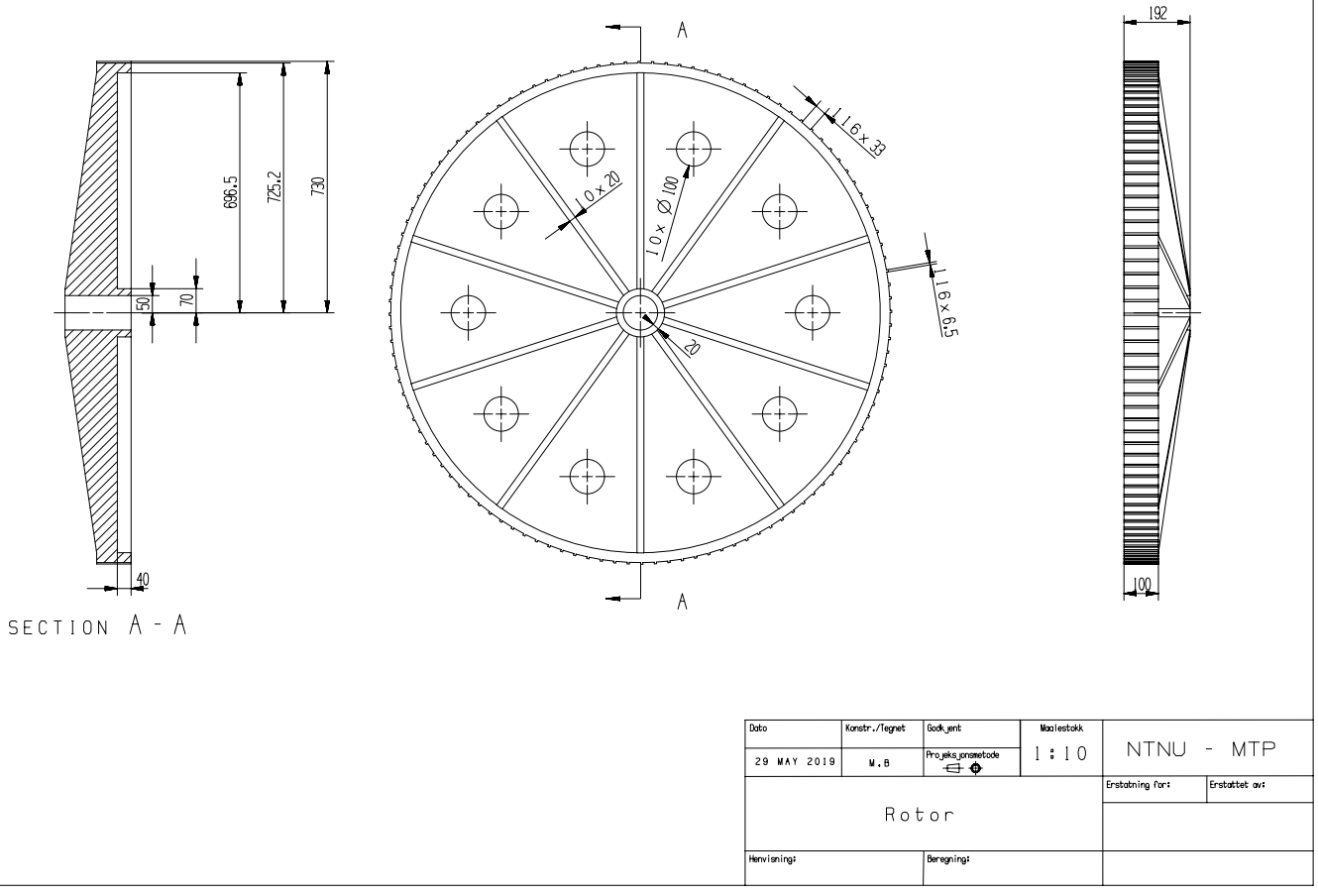


Figure D.2: Rotor

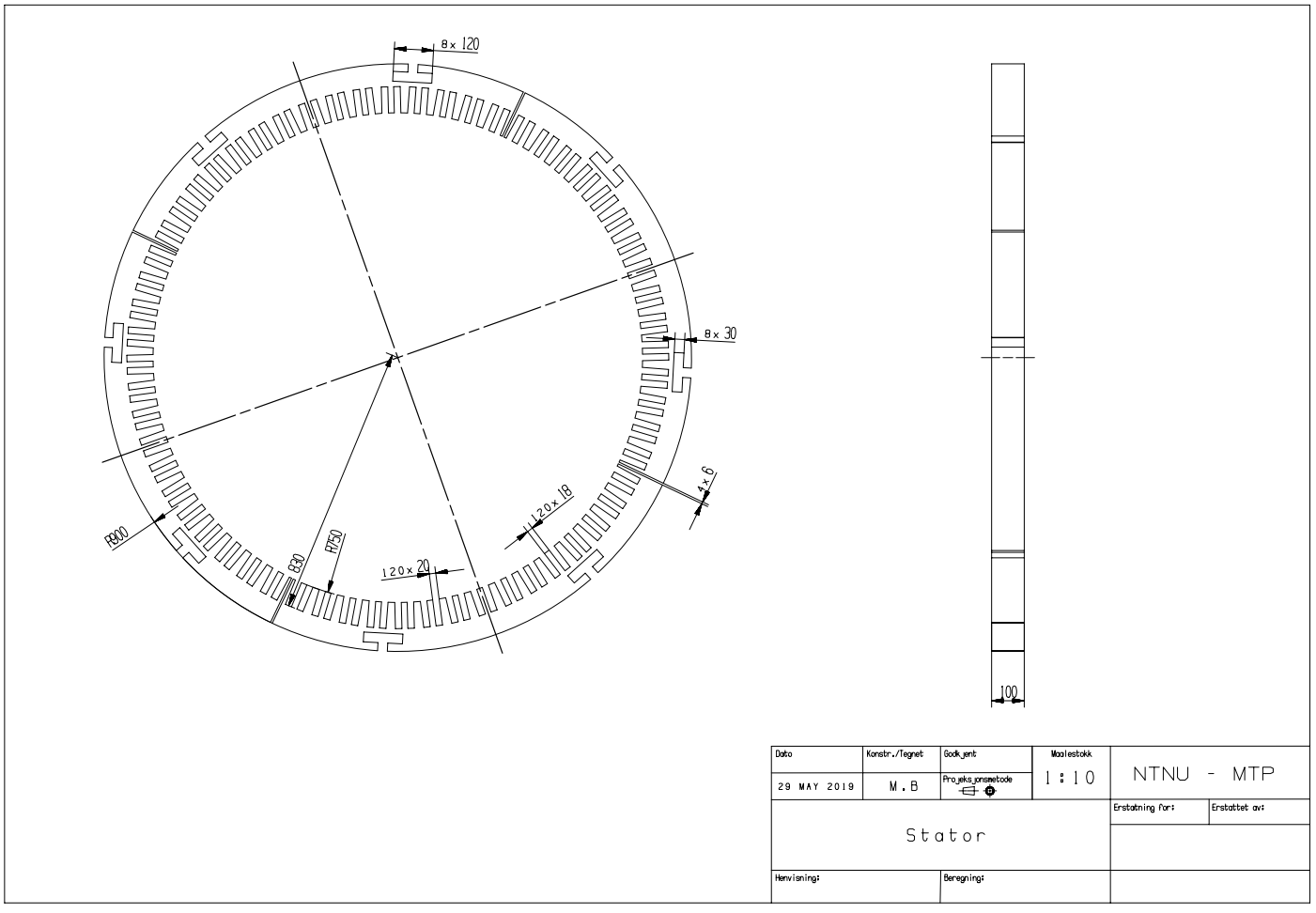


Figure D.3: Stator

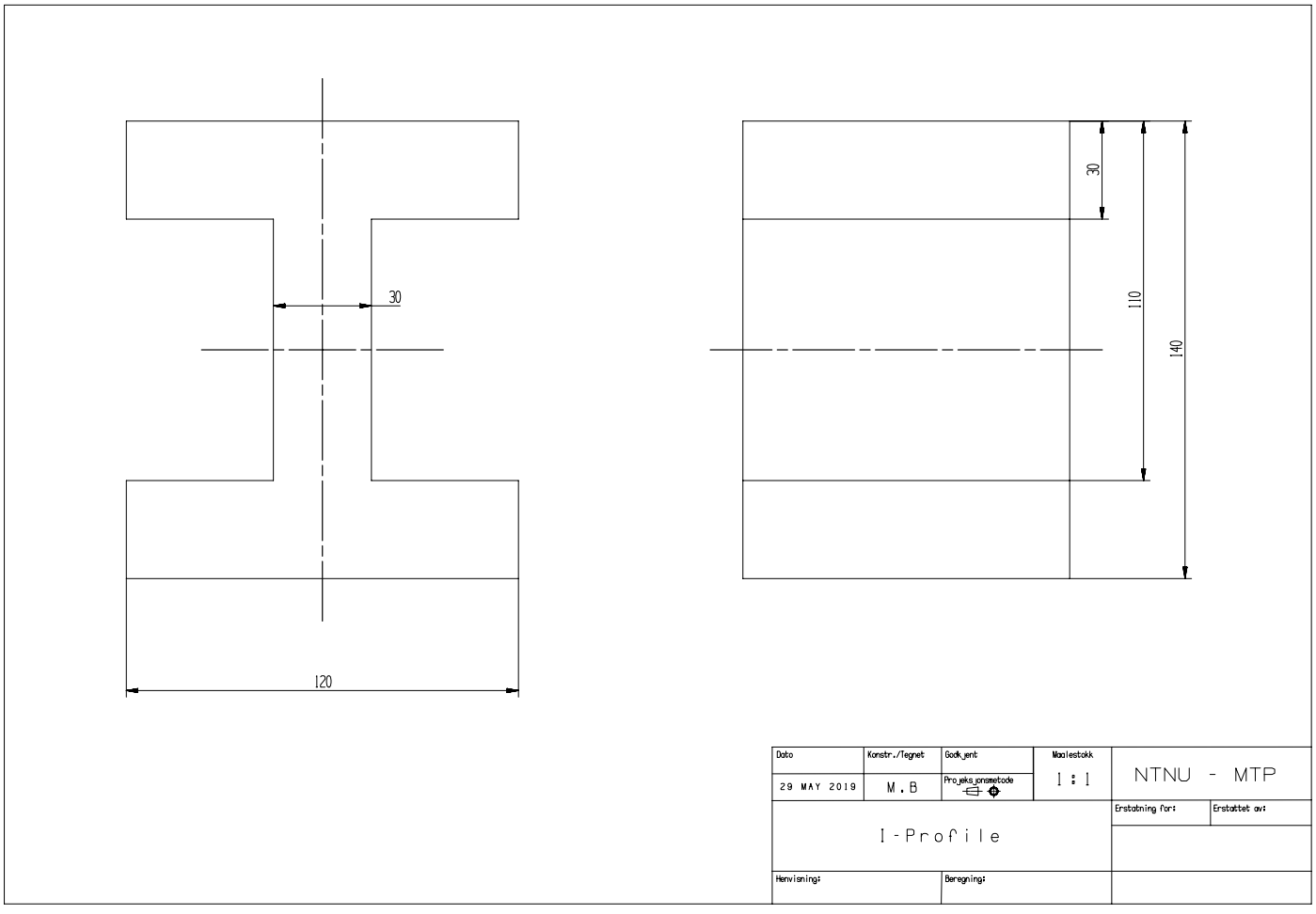


Figure D.4: I-profile

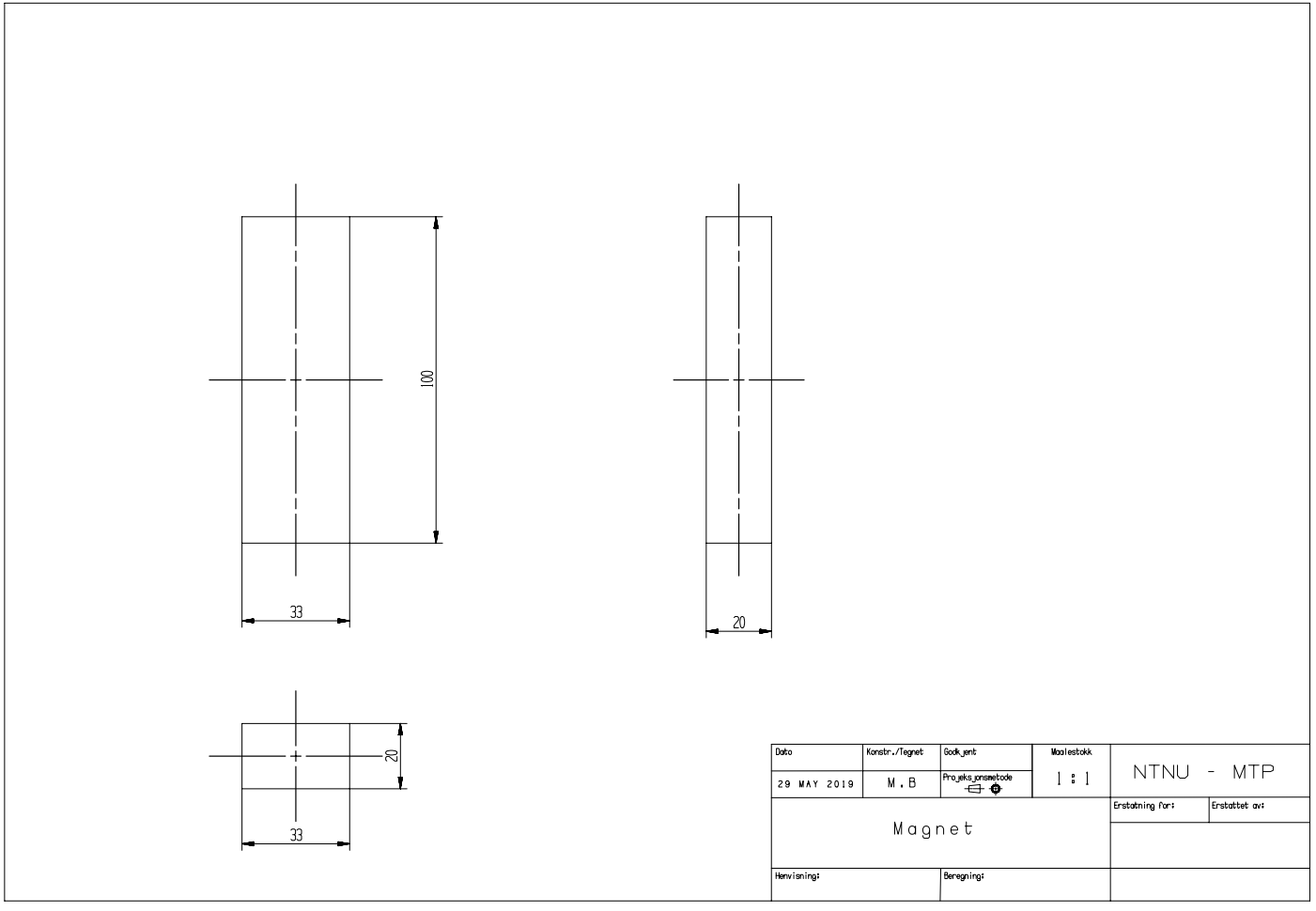


Figure D.5: Magnet

E

Explanatory Models

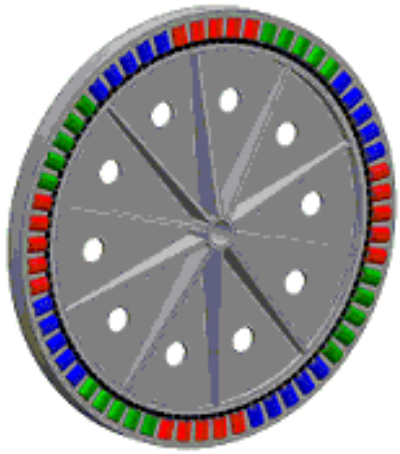


Figure E.1: Generator.

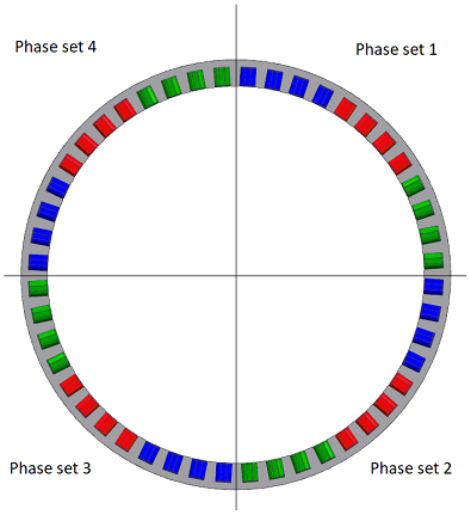


Figure E.2: Phase sets

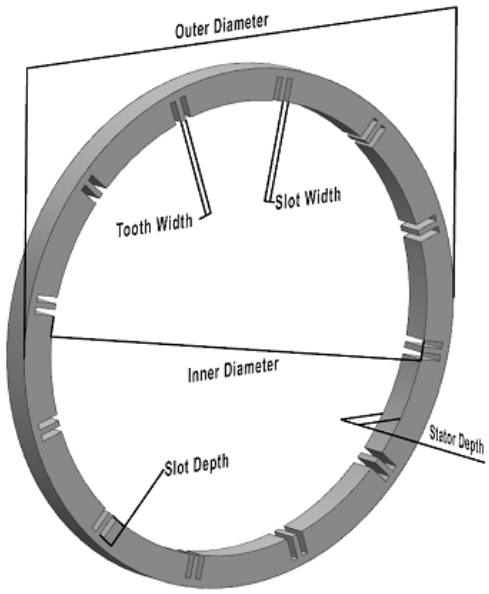


Figure E.3: Stator

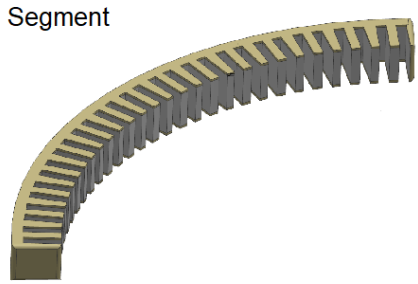


Figure E.4: Segment

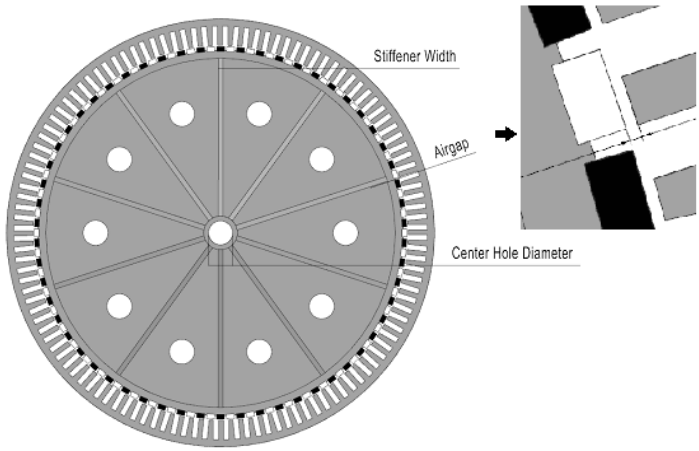


Figure E.5: Rotor

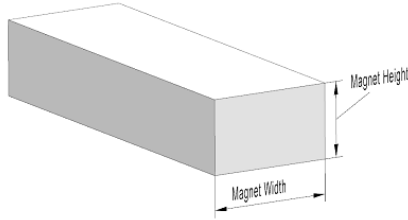


Figure E.6: Magnet

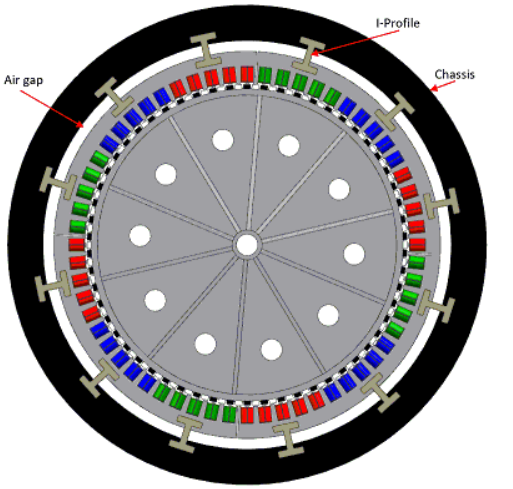


Figure E.7: Fastening concept 1

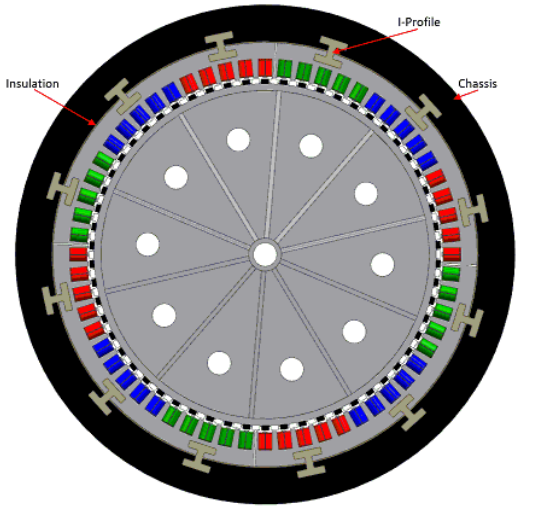


Figure E.8: Fastening concept 2

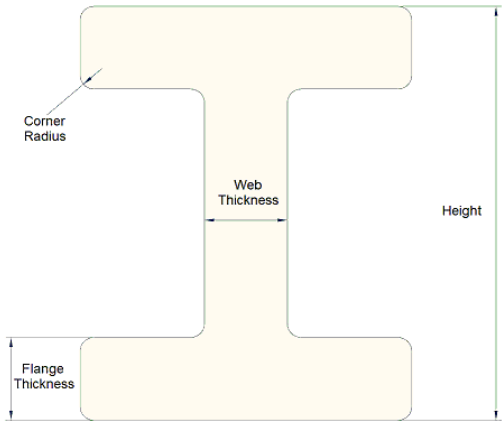


Figure E.9: I-profile

Figure E.10: Insulation ring

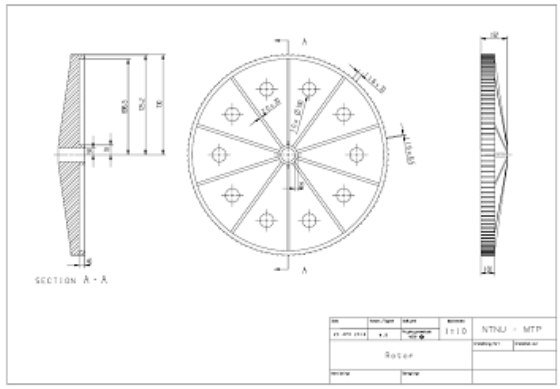


Figure E.11: Manufacturing drawing

F

Drawing Template Palette File

```
<?xml version="1.0" encoding="UTF-8"?>

<Palette
 ↪  xmlns="http://www.ugsolutions.com/Schemas/2001/UGPalettes"
 ↪  schemaVersion="1.0">

  <Presentation name="Drawing Templates (Inch)"
   ↪  bitmap="drawing_new.bmp" FileNewTab="NTNU Drawing
   ↪  Templates" application="All" UsesMasterModel="Yes"/>



   <PaletteEntry id="d15">
    <References/>
    <Presentation name="NTNU A3 Template"
     ↪  description="Creates 297mm x 420mm size drawing"
     ↪  tooltip="This NX template example creates an A3
     ↪  size drawing that does not reference any existing
     ↪  models.">
      <PreviewImage type="UGPart"
       ↪  location="drawing_template.jpg"/>
    </Presentation>
    <ObjectData class="DrawingTemplate">
        <TemplateFileType>none</TemplateFileType>
        <Filename>NTNU-IPM_a3_template.prt</Filename>
        <Units>Metric</Units>
    </ObjectData>
```

```
  </PaletteEntry>

   <PaletteEntry id="d16">
    <References/>
    <Presentation name="NTNU A4 Template"
     ↪ description="Creates 210mm x 297mm size drawing"
     ↪ tooltip="This NX template example creates an A4
     ↪ size drawing that does not reference any existing
     ↪ models.">
      <PreviewImage type="UGPart"
       ↪ location="drawing_template.jpg"/>
    </Presentation>
    <ObjectData class="DrawingTemplate">
        <TemplateFileType>none</TemplateFileType>
        <Filename>NTNU-IPM_a4_template.prt</Filename>
        <Units>Metric</Units>
    </ObjectData>
  </PaletteEntry>


</Palette>
```

G

Code for PDF Export

G.1 cmd_PDF_call.vb

```
1  Option Strict Off
2  Imports System
3  Imports NXOpen
4
5  Imports System.Diagnostics
6
7  Module NXJournal
8      Sub Main(ByVal args() As String)
9
10         Dim theSession As Session = Session.GetSession()
11         Dim theUI As UI = UI.GetUI()
12
13         'Getting the file path for the executable file
14
15         Dim strPath As String = GetInputFilePath()
16         Dim inputArgs As String = " /c " + strPath
17
18         'Starting Command Prompt with the file path for the
               executable as input
19
20         Dim startInfo As New ProcessStartInfo("cmd.exe")
21         startInfo.WindowStyle = ProcessWindowStyle.Normal
22         startInfo.Arguments = inputArgs
23         Process.Start(startInfo)
24
25     End Sub
26     ' Searching for the executable file "PDF_Exporter.exe, in all
           folders at the desktop
```

```
27      ' The folder containing the cad files has to be placed on the
            desktop for the program to find it
28
29      Function GetInputFilePath()
30
31          'Searching through all files at the desktop.
32
33          For Each filename As String In IO.Directory.GetFiles(My.
                Computer.FileSystem.SpecialDirectories.Desktop, "
                PDF_Exporter.exe", IO.SearchOption.AllDirectories)
34
35
36              Return filename
37
38              Exit For
39          Next
40
41      End Function
42  End Module
```

G.2 PDF_Exporter.exe

```
1   Option Strict Off
2   Imports System
3   Imports NXOpen
4   Imports NXOpen.Drawings
5   Imports NXOpen.UF
6
7
8
9   Module PRT_to_PDF
10
11      Dim session As Session = Session.GetSession()
12      Dim ufs As UFSession = UFSession.GetUFSession()
13      Dim lw As ListingWindow = session.ListingWindow
14
15      Sub Main(ByVal args As String())
16
17          'Getting the file path for generator.prt
18
19          Dim filename As String = "Generator.prt"
20
21          Dim filePath As String = GetInputFilePath(filename)
22
23          'Getting the output file path
24
25          Dim outPath As String = GetOutputFilePath
26
```

```
27          ' Loading part
28
29          Dim theSession As NXOpen.Session = NXOpen.Session.
                GetSession()
30          Dim markId1 As NXOpen.Session.UndoMarkId = Nothing
31          markId1 = theSession.SetUndoMark(NXOpen.Session.
                MarkVisibility.Visible, "Load Part")
32
33          Dim basePart1 As NXOpen.BasePart = Nothing
34          Dim partLoadStatus1 As NXOpen.PartLoadStatus = Nothing
35          basePart1 = theSession.Parts.OpenActiveDisplay(filePath,
                NXOpen.DisplayPartOption.AllowAdditional,
                partLoadStatus1)
36
37          'Set loaded part as workpart
38
39          Dim workPart As NXOpen.Part = theSession.Parts.Work
40
41          Dim displayPart As NXOpen.Part = theSession.Parts.Display
42
43          'Entering the drafting application
44
45          theSession.ApplicationSwitchImmediate("UG_APP_DRAFTING")
46
47          'Export each drawing to PDF
48
49          ProcessDrawings(session.Parts.Display(), filename, outPath
                )
50
51          ' Close open part files
52
53          session.Parts.CloseAll(BasePart.CloseModified.
                CloseModified, Nothing)
54
55          ufs.Undo.DeleteAllMarks()
56
57      End Sub
58
59          ' The following subroutine performs the PDF exporting
                process.
60
61      Sub ProcessDrawings(ByVal thePart As Part, ByVal fileName As
            String, ByVal outPath As String)
62
63          Dim workPart As Part = thePart
64
65          ' Returns if no part is not found
66
67          If thePart.Tag = Tag.Null Then Return
68
```

```
69              'Defining suppression expressions for the two
                    fastening concepts
70
71              Dim ExpConcept1 As Expression
72              Dim ExpConcept2 As Expression
73
74              'Getting the expressions values for the concept
                    suppression states.
75
76              ExpConcept1 = workPart.Expressions.FindObject("
                    DS_Fastening_Concept1_Suppression_State")
77              ExpConcept2 = workPart.Expressions.FindObject("
                    DS_Fastening_Concept2_Suppression_State")
78
79              'Defines drawing sheet as object
80
81              Dim drawingSheets As DrawingSheet() = thePart.
                    DrawingSheets.ToArray
82              Dim mySheet As DrawingSheet = Nothing
83
84              'Getting the part name to string
85
86              Dim partName As String = thePart.Leaf.ToString()
87
88                  'Looping through all drawing sheets in the part
89
90              For Each mySheet In drawingSheets
91
92                  'Skips the fastener drawings if the concepts are
                        not applied
93
94                  If (ExpConcept1.Value = 0 And mySheet.Name = "I-
                        Profile") Or (ExpConcept2.Value = 0 And
                        mySheet.Name = "Ring_with_profiles") Then
95                      Continue For
96                  End If
97
98                  'Opening sheet
99
100                 mySheet.Open()
101
102                 'Updates the part list and balloon view in the
                        assembly drawing.
103
104             If(mySheet.Name = "Sammenstilling") Then
105
106                  Dim n As Integer = 0
107                  Dim myPlists() As Tag
108                  Dim pTag As Tag
109
```

```
110                 ufs.Plist.AskTags(myPlists, n)
111
112                 For Each pTag In myPlists
113                 ufs.Plist.Update(pTag)
114                 Next
115
116             End If
117
118             'Using the file name and sheet names to create output
                    names
119
120             Dim pdf_output_name As String = ""
121             pdf_output_name = fileName
122             pdf_output_name = pdf_output_name.Replace(".prt", "")
123             Dim sheetName As String = mySheet.Name
124             pdf_output_name = String.Concat(pdf_output_name, "_",
                    sheetName, ".pdf")
125
126                 'Defining PDF builder
127
128                 Dim printPDFBuilder1 As PrintPDFBuilder
129                 printPDFBuilder1 = workPart.PlotManager.
                        CreatePrintPdfbuilder()
130
131                 'Defining metric units
132
133                 Dim dwgUnits As DrawingSheet.Unit =
                        PrintPDFBuilder.UnitsOption.Metric
134
135                 'Defining PDF setup
136
137                 printPDFBuilder1.Scale = 1.0
138                 printPDFBuilder1.Action = PrintPDFBuilder.
                        ActionOption.New
139                 printPDFBuilder1.Action = PrintPDFBuilder.
                        ActionOption.Native
140                 printPDFBuilder1.Append = False
141                 printPDFBuilder1.Colors = PrintPDFBuilder.Color.
                        BlackOnWhite
142                 printPDFBuilder1.Widths = PrintPDFBuilder.Width.
                        StandardWidths
143                 printPDFBuilder1.Size = PrintPDFBuilder.SizeOption
                        .ScaleFactor
144                 printPDFBuilder1.XDimension = mySheet.Length()
145                 printPDFBuilder1.YDimension = mySheet.Height()
146                 printPDFBuilder1.RasterImages = True
147                 printPDFBuilder1.ImageResolution = PrintPDFBuilder
                        .ImageResolutionOption.Medium
148                 printPDFBuilder1.ShadedGeometry = True
149
```

```
150                 Dim nXObject1 As NXObject
151                 Dim sheets1(0) As NXObject
152                 sheets1(0) = mySheet
153
154                 'Processing sheet
155
156                 printPDFBuilder1.SourceBuilder.SetSheets(sheets1)
157                 printPDFBuilder1.Filename = outPath +
                        pdf_output_name
158
159             nXObject1 = printPDFBuilder1.Commit()
160
161             printPDFBuilder1.Destroy()
162
163         Next
164
165     End Sub
166
167     'This function finds the file path to a part named Generator.
            prt, in all folders saved at the desktop
168     Function GetInputFilePath(fName As String)
169
170         For Each filename As String In IO.Directory.GetFiles(My.
                Computer.FileSystem.SpecialDirectories.Desktop, fName,
                 IO.SearchOption.AllDirectories)
171
172             Return filename
173
174             Exit For
175         Next
176
177     End Function
178
179     'This function finds the file path to a folder namet "Machine
            Drawings", in all folders saved at the desktop
180     Function GetOutputFilePath()
181
182         For Each dirName As String In IO.Directory.GetDirectories(
                My.Computer.FileSystem.SpecialDirectories.Desktop, "
                Machine Drawings", IO.SearchOption.AllDirectories)
183
184             Return dirName + "\"
185
186             Exit For
187         Next
188     End Function
189
190 End Module
```

H

Code for Automatic Analysis

H.1 Modal Analysis, Concept 1

```
1
2  Imports System
3  Imports NXOpen
4  Imports NXOpen.UF
5  Imports System.Collections.Generic
6
7  Module NXJournal
8      Sub Main(ByVal args() As String)
9
10         Dim theSession As Session = Session.GetSession()
11
12         ' The script is opened from a .sim file
13         ' In order to mesh, mesh mate and assign materials, we
               need to enter the .fem file
14
15         If IsNothing(theSession.Parts.BaseWork) Then
16             'active part required
17             Return
18         End If
19
20         Dim mySimPart As CAE.SimPart
21         Try
22             mySimPart = CType(theSession.Parts.BaseWork, CAE.
                   SimPart)
23         Catch ex As Exception
24             'not a sim part
25             Return
26         End Try
27
```

```
28          'switch to FEM part
29          MsgBox("Opening FEM part")
30          theSession.Parts.SetWork(mySimPart.FemPart)
31
32          ' Starting to work on the FEM part
33
34          Dim theUfSession As UFSession = UFSession.GetUFSession()
35
36          ' Setting the FEM part as workpart
37
38          Dim workFemPart As NXOpen.CAE.FemPart = mySimPart.FemPart
39
40          ' Defining expressions for materials, element size and
                number of stator fasteners
41
42          Dim materialFastenerExp As Expression
43          Dim materialStatorExp As Expression
44          Dim materialChassisExp As Expression
45          Dim fastenerExp As Expression
46          Dim elementSizeExp As Expression
47
48          ' Assigning the expression values
49
50          materialFastenerExp = workFemPart.Expressions.FindObject("
                Fastener_Material")
51          materialStatorExp = workFemPart.Expressions.FindObject("
                Stator_Material")
52          materialChassisExp = workFemPart.Expressions.FindObject("
                Chassis_Material")
53
54          fastenerExp = workFemPart.Expressions.FindObject("
                Fasteners_pr_Segment")
55          elementSizeExp = workFemPart.Expressions.FindObject("
                Element_Size")
56
57          ' Getting the string valus from the material expressions
58
59          Dim matFast As String = "PhysicalMaterial[" +
                materialFastenerExp.StringValue + "]"
60          Dim matStat As String = "PhysicalMaterial[" +
                materialStatorExp.StringValue + "]"
61          Dim matChas As String = "PhysicalMaterial[" +
                materialChassisExp.StringValue + "]"
62
63          ' Getting the element size value from the element size
                expression
64
65          Dim elementSize As Integer = elementSizeExp.Value
66
67          ' Defining the FE model
```

```
68
69          Dim fEModel1 As NXOpen.CAE.FEModel = CType(workFemPart.
                FindObject("FEModel"), NXOpen.CAE.FEModel)
70
71          Dim meshManager1 As NXOpen.CAE.MeshManager = CType(
                fEModel1.Find("MeshManager"), NXOpen.CAE.MeshManager)
72
73          Dim nullNXOpen_CAE_Mesh3d As NXOpen.CAE.Mesh3d = Nothing
74
75          ' Defining mesh builder
76
77          Dim mesh3dTetBuilder1 As NXOpen.CAE.Mesh3dTetBuilder =
                Nothing
78          mesh3dTetBuilder1 = meshManager1.CreateMesh3dTetBuilder(
                nullNXOpen_CAE_Mesh3d)
79
80          ' Choosing element type
81
82          mesh3dTetBuilder1.ElementType.ElementTypeName = "CTETRA
                (10)"
83
84          ' Defining milimeter as desired element unit
85
86          Dim unit1 As NXOpen.Unit = CType(workFemPart.
                UnitCollection.FindObject("MilliMeter"), NXOpen.Unit)
87
88          ' Setting mesh properties
89
90          mesh3dTetBuilder1.PropertyTable.
                SetBaseScalarWithDataPropertyValue("quad mesh overall
                edge size", elementSize, unit1)
91
92          Dim nullNXOpen_Unit As NXOpen.Unit = Nothing
93
94          mesh3dTetBuilder1.PropertyTable.
                SetBaseScalarWithDataPropertyValue("small feature
                value", "2", nullNXOpen_Unit)
95
96          ' Defining material builder
97
98          Dim physicalMaterialListBuilder1 As NXOpen.PhysMat.
                PhysicalMaterialListBuilder = Nothing
99          physicalMaterialListBuilder1 = workFemPart.MaterialManager
                .PhysicalMaterials.CreateListBlockBuilder()
100
101         Dim physicalMaterialAssignBuilder1 As NXOpen.PhysMat.
                PhysicalMaterialAssignBuilder = Nothing
102         physicalMaterialAssignBuilder1 = workFemPart.
                MaterialManager.PhysicalMaterials.
                CreateMaterialAssignBuilder()
```

```
103
104         ' Defining material for the fasteners, stator and chassis
105
106         Dim physicalMaterialFastener As NXOpen.PhysicalMaterial =
                CType(workFemPart.MaterialManager.PhysicalMaterials.
                FindObject(matFast), NXOpen.PhysicalMaterial)
107
108         Dim physicalMaterialStator As NXOpen.PhysicalMaterial =
                CType(workFemPart.MaterialManager.PhysicalMaterials.
                FindObject(matStat), NXOpen.PhysicalMaterial)
109
110         Dim physicalMaterialChassis As NXOpen.PhysicalMaterial =
                CType(workFemPart.MaterialManager.PhysicalMaterials.
                FindObject(matChas), NXOpen.PhysicalMaterial)
111
112         ' Defining mesh mate builder
113
114         Dim nullNXOpen_CAE_MeshMate As NXOpen.CAE.MeshMate =
                Nothing
115
116         Dim mMCCreateBuilder1 As NXOpen.CAE.MMCCreateBuilder =
                Nothing
117         mMCCreateBuilder1 = fEModel1.MeshControls.
                CreateMmcCreateBuilder(nullNXOpen_CAE_MeshMate)
118
119         ' Mesh mate search distance
120
121         mMCCreateBuilder1.DistTolerance.RightHandSide = "0.0254"
122
123         mMCCreateBuilder1.DistTolerance.RightHandSide = "0.0001"
124
125         'Defining counters
126
127         Dim counter As Integer = 0
128
129         Dim subCounter As Integer = 0
130
131         'Defining object list for the fastener meshing
132
133         Dim objects1(fastenerExp.Value - 1) As NXOpen.NXObject
134
135         'Defining list for applying mesh mating conditions to all
                bodies
136
137         Dim objects2(fastenerExp.Value + 1) As Boolean
138
139         'Defining single object for material assginment of the
                stator and chassis
140
141         Dim object0(0) As NXOpen.NXObject
```

```
142
143         Dim ufs = NXOpen.UF.UFSession.GetUFSession
144         Dim thisObject As NXOpen.NXObject
145         Dim thisTag As NXOpen.Tag = NXOpen.Tag.Null
146
147         ' Looping through all objects, looking for bodies
148
149         Do
150             thisTag = ufs.Obj.CycleAll(workFemPart.Tag, thisTag)
151             If thisTag <> NXOpen.Tag.Null Then
152                 thisObject = NXOpen.Utilities.NXObjectManager.Get(
                        thisTag)
153
154                 If thisObject.JournalIdentifier.contains("CAE_Body
                        ") Then
155
156                     ' Defining cAEBody1 as CAE body object
157
158                     Dim cAEBody1 As NXOpen.CAE.CAEBody = CType(
                            workFemPart.FindObject(thisObject.
                            JournalIdentifier), NXOpen.CAE.CAEBody)
159
160                     ' Assigning material to the chassis
161
162                     If counter = 0 Then
163                         object0(0) = cAEBody1
164                         physicalMaterialChassis.AssignObjects(
                                object0)
165                     End If
166
167                     ' Assigning material to the stator
168
169                     If counter = 1 Then
170                         object0(0) = cAEBody1
171                         physicalMaterialStator.AssignObjects(
                                object0)
172                     End If
173
174                     ' Creates a list of all fastener bodies, for
                            material assignment
175                     ' Adding all fastener bodies to a list for
                            meshing
176
177                     If counter > 1 Then
178
179                         objects1(subCounter) = cAEBody1
180
181                         mesh3dTetBuilder1.SelectionList.Add(
                                cAEBody1)
182
```

```
183                         subCounter = subCounter + 1
184
185                     End If
186
187                     ' Adding all bodies for Auto Selection Mesh
                            Mating
188
189                     objects2(counter) = mMCCreateBuilder1.
                            AutoSelection.Add(cAEBody1)
190
191                     counter = counter + 1
192
193                 End If
194             End If
195         Loop Until thisTag = NXOpen.Tag.null
196
197         ' Assign Material to the fastner bodies
198
199         physicalMaterialFastener.AssignObjects(objects1)
200
201         ' Commits mesh mate conditions
202
203         Dim mmcs1() As NXOpen.CAE.MeshMate
204         mmcs1 = mMCCreateBuilder1.CommitMmcs()
205
206         Dim destinationCollectorBuilder1 As NXOpen.CAE.
                DestinationCollectorBuilder = Nothing
207         destinationCollectorBuilder1 = mesh3dTetBuilder1.
                ElementType.DestinationCollector
208
209         mesh3dTetBuilder1.PropertyTable.
                SetBaseScalarWithDataPropertyValue("quad mesh overall
                edge size", "20", unit1)
210
211         ' Commits mesh
212
213         Dim meshes1() As NXOpen.CAE.Mesh
214         meshes1 = mesh3dTetBuilder1.CommitMesh()
215
216         mesh3dTetBuilder1.Destroy()
217
218         physicalMaterialAssignBuilder1.Destroy()
219
220         physicalMaterialListBuilder1.Destroy()
221
222         mMCCreateBuilder1.Destroy()
223
224         ' Updating the model after mesh mating conditions has been
                applied
225
```

```
226         Dim markId4 As NXOpen.Session.UndoMarkId = Nothing
227         markId4 = theSession.SetUndoMark(NXOpen.Session.
                MarkVisibility.Visible, "Update FE Model")
228
229         fEModel1.UpdateFemodel()
230
231         'switching back to SIM part
232
233         MsgBox("Switching back to SIM part")
234         theSession.Parts.SetWork(mySimPart)
235
236     End Sub
237
238 End Module
```

H.2 Modal Analysis, Concept 2

```
1  Imports System
2  Imports NXOpen
3  Imports NXOpen.UF
4  Imports System.Collections.Generic
5
6  Module NXJournal
7      Sub Main(ByVal args() As String)
8
9          ' The script is opened from a .sim file
10         ' In order to mesh, mesh mate and assign materials, we
               need to enter the .fem file
11
12         Dim theSession As Session = Session.GetSession()
13         If IsNothing(theSession.Parts.BaseWork) Then
14             'active part required
15             Return
16         End If
17
18         Dim mySimPart As CAE.SimPart
19         Try
20             mySimPart = CType(theSession.Parts.BaseWork, CAE.
                   SimPart)
21         Catch ex As Exception
22             'not a sim part
23             Return
24         End Try
25
26         'switch to FEM part
27         MsgBox("Opening FEM part")
28         theSession.Parts.SetWork(mySimPart.FemPart)
29
```

```
30          ' Starting to work on the FEM part
31
32          Dim theUfSession As UFSession = UFSession.GetUFSession()
33
34          ' Setting the FEM part as workpart
35
36          Dim workFemPart As NXOpen.CAE.FemPart = mySimPart.FemPart
37
38          ' Defining expressions for the materials
39
40          Dim materialFastenerExp As Expression
41          Dim materialStatorExp As Expression
42          Dim materialChassisExp As Expression
43
44          ' Assigning the expression values
45
46          materialFastenerExp = workFemPart.Expressions.FindObject("
                Fastener_Material")
47          materialStatorExp = workFemPart.Expressions.FindObject("
                Stator_Material")
48          materialChassisExp = workFemPart.Expressions.FindObject("
                Chassis_Material")
49
50          ' Getting the string valus from the material expressions
51
52          Dim matFast As String = "PhysicalMaterial[" +
                materialFastenerExp.StringValue + "]"
53          Dim matStat As String = "PhysicalMaterial[" +
                materialStatorExp.StringValue + "]"
54          Dim matChas As String = "PhysicalMaterial[" +
                materialChassisExp.StringValue + "]"
55
56          ' Defining the FE model
57
58          Dim fEModel1 As NXOpen.CAE.FEModel = CType(workFemPart.
                FindObject("FEModel"), NXOpen.CAE.FEModel)
59
60          ' Defining material builder
61
62          Dim physicalMaterialListBuilder1 As NXOpen.PhysMat.
                PhysicalMaterialListBuilder = Nothing
63          physicalMaterialListBuilder1 = workFemPart.MaterialManager
                .PhysicalMaterials.CreateListBlockBuilder()
64
65          Dim physicalMaterialAssignBuilder1 As NXOpen.PhysMat.
                PhysicalMaterialAssignBuilder = Nothing
66          physicalMaterialAssignBuilder1 = workFemPart.
                MaterialManager.PhysicalMaterials.
                CreateMaterialAssignBuilder()
67
```

```
68          ' Defining material for the fasteners, stator and chassis
69
70          Dim physicalMaterialFastener As NXOpen.PhysicalMaterial =
                CType(workFemPart.MaterialManager.PhysicalMaterials.
                FindObject(matFast), NXOpen.PhysicalMaterial)
71
72          Dim physicalMaterialStator As NXOpen.PhysicalMaterial =
                CType(workFemPart.MaterialManager.PhysicalMaterials.
                FindObject(matStat), NXOpen.PhysicalMaterial)
73
74          Dim physicalMaterialChassis As NXOpen.PhysicalMaterial =
                CType(workFemPart.MaterialManager.PhysicalMaterials.
                FindObject(matChas), NXOpen.PhysicalMaterial)
75
76          ' Defining mesh mate builder
77
78          Dim nullNXOpen_CAE_MeshMate As NXOpen.CAE.MeshMate =
                Nothing
79
80          Dim mMCCreateBuilder1 As NXOpen.CAE.MMCCreateBuilder =
                Nothing
81          mMCCreateBuilder1 = fEModel1.MeshControls.
                CreateMmcCreateBuilder(nullNXOpen_CAE_MeshMate)
82
83          ' Mesh mate search distance
84
85          mMCCreateBuilder1.DistTolerance.RightHandSide = "0.0254"
86
87          mMCCreateBuilder1.DistTolerance.RightHandSide = "0.0001"
88
89          'Defining counter
90
91          Dim counter As Integer = 0
92
93          'Defining single object for material assginment of the
                stator and chassis
94
95          Dim object0(0) As NXOpen.NXObject
96
97          'Defining list for applying mesh mating conditions to all
                bodies
98
99          Dim objects1(2) As Boolean
100
101         Dim ufs = NXOpen.UF.UFSession.GetUFSession
102         Dim thisObject As NXOpen.NXObject
103         Dim thisTag As NXOpen.Tag = NXOpen.Tag.Null
104
105          'Looping through all objects, looking for bodies
106
```

```
107         Do
108             thisTag = ufs.Obj.CycleAll(workFemPart.Tag, thisTag)
109             If thisTag <> NXOpen.Tag.Null Then
110                 thisObject = NXOpen.Utilities.NXObjectManager.Get(
                        thisTag)
111                 If thisObject.JournalIdentifier.contains("CAE_Body
                        ") Then
112
113                     'Defining cAEBody1 as CAE body object
114
115                     Dim cAEBody1 As NXOpen.CAE.CAEBody = CType(
                            workFemPart.FindObject(thisObject.
                            JournalIdentifier), NXOpen.CAE.CAEBody)
116
117                     ' Assigning material to the ring with
                            fasteners
118
119                     If counter = 0 Then
120                         object0(0) = cAEBody1
121                         physicalMaterialFastener.AssignObjects(
                                object0)
122                     End If
123
124                     ' Assigning material to the chassis
125
126                     If counter = 1 Then
127                         object0(0) = cAEBody1
128                         physicalMaterialChassis.AssignObjects(
                                object0)
129                     End If
130
131                     ' Assigning material to the stator
132
133                     If counter = 2 Then
134                         object0(0) = cAEBody1
135                         physicalMaterialStator.AssignObjects(
                                object0)
136                     End If
137
138                      'Adding all bodies for Auto Selection Mesh
                            Mating
139
140                     objects1(counter) = mMCCreateBuilder1.
                            AutoSelection.Add(cAEBody1)
141
142                     counter = counter + 1
143
144                 End If
145             End If
146         Loop Until thisTag = NXOpen.Tag.null
```

```
147
148         ' Commits mesh
149
150         Dim mmcs1() As NXOpen.CAE.MeshMate
151         mmcs1 = mMCCreateBuilder1.CommitMmcs()
152
153         physicalMaterialAssignBuilder1.Destroy()
154
155         physicalMaterialListBuilder1.Destroy()
156
157         mMCCreateBuilder1.Destroy()
158
159         ' Updating the model after mesh mating conditions has been
                applied
160
161         Dim markId4 As NXOpen.Session.UndoMarkId = Nothing
162         markId4 = theSession.SetUndoMark(NXOpen.Session.
                MarkVisibility.Visible, "Update FE Model")
163
164         fEModel1.UpdateFemodel()
165
166         'Switching back to SIM part
167
168         MsgBox("switching back to SIM part")
169         theSession.Parts.SetWork(mySimPart)
170
171     End Sub
172
173 End Module
```

I

Code for STEP Export

This code has been taken directly from www.NXJournaling.com [27].

```
1  Option Strict Off
2  Imports System
3  Imports System.Collections.Generic
4  Imports NXOpen
5
6  Module Module2
7
8      Dim theSession As Session = Session.GetSession()
9      Dim workPart As Part = theSession.Parts.Work
10     Dim displayPart As Part = theSession.Parts.Display
11     Dim lw As ListingWindow = theSession.ListingWindow
12
13
14     Sub Main()
15
16         lw.Open()
17
18         Dim mySelectedObjects As New List(Of NXObject)
19         Dim step214File As String
20         Dim outputPath As String = IO.Path.GetDirectoryName(
               workPart.FullPath)
21         Dim outputFile As String = IO.Path.
               GetFileNameWithoutExtension(workPart.FullPath)
22         outputFile = IO.Path.Combine(outputPath, outputFile & ".
               stp")
23
24         Dim STEP214UG_DIR As String = theSession.
               GetEnvironmentVariableValue("STEP214UG_DIR")
25         step214File = IO.Path.Combine(STEP214UG_DIR, "ugstep214.
               def")
```

```
26
27          If Not IO.File.Exists(step214File) Then
28              MsgBox("The step214 settings file (ugstep214.def) was
                    not found." & vbCrLf & _
29                  "This journal will now exit.", vbOKOnly +
                        vbCritical)
30              Exit Sub
31          Else
32              'lw.WriteLine("STEP214 definition file found at: " &
                    step214File)
33          End If
34
35          'output all solids in the file to STEP
36          For Each temp As Body In workPart.Bodies
37              If temp.IsSolidBody Then
38                  mySelectedObjects.Add(temp)
39              End If
40          Next
41
42          'export to STEP214 file
43          Dim step214Creator1 As Step214Creator
44          step214Creator1 = theSession.DexManager.
                CreateStep214Creator()
45
46          step214Creator1.SettingsFile = step214File
47          step214Creator1.ObjectTypes.Solids = True
48          step214Creator1.LayerMask = "1-256"
49          step214Creator1.InputFile = workPart.FullPath
50          step214Creator1.OutputFile = outputFile
51          step214Creator1.FileSaveFlag = False
52          step214Creator1.ExportSelectionBlock.SelectionScope =
                ObjectSelector.Scope.SelectedObjects
53
54          Dim added1 As Boolean
55          added1 = step214Creator1.ExportSelectionBlock.
                SelectionComp.Add(mySelectedObjects.ToArray)
56
57          Dim nXObject1 As NXObject
58          nXObject1 = step214Creator1.Commit()
59
60          step214Creator1.Destroy()
61          lw.Close()
62
63      End Sub
64
65      Sub AddSolidsFromLayer(ByVal layer As Integer, ByRef objList
            As List(Of NXObject))
66
67          Dim tempselectedobjects() As NXObject
68          Dim tempSolidObj As Body
```

```
69
70          tempselectedobjects = workPart.Layers.GetAllObjectsOnLayer
                (layer)
71          'filter out the solid bodies on the layer and add them to
                the export list
72          For Each obj As NXObject In tempselectedobjects
73              If TypeOf obj Is Body Then
74                  tempSolidObj = CType(obj, Body)
75                  If tempSolidObj.IsSolidBody Then
76                      objList.Add(tempSolidObj)
77                  End If
78              End If
79          Next
80
81
82      End Sub
83
84      Function SelectObjects(ByVal prompt As String, _
85             ByRef selObj As NXObject()) As Selection.Response
86
87          Dim theUI As UI = UI.GetUI
88          Dim typeArray() As Selection.SelectionType = _
89              {Selection.SelectionType.All}
90
91          Dim resp As Selection.Response = theUI.SelectionManager.
                SelectObjects( _
92                  prompt, "Select Objects for STEP Export", _
93                  Selection.SelectionScope.WorkPart, _
94                  False, typeArray, selObj)
95
96          If resp = Selection.Response.ObjectSelected Or _
97                  resp = Selection.Response.ObjectSelectedByName Or
                        _
98                  resp = Selection.Response.Ok Then
99              Return Selection.Response.Ok
100         Else
101             Return Selection.Response.Cancel
102         End If
103
104     End Function
105
106     Public Function GetUnloadOption(ByVal dummy As String) As
            Integer
107
108         'Unloads the image when the NX session terminates
109         GetUnloadOption = NXOpen.Session.LibraryUnloadOption.
                AtTermination
110
111     End Function
112
```

```
113 End Module
```