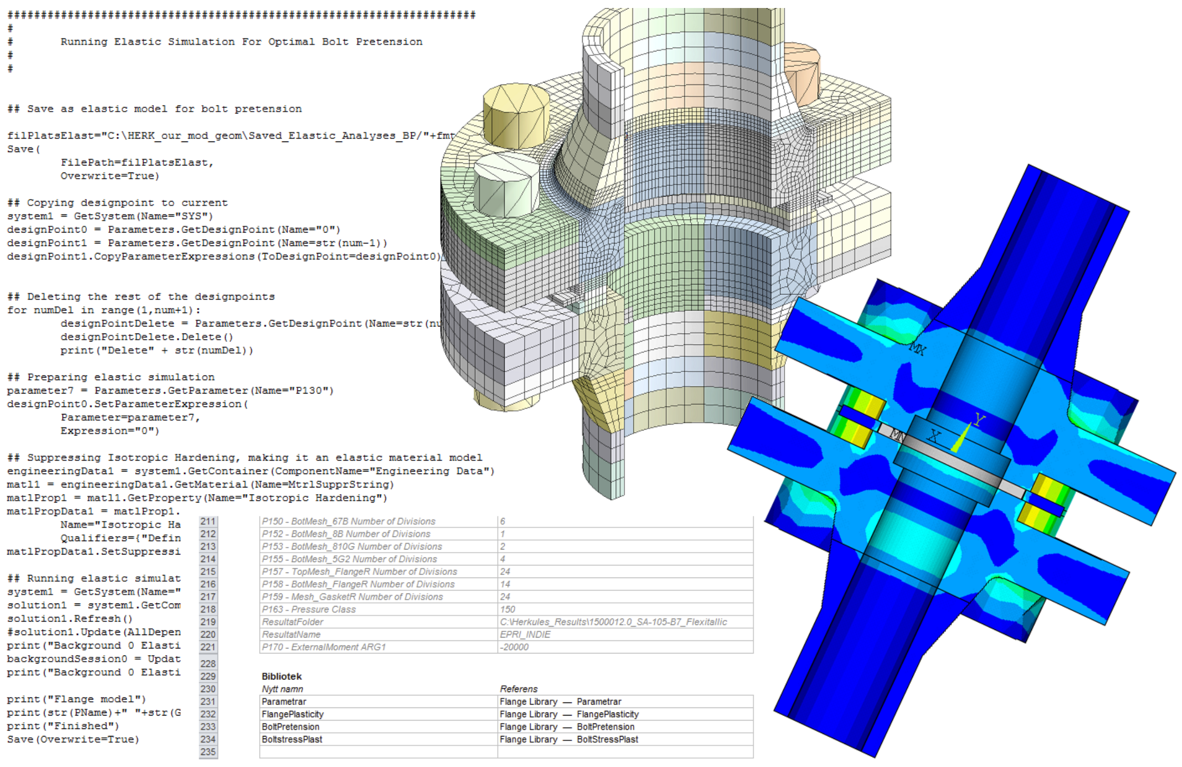
CHALMERS
UNIVERSITY OF TECHNOLOGY



```
######################################################################
#
#       Running Elastic Simulation For Optimal Bolt Pretension
#
#


## Save as elastic model for bolt pretension

filPlatsElast="C:\HERK_our_mod_geom\Saved_Elastic_Analyses_BP/"+fmt
Save(
        FilePath=filPlatsElast,
        Overwrite=True)

## Copying designpoint to current
system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
designPoint1 = Parameters.GetDesignPoint(Name=str(num-1))
designPoint1.CopyParameterExpressions(ToDesignPoint=designPoint0)


## Deleting the rest of the designpoints
for numDel in range(1,num+1):
        designPointDelete = Parameters.GetDesignPoint(Name=str(nu
        designPointDelete.Delete()
        print("Delete" + str(numDel))

## Preparing elastic simulation
parameter7 = Parameters.GetParameter(Name="P130")
designPoint0.SetParameterExpression(
        Parameter=parameter7,
        Expression="0")

## Suppressing Isotropic Hardening, making it an elastic material model
engineeringData1 = system1.GetContainer(ComponentName="Engineering Data")
matl1 = engineeringData1.GetMaterial(Name=MtrlSupprString)
matlProp1 = matl1.GetProperty(Name="Isotropic Hardening")
matlPropData1 = matlProp1.
        Name="Isotropic Ha
        Qualifiers={"Defin
matlPropData1.SetSuppressi

## Running elastic simulat
system1 = GetSystem(Name="
solution1 = system1.GetCom
solution1.Refresh()
#solution1.Update(AllDepen
print("Background 0 Elasti
backgroundSession0 = Updat
print("Background 0 Elasti

print("Flange model")
print(str(PName)+" "+str(G
print("Finished")
Save(Overwrite=True)
```

P150 - BotMesh_67B Number of Divisions	6
P152 - BotMesh_8B Number of Divisions	1
P153 - BotMesh_810G Number of Divisions	2
P155 - BotMesh_5G2 Number of Divisions	4
P157 - TopMesh_FlangeR Number of Divisions	24
P158 - BotMesh_FlangeR Number of Divisions	14
P159 - Mesh_GasketR Number of Divisions	24
P163 - Pressure Class	150
ResultatFolder	C:\Herkules_Results\1500012.0_SA-105-B7_Flexitallic
ResultatName	EPRI_INDIE
P170 - ExternalMoment ARG1	-20000

Bibliotek

Nytt namn	*Referens*
Parametrar	Flange Library — Parametrar
FlangePlasticity	Flange Library — FlangePlasticity
BoltPretension	Flange Library — BoltPretension
BoltstressPlast	Flange Library — BoltStressPlast

Method development for optimizing the pre-tension of flange joints

Master's thesis in Applied Mechanics

PETER OSTROWSKIS
JONAS OHLSSON

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016:19

Method development for optimizing the pre-tension of flange joints

PETER OSTROWSKIS
JONAS OHLSSON

Department of Applied Mechanics
Division of Material and Computational Mechanics
Chalmers University of Technology
Gothenburg, Sweden 2016

Method development for optimizing the pre-tension of flange joints
PETER OSTROWSKIS
JONAS OHLSSON

Supervisor: Johan Hemström, Ringhals
Supervisor: Torkel Davidsson, Uniso Technologies
Examiner: Martin Fagerström, Department of Applied Mechanics

Master's Thesis 2016:19
ISSN 1652-8557
Department of Applied Mechanics
Division of Material and Computational Mechanics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Compilation of images from the various software programs used.

Gothenburg, Sweden 2016

Method development for optimizing the pre-tension of flange joints
PETER OSTROWSKIS
JONAS OHLSSON
Department of Applied Mechanics
Chalmers University of Technology

Abstract

At this moment flange joints are designed with guidance from handbooks according to criteria set up by standards from The American Society of Mechanical Engineers (ASME) or the European Committee for Standardization (CEN). These standards are generalized and take the geometric factors and load cases into account in a rather simplified manner. Designing the piping systems according to these standards can in some cases lead to leakage in the flange joint, which is assumed to be a consequence of not applying a sufficient bolt pre-tension. In this thesis, as an attempt of increasing the pre-tension of the bolts, the allowable loads were evaluated using FE-analysis. By intertwining the FE-software Ansys Workbench and the prepost software Herkules with a Python script, a new way of computing the design loads of flange joints was developed. Instead of designing the flange joints in accordance with the ASME-standard, the FE-analyzes estimate the allowable loads in an automated process. This automated process uses a few parametrized FE-models that can be transformed into any flange joint when inserting the correct parameters. Apart from the geometrical features, the input parameters also control the mesh and the magnitude of the loads. And by establishing output parameters, the load stepping in the automated process can be modified during the FE-analysis. By implementing this method when designing piping systems, a more accurate estimation is obtained of which flange joint will be suitable for the piping system. Therefore, it is recommended that this method is used as a new design standard.

Preface

This Master thesis was performed for the Department of Applied Mechanics and the Division of Material and Computational Mechanics at Chalmers. The work during this Master thesis was carried out at the Swedish nuclear power plant Ringhals and at the consulting firm Uniso Technologies. It was carried out as an attempt of developing a new method to determine the flange joint design loads and the maximum allowed bolt pre-tensions. This report contains work done from Januari to June 2016. Our supervisors on the project has been Johan Hemström, computational engineer at Ringhals, and Torkel Davidsson, computational engineer at Uniso Technologies. Martin Fagerström, Associate Professor at Chalmers, acted as our examiner. However, the thesis has been made solely by the authors.

Acknowledgements

First and foremost we would like to thank our two supervisors Johan Hemström and Torkel Davidsson who have helped us very much during this Master thesis. They assumed the role of being both technical sounding boards, but also flange experts with extensive knowledge in mechanics. This Master thesis would never have been successful without their involvement.

Martin Fagerström has been our examiner and we thank him for his guidance through this Master thesis.

A thank you also goes out to the personnel at Ringhals and Uniso Technologies for being supportive, compassionate, and having contributed to a pleasant working environment.

Jonas Ohlsson & Peter Ostrowskis, Gothenburg, June 2016

Contents

1 Introduction

1.1 Background

A flange joint consists of two pipes connected via two flanges which are bolted together using a specific number of bolts depending on the pressure class and the size of the flange. A gasket is placed between the faces of the flanges before tightening the bolts in order to prevent leakage of the joint system. The purpose of the flange joint is to facilitate construction and easier demounting of certain parts of the pipelines. Since the maximum length of pipes are limited due to manufacturing processes, flanges enable easier construction as well. Demounting of a pipe lining is easier when using flange joints, since loosening bolts is more convenient than to cut pipes and then weld them back together.

Having a proper setup of flanges, gaskets and bolts is very critical since there often can be an extensive pressure inside the pipelines. A correct setup is very important in order to avoid malfunctioning. The current procedure of choosing the correct setup is by comparing certain values to criteria based on the ASME-norm, and these criteria vary for different pressure classes.

The flange geometry in this study will be a raised-face welded-neck (RF WN) flange. Raised-face implies that the gasket surfaces are raised above the bolting surfaces and welded-neck means that the upper side of the flange is welded on to the pipe according to box 2 in Figure 1.1 below. The basic geometry of the flange will be of this sort in all analyzes, but will differ depending on flange dimension, gasket type, material, pressure level and number of bolts.

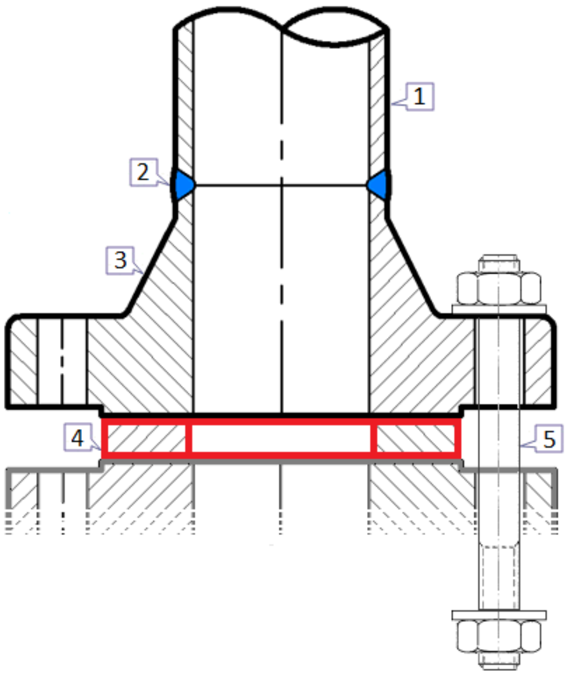


Figure 1.1: Schematic picture of flanged joint containing; [1] Pipe, [2] Weld, [3] Flange, [4] Gasket, and [5] Bolt.

When constructing the flange joint system it is of utter importance to choose the correct parts according to standards since there often is a large inner pressure that the flange system is supposed to be able to withstand. However, the inner pressure is not the only factor to take into account since the flange has to be able to withstand other loadings as well, such as the pre-tension of the bolts, temperature, and external moments from the connected pipes.

Flange joints often occur in the nuclear power industry and this study is carried out on behalf of the Swedish power plant Ringhals. The study is a part of the continuous work in progress to find methods for optimizing the dimensioning of these joint constructions, in order to increase the pre-tension of the bolts. A similar study has been carried out previously by one of the employees at Ringhals where the problem-approach was the same but the analyzes were made using hand calculations from the ASME-norms in *Mathcad*. However, the results obtained from the hand calculations gave a lower allowable pre-tension compared to the best possible flange setup obtained in tests performed by ASME and EPRI. Therefore this project aims to setup FE-analyzes of the problem, to obtain a more accurate result.

1.2 Aim of the project

At this moment, flange joints are designed with guidance from handbooks according to criteria in ASME- or EN-standards at Ringhals. These standards are generalized and take the geometric factors and load cases into account in a rather simplified manner, which results in the pre-tension of the bolts being too conservative, i.e. not tightening the bolts hard enough. These conservative pre-tensions do in some cases lead to leakage in the flange joints.

In order to make it easier for Ringhals to select which flange joint setup, and what pre-tension of the bolts that is most suitable for a certain load case, FE-analyzes will be performed. The goal is to develop a method that Ringhals can use to determine optimal pre-tensions for different flange joint setups. Using FEM would be a more accurate method than the ASME-standard based hand calculations currently used. This method would also allow Ringhals to increase the pre-tension of the bolts so that the flange joint is better able to withstand external forces and moments, as well as prevent possible leakage.

1.3 Delimitations

During this project, there has been quite many different factors to take into account while running the analyzes. A lot of parameters concerning different parts of the flange joint construction have been combined, which entails that boundaries had to be made regarding the possible maximum combinations. Therefore, the following governing parameters are the chosen ones to be considered during the FE-analyzes.

- Only the following pressure levels according to the ASME-norms will be included in the FE-analyzes; *150#*, *300#*, *400#*, *600#*, *900#* and *1500#*.
- There will only be four different flange materials included, which are *SA-105*, *SA-182 F316*, *SA-182 F316L* and *SA-182 F304*.
- When increasing the pressure class and flange size dimensions the number of connecting bolts will differ as: *4*, *8*, *12*, *16* and *20*.
- There will only be one bolt material used, i.e. *SA-193 B7*.
- Only the gaskets; *Grafex*, *Novapress 815* and *Flexitallic* will be considered.
- Only four values of pre-tension will be considered when evaluating the maximum external moments allowed for all flanges. These are *80%*, *85%*, *90%*, and *95%* of the computed maximum pre-tension value.
- No CFD-analysis will be performed concerning the behaviour of the fluid inside the pipe linings.
- Temperature transients will not be considered in the automatized FE-analyzes.

1.4 Software

Below are short descriptions of the different software which have been used during this Master Thesis.

Ansys Workbench 16 A finite element analysis tool with focus on a user-friendly graphical user interface, which has been used when creating the finite element models of flanges.

Ansys Classic This software uses the parametric design language, APDL, and works as the basis for all actions being performed in Ansys Workbench 16, such as pre-processing, solving and post-processing. The APDL commands one can insert into Ansys Workbench 16 are created by the use of APDL syntax.

Microsoft Excel with Herkules 2 Add-In When referring to Herkules in this Master Thesis it means that Microsoft Excel together with the Herkules 2 Add-In is being used. Herkules has been developed at Ringhals where one of its uses is that it lets the user load text-files and figures into Excel worksheets for easy presentation when having a lot of data content.

Python A general-purpose programming language which is compatible with running and controlling both Ansys Workbench 16.0 and Microsoft Excel. It has been used to combine the parameter library in Herkules with the parametrized Ansys Workbench models as well as control the optimization method.

Mathcad 15 The primary uses of Mathcad 15 is to verify and validate engineering calculations. This was previously used when designing the flanges in the pipe line systems.

1.5 ASME-norm

The ASME design norm criteria currently used determines the geometries of the flange joint components and how high stresses are allowed in them. In accordance with ASME III Appendix XI [1] the allowable stresses are determined with respect to the regulations stated in ASME II Materials - Part D [2]. In ASME II Materials Part D different material data are stated depending on what safety class the allowable stresses are computed for. For class 2 and class 3 components in the nuclear industry, the allowable stresses in the flange are to be calculated from 2/3 of the materials yield strength. When it comes to bolts, this value is only set to 1/5 of the materials tensile strength or 1/4 of the yield strength.

However, since ASME III Appendix XI calculates the stresses in the flange in a very conservative manner and also have very high requirements regarding the stresses in the bolts, the allowable pre-tension can in some cases be too low, which can cause leakage when the pipesystem is subjected to high external loads. Therefore, the total safety of the flange-joint can be increased by using FE-analysis together with the guidelines stated in ASME PCC-1-2010 [3], ASME Appendix XII-XIII and EPRI [8], instead of Appendix XI. When interpreting ASME PCC-1-2010 the bolt pre-tension should be within the range of 20 to 70 % of the bolt yield strength at operating temperature, in order to prevent that leakage occur. This norm also states that the gasket pressure and flange rotation must be taken into account as well. This is why the maximum and minimum gasket pressure is set as a criteria when evaluating the flange joint. When determining a limit for the flange rotation 1 degree was selected, since this was the upper limit stated in ASME PCC-1-2010 [3]. The other design norm criteria established for the flange joint are adopted from EPRI which limits the plasticity in the radius of the flange to 1% plasticity.

1.5.1 Previous work

ASME Hand Calculations

As a precursor to this Master Thesis a Mathcad document was used when calculating the allowable bolt pre-tension and allowable external moment. The magnitudes of these two loads were calculated for ASME flanges with respect to several ASME criteria. Pursuant to Appendix XI in ASME III, the allowed pre-tensions and external moments were determined for the various flange joints using analytical hand calculations in the computational software *Mathcad*. All the ASME flanges evaluated in *Mathcad* are approved with respect to the criteria in ASME III Appendix XI [1].

The allowable external moments are determined in accordance to ASME III NC-3658 [5], and the flange rotation is according to 2-14 in ASME VIII Appendix 2 [6] with influence from ASME PCC-1-2010 [3]. The complete explanation of how these analytical calculations were performed is presented in an internal report written by the supervisor Johan Hemström [4]. The full content of this report and the explanation of the analytical calculations performed will not be presented here.

EPRI

The American institute Electric Power Research Institute, known as EPRI, wrote a report in year 2000 with the title *Bolt Preload Stress for ANSI Raised-Face Flanges Using Spiral-Wound Gaskets* [8]. This report intends to establish recommended bolt pre-tensions for ANSI B16.5 raised-face flanged joints with spiral wound gaskets using FE-analysis. In order to approve the bolt preload stress they evaluate the flange joint using five different criteria concerning; *Flange rotation*, *Flange plastic strain*, *Bolt membrane stress*, *Bolt membrane + bending stress*, and *Plastic strain in gasket outer ring.* The limit of respective criteria is presented in Table 1.1.

Table 1.1: Limits of the criteria set to evaluate whether the bolt pre-tension is allowable. G_o and G_i represent the outer and inner radius of the gasket. S_m is one third of the bolt material's yield stress at a certain temperature.

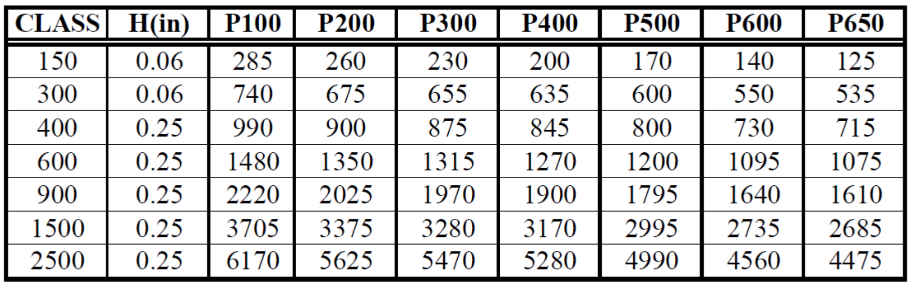
Criterion	Limit
Flange rotation	$< \left(\dfrac{0,015}{G_o - G_i}\right)$
Flange plastic strain	1%
Bolt membrane stress	$2S_m$
Bolt membrane + bending stress	$3S_m$
Plastic strain in gasket outer ring	2%

The acceptance criteria set to ensure that the bolt pre-tension are valid originates from the ASME Code, Section III in Subsection NB, and Appendix XI - XIII. In EPRI's earlier manual, *EPRI Good Bolting Practices* [12], bolt pre-tensions up to 85% of the bolt yield strength are permitted. However, the limit set for the bolt membrane stress (tensile stress in the axial direction) is not based on this manual. Section III in Subsection NB of the ASME Code limits the bolt membrane stress to 2/3 of the material yield strength. And in the same section, the *Bolt membrane + bending stress* is limited to the material yield strength which is up to 105 ksi (723 MPa) at room temperature for A193, Grade B7.

EPRI states that the criteria that limits the amount of plastic strain in the flange to 1%, and the plastic strain in the Gasket Outer Ring to 2%, are in accordance with ASME III Appendix XII. However the flange rotation criterion is based on EPRI's own manual, which limits the flange rotation to 0.015 inch (0.381 mm) across the active width of the gasket. This means that the flange is allowed to rotate such that the height difference between the inner and outer gasket radius is at most 0.015 inch. Using this criterion instead of the 1 degree limit stated in ASME enables a higher rotation angle for small flange joints with a smaller gasket width. Since the rotation of the flange is equal to 0.015 divided by the gasket width. The allowed flange rotation for larger flange joint are, however, closer to half a degree.

By using an Ansys FE-model, EPRI computed the maximum recommended bolt pre-tensions for 137 various ANSI B16.5 flanges. The majority of the evaluated flanges was of the type welded neck but a few socket welded flanges were analyzed as well. For the welded neck flanges all sizes from 1/2 inch to 24 inch were evaluated for all pressure classes from 150# to 2500#. The socket welded flange sizes from 1/2" to 2" were evaluated for the pressure classes 150#, 300#, 600#, and 1500#.

The typical FE-model of a welded neck flange is presented in Figure 1.3. The dimensions set as input for the flanges were consistent with the ASME B16.5 dimensions, except for the height of the raised face. The height of the raised face was instead set as 0.06 inch (1.524 mm) for the pressure classes 150 and 300, and 0.25 inch (6.35 mm) for the remaining pressure classes. The analysis procedure of evaluating the flanges involved increasing the bolt pre-tension in steps of 5 ksi (34.5 MPa), from 0 to 90 ksi (620 MPa), at room temperature. Thereafter the maximum allowable working pressures for temperatures between 100 to 650°F (38 to 343°C) were evaluated for bolt preload stresses at 30, 45, 60, 75, and 90 ksi. In Figure 1.2, the ANSI B16.5 allowable working pressures for various temperatures in Fahrenheit (100, 200, 300, 400, 500, 600, 650) and pressure classes (150#, 300#, 400#, 600#, 900#, 1500#, 2500#) are presented.



CLASS	H(in)	P100	P200	P300	P400	P500	P600	P650
150	0.06	285	260	230	200	170	140	125
300	0.06	740	675	655	635	600	550	535
400	0.25	990	900	875	845	800	730	715
600	0.25	1480	1350	1315	1270	1200	1095	1075
900	0.25	2220	2025	1970	1900	1795	1640	1610
1500	0.25	3705	3375	3280	3170	2995	2735	2685
2500	0.25	6170	5625	5470	5280	4990	4560	4475

Figure 1.2: The ANSI B16.5 allowable working pressures (ksi) for different temperatures (°F) and pressure classes [9]. The different heights of the raised face (inch) are stated in the second column of the table.

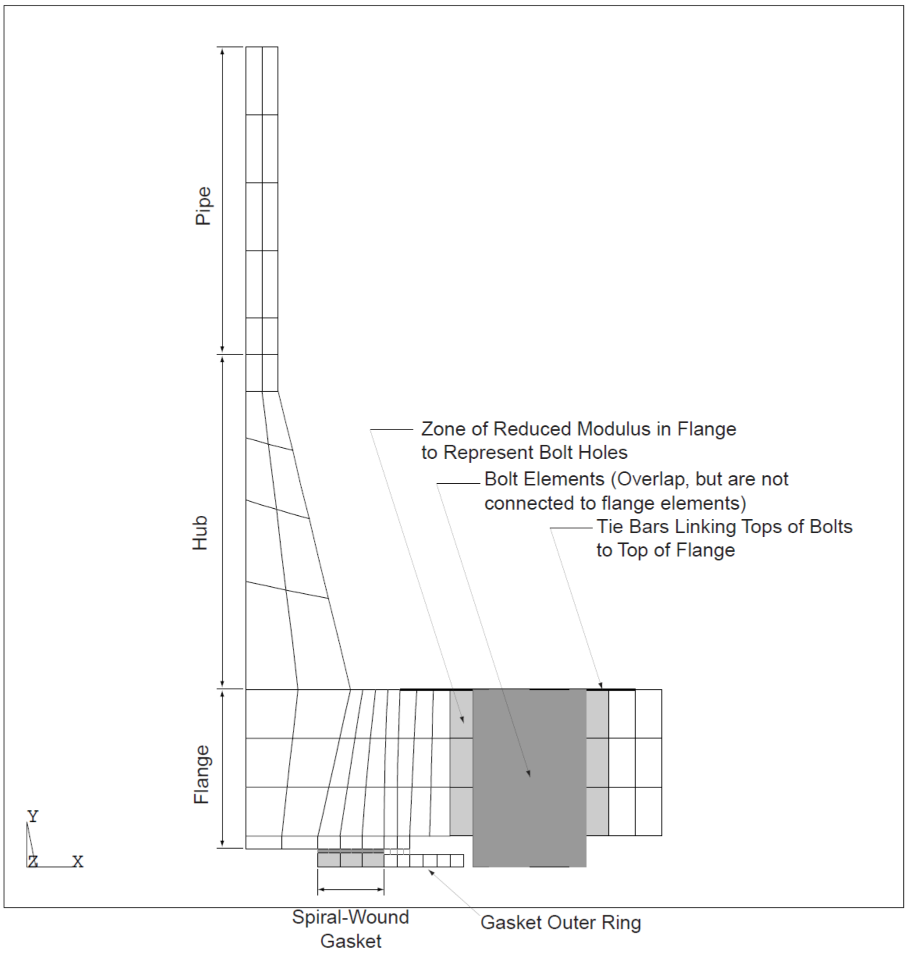

Pipe
Hub
Flange
Zone of Reduced Modulus in Flange to Represent Bolt Holes
Bolt Elements (Overlap, but are not connected to flange elements)
Tie Bars Linking Tops of Bolts to Top of Flange
Y
Z
X
Spiral-Wound Gasket
Gasket Outer Ring

Figure 1.3: The FE-model used by EPRI when analysing the ANSI B16.5 welded neck flanges [10].

As can be seen in Figure 1.3 the flange joint consists of a Pipe, Hub, Flange, Spiral-Wound Gasket, and Gasket Outer Ring. The bolts are modeled as a series of elastic beam elements, connected to the top of the flange by a series of rigid "tie-bar" elements connected to the upper surface of the flange.

The Pipe, Hub, Flange, and the Gasket Outer Ring are all assumed to consist of an elastic-plastic "low carbon steel", SA-105. In order to simulate the non-linear material properties of the Spiral-Wound Gasket, its modeled using two materials with different stiffness and closure gaps to the flange. This material is illustrated by the black curve in Figure 1.4. The softest of the two gasket materials reach 60 to 70% of full compression at a bolt pre-tension of 25 ksi (172 MPa). When the gasket exceeds 80% compression its stiffness increase significantly and when the bolt pre-tension approaches 45 ksi (310 MPa) the Gasket Outer Ring comes in full contact with the Raised Face. In the same manner, the curve of the unloading is initially very steep but once the Bolt Stress approaches 20 ksi (138 MPa) the slope decreases. Since only one flange is modeled, symmetry boundary conditions were added at the bottom line of the gasket to simulate an identical flange opposing it. Therefore the gasket was modeled with half its height.

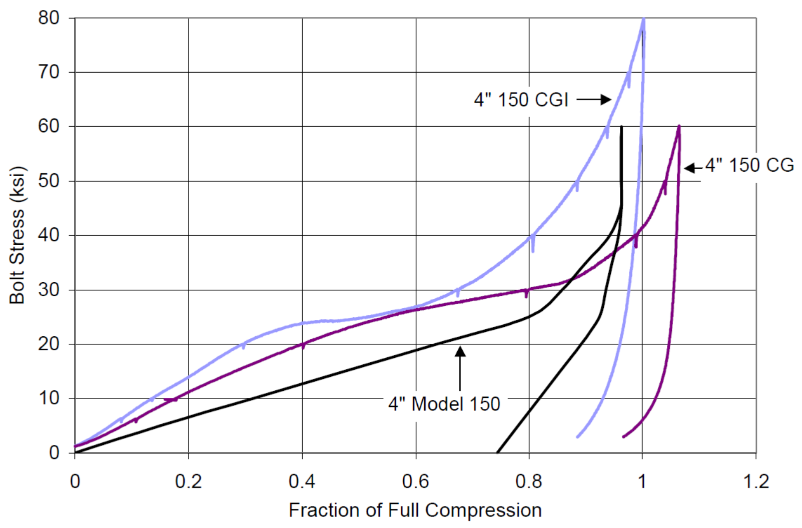
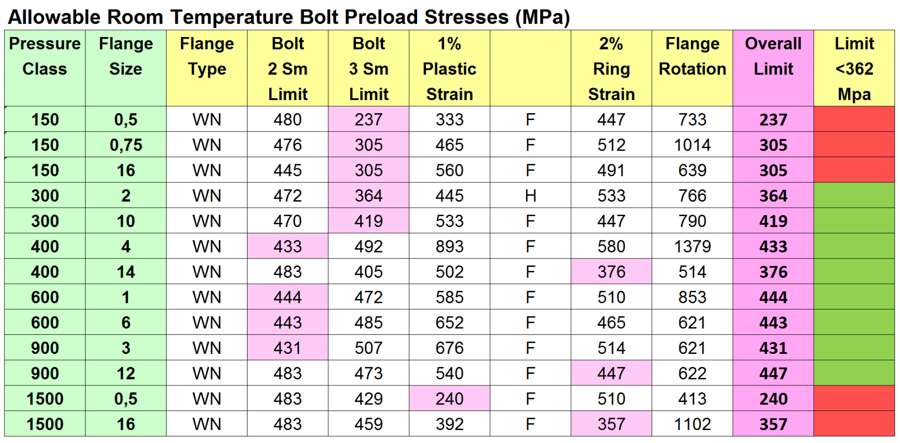


Figure 1.4: The material model used to simulate the Spiral-Wound Gasket (black). The light purple curve shows the experimental data obtained by EdF for a spiral-wound gasket with both inner and outer ring. The dark purple curve shows Edf's experimental data for a spiral-wound gasket without inner ring [11].

The complete result obtained by EPRI for all 137 flanges can be studied in their report, only the result of 13 selected flanges will be compared in this report. The result of these selected flanges are presented in Figure 1.5 below. EPRI wanted to determine a universal bolt pre-tension that suited all flange joints. Therefore, a general bolt pre-tension limit of 52.5 ksi (362 Mpa) was set for all flanges, and if the specific flange model didn't reach this value it was noted in the results by a red marker.


Allowable Room Temperature Bolt Preload Stresses (MPa)

Pressure Class	Flange Size	Flange Type	Bolt 2 Sm Limit	Bolt 3 Sm Limit	1% Plastic Strain		2% Ring Strain	Flange Rotation	Overall Limit	Limit <362 Mpa
150	0,5	WN	480	237	333	F	447	733	237	🟥
150	0,75	WN	476	305	465	F	512	1014	305	🟥
150	16	WN	445	305	560	F	491	639	305	🟥
300	2	WN	472	364	445	H	533	766	364	🟩
300	10	WN	470	419	533	F	447	790	419	🟩
400	4	WN	433	492	893	F	580	1379	433	🟩
400	14	WN	483	405	502	F	376	514	376	🟩
600	1	WN	444	472	585	F	510	853	444	🟩
600	6	WN	443	485	652	F	465	621	443	🟩
900	3	WN	431	507	676	F	514	621	431	🟩
900	12	WN	483	473	540	F	447	622	447	🟩
1500	0,5	WN	483	429	240	F	510	413	240	🟥
1500	16	WN	483	459	392	F	357	1102	357	🟥

Figure 1.5: The allowed bolt pre-tension obtained by EPRI for 13 selected flanges. The permissible bolt pre-tension values with respect to the certain criteria are presented as well as the overall limit for that specific flange.

Summer work

When this Master Thesis was started there had already been some previous work done regarding the optimization process [7]. There was an ANSYS Workbench model meant to work as a main model which could be fed with parameters from a Python script. Also some work had been done concerning the Herkules part where there were a project file as well as a few analysis files from which the Python script could fetch parameters. Results could also be presented in worksheets of the Herkules analysis files.

When initially trying to use the old ANSYS main model together with the Python script it was found that the model was not able to construct new flange geometries by receiving parameters from the Herkules analysis files through the Python script. When investigating the ANSYS model further it was clear that it would not be able to build up different geometries based on the number of bolts. This is why a new model was created from scratch.

When it comes to differences in the construction of the new main ANSYS Workbench models used compared to the older ones that existed when the Master Thesis was started, this will be explained in the *Methods* chapter, Section 2.

Replacing the parametrized FE-model led to the fact that the existing Python script had to be studied thoroughly in order for a new modified script to be created. The old ANSYS main model also included several APDL commands which also had to be modified. These control both parts of the pre-processing and post-processing of the ANSYS analysis.

1.6 Outline of the report

This report will give the reader insight into how the optimization method for determining bolt pre-tensions was developed. The chapter *Methods* explains the steps taken to end up with the final optimization method. In *Methods* the choices that led to the features of the FE-model are explained. The reader gets an overview of what caused problems and how these problems were solved. There are also descriptive explanations of the Python script and the intertwined software programs used for the final automated process. The analyzes carried out during the progression of the project are presented in the chapter *Results*. In this chapter there are also figures illustrating how the final result obtained with the automatized Python script will look. In the chapter, *Discussion*, the various modifications are presented that were attempted but could not be implemented. This is also where suggestions of future work are presented. And finally, the Master Thesis is summarized, in the last chapter *Conclusions*.

2
Methods

2.1 Prestudy

As an introduction to the thesis project other previous work was studied. Initially the main focus was on understanding the ASME-norm for flange joints in the nuclear industry and how the flange joint design criteria determined the allowable loads by analytical hand calculations. These analytical hand calculations were carried out in the computational software *Mathcad.*

Apart from the analytical design norm criteria, a report written by EPRI evaluated the allowable bolt pre-tensions using parametrized FE-models. The results obtained by EPRI were studied as well as the method used when developing the FE-model.

As an introduction to the FE-software ANSYS Workbench, which this project was to be carried out in, two courses in the FE-software was attended. The first course covered the basics of the program, such as sketching, various perks in ANSYS Mechanical, and some Mesh optimization. The second course covered how to deal with non-linear structures in ANSYS. Both of these courses worked as a stepping stone to the FE-modeling in ANSYS. The summer work previously carried out at Ringhals was not included in the prestudy per se, but it did however come to use as an introduction to the Python scripting and how to intertwine the software programs. Along with the summer work there was also a Herkules project file and several Herkules analysis files that were used in order to understand how the developed Excel add-in worked.

2.2 ANSYS Workbench model

The design of the FE-model was mainly targeted towards parametrization. Early on in the project the idea was that one model would be used to generate all other flange joints possible. However, restrictions in ANSYS Workbench complicated that objective. Therefore, 24 sub-models had to be created, 16 for flat gaskets and eight for spring wound gaskets. In the subsections below the different steps in the process of creating the general models will be explained.

2.2.1 Geometry

Prestudy to designing the general FE-model

Before creating a general template model, the already existing FE-model of the flange joint was studied. This FE-model was divided into only three parts, which can be seen in Figure 2.1, and had a hole pattern that entailed to that two of the bolts were cut in half when implementing symmetry to the model. However, since this FE-model was not able to generate flange joint geometries that varied in number of bolts, another FE-model was created from scratch. Still, a lot of the features from the old model were implemented in the new model as well. One of which was the Bolt Slice, shown in the figure below (9), which is created by an incision on the Bolt precisely below the Bolt Head. The Bolt Slice serves to separate the complex pre-tension applied to the Bolt-bodies from the contact that is present in between the upper surface of the Flange and the Bolt Heads. This is assumed to simplify the solution to the model.

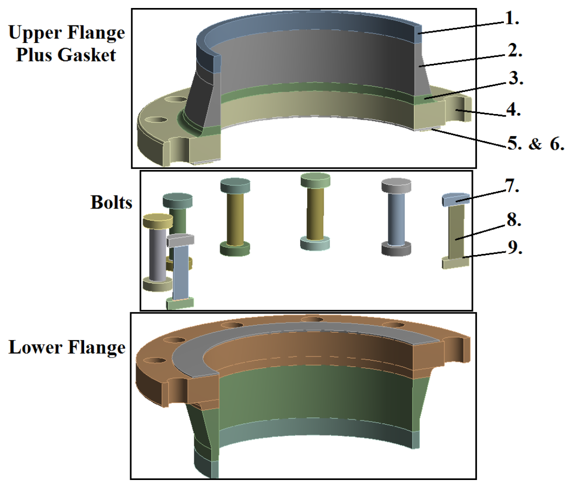


Figure 2.1: The implementation of the old ANSYS Workbench model, divided into three parts. The bodies in the model are; 1. Pipe, 2. Upper Hub, 3. Lower Hub, 4. Flange, 5. Raised Face, 6. Gasket, 7. Bolt Head, 8. Bolt, and 9. Bolt Slice.

Sensitivity analyzes

While designing the template FE-model several other optimizations and sensitivity analyzes were carried out in order to get a hint of which characterizations of the flange joint where the most important to focus on. In one of the first sensitivity analyzes, the hole pattern of the flange in relation to the symmetry plane was studied. The analysis was performed in order to investigate whether there was any difference between the original hole pattern and a pattern where the bolt was not sliced by the symmetry plane. The physical difference between the two hole patterns are illustrated in Figure 2.2. Intuitively, it seemed that hole pattern 1 was better since the bolts was placed such that they would experience more stress from the external moment than the bolts in pattern 2 would. However, when testing the two hole patterns in ANSYS Workbench it showed that the area between the bolts was far more sensitive to the external moment and would experience more plasticity. These results are presented in Figures 3.1a and 3.1b in the Results subsection, Sensitivity Analyzes. In order to capture this behaviour, hole pattern 2 was selected for the new FE-model.

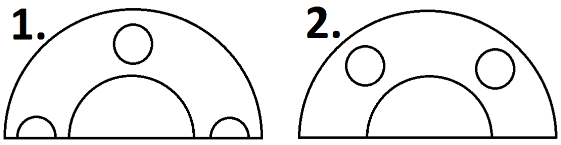


Figure 2.2: The two hole patterns evaluated. Hole pattern 1. was used in the old flange model while hole pattern 2. was implemented in the new model.

Another feature that was analysed due to uncertainties of its influence on the results was the length of the pipe connected to the flange joint. Since the external moment was to be implemented on the top surface of the pipe, it seemed that higher stresses would occur in the flange if the pipe connected to it was longer.

An investigation of what impact a pure external moment (PEM) would have on the flange was conducted. The influence of a PEM was then compared to the influence that an external moment arising from an external force (EFM) has. These loads will not impact the flange at the same time. The theory states that a pure external moment can be placed anywhere on a part in a certain plane, and the impact from the load will always be the same. An external moment arising from an external force on the other hand, will increase by the length of its own lever.

In Figure 2.3, Picture 1 and 2 show a flange joint with a pipe connected to its lower respective upper edge. Picture 3 and 4 illustrate the distribution of the PEM respective EFM load. Note that the illustration of the load distributions are simplified and only accounts for a homogeneous cross-sectional area. However, Picture 4 indicates that more plasticity will arise in the flange joint in Picture 2 compared to the one in Picture 1 when applying an EFM.

The conclusion was drawn that a pure moment would be most suitable to represent an unexpected load since this is not affected by the length of the pipe. Therefore, the length of the pipe was not changed for any flange model. Results from the investigation can be seen in Table 3.1.

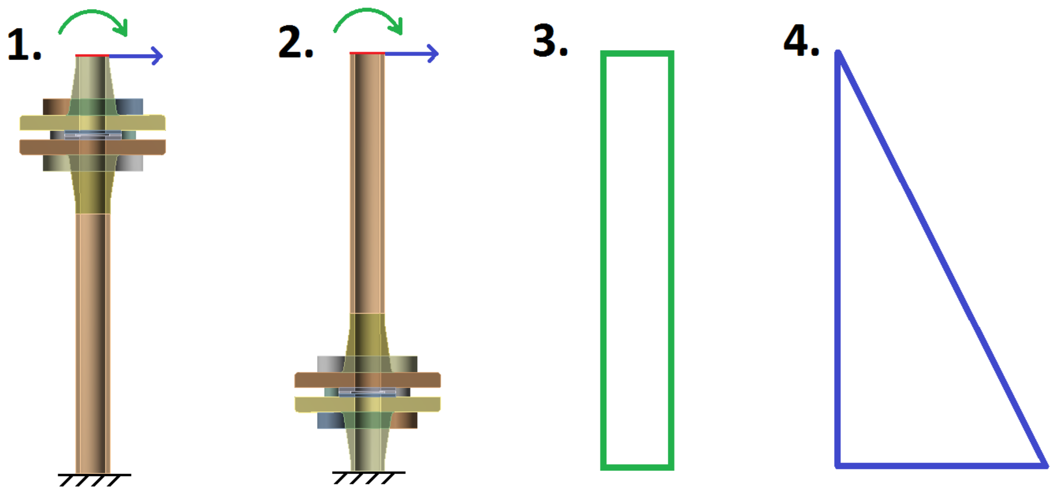


Figure 2.3: Pictures 1 and 2 represent the flanges in the investigation having a lower and an upper pipe part attached to it, respectively. Picture 3 and 4 shows a simplified theoretical distribution of a PEM and a horisontal EFM, respectively, imposed on the upper face of flange 1 and 2.

Circular sector of a flange

The two sensitivity analyzes previously described, changed the physical appearance of the FE-model. But in order to avoid other problems that arose during the progression of the project, many other changes was made as well. Although the major change of the model was how the construction process was constituted. One of the things that could not be modified in the initial FE-model was the number of bolts in the model. Adding more bolts resulted in that ANSYS workbench did not know how the added bodies were to be implemented in the rest of the model. This problem was initially attempted to be solved by assigning all bodies to selected parts by writing code in Visual Basic (the code that ANSYS Design modeler is written in). However there is a much simpler way of solving the problem. By creating a flange model that includes more parts than the other models that it will transform into will have. Once the bodies in that model are selected into the correct part they will stay within that part even if the number of parts change. That is, as long as no new parts are added to the other models. From this realisation the general template model was built.

Initially just a circular sector of the upper flange joint was created. This sector included all parts that would be present in the flange joint, such as a flange, the upper part of a bolt, a pipe connected to the top of the flange, and a gasket. Since all other flange models have to function after being generated from the template model, the assignment and the selections were defined by assigning named selections to the circular sector.

The flange was divided in to several sketches to make the parametrization process simpler. In order to make sure that the program does not dismiss the changes in the model, all constraints have been removed in-between the sketches such that dimensional parameters control the appearance of each sketch and its position in space. Once every sketch is modified according to the current flange dimension, the sketches are revolved a certain number of degrees depending on the number of bolts in the flange model.

The more bolts present in the current model, the smaller the revolved flange sector will be. This ratio is presented in Equation (2.1) where ϕ is the rotation angle and n is the number of bolts. When using revolve to create the circular sector of the upper flange, all sketches are merged into one body which then have to be sliced in order to assign various materials to the different bodies.

$$\phi = \frac{2\pi}{n} \tag{2.1}$$

Slicing the upper flange into several individual bodies was made by revolve as well, and with the same sketches that was previously used. After adding a radius to the transition between the hub and the upper flange, the correct appearance of the upper flange sector was established and a mirroring function was used to create a sector of the lower flange. Thereafter the gasket was merged together with its mirrored body. In the same way, the bolt was merged into one body.

The complete circular sector of the flange joint is presented in Figure 2.4. In the caption of this figure there is a nomenclature stating the denotations of the bodies included in each part. This nomenclature only covers the names of bodies in the upper parts of the flange joint, but these names are the same for the bodies in the lower parts as well, except that a specific body are described as the *Bottom* body of that kind. Whereas that specific body in the upper flange joint are described as the *Top* body of that kind.

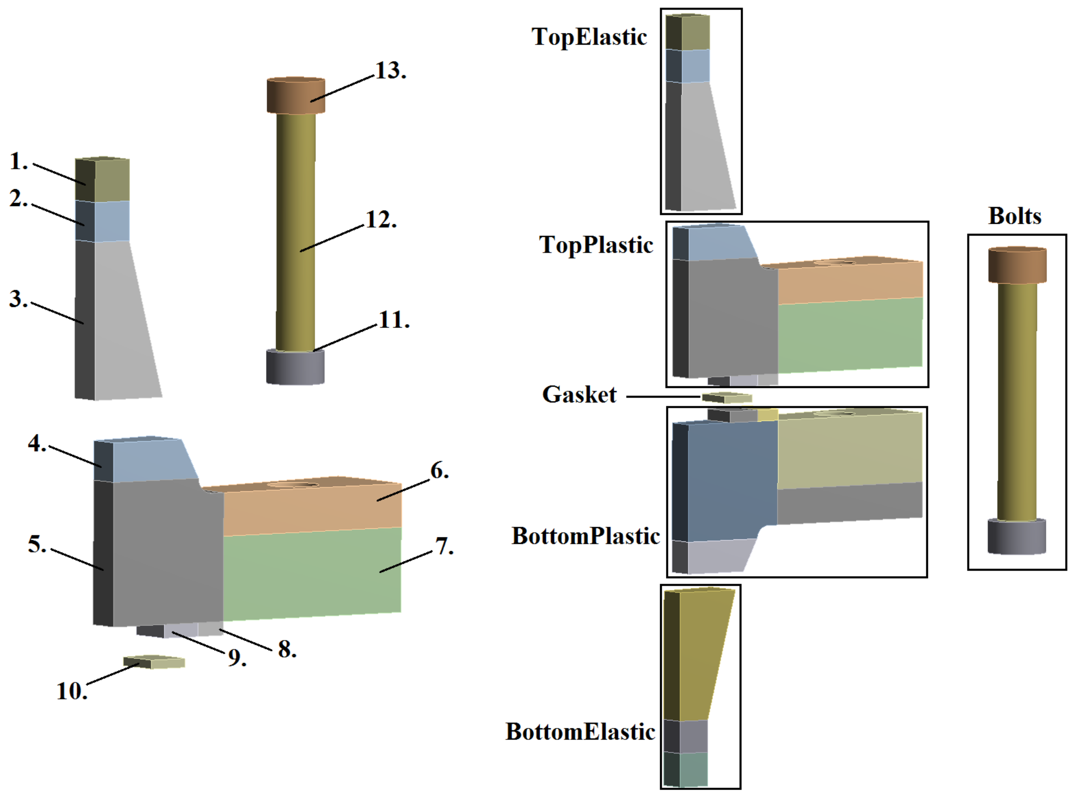


Figure 2.4: The implementation of the new general ANSYS Workbench model. The bodies in the model are; 1. Upper Pipe, 2. Lower Pipe, 3. Upper Hub, 4. Lower Hub, 5. Center Flange, 6. Upper Flange, 7. Lower Flange, 8. Outer Raised Face, 9. Center Raised Face, 10. Gasket, 11. Bolt Slice, 12. Bolt, and 13. Bolt Head. The names are also supplemented by a prefix (Top or Bottom).

From this flange joint sector several named selection were created which would be used to control every other flange model generated from it. Named selection were created to control the mesh, where to apply contacts, how to apply the loads, and where to obtain certain results.

Generating the complete flange and implementing parameters

After assigning the various entities to their named selection, the flange joint sector was duplicated using ANSYS Workbench pattern function such that it formed a 360 degrees complete flange joint. By creating named selections prior to using the pattern function, all entities related to the original selection of the named selection were implemented into the same group. This trick had a big influence on the automation process and made it easier to micromanage the generated flange models.

Lastly, the created bodies were divided into parts. Six different parts were created, *TopElastic*, *TopPlastic*, *Gasket*, *BotPlastic*, *BotElastic*, and *Bolts*, which are presented in Figure 2.4. The part *GRing*, which represents the outer ring of a gasket, is only present for the flange models with spiral wound gaskets. Although that the parts are defined as eg. *BotPlastic*, *BotElastic*, and so on, the names of the parts does not correctly reflect the material features applied to them. This is explained further in the section *Materials* below.

The reason that the FE-model is divided into so many different bodies is due to that another type of parametrized mesh was attempted in an earlier stage of the project. But after finalising the optimization process in the matter of increasing the convergence rate of the model this mesh parametrization was substituted by another one, this is explained further under the section concerning the Mesh of the model.

Even though the mesh parametrization was dismissed, the division of the bodies remained as they were since redesigning the current model would only take time from more important issues. Although the model contain many various bodies there are some bodies that have been disregarded. One of these bodies that have been disregarded is the inner ring included in the spiral wound gasket, Flexitallic. This part of the Flexitallic gasket was disregarded since the majority of the compressive stress will act on the outer ring of the gasket which will break before the inner ring does. Therefore, an assessment that the inner ring could be disregarded was made. The same reasoning was used when deciding to disregard the part of the Raised Face closest to the pressure-passage of the flange. This part of the Raised Face caused problems during the meshing process, which is described further under the Mesh section, and this is the main reason that the body was not implemented in the model.

In the beginning of creating a new flange model, not all geometrical parameters was featured. Therefore, the missing dimensions had to be searched for, and the ones that were not found had to be invented. After contacting the company *Nordic Flanges*, help was received concerning the radius of the transition between the hub and the upper flange. The radius of this transition was not stated in the ASME code but it could be obtained in SS-EN-1759-1 [13], which *Nordic Flanges* uses when manufacturing ANSI B16.5 flanges.

Another geometrical feature that was not described by any dimensions was the inclination of the hub. The height of the hub in relation to the total height of the flange determines the angle of this inclination. Since no dimension describing the height of the hub was found, a new dimension was conceived in order to describe the general appearance of the hub. Along with conceiving this dimension, the relations between other dimensions had to be developed as well. These dimensions, illustrated in Figure 2.5, are derived in the following section.

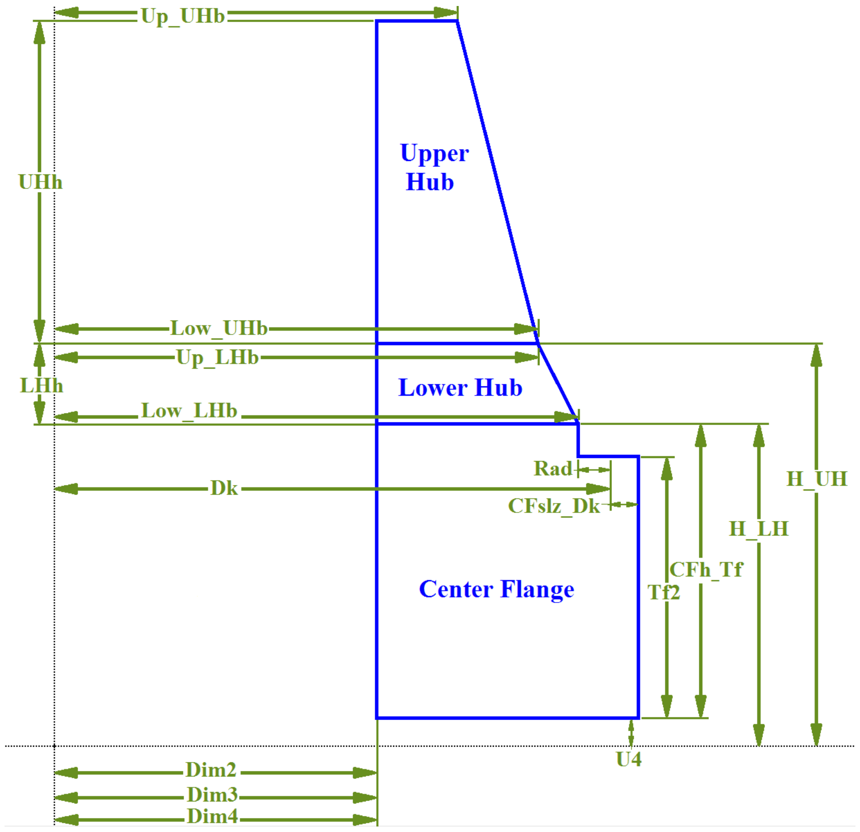


Figure 2.5: The dimensions assigned to the sketches controlling the Upper Hub, Lower Hub, and the Center Flange. Some of which were predefined.

Since the hub was to be sliced, two dimension had to be formulated to describe the height of the upper hub and the height of the lower hub. The height of the upper hub, *UHh*, was defined using four of the already existing dimensions; the total height of the flange H, the thickness of the flange T_f, the height of the raised face U, and the flange radius Rad.

$$UHh = \frac{H - (T_f + U + Rad)}{2} \tag{2.2}$$

$$LHh = \frac{UHh}{3} \tag{2.3}$$

Even though the hub was sliced and divided into two parts, the angle of the inclination should be the same for the two hub sections. Therefore the following equations were used to derive the dimensions controlling the upper and lower edges of the two hub sections.

$$LowUHb = UpLHb = \frac{(LowLHb \cdot UHh) + (UpUHb \cdot LHh)}{(LHh + UHh)} \tag{2.4}$$

2.2.2 Materials

Not only the geometrical parameters will vary from flange to flange but also the materials of the parts. Three different gasket materials will be evaluated and four different materials will be assigned to the flange and the pipe. But just one bolt material will be evaluated.

In order to obtain credible results from the FE-analyzes, the material setup had to be carefully chosen. Therefore a sensitivity analysis was carried out, comparing the stresses in the bolts after pre-tension, and the plasticity obtained in the radius. When applying the bolt pre-tension, the tensile stresses in the radius were influenced by large contact pressures from the surface between the bolt heads and the upper flange. To capture this behaviour, various modifications of material setups were evaluated and compared.

Four flanges were compared, that had the same geometry and was subjected to the exact same loads. But the setup of the plastic material varied between them. The selected bolt pre-tension acting on the flanges was set to 55494 N. This pre-tension was selected since it was supposed to give a plasticity close to 1% in the radius for this flange model, according to EPRI. The results obtained from this sensitivity analysis are presented under the subsection *Flange material* in the *Results*-chapter.

After evaluating the four different material setups, one of them was selected as the most realistic one with the most credible results. The selected setup only contained plastic material in the bodies *Lower Hub*, *Center Flange*, and *Lower Flange*, included in the part *TopPlastic*. The remaining bodies in *TopPlastic* were simulated with elastic material. All other parts were also assigned elastic material. However, what type of elastic material the parts were assigned varied.

Table 2.1: The various materials that can be selected for the different parts.

Part	Materials	Material Model
Bolts	SA-193 B7	Linear Elastic
Gasket	Grafex Novapress 815 Flexitallic	Non-Linear
Flange and Pipe	SA-105 SA-182-F304 SA-182-F316L SA-182-F316	Ideal Elastic Plastic

The *Bolts*-part was assigned elastic *SA-193, Grade B7*. The *Gasket* was assigned a material model called *Non linear gasket unloading* which material properties were changed depending on what gasket material it was meant to simulate. All other parts belonging to the flange joint were assigned either of the four flange materials; *SA-105*, *SA-182-F304*, *SA-182-F316*, or *SA-182-F316L*. The materials and the features of their respective material models are presented in Table 2.1.

All material models already existed except for the gasket materials *Novapress 815* and *Flexitallic*, and for the flange and pipe material *SA-182-F316*. In order to analyze the flange joints with the flange material *SA-182-F316*, the material model have to be created in ANSYS. However, since all flange material models are similar, creating a new material model only involves changing a few material properties.

When creating the gasket materials, the approach was to use the *Grafex* material model (shown in Figure 2.6) as a template. The material models for *Novapress 815* and *Flexitallic* were then rescaled by changing the pressure- and closure-values describing the point where the red and green curves meet. So that all gasket material models have the same appearance but the values on the axes vary. These values were obtained in the manufacturer's material data.

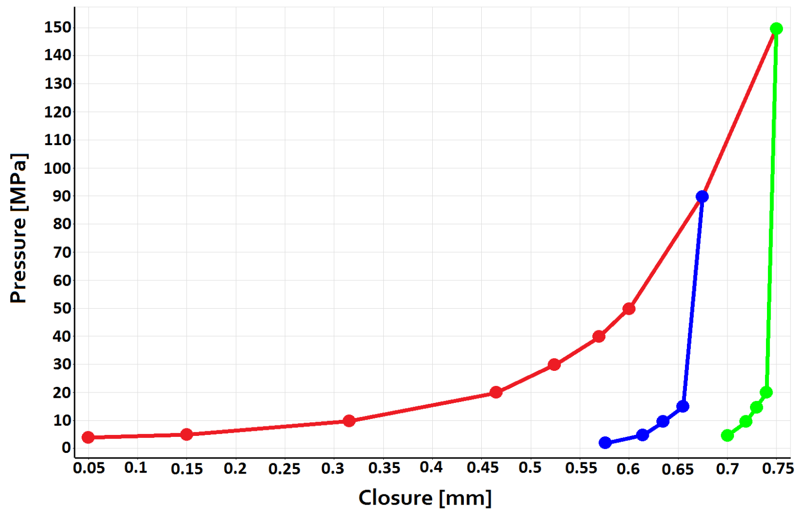


Figure 2.6: The Grafex material model implemented in ANSYS Workbench. The red curve represents compression, whereas the blue and green curve show the non-linear unloading from two different closure values.

2.2.3 Contacts

By adding contacts to the flange model, the difficulty of obtaining convergence is increased. But in order to make the model simpler to parametrize it has to be divided into parts, which have to be connected with contacts. As presented in Figure 2.7, there are three different contacts that were used in the model and these are; Bonded contact using Normal Lagrange (red), Bonded contact using Pure penalty (blue), and Rough contact using Pure penalty (green). The difference between pure penalty and normal Lagrange is that Pure penalty allows some penetration as long as it is not too big while normal Lagrange enforces the contact with no penetration by adding an extra degree of freedom (contact pressure) which is solved for explicitly. The third contact type, Rough contacts, allows the surfaces to separate but once they are in contact no sliding is allowed.

Between the bolt heads and the upper flange, a bonded contact using normal Lagrange was added. Since the bolt pre-tension function in ANSYS Workbench works in a special way that is explained further down in the report it is crucial that there is practically no penetration of the bolt heads into the flange. When using bonded contacts with normal Lagrange, no penetration will occur. This type of contact might affect the computational time but the obtained results will be more correct.

To minimize the computational time, the bonded contacts on the blue surfaces use pure penalty instead of normal Lagrange. However, in the contacts between the gasket and the flanges, and in the contact between the gasket outer ring and the top flange, additional penetration tolerances were added to ensure that valid results are obtained. The third contact type, Rough contacts, were added on the green surfaces in Figure 2.7 in order to ensure that the surfaces detect each other during contact.

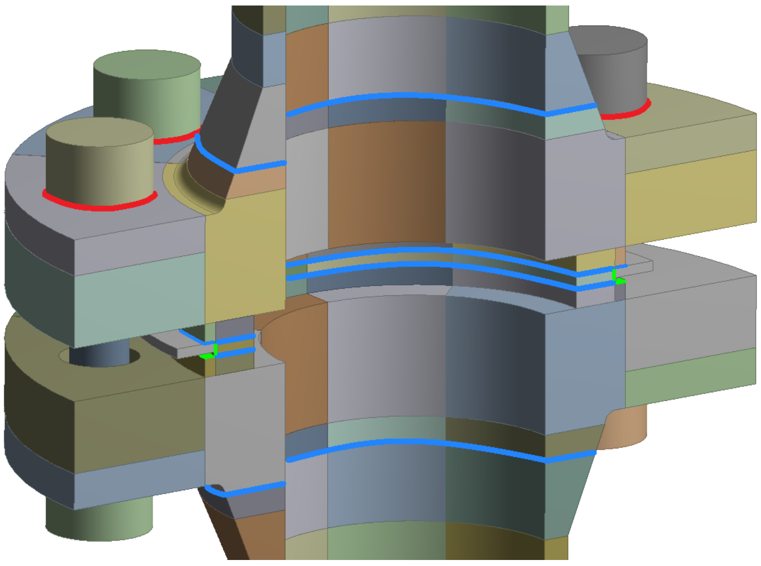


Figure 2.7: The three different contacts that were used in the FE-model; Bonded contact using Normal Lagrange (red), Bonded contact using Pure penalty (blue), and Rough contact using Pure penalty (green).

2.2.4 Mesh

First approach

Initially, the mesh was only intended to be refined in the areas close to the critical geometries, such as the radius and the raised face. It was due to this intention that the flange model was divided into all the different sections shown in Figure 2.4. A very detailed parametrized mesh was created for this design which demanded that the Center Flange was sliced into an individual part, separated from the other parts associated to the flange. However, designing the flange model in this manner entailed that more contacts had to be added between the center flange and its surrounding parts. But applying additional contacts to the flange resulted in that it was harder for the model to converge. Therefore the developed parametrization had to be discarded and replaced with another one.

Second approach

In order to keep the refined mesh grid close to the radius, the second mesh had to increase in number of elements. A large number of elements and nodes affects the solution time. If the mesh grid is too dense, there might be to many elements so that the computer will not be able to solve the model using the *Direct Solver*, which is explained further in the section *Analysis settings*. And this increases the solution time substantially.

With regard to these matters, it was decided that only the Top Flange was to obtain a refined mesh. Since the Upper Pipe of the Bottom Flange is constrained by boundary conditions, it will influence the stiffness of the entire Bottom Flange. The stiffness of the Bottom Flange might not increase very much but because of its constraints it is assumed that the Top Flange will deform more when applying the loads to the flange model. This also affected the decision of only refining the mesh of the Top Flange, which can be seen in Figure 2.8.

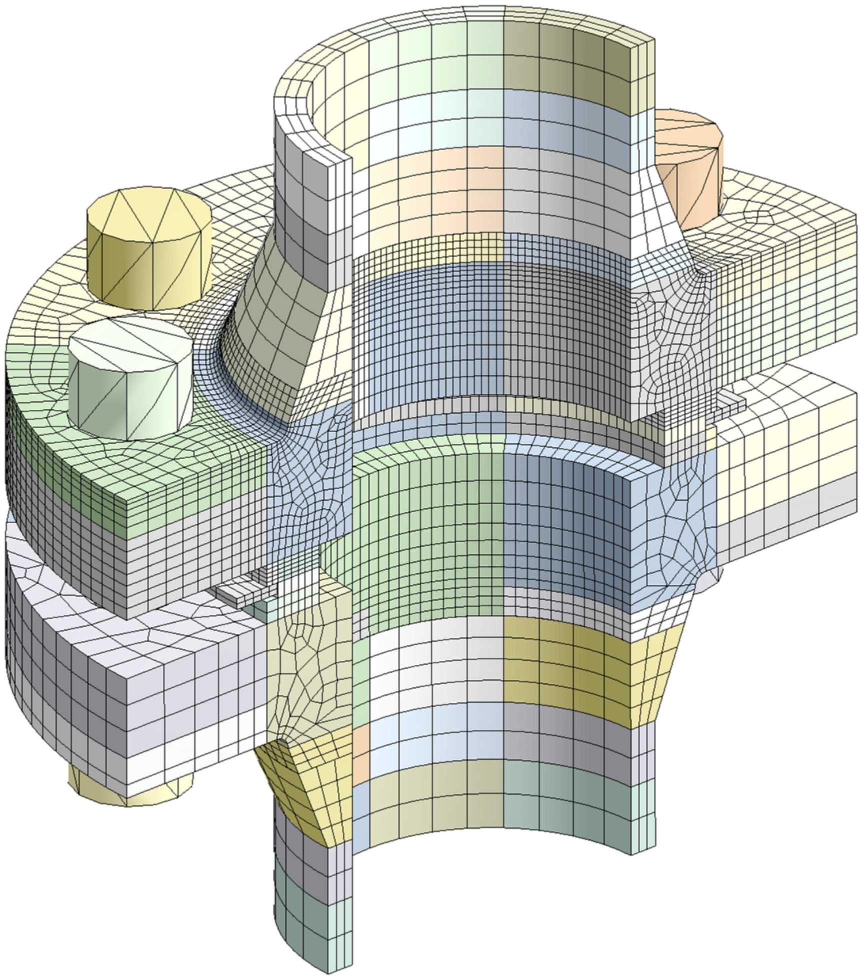


Figure 2.8: The mesh of the model is more coarse in the lower flange. By assuming that only the upper part of the flange joint have to be studied, the mesh can be more coarse in the lower parts. And reducing the number of elements decreases the computational time.

Mesh implementation

By using the mesh control function *Edge Sizing* together with the named selections the mesh was parametrized, and the number of elements on edges could be changed in great detail. An edge sizing could be assigned to a body in three different directions, vertical, horizontal, and through thickness. But since the mesh on bodies in the same part are dependent of each other some parameters controlled the mesh on several bodies.

Apart from controlling the number of elements with edge sizing, the mesh control Inflation was used. Inflation is a way of refining the mesh close to a specific geometry and was used close to the radius as seen in Figure 2.9. By refining the mesh in this area, a more accurate result can be obtained when controlling the plasticity on the radius. An inflation was also added to the top surface of the upper flange to ensure that the contact pressure from the bolts didn't affect the stress in the radius.

In order to match the mesh grid of the bolts to the bolt pre-tension function in ANSYS, a Mesh Control named Face Sizing was added to the cylindrical surface of the bolts. By using Face Sizing the element size on that surface can be specified. For the bolt pre-tension to be implemented correctly, it is required that nodes are generated in the mid plane of the bolts. Therefore, the element size for the Face Sizing had to be equal to the height of the bolt divided by an even number. After some testing, the element size on the bolt surface was determined as; the height of the bolt divided by twelve.

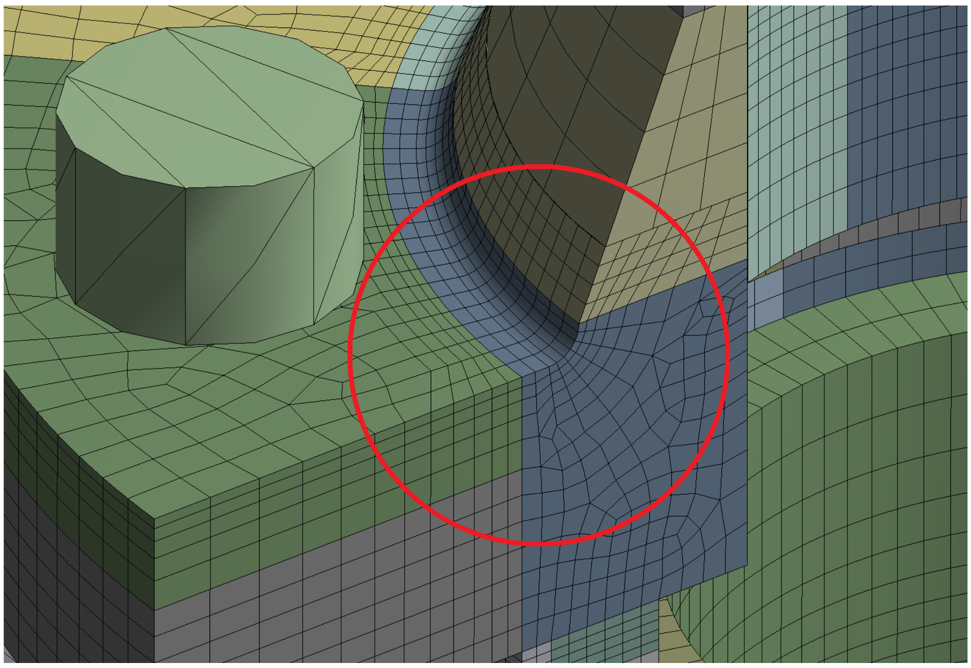


Figure 2.9: The mesh is refined closer to the radius using the Mesh Control Inflation since the plasticity in this area have to be carefully observed.

2.2.5 Loads and boundary conditions

Chronological load application

Five different loads were added to the model, bolt pre-tension, internal pressure, temperature load, external force, and external moment. The chronological order in which these five loads are applied is presented in Table 2.2. The external force is added to simulate the stresses that occur in the axial direction of the pipe due to the internal pressure. Whereas the external moment is supposed to simulate the load acting on the flange joint due to movement of the connecting pipes.

Table 2.2: The loads and at what time they are fully applied.

Load	Applied at Time
Bolt Pre-Tension	1
Pressure	2
Temperature	2
External Force	2
External Moment	3

Bolt pre-tension

When the pre-tensions of all the bolts were to be implemented in the model, the idea was to apply them as one unit. However, in order to apply this load on all bolts simultaneously, a sensitivity analysis had to be carried out. In this sensitivity analysis, the bolt stress intensity and the deformation of the bolts were compared for four different implementations of the load which are illustrated in Figure 2.10. Implementation A implies that all bolts are selected and unified into one bolt pre-tension that is defined in a coordinate system with origin set in the global origin. Case B is implemented in the same way except that the selected coordinate system, in which the bolt pre-tension is defined, has its origin in the bolts center of mass. Both of these implementations are compared to each other but also to the two other, C and D, which are constituted by individual bolt pre-tensions. In C, all bolt pre-tensions are defined in the origin of the global coordinate system. In D, each bolt pre-tension has its own coordinate system with an origin in the mass center of its respective bolt.

In the original model, the pre-tension was applied for each bolt using a complex APDL-command, but the results obtained through the sensitivity analysis (which are presented in the *Result* section *Sensitivity analyzes*) enabled the possibility of applying the pre-tension to all bolts as in case A. Implementing the bolt pre-tensions in this manner had a large impact on the progression of the project.

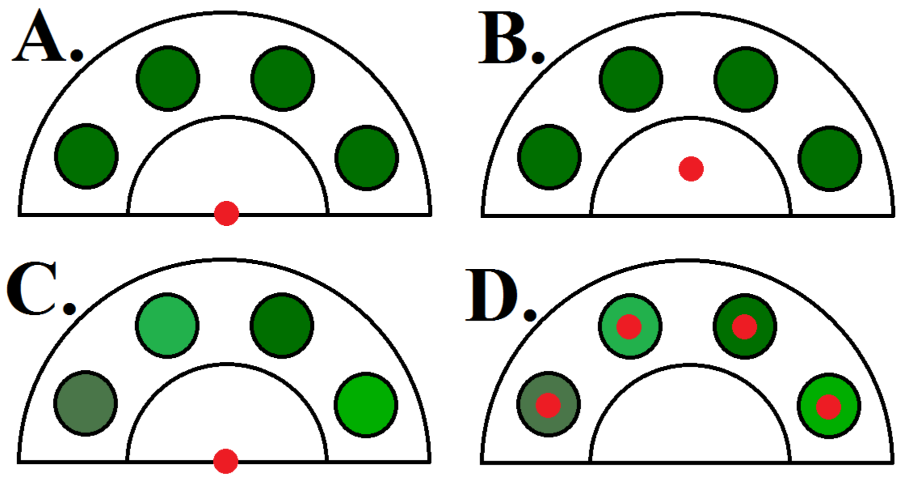


Figure 2.10: The four different implementations of the bolt pre-tension that were examined in the sensitivity analysis. The red dot represents the origin of the co-ordinate system in which the bolt pre-tension(s) was/were defined. The different green colours represent the various bolt pre-tensions that were used. For A and B only one green colour is used which implies that only one bolt pre-tension was used for all bolts. In C and D several different green colours marks the bolts since four individual bolt pre-tensions were used.

Pressure

When applying the internal pressure, a few problems concerning the gasket occurred. In the original model the internal pressure was applied over the entire inner surface of the flange joint, which intuitively seemed to be the correct way of applying it. However, the original model was created in an older version of ANSYS, and inserting the internal pressure in this manner in ANSYS 16.0 made the gasket elements deform heavily. Therefore, the internal pressure was applied on all surfaces except on the inner surface of the gasket, since the gasket-elements only have one degree of freedom in compression.

External force and moment

The external force was added on the top surface of the upper pipe and were added as a complement to the internal pressure. Since only a small part of the pipe is included in the model and not the entire piping system, a force is added on its top surface to account for the axial load that the internal pressure induce. This surface is also where the external moment is applied.

Temperature

The temperature load acting on the flange joint was applied on all bodies and the ascension of the temperature was assumed to be equal everywhere in the model. However assuming no thermal transients exist is a big simplification of the temperature load.

Parametrization

In order to update the value of the load parameters, APDL-commands were added for each of them. In these commands input arguments were implemented such that the loads that were defined in a table could be parametrized as well. Normally, in ANSYS Workbench, loads that vary over time can not be controlled by appointing a parameter to it. Therefore, it has to be controlled by a command and the different arguments selected in it. All these commands are written such that they all work in the same way. First, the values of the load are changed. This is done by assigning new values to the elements in a load variable matrix created by the underlying functions in ANSYS Workbench. After changing the load values in the matrix, the area of where the load will be applied is specified ones more such that the old values are replaced by the new ones in the load function as well. This is what happens in all load commands except for some more advanced allocations that had to be implemented for the external force. In the load command for the external force the inputted values of the load have to be divided by the number of nodes on the surface that it will be distributed upon. The commands controlling the various loads are presented and explained in the subsection *APDL commands for loads*.

2.2.6 Analysis settings

In order to improve the computational time of the flange model, changes to the analysis settings can be made. However, these settings can also be tweaked to avoid convergence issues. For the general template model both aspects have been considered. Selecting *Direct solver* as *Solver Type* allows all iterations to be solved in the RAM memory which makes the analysis more time efficient. By modifying the number of substeps that are used when solving the various load cases, divergence issues can be avoided. The number of substeps entered in the *Analys Settings* determines in how many substeps that the load will be applied. If the parameter for initial substeps is set to 20 this means that during the first load substep, 5% of the total load will be applied. By setting the parameter for the initial number of substeps to 20, the minimum number of substeps to 20, and the maximum number of substeps to 200 all load steps are expected to converge. It is very important that the initial substep is not too large, especially if the current load that is to be applied is large. And if the initial substep is too large, it is of even greater importance that the value of maximum substeps is high such that it allows smaller load steps during the solution.

2.3 Model validation

To ensure that the results obtained by the parametrized flange model are reasonable, 13 flanges evaluated in the EPRI report were selected and analyzed. Three flange sizes were selected from the lowest pressure class, and from every other pressure class two flange sizes were selected. In the EPRI report, the maximum allowable bolt pre-tensions obtained with respect to the various criteria were stated. And the criteria concerning the bolt stresses and the 1% plasticity in the flange were selected as the ones to compare.

In order to compare EPRI's results to the ones obtained by the ANSYS Workbench model, several changes were made in the Workbench model such that it resembled the EPRI model as much as possible. The material of the flange in the Workbench model was changed to *SA-105*, the same material that EPRI used. However not all parts of the flange were assigned plastic material. The plastic material in the model was distributed in the same way as earlier mentioned. In the Center Flange, Lower Flange, and Lower Hub, in the Top Flange, *SA-105* with linear plastic hardening was implemented. The plastic hardening was added to the material model in order to mimic the material model used by EPRI. All other parts were modelled with elastic *SA-105*, except for the bolts which were assigned *SA-193, Grade B7*, which was the bolt material used by EPRI.

Apart from assigning other materials to the FE-model, the height of the Raised Face was also modified such that it matched EPRI's dimensions. Another dimension that separated EPRI's model from the Workbench model was the thickness of the connected pipe. However, this dimension was not changed in the Workbench model since the analysis of the pipe is carried out using another program called *Pipestress.*

The mesh of the EPRI model, shown in Figure 1.3, are much more coarse than the mesh generated in the Workbench model. A much more coarse mesh increases the stiffness of the model, but despite the differences in mesh grid size the refined mesh of the Workbench model was kept.

After making these changes to the model, the results obtained by EPRI were used in order to calculate what bolt pre-tensions were to be applied for the various flange models. Since the results concerning the allowable bolt pre-tensions were given as a pressure (R_P) they had to be transformed into a force (R_F) in order to insert them into the Workbench function governing the bolt pre-tension. This transformation of the results are explained in Equation 2.5, where r_B is the radius of the bolt and n_B is the number of bolts present in the various flange models.

$$R_F = R_P \cdot r_B^2 \cdot \pi \cdot n_B \tag{2.5}$$

Once the value of the maximum allowable bolt pre-tension determined by EPRI has been transformed into a force, it was inserted among the parameter list of that specific flange model. During the analysis, the pre-tension of the bolts was implemented at room temperature first. After time step one, when the full pre-tension have been applied, the ambient temperature, internal pressure, and external force were increased until the end of load step two. The plasticity obtained in the radius after load step two was then stored and compared to the plasticity value obtained by EPRI, which was 1 % for every model. The comparison between these results are presented in Figure 3.4 included under the section *EPRI* in the *Results* section.

Initially a Herkules project file was created for the EPRI comparison, however since the applied bolt pre-tensions gave such high plasticity for certain flange models errors occurred during the analyzes. Effort was put into making ANSYS Workbench proceed to the next flange model if an error occurred in the current analysis. The APDL-command `/NCNV,2` was recommended as a solution to the problem but regardless of where this command was inserted in Workbench the problem still remained. So in order to get around the problem the parameters for each respective flange model were inserted as separate design points into ANSYS Workbench's parameter set. Although, to ensure that the model settings are not changed the flange model design points have to be inserted such that the number of bolts are placed in descending order. If the smallest model is inserted in the first design point the additional parts created in the following design points will be assigned structural steel material instead of the material that the part was intended to have. As long as the initial design point contains more or as many parts as the following design points all material assignments earlier made will be intact.

A comparison between the stresses obtained in the bolts was meant to be carried out as well. However for this comparison, only elastic materials were to be used. This comparison was not performed since the plasticity comparison took much longer time than what was expected. Several attempts were required before any results could be obtained, both with Python scripts and by using design points in ANSYS Workbench. However, due to convergence issues and strange results, more effort was put into the plastic comparison.

2.4 Optimization method

2.4.1 Load validation criteria

The final criteria chosen to validate the loads applied to the flange joint during the automated process are presented in Table 2.3 below. In order to show what the various criteria represent, they are also illustrated in Figure 2.11. In accordance with the EPRI and ASME-standards presented in the *Introduction* these criteria set sufficient limits. The maximum *Bolt membrane stress* that is allowed for the material *SA-193, Grade B7* is equal to 482 MPa for the bolt pre-tension and 380 MPa at operating temperature. The maximum *Bolt membrane + bending stress* that is allowed is equal to 570 MPa at operating temperature and 743 MPa when pre-tensioning the bolts.

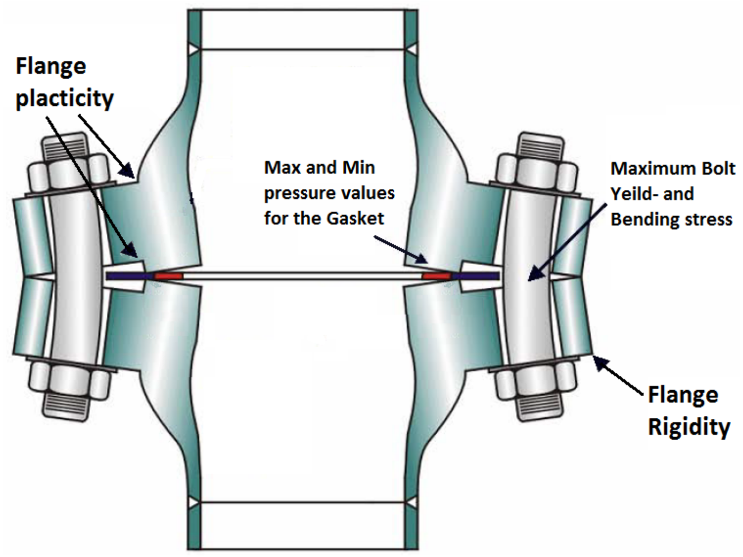

Flange placticity
Max and Min pressure values for the Gasket
Maximum Bolt Yeild- and Bending stress
Flange Rigidity

Figure 2.11: The criteria limiting the flange joint components [14].

Table 2.3: Final ASME-criteria which will be considered in the automated optimization process. σ_y represents the bolt material's yield strength.

Parameter	Criterion
Flange Plasticity	1%
Flange Rigidity	1 degree
Bolt Memb Stress, $\sigma_{M,max}$	$\leq\ 2 \cdot S_m$ ($S_m \approx \frac{1}{3}\sigma_y$ acc. to ASME-norms)
Bolt Memb + Bend Stress, $\sigma_{M+B,max}$	$\leq\ 3 \cdot S_m$
Grafex Min Stress	10-25 MPa (depending on pressure level)
Grafex Max Stress	100-120 MPa (depending on temperature)
Novapress 815 Min Stress	10-22 MPa (depending on pressure level)
Novapress 815 Max Stress	15-330 MPa (depending on temperature)
Flexitallic Min Stress	69 MPa
Flexitallic Max Stress	220-300 MPa (depending on temperarure)

2.4.2 Bolt pre-tension optimization

The bolt pre-tension optimization procedure is done in order to get the highest possible allowed bolt pre-tension without violating any of the criteria stated in Table 2.3. The idea is to run an analysis with bolt pre-tension and the other loads at their specific time steps, which can be seen in Table 2.2, and then evaluate whether any criteria was violated. If not, then the analysis will be run again where the bolt pre-tension has been set to a higher value and the rest of the loads will have the same magnitude as the previous run. By having this iterative process, the maximum allowable bolt pre-tension will eventually be evaluated and the violated criteria will be acknowledged.

The decision was made that the multiplier to decide of what increase the bolt pre-tension would receive was the amount of plasticity that had arisen at the radius of the flange during a previous run. The multipliers were set to what can be seen in Table 2.4 below.

Table 2.4: The bolt pre-tension multiplier depending on plasticity in radius.

Plasticity interval	Bolt pre-tension multiplier
Plasticity $<$ 0.5 %	1.2
0.5 $\leq$ Plasticity $<$ 0.75 %	1.1
0.75 $\leq$ Plasticity $<$ 0.9 %	1.05
Plasticity $\geq$ 0.9 %	1.01

No study has been performed concerning the multipliers suitability. This could perhaps be tweaked advantageously, but in several test runs the bolt pre-tension seemed to be stepped up fairly well and were not overshooting. There is also a compromize between having very fine bolt pre-tension increase steps versus the total analysis time. This since the model is solved for two complete time steps every time the new bolt pre-tension is induced and a new run is initiated.

2.4.3 Maximum allowed external moment

The maximum allowed external moment is evaluated after an optimal bolt pre-tension has been established for a certain flange set. There are four different cases with different bolt pre-tension magnitudes for which the maximum external moment is sought for. These are factors of 0.80, 0.85, 0.90 and 0.95 which are multiplied to the bolt pre-tension.

The four different cases make it possible to investigate to what extent the maximum allowed external moment would be affected, by having the flange pre-tensioned with a certain lower magnitude. There could very well be that a certain bolt pre-tension factor is optimal for one flange set and not for another set.

The external moment is applied in a third time step when all previously mentioned loads have already been applied. Now, a similar stepping up process is used as for the one stepping up the bolt pre-tension. The bolt pre-tension is fixed while an increasing external moment is used. The step-up multipliers is arranged the same way and according to the same criteria as the one in Table 2.4.

2.4.4 Analysis automatization process

The purpose of the automatization of FEM-analyzes is to be able to use a few parametrized flange models constructed in ANSYS Workbench and use these together with other software products to produce FEM-analyzes of basically every possible flange set variant that Ringhals uses today. The FEM-analyzes are not only meant to be run in an automatic fashion, but unique results, in the shape of both figures and data, are to be extracted from each individual run as well. These results will then be presented and evaluated in the respective Herkules analysis file, which is explained more below. For every FE-model there is a Herkules project file containing analysis files for every flange size in each pressure class. These individual project files were created such that the analysis files would be easier to handle, instead of placing all 1512 analysis files in one project file. The reason this many analysis files had to be created is that the material assigned to the various parts could not be parametrized.

Intertwined software

This section is meant to generally describe how the different software products are linked to each other and their different purposes. The flow of the automated process is illustrated in Figure 2.12. The process of one automated analysis will be gone through step by step. More detailed reporting of what has been created within Herkules, Python and ANSYS APDL can be viewed in Appendices A, B, and C.

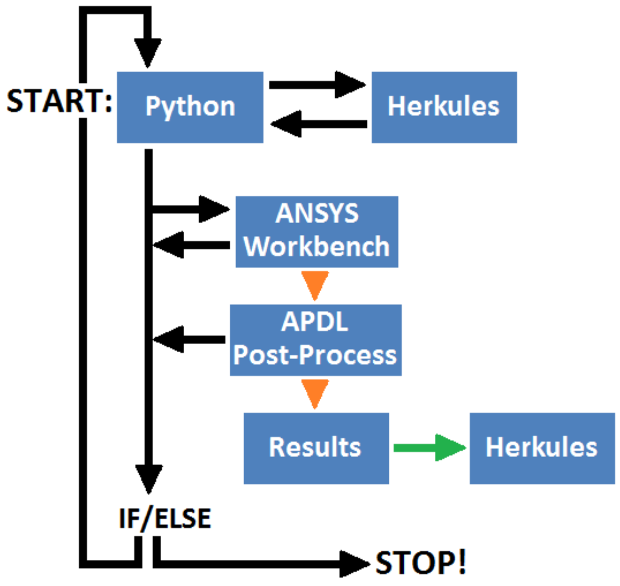


Figure 2.12: A schematic figure of the intertwining of the different software.

First of all there is a python script created that serves as a base for most operations, which is run inside ANSYS Workbench. From now on the word *script* will refer to the *python-script* during this section.

When the script is started the first thing that happens is that a chosen ANSYS main file is opened in ANSYS Workbench. Next, the script opens the Herkules project file where all parameters of the different flange models are stated, see Figure in Appendix A.4. The script then reads the name of the analysis file which is stated in the third column of the Herkules project file (blue text in Figure A.4). The number of different flange sets to be run is predefined at the beginning of the script.

The script then opens the Herkules analysis file, created for a specific flange set, and starts loading all parameters, see Figure A.3. If two or more flange sets are set to be analyzed, then the script will fetch parameters for all flange sets. After this is done, the same parameters are inserted into the parameter library in ANSYS Workbench. Special ANSYS syntax for python is used to perform these operations. The basic flange model in ANSYS is then saved as a new model with a unique name that corresponds to the actual flange set.

A first analysis is then being run with a model of a flange having a certain set of parameters for both dimensions, loads and various other data necessary. This is basically a normal solving session performed in ANSYS Workbench, but with additional modifications both concerning the pre-processing and post-processing by the use of APDL commands that originates from the ANSYS Classic syntax. The pre-processing includes APDL commands which makes it possible to use parameterized loads.

The other modification of the ANSYS Workbench model concerns the post-processing. The purpose of this is to generate figure- and text-files of the analysis results, where the text files include result data from the analysis. The figures and result data are later used along with the results presentation inside the Herkules analysis files. Also, some of the result data inside the text-files are fetched by the Python script to check it against the different criteria. Since these functionalities do not exist in ANSYS Workbench, a special APDL command has been created to satisfy this need.

Now the basic procedure of one flange analysis run with a python script is at its end. However, Herkules can now load these figures and result data into the Herkules analysis files corresponding to the same flange set, and then present it. Example figures of what the presentation looks like can be seen in Figures A.5, A.6 and A.7 in Appendix A. This is where the the end result of every analysis is being presented.

3
Results

3.1 Sensitivity analyzes

The results from the main influencing sensitivity analyzes that were performed in order to create an adequate FE-model are presented in this section. These results will not only give understanding to the choices made for the FE-model, but will also give insight into certain operations in ANSYS Workbench.

3.1.1 Pipe length impact

When the investigation was conducted, five different models were used. One flange model with no pipes connected at all with vertical length of 97.5 mm, two flange models having a lower pipe part of lengths 80 mm and 160 mm and two flange models having an upper pipe part of lengths 80 mm and 160 mm. A pure external moment (PEM) and an external moment arising from a horizontal external force (EFM) was applied to the upper surfaces of the flange models 1 and 2 in Figure 2.3. The PEM magnitude was 200 Nm and the EFM magnitude was 600 N. These magnitudes were set to show the impact from the loads on the flanges and could have been set to something different. Pictures 3 and 4 in Figure 2.3 shows the theoretical distribution along the length of the flange pipe set when having a fixed support at the bottom surface.

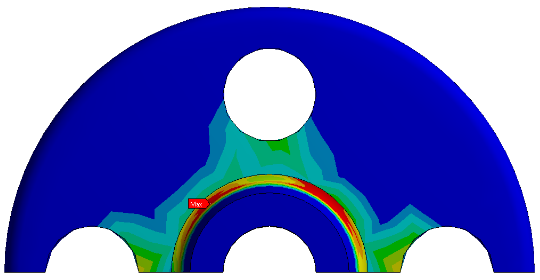
The maximum plasticity in the flange is then checked for the five cases, which is higher for a more extensive load impact, and the results can be seen in Table 3.1. The conclusion is that the results correlate well to the theory.

Table 3.1: Plasticity results in models with different pipe lengths.

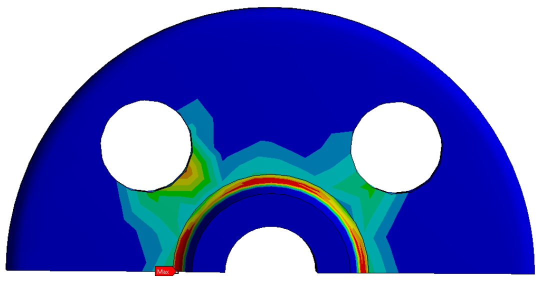
Lengths Of Pipes [mm]	EFM 600 [N]	PEM 200 [Nm]
Upper = 0, Lower = 0	0,04%	1.1%
Upper = 0, Lower = 80	0,04%	1.1%
Upper = 0, Lower = 160	0,04%	1.1%
Upper = 80, Lower = 0	0,11%	1.1%
Upper = 160, Lower = 0	0,21%	1.1%

3.1.2 Hole pattern

The results for the investigation concerning the suitability of the hole pattern show that the hole pattern in Figure 3.1b is more appropriate to use than the hole pattern in Figure 3.1a. The red in the figures show where the most plasticity has arisen during a bolt pre-tension. The applied pure external moment will be around a vertical axis going through the middle of the flange (when watching the flange from the perspective of the figures in this section). This means that it would be beneficial if the rotation of the external moment was located in the same plane as the maximum plasticity occurs in order to get the worst case scenario of the external moment. The results show that this will be the case when having changed into the new hole pattern.



(a) The plasticity pattern from bolt pre-tension with old hole pattern.



(b) The plasticity pattern from bolt pre-tension with new hole pattern.

Figure 3.1: The old and new hole pattern of the half flange model.

3.1.3 Bolt pre-tension

As previously mentioned in the method, a sensitivity analysis was carried out in order to investigate whether the pre-tension of all bolts could be gathered into one ANSYS Workbench function. Ultimately, this implementation was found to be valid which, along with the decision of changing the hole pattern, simplified the parametrization significantly. This sensitivity analysis also brought light to how the function *Bolt Pre-Tension* in ANSYS Workbench can be used.

As seen in Figure 3.2 below, there is no differences at all between the various implementations of the bolt pre-tension. The results of the adjustment reaction and the preload reaction of the bolt pre-tension were obtained adding a *probe* for the bolt pre-tension into the solution-part of ANSYS Mechanical.

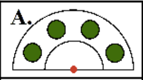
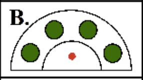
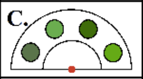
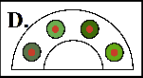
		Adjustment Reaction	Preload Reaction	
One Global Bolt Pre-Tension	Created from Global Coordinate System in Global Origo	0,02193 mm	31,1 MPa	 A.
One Global Bolt Pre-Tension	Created from Global Coordinate System in Mass Center of Model	0,02193 mm	31,1 MPa	 B.
Several Individual Bolt Pre-Tensions	Created from Global Coordinate System in Global Origo	0,02193 mm	31,1 MPa	 C.
Several Individual Bolt Pre-Tensions	Created from Individual Coordinate Systems in Mass Center of each Bolt	0,02193 mm	31,1 MPa	 D.

Figure 3.2: The bolt deformations and stresses obtained when comparing the various ways of implementing the bolt pre-tension.

3.1.4 Flange material

In Figure 3.3, the plasticity obtained in the radius for four different material setups of the smallest flange (150# 1/2”) are presented. The purple bodies in the figure are the ones assigned with plastic material. During the analysis, a bolt pre-tension of 55494 N was applied, since the plasticity in the radius for this bolt pre-tension will be approximately 1% according to EPRI’s results.

After evaluating the four material setups, the fourth setup was selected to be implemented into the FE-model. The fourth material setup gave stresses in the radius close to 1%, obtained values just slightly below the bolt pre-tension when examined with a probe, and unlike the first material setup, the plasticity in the radius was not affected by the contact pressure acting on the Upper Flange.

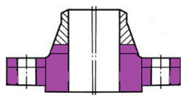
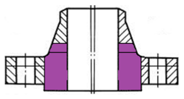
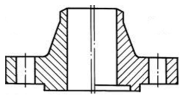
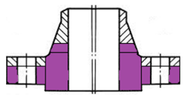
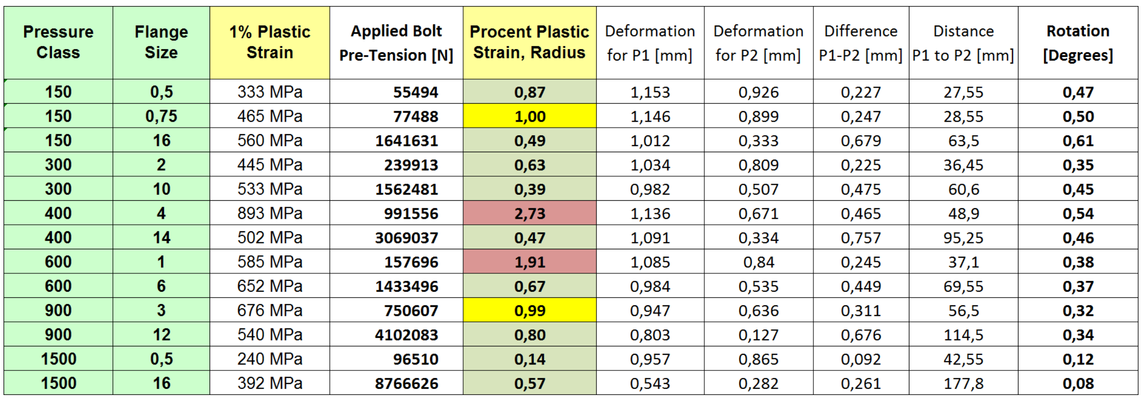
	Bolt Tool		Gasket Normal Pressure		Plasticity at various times				
	Bolt Pre-Tension [mm]	Bolt Pre-Tension [N]	Upper Face Gasket Pressure [Mpa]	Lower Face Gasket Pressure [Mpa]	Geometry	Plasticity at 0,5 s	Plasticity at 0,7 s	Plasticity at 0,8 s	Plasticity at 0,95 s
	Coarse Mesh								
	1,5348	54034	312	329	Radius	0,09%	0,32%	0,61%	1,65%
	Fine Mesh								
	1,553	54030	310	329	Radius	0,10%	0,33%	0,65%	1,78%
	Coarse Mesh								
	1,2526	54862	295	306	Radius	0,09%	0,29%	0,49%	0,91%
	Fine Mesh								
	1,2525	54861	295	306	Radius	0,09%	0,30%	0,49%	0,91%
	Coarse Mesh								
	1,1506	55216	264	264	-	-	-	-	-
	Fine Mesh								
	1,2767	54744	296	309	Radius	0,09%	0,30%	0,49%	0,95%

Figure 3.3: A comparison between four different material setups. The purple areas of the flanges are where plastic material are applied.

3.2 EPRI comparison

The results obtained when applying the bolt pre-tensions that according to EPRI gave 1% plasticity in the radius can be studied in Figure 3.4. The majority of the flange models analyzed obtained plasticity in the radius below 1%. Two flange models obtained values above the 1% criteria stated by EPRI, and two models came very close to 1% plasticity in the radius. And none of the flange models analyzed obtained any flange rotation above 1 degree. The maximum flange rotation that could be seen was 0.61 degrees. The flange rotation for the bolt pre-tensions stated below was not presented by EPRI but it was considered an interesting result to compare to the plasticity in the radius, which is why it was also presented.



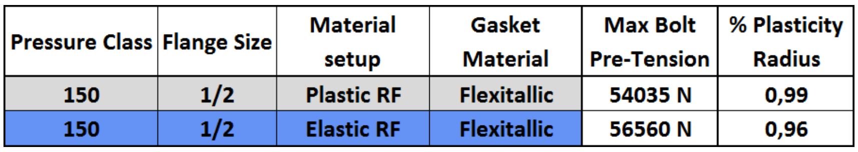
Pressure Class	Flange Size	1% Plastic Strain	Applied Bolt Pre-Tension [N]	Procent Plastic Strain, Radius	Deformation for P1 [mm]	Deformation for P2 [mm]	Difference P1-P2 [mm]	Distance P1 to P2 [mm]	Rotation [Degrees]
150	**0,5**	333 MPa	**55494**	**0,87**	1,153	0,926	0,227	27,55	**0,47**
150	**0,75**	465 MPa	**77488**	**1,00**	1,146	0,899	0,247	28,55	**0,50**
150	**16**	560 MPa	**1641631**	**0,49**	1,012	0,333	0,679	63,5	**0,61**
300	**2**	445 MPa	**239913**	**0,63**	1,034	0,809	0,225	36,45	**0,35**
300	**10**	533 MPa	**1562481**	**0,39**	0,982	0,507	0,475	60,6	**0,45**
400	**4**	893 MPa	**991556**	**2,73**	1,136	0,671	0,465	48,9	**0,54**
400	**14**	502 MPa	**3069037**	**0,47**	1,091	0,334	0,757	95,25	**0,46**
600	**1**	585 MPa	**157696**	**1,91**	1,085	0,84	0,245	37,1	**0,38**
600	**6**	652 MPa	**1433496**	**0,67**	0,984	0,535	0,449	69,55	**0,37**
900	**3**	676 MPa	**750607**	**0,99**	0,947	0,636	0,311	56,5	**0,32**
900	**12**	540 MPa	**4102083**	**0,80**	0,803	0,127	0,676	114,5	**0,34**
1500	**0,5**	240 MPa	**96510**	**0,14**	0,957	0,865	0,092	42,55	**0,12**
1500	**16**	392 MPa	**8766626**	**0,57**	0,543	0,282	0,261	177,8	**0,08**

Figure 3.4: The plasticity obtained when applying the EPRI bolt pre-tensions that gave 1% plasticity. The rotation of the flange, calculated from displacements of two vertexes, did not exceed 1 degree for any flange model.

It is hard to say whether EPRI's results are more correct than the ones obtained in this analysis. The complete method describing EPRI's FE-model was not available, which makes it difficult to know exactly what the results are compared to.

3.3 Results from the automated process

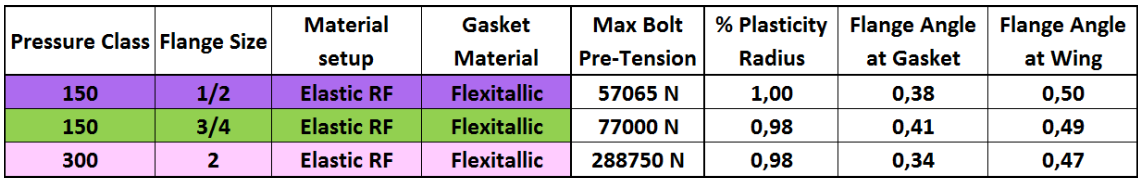
In Figure 3.5, the maximum bolt pre-tensions obtained for one flange model with two different material setups are presented. The flange model with plastic material in the raised face obtained a higher plasticity in the radius than the flange model with elastic material in the raised face.



Pressure Class	Flange Size	Material setup	Gasket Material	Max Bolt Pre-Tension	% Plasticity Radius
150	1/2	Plastic RF	Flexitallic	54035 N	0,99
150	1/2	Elastic RF	Flexitallic	56560 N	0,96

Figure 3.5: The maximum bolt pre-tensions obtained for one flange model with two different material setups.

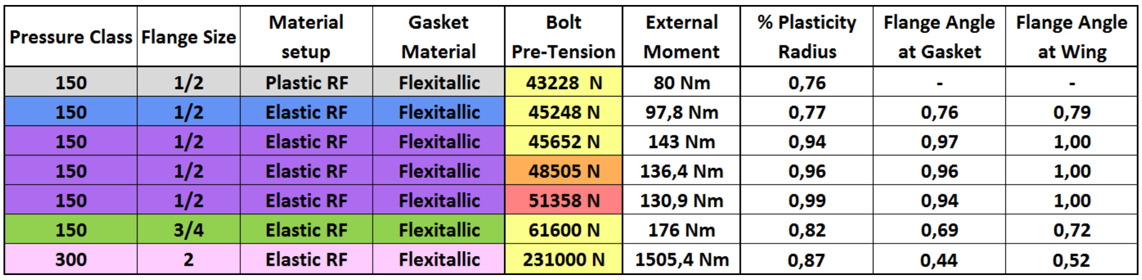
The maximum bolt pre-tensions obtained for three different flange models are presented in Figure 3.6. As shown in the figure, the plasticity in the radius is the bottleneck for these three flange models.



Pressure Class	Flange Size	Material setup	Gasket Material	Max Bolt Pre-Tension	% Plasticity Radius	Flange Angle at Gasket	Flange Angle at Wing
150	1/2	Elastic RF	Flexitallic	57065 N	1,00	0,38	0,50
150	3/4	Elastic RF	Flexitallic	77000 N	0,98	0,41	0,49
300	2	Elastic RF	Flexitallic	288750 N	0,98	0,34	0,47

Figure 3.6: The maximum bolt pre-tensions obtained for three different flange models.

In Figure 3.7, the maximum external moments obtained for the various flange models are presented. The color coding of the flange model properties correlate to the various analyzes of the flange models. Therefore, there are two different results for the 150# 1/2" flange. The yellow bolt pre-tensions represent 80% of the maximum bolt pre-tension obtained in Figure 3.6. The orange and red bolt pre-tensions represent 85% respective 90% of the maximum bolt pre-tensions obtained for the flange model. For the majority of the flange models, the flange wing angle acted as the bottleneck.



Pressure Class	Flange Size	Material setup	Gasket Material	Bolt Pre-Tension	External Moment	% Plasticity Radius	Flange Angle at Gasket	Flange Angle at Wing
150	1/2	Plastic RF	Flexitallic	43228 N	80 Nm	0,76	-	-
150	1/2	Elastic RF	Flexitallic	45248 N	97,8 Nm	0,77	0,76	0,79
150	1/2	Elastic RF	Flexitallic	45652 N	143 Nm	0,94	0,97	1,00
150	1/2	Elastic RF	Flexitallic	48505 N	136,4 Nm	0,96	0,96	1,00
150	1/2	Elastic RF	Flexitallic	51358 N	130,9 Nm	0,99	0,94	1,00
150	3/4	Elastic RF	Flexitallic	61600 N	176 Nm	0,82	0,69	0,72
300	2	Elastic RF	Flexitallic	231000 N	1505,4 Nm	0,87	0,44	0,52

Figure 3.7: The maximum external moments obtained for several different flange models.

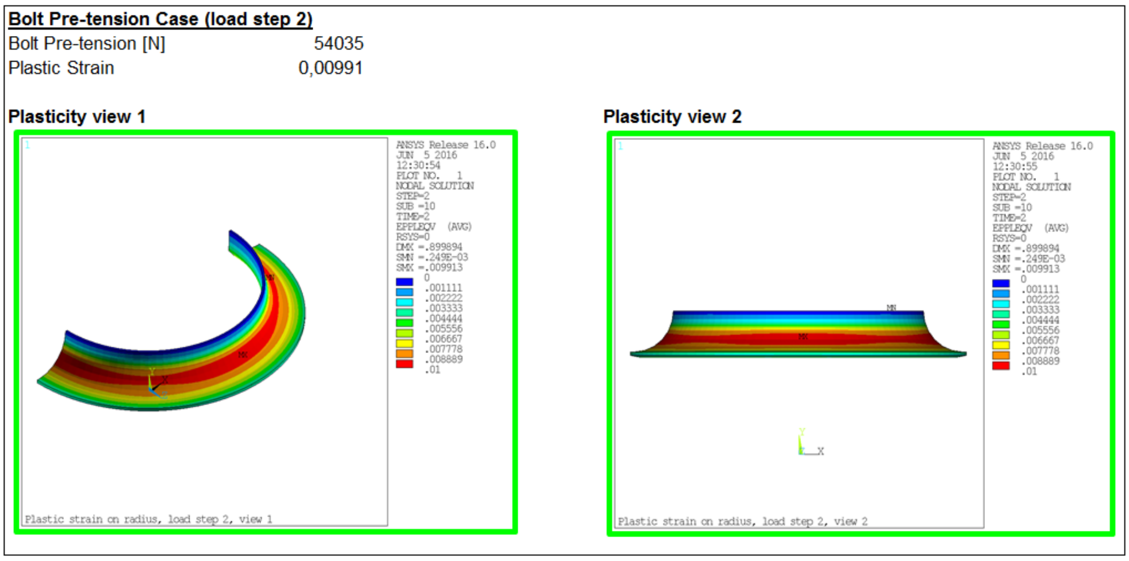
The results presented in Figure 3.8, show the maximum bolt pre-tensions obtained by the FE-models and the maximum allowed bolt pre-tensions according to the analytical calculations from the ASME norm. As seen when studying the results, there is definitely room for increasing the bolt pre-tensions without damaging the flange.

Pressure Class	Flange Size	Material setup	Gasket Material	Max Bolt Pre-Tension FE-model	Max Bolt Pre-Tension Analytical	Increase factor
150	1/2	Elastic RF	Flexitallic	57065 N	14000 N	4,08
150	3/4	Elastic RF	Flexitallic	77000 N	18000 N	4,28
300	2	Elastic RF	Flexitallic	288750 N	84000 N	3,44

Figure 3.8: The maximum external moments obtained for several different flange models.

Herkules presentation

When having run flange analyzes and result data has been output, the Herkules analysis files will be able to present this in a predetermined manner. Figures 3.9, 3.10 and 3.11 are shown here to give the reader a feeling of what the presentation will look like for each flange set in Herkules. These represent the maximum plasticity at the radius, the two flange rotation angles and maximum/mimumum gasket pressure, respectively. In this case *150# 1/2" SA-105 Flexitallic* was the flange used and the result values are for the maximum allowed bolt pre-tension. The plastic strain of 0.00991 is seen in Figure 3.9. The plasticity criteria of 0.01 was exceeded during the next run when stepping up the bolt pre-tension, which resulted in optimal bolt pre-tension to be 54035 N. Full Herkules result worksheets can be seen in Figures A.5, A.6 and A.7.



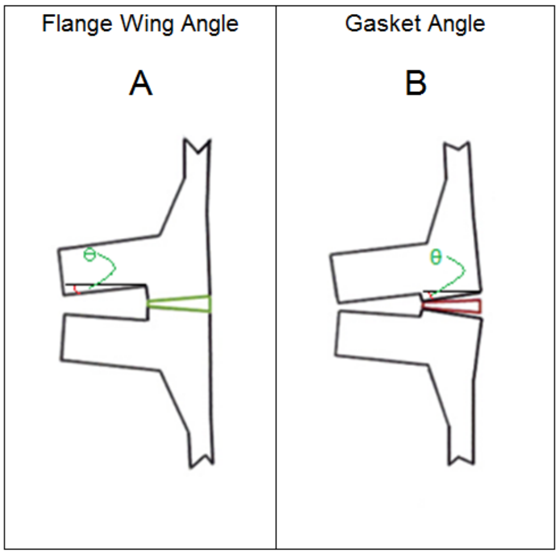
Bolt Pre-tension Case (load step 2)

Bolt Pre-tension [N]	54035
Plastic Strain	0,00991

Plasticity view 1

Plasticity view 2

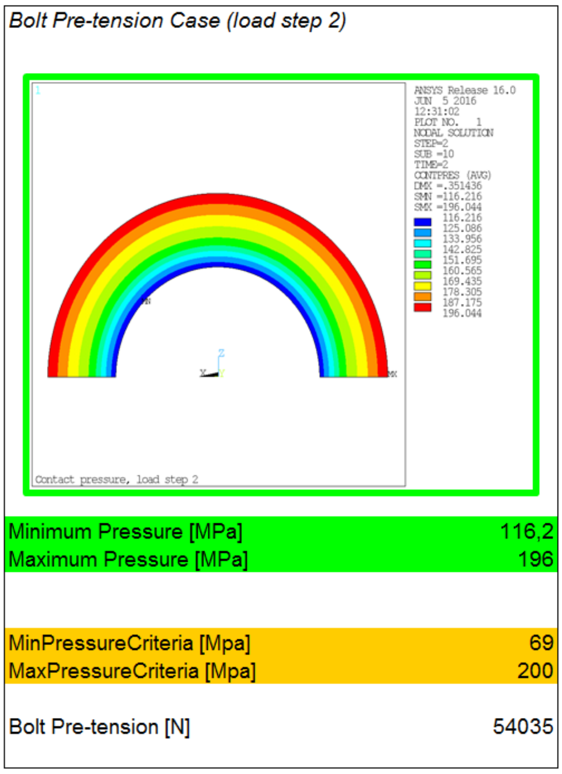
Figure 3.9: The plasticity at the radius for Flange - 150# 1/2" SA-105 Flexitallic.



Bolt Pre-tension Case (load step 1)	
Flange Wing Angle [deg]	0,558
Flange Gasket Angle [deg]	0,634
Bolt Pre-tension [N]	54035

Bolt Pre-tension Case (load step 2)	
Flange Wing Angle [deg]	0,673
Flange Gasket Angle [deg]	0,691
Bolt Pre-tension [N]	54035

Figure 3.10: The two flange angles for Flange - 150# 1/2” SA-105 Flexitallic.



Bolt Pre-tension Case (load step 2)

```
ANSYS Release 16.0
JUN  5 2016
12:31:02
PLOT NO.   1
NODAL SOLUTION
STEP=2
SUB =10
TIME=2
CONTPRES (AVG)
DMX =.351436
SMN =116.216
SMX =196.044
🟦    116.216
🟦    125.086
🟦    133.956
🟩    142.825
🟩    151.695
🟩    160.565
🟨    169.435
🟨    178.305
🟧    187.175
🟥    196.044
```

Contact pressure, load step 2

Minimum Pressure [MPa]	116,2
Maximum Pressure [MPa]	196
MinPressureCriteria [Mpa]	69
MaxPressureCriteria [Mpa]	200
Bolt Pre-tension [N]	54035

Figure 3.11: The gasket pressure for Flange - 150# 1/2” SA-105 Flexitallic.

4
Discussion

4.1 ANSYS Workbench model

During the development of the parametrized FE-model used in the optimization process, many different versions were evaluated before finally concluding that several models have to be used during the total flange analysis. A lot of effort was put into designing just one parametrized model. However, certain limitations in ANSYS Workbench prohibited this.

The two main limitations that made it necessary to create several FE-models were that the materials assigned to the various parts could not be parametrized, and that the FE-model could not handle that more bodies were added to the model. If there is a solution to these two problems, only one FE-model would be needed for the automated process.

Another problem that occurred repeatedly and without any obvious reason was that the script gave errors and failed to update the geometry. Eventually, the cause to the problem was found to be the constraints in-between the sketches used when revolving the bodies of the circular sector. Apparently, ANSYS Workbench could not handle the geometrical transformation when constraints were present in the sketches. Since the parameters were not changed all at once, but one at the time, the constraints prohibited the sketches to change. And the radius was often implemented in the wrong way when parametrized as part of a sketch. Therefore it was added afterwards using a *Blend*-function instead.

A lot of effort was also put into optimizing the mesh in order to reduce the computational time. However, splitting the flange into several parts and connecting these parts with contacts was not the right way. The contacts in ANSYS Workbench did not work as well as one could hope and as a result of this, the convergence rate decreased. In hindsight, it would have been better to just model the bottom flange as one solid circular plate instead. This would have reduced the number of elements significantly and since the criteria only focuses on the upper part of the flange joint, this would not have affected the results.

4.2 Optimization process

During the development of the optimization software there have been setbacks and acknowledged difficulties which have resulted in the shape not being as originally expected. Below are sections which go into the difficulties more in detail.

4.2.1 Python script

At this moment, there is no clear idea of the initial guess of both the bolt pre-tension and external moment for each flange set. Having fairly accurate initial guesses, result in that the analyzes run more smoothly. There are results for both of these loads for each flange set from the Mathcad documents used which at this moment contains the suitable optimal and maximum values. These are however recognized as probably being very conservative so perhaps some kind of factor multiplied with these values could lead to good starting guesses.

When it comes to the initial external moments being used for each flange analysis there is good reason to believe that there should different initial guesses for e.g. case A where the optimal bolt pre-tension is multiplied with 0.8 compared to case D where the optimal bolt pre-tension is multiplied with 0.95. Now there is no difference between the initial external moment for these inside the Python script since time has not allowed for appropriate studies of this to be conducted. Although, there are indications that e.g. case A would allow more external moment compared to case D from a few analyzes performed.

Another thing worth considering the optimization process is to investigate whether there exist suitable ANSYS-python commands that lets the operator modify the analysis settings such that the bolt pre-tension could be locked in an earlier stage during its own load step. Since it is stepped up linearly during this load step, one could perhaps use Python syntax to use "restart controls" in ANSYS, which would allow modification of a previously run analysis, and "lock" the bolt pre-tension at an earlier time, e.g. at time 0.8 seconds. This would result in a bolt pre-tension with a magnitude of 80 % of the original one. Same thing goes for the external moment if you can use restart controls from time 2 seconds, when the third time step including the external moment starts, and use this as a starting point using a new lower external moment.

It was expected that the material models of both the flange material and the gasket material would be able to be altered inside the ANSYS Workbench models by the use of python scripting. This however turned out not to be the case. After a few tries the ANSYS support group, EDR Medeso, was contacted and confirmed that it was not possible to change material in the intended way. A solution would be to create an ACT-module, which is a more advanced python based application for ANSYS, but with the time perspective of this Master Thesis it never really became an option. This is what led to the setup of the 24 different main ANSYS models instead of only two.

4.2.2 Quantity of results

The initial expectations regarding the quantity of the final results were set very high, but due to several unexpected challenges associated with the programs used, only a few of these results could be obtained. However, since there are somewhere above 5000 different flange set combinations, neither the time for running the analyzes, nor the room to present all the results would be sufficient.

5
Conclusions and future work

5.1 Conclusions

During this master thesis, the main scope was to develop an automated process in order to determine the maximum allowable bolt pre-tensions and external moments for a large number of flange models. This goal have been achieved, and the developed method will come to great use when selecting suitable flange joints in the future.

The framework of the automated process has been successfully developed, and with some modifications it will be completely finished. Getting the intertwined software to cooperate without errors proved to be a very time-demanding task, but in the final steps of the project it performed accordingly. Therefore, the optimization method can be considered to be a success. The results evidently show that a step-wise increase of both the bolt pre-tension and the external moment works for several different flange models. Herkules serves as a parameter library but also as the post processor it was intended to be at the start of this project.

When comparing the maximum allowed bolt pre-tensions obtained with our FE-models to the ASME-norm based hand calculations, there is obviously a large difference. The FE-models allow up to four times higher bolt pre-tension than the ASME-norm. This shows that the bolt pre-tension can be increased without any risk of damaging the flange.

The comparison made to EPRI's results does not show an exact correlation of the maximum allowed bolt pre-tension we obtained. However, the method EPRI used was hard to interpret and we could not get access to their complete report. Therefore, there might be a lot of unknown differences between the two models.

5.2 Future work

In order for future users to utilize the developed method, parts of this report have been written in a very detailed and descriptive way. To facilitate the workload in the future, here are some things to consider.

- In order for the complete automated process to work for all flange model setups, all 24 ANSYS FE-models shown in Figure A.1 and Figure A.2 have to be created. For the 24 FE-models, twelve Herkules project files have to be created, which have to have 84 unique analysis files. However, if the material and suppressing parts could be parametrized only one Herkules project file would be necessary. The project file would then need 1008 analysis files.

- Further investigations that evaluate the effect of plastic material in the raised face should be conducted. There might also be in ones interest to create gasket material models that are more consistent with the real gasket material features.

- Before running an analysis of the maximum bolt pre-tension of a flange model, it is of importance that a decent starting guess is chosen. This also applies to the starting guess of the maximum external moment. During the end of the project it became evident that the analyzes would take a very long time if the starting guess would be too low. Therefore a more efficient step-wise increase would be beneficial in order to make the automated process more time efficient.

- Having a too high initial guess of starting load (either bolt pre-tension or external moment) could result in the analysis diverging. At this moment the APDL command `NCNV,2`, which is meant to erase the termination of the analysis if convergence issues occur, is not implemented correctly in ANSYS at this moment. It seems to become active too late and should instead be set active immediately when starting an analysis. In other words, it should be placed directly after the *Solution* section inside the input file of an ANSYS analysis.

Bibliography

[1] ASME Boiler & Pressure Vessel Code III, Division 1 – Appendices: Appendix XI (2010)

[2] ASME Boiler & Pressure Vessel Code II, Part D – Materials (2010)

[3] ASME PCC-1-2010, Guidelines for Pressure Boundary Bolted Flange Joint Assembly

[4] Hemström J. (2013) Beräkningar av ASME flänsar enligt ASME III Appendix XI med externa moment enligt ASME III NC-3658 (2226600 / 1.5). Unpublished.

[5] ASME Boiler & Pressure Vessel Code III, Division 1 – Subsection NC (2010)

[6] ASME Boiler & Pressure Vessel Code VIII, Division 1 – Rules for construction of pressure vessels - Appendix 2 (2010)

[7] Ericson J. (2015) Manual Ansys ASME-Flänsberäkningar. Unpublished.

[8] Bolt Preload Stress for ANSI Raised-Face Flanges Using Spiral-Wound Gaskets: Sealing Technology & Plant Leakage Reduction Series. EPRI, Palo Alto, CA: 2000. 000000000001000066.

[9] Picture from page 52 in Bolt Preload Stress for ANSI Raised-Face Flanges Using Spiral-Wound Gaskets: Sealing Technology & Plant Leakage Reduction Series. EPRI, Palo Alto, CA: 2000. 000000000001000066.

[10] Picture from page 51 in Bolt Preload Stress for ANSI Raised-Face Flanges Using Spiral-Wound Gaskets: Sealing Technology & Plant Leakage Reduction Series. EPRI, Palo Alto, CA: 2000. 000000000001000066.

[11] Picture from page 55 in Bolt Preload Stress for ANSI Raised-Face Flanges Using Spiral-Wound Gaskets: Sealing Technology & Plant Leakage Reduction Series. EPRI, Palo Alto, CA: 2000. 000000000001000066.

[12] Good Bolting Practices, A Reference Manual for Nuclear Power Plant Personnel, Volume 1: Large Bolt Manual. EPRI, Palo Alto, CA: 1987. Report NP-5067.

[13] EN 1759-1, Flanges and their joints - Circular flanges for pipes, valves, fittings and accessories, Class designated - Part 1: Steel flanges, NPS 1/2 to 24 (EN 1759-1:2004)

[14] Picture from page 44 in chapter B Training. Nuclear Maintenance Applications Center: Assembling Gasketed, Flanged Bolted Joints. EPRI, Palo Alto, CA: 2007. 1015337.

A

Appendix 1 - Herkules

A.1 Herkules description

There are quite many settings as well as underlying VBA (Visual Basic for Applications) Excel programming for Herkules to work. Therefore, only the most basic functions which have been used in this Master Thesis will be explained. This was not created from scratch, but has been modified to serve the needs of this Master thesis.

Herkules project files The project file serve as a base for all analysis files for a certain flange group, and this is where all parameters are initially stated. The flange group is based on flange material and gasket material. There are four different flange materials and three different gasket materials, which results in 12 different flange groups and Herkules project files. But for the automated process, two Workbench FE-models have to be created for each flange group, making the total amount of FE-models 24. These models are represented by red dots in Figure A.1 and Figure A.2. The two FE-models differ in number of load cases. One is used when determining the maximum bolt pre-tension, and the other one when determining the maximum external moment. However, both FE-models use the same Hercules project file when loading their parameters.

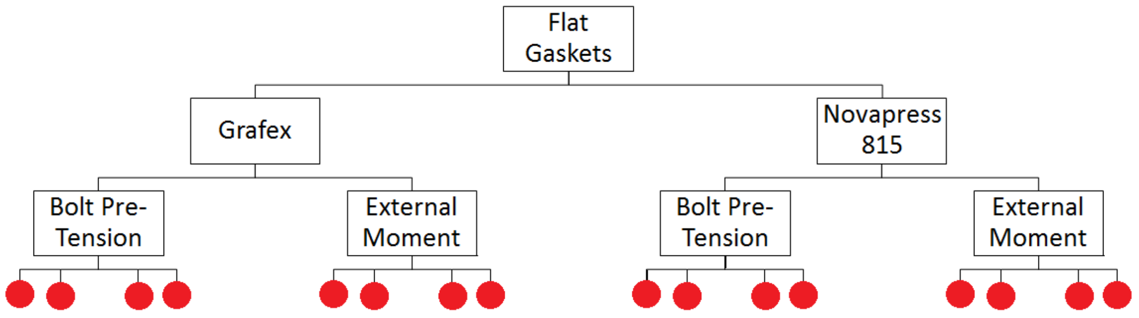


Figure A.1: A schematic figure of the FE-models with flat gaskets, needed for the automated process. The red dots represent the four flange materials.

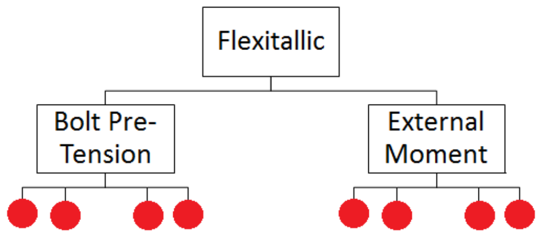


Figure A.2: A schematic figure of the FE-models with Flexitallic gaskets, needed for the automated process. The red dots represent the four flange materials.

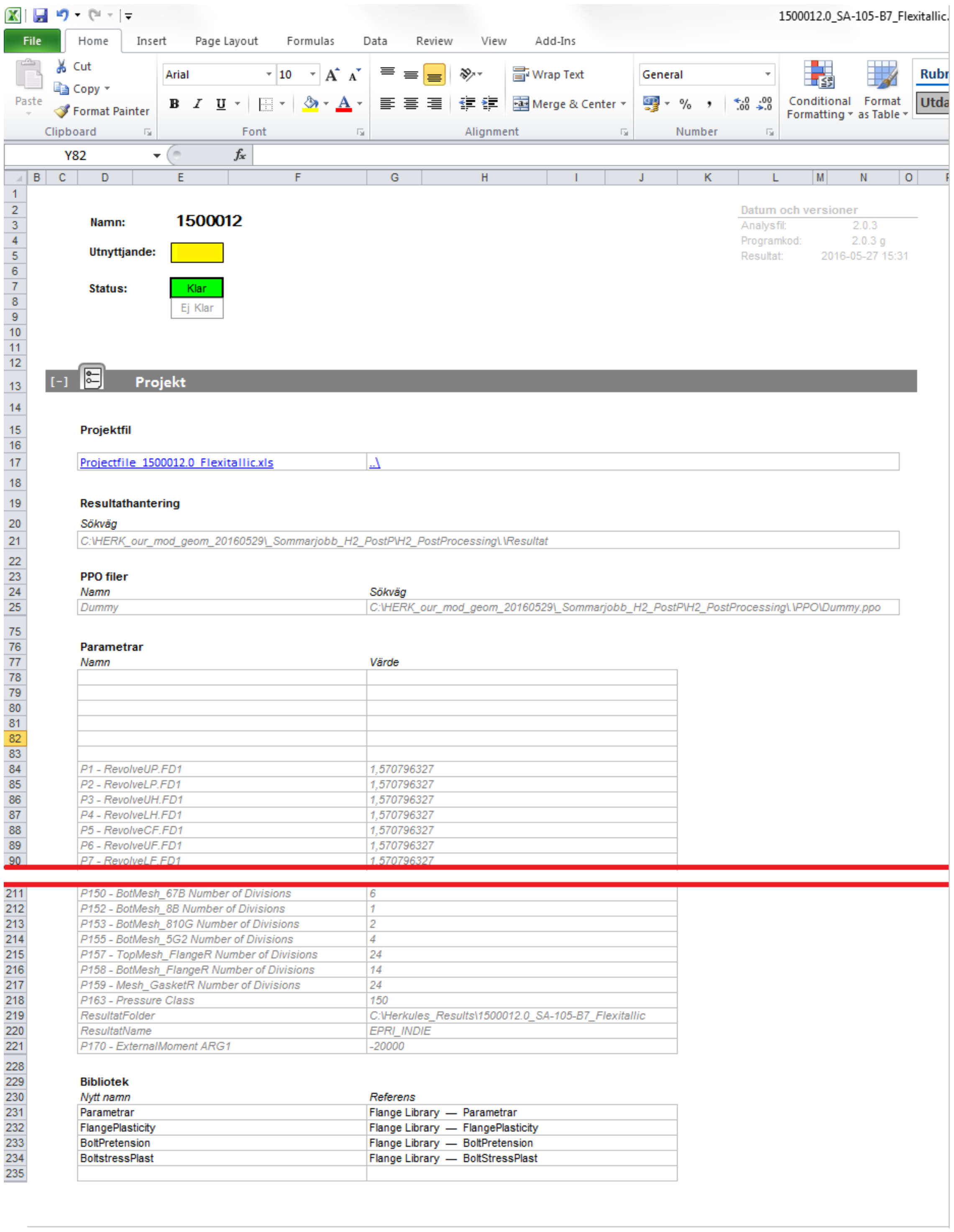
A picture of a typical Herkules project file can be seen in the Figure A.4 in Appendix A, and the following explanation of the project file will be based on this figure. In the upper left corner of the Excel document there are two buttons. The one marked with a cross-sign lets one select a certain flange analysis file and add this in the *C*-column. The name of the analysis file will be stated here and the blue text indicates that it also functions as a link directly to the analysis file. The cells to the right of this cell are all parameters that are used when running automated analyzes. These parameters determine both geometry, load magnitudes in different time steps as well as input for the APDL commands inside the Workbench model that controls the post-processing. The other button marked with a curved arrow lets the operator choose if any analysis files should be updated. This must be done in case a parameter is changed in the parameter row for a certain analysis file.

Herkules analysis files The Herkules analysis files are unique for each combination of flange size, flange material and gasket type. This is the file that the python script opens and from which it extracts the parameters for a flange set during a run. Apart from holding the parameters the Herkules analysis file also function as a presenter of the results being output from the post-process APDL command in ANSYS Workbench. This will be explained in more detail further down in this section.

A picture of the first and main worksheet of a typical Herkules analysis file can be seen in Figure A.3. The name of the file is stated at the top and further down below the *Parametrar* cell all parameters are placed row-wise with their names to the left and values to the right. These are the same parameters for a specific flange set as were lined column-wise in the Herkules project file mentioned earlier. This is the pre-processing part of the Herkules usage.

The post-processing part presents figures and values from all runs where the values also can be checked against criteria. Example figures of what the presentation looks like can be seen in Figures A.5, A.6 and A.7. The bolt membrane and bending stress criteria are evaluated here and do not exist as a stop criteria in the python script. The results are presented in worksheet nr. 2 and above. These worksheets are templates that are loaded from an external Excel file where the input for loading of results and pictures inside the worksheets have already been done. The benefit of doing this is that if some kind of change has to be made concerning the results presentation inside the worksheets, the change can be made inside this Excel document and then be automatically updated for all Herkules analysis files.

A.2 Herkules figures



Namn: 1500012

Utnyttjande:

Status: Klar / Ej Klar

Datum och versioner

Analysfil:	2.0.3
Programkod:	2.0.3 g
Resultat:	2016-05-27 15:31

[-] Projekt

Projektfil

Projectfile_1500012.0_Flexitallic.xls ..\

Resultathantering

Sökväg

C:\HERK_our_mod_geom_20160529_Sommarjobb_H2_PostP\H2_PostProcessing\.\Resultat

PPO filer

Namn	*Sökväg*
Dummy	C:\HERK_our_mod_geom_20160529_Sommarjobb_H2_PostP\H2_PostProcessing\.\PPO\Dummy.ppo

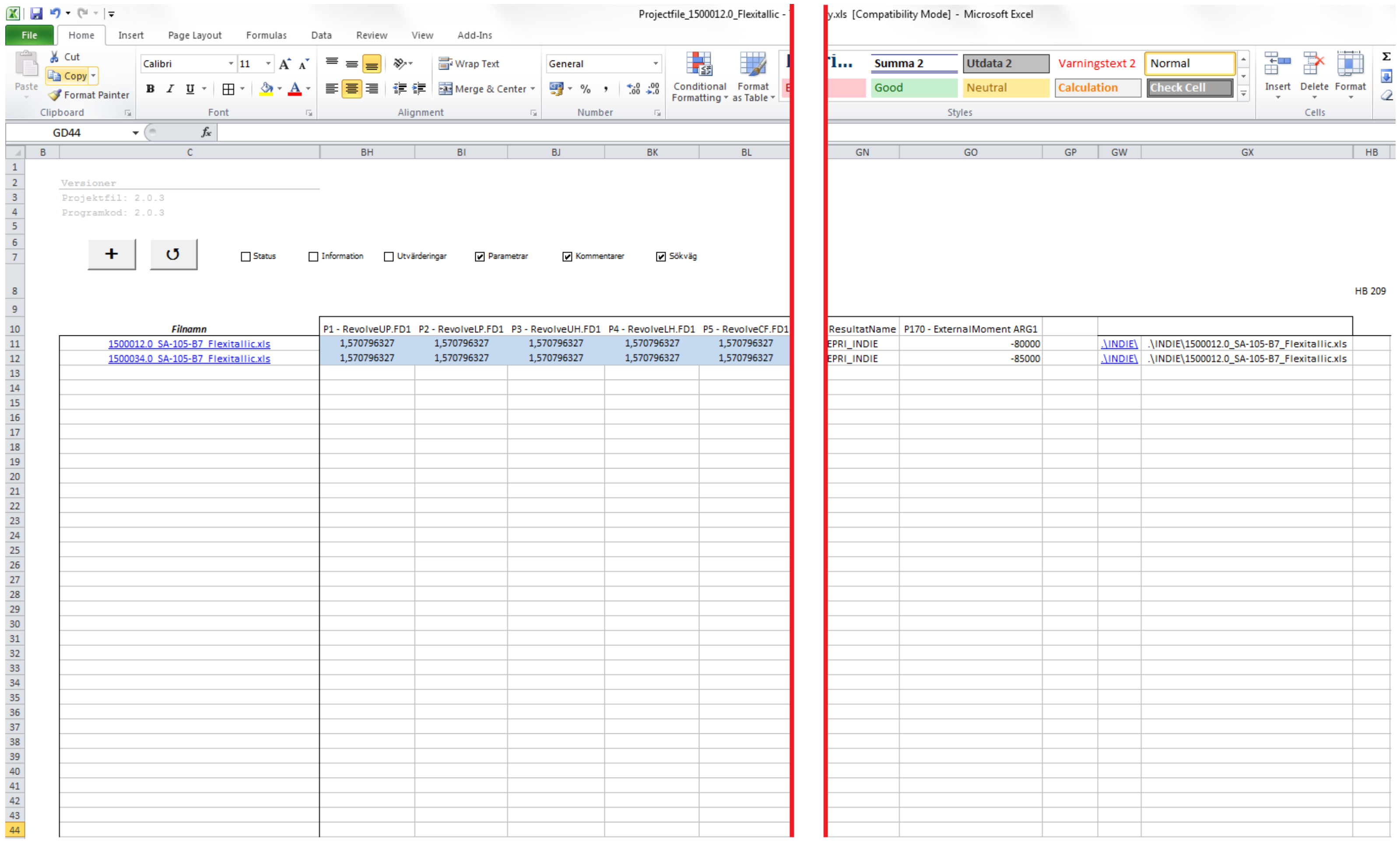
Parametrar

Namn	*Värde*
P1 - RevolveUP.FD1	1,570796327
P2 - RevolveLP.FD1	1,570796327
P3 - RevolveUH.FD1	1,570796327
P4 - RevolveLH.FD1	1,570796327
P5 - RevolveCF.FD1	1,570796327
P6 - RevolveUF.FD1	1,570796327
P7 - RevolveLF.FD1	1,570796327
P150 - BotMesh_67B Number of Divisions	6
P152 - BotMesh_8B Number of Divisions	1
P153 - BotMesh_810G Number of Divisions	2
P155 - BotMesh_5G2 Number of Divisions	4
P157 - TopMesh_FlangeR Number of Divisions	24
P158 - BotMesh_FlangeR Number of Divisions	14
P159 - Mesh_GasketR Number of Divisions	24
P163 - Pressure Class	150
ResultatFolder	C:\Herkules_Results\1500012.0_SA-105-B7_Flexitallic
ResultatName	EPRI_INDIE
P170 - ExternalMoment ARG1	-20000

Bibliotek

Nytt namn	*Referens*
Parametrar	Flange Library — Parametrar
FlangePlasticity	Flange Library — FlangePlasticity
BoltPretension	Flange Library — BoltPretension
BoltstressPlast	Flange Library — BoltStressPlast

Figure A.3: Herkules Analysis File. The red lines represent a large part being cut off.



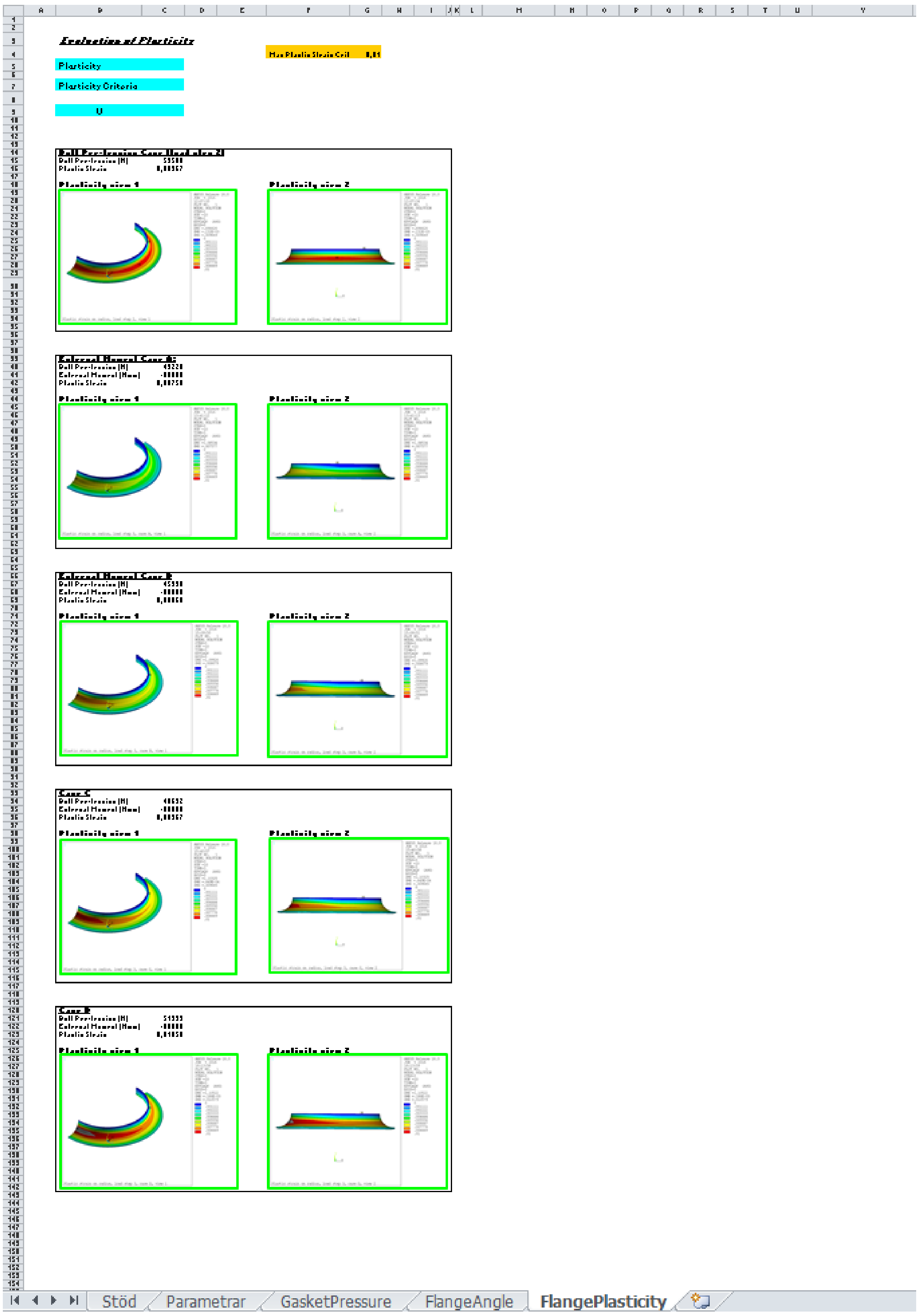
```
Versioner
Projektfil: 2.0.3
Programkod: 2.0.3
```

☐ Status ☐ Information ☐ Utvärderingar ☑ Parametrar ☑ Kommentarer ☑ Sökväg

HB 209

Filnamn	P1 - RevolveUP.FD1	P2 - RevolveLP.FD1	P3 - RevolveUH.FD1	P4 - RevolveLH.FD1	P5 - RevolveCF.FD1	ResultatName	P170 - ExternalMoment ARG1		
1500012.0_SA-105-B7_Flexitallic.xls	1,570796327	1,570796327	1,570796327	1,570796327	1,570796327	EPRI_INDIE	-80000	.\INDIE\	.\INDIE\1500012.0_SA-105-B7_Flexitallic.xls
1500034.0_SA-105-B7_Flexitallic.xls	1,570796327	1,570796327	1,570796327	1,570796327	1,570796327	EPRI_INDIE	-85000	.\INDIE\	.\INDIE\1500012.0_SA-105-B7_Flexitallic.xls

Figure A.4: Herkules Project File. The red lines represent a large part being cut off.



Evaluation of Plasticity

Max Plastic Strain Crit 0,01

Plasticity

Plasticity Criteria

U

Bolt Pre-tension Case (load step 2)

Bolt Pre-tension [N]	59500
Plastic Strain	0,00567

Plasticity view 1

Plasticity view 2

External Moment Case A

Bolt Pre-tension [N]	49220
External Moment [Nmm]	-00000
Plastic Strain	0,00750

Plasticity view 1

Plasticity view 2

External Moment Case B

Bolt Pre-tension [N]	45990
External Moment [Nmm]	-00000
Plastic Strain	0,00060

Plasticity view 1

Plasticity view 2

Case C

Bolt Pre-tension [N]	40692
External Moment [Nmm]	-00000
Plastic Strain	0,00567

Plasticity view 1

Plasticity view 2

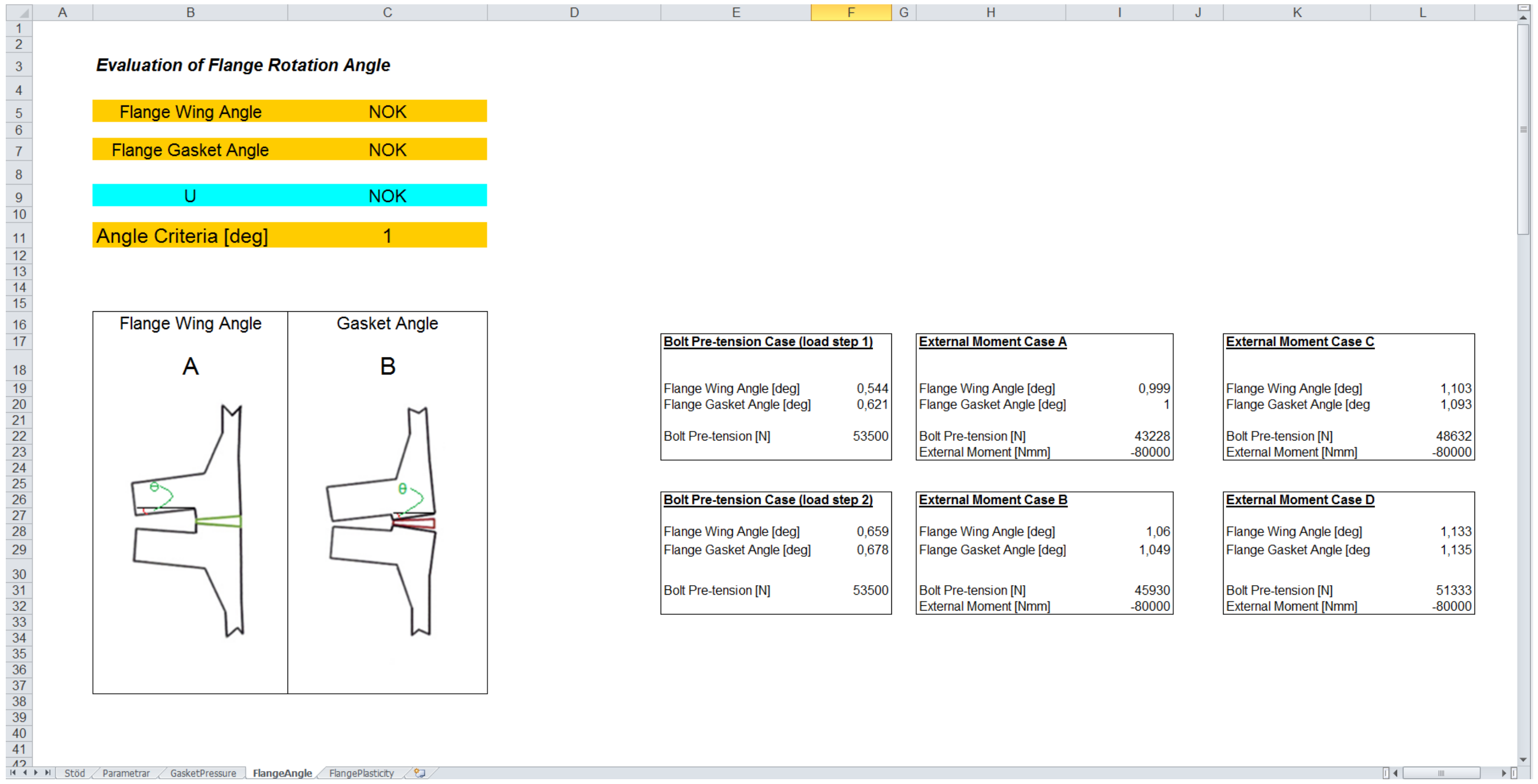
Case D

Bolt Pre-tension [N]	51999
External Moment [Nmm]	-00000
Plastic Strain	0,01050

Plasticity view 1

Plasticity view 2

Figure A.5: The full worksheet of the radius plasticity results presented in Herkules for both the bolt pre-tension case as well as external moment cases A-D.



Evaluation of Flange Rotation Angle

Flange Wing Angle	NOK
Flange Gasket Angle	NOK
U	NOK
Angle Criteria [deg]	1

Bolt Pre-tension Case (load step 1)	
Flange Wing Angle [deg]	0,544
Flange Gasket Angle [deg]	0,621
Bolt Pre-tension [N]	53500

External Moment Case A	
Flange Wing Angle [deg]	0,999
Flange Gasket Angle [deg]	1
Bolt Pre-tension [N]	43228
External Moment [Nmm]	-80000

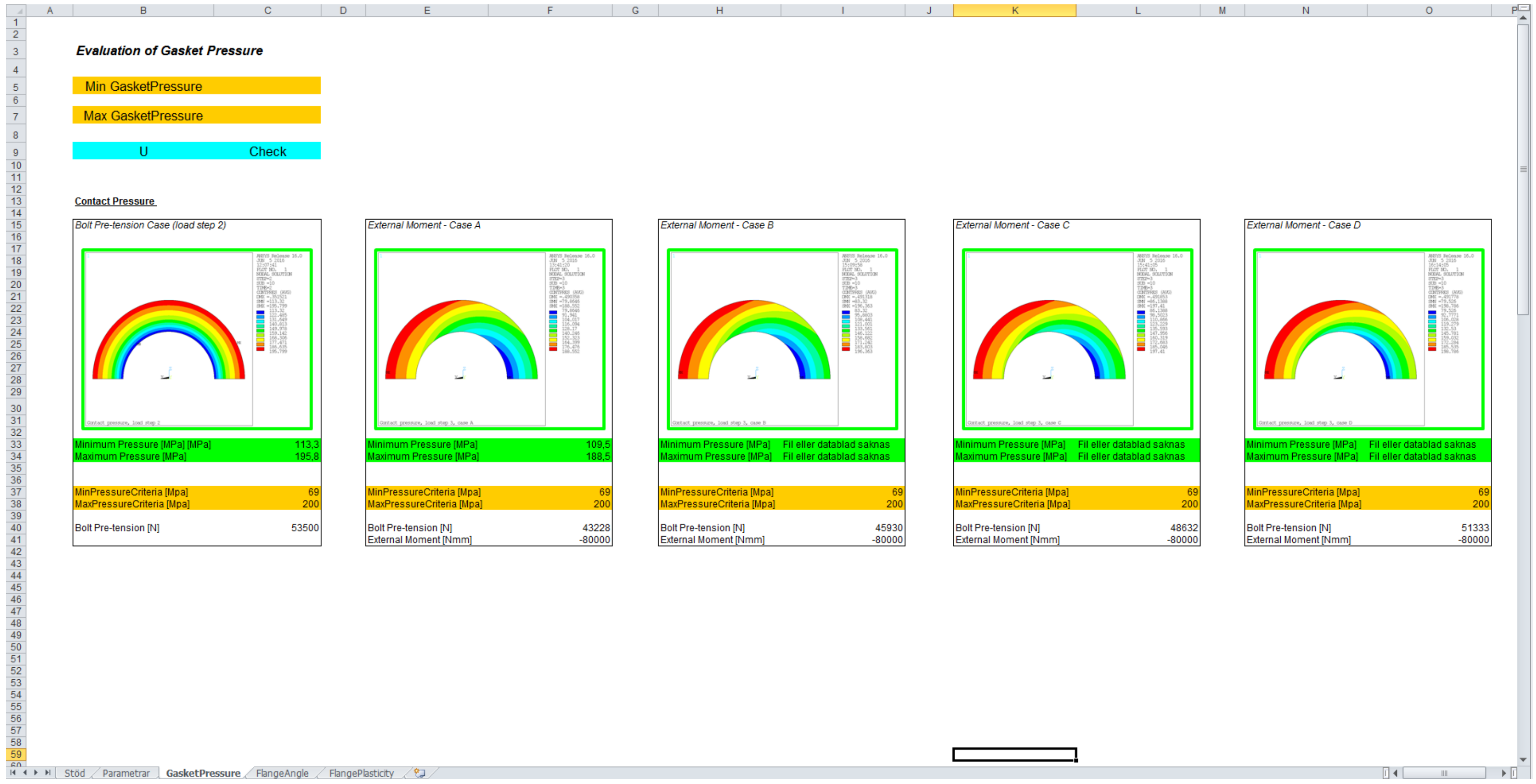
External Moment Case C	
Flange Wing Angle [deg]	1,103
Flange Gasket Angle [deg	1,093
Bolt Pre-tension [N]	48632
External Moment [Nmm]	-80000

Bolt Pre-tension Case (load step 2)	
Flange Wing Angle [deg]	0,659
Flange Gasket Angle [deg]	0,678
Bolt Pre-tension [N]	53500

External Moment Case B	
Flange Wing Angle [deg]	1,06
Flange Gasket Angle [deg]	1,049
Bolt Pre-tension [N]	45930
External Moment [Nmm]	-80000

External Moment Case D	
Flange Wing Angle [deg]	1,133
Flange Gasket Angle [deg	1,135
Bolt Pre-tension [N]	51333
External Moment [Nmm]	-80000

Figure A.6: The full worksheet of the flange rotation angle results presented in Herkules for both the bolt pre-tension case as well as external moment cases A-D.



Evaluation of Gasket Pressure

Min GasketPressure

Max GasketPressure

U | Check

Contact Pressure

Bolt Pre-tension Case (load step 2)

Minimum Pressure [MPa] [MPa]	113,3
Maximum Pressure [MPa]	195,8
MinPressureCriteria [Mpa]	69
MaxPressureCriteria [Mpa]	200
Bolt Pre-tension [N]	53500

External Moment - Case A

Minimum Pressure [MPa]	109,5
Maximum Pressure [MPa]	188,5
MinPressureCriteria [Mpa]	69
MaxPressureCriteria [Mpa]	200
Bolt Pre-tension [N]	43228
External Moment [Nmm]	-80000

External Moment - Case B

Minimum Pressure [MPa]	Fil eller datablad saknas
Maximum Pressure [MPa]	Fil eller datablad saknas
MinPressureCriteria [Mpa]	69
MaxPressureCriteria [Mpa]	200
Bolt Pre-tension [N]	45930
External Moment [Nmm]	-80000

External Moment - Case C

Minimum Pressure [MPa]	Fil eller datablad saknas
Maximum Pressure [MPa]	Fil eller datablad saknas
MinPressureCriteria [Mpa]	69
MaxPressureCriteria [Mpa]	200
Bolt Pre-tension [N]	48632
External Moment [Nmm]	-80000

External Moment - Case D

Minimum Pressure [MPa]	Fil eller datablad saknas
Maximum Pressure [MPa]	Fil eller datablad saknas
MinPressureCriteria [Mpa]	69
MaxPressureCriteria [Mpa]	200
Bolt Pre-tension [N]	51333
External Moment [Nmm]	-80000

Figure A.7: The full worksheet of the gasket pressure results presented in Herkules for both the bolt pre-tension case as well as external moment cases A-D.

B
Appendix 2 - Python script

B.1 Python script description

The Python script will be gone through rather thoroughly in this Appendix. A good idea is to have a glance at the script simultaneously while reading this chapter. The following subsections will be titled the same as the corresponding sections in the Python script, in chronological order. Everything inside the script that is stated inside the `print()` command is being printed out in the command window during a run of the script.

Defining and opening ANSYS main model The initial actions to take place in the Python script is to determine for what flange material the analyzes shall be solved for. This is done with parameters `mtrl` and `gsk`. The first if statement handles the choice of flange material and by choosing one of the four options, two text-strings and one parameter are created. The first text string is used for creating an individual filename when saving a new ANSYS model created by parameters. `PostCommParam` is a parameter that sets the material in the APDL command for post-processing inside the ANSYS model. `MtrlSupprString` is used when suppressing the plastic material properties of the flange material when running the elastic analyzes. Then the gasket type is defined and another text string for creating a unique file name is obtained.

These text strings are also used when, later in the script, loading a Herkules project file for this flange group combination. There are 12 flange group combinations (Separate ANSYS main models had to be used for these 12 groups, since there does not exist a python script command to change the material from one to another inside the material library of an ANSYS Workbench file).

Criteria, loads and output Different criteria such as, gasket minimum/maximum pressure, maximum flange angle and maximum plasticity are stated here. The criteria for all different gasket types are depending on which gasket is used in the opened main model. The correct gasket criteria will be chosen further down in the script. The load parameters refer to bolt pre-tension and external moment which are the loads being stepped up in the two different analyzes of a flange model. The output parameters are values being compared against the criteria and are being fetched after each run during an analysis.

Predefining parameter matrices Number of load cases decides how many times the load parameter will be stepped up after an initial run. This will be either the bolt pre-tension or the external moment. `FirstFlangeRow` and `LastFlangeRow` represent the row numbers in the Herkules project file which in turn represent what Herkules analysis files are being used. This is the flange set interval that constitutes the number of flange sets that will be analyzed. The Herkules analysis file contains the parameters that will be collected, which is mentioned in the next section. `NumberOfParameters` represents the number of parameters that ANSYS will use as input and not the total number of parameters that is of use in the Herkules analysis file. `PValue` and the code beneath are matrices and vectors which are being pre-defined for the next section where the parameters are fetched from Herkules.

Collecting parameters from Herkules Here a for-loop is initiated over the flange set interval mentioned in the earlier section. It starts off with opening Excel and then opening the Herkules project file. Next it sets the first worksheet as active and then fetches the name of the Herkules analysis file. Now a file path is constructed using the name of the Herkules analysis file and the folder structure of where it is located.

The Herkules analysis file is then opened and the project file is closed. Next are several for-loops where all have the same characteristics and this is where the actual collecting of the parameters occur. Here the predefined parameter matrices and vectors are filled with parameters. First `Ps` is constructed as a text string that always consist of a `P` + `i` which for the first iteration in the first for-loop becomes `P1`. The ranges of the for-loops are constructed in a way that the parameters always gets the same name as the ones predefined in the parameter library inside the ANSYS main model. The first for-loop is now described as an example.

Matrix `PValue` for instance receives the value of the contents of the cell placed in column 3 and row (1+83) in the Herkules analysis file, see Figure A.3. This is repeated for all parameters in the Herkules analysis file that will be used in the new ANSYS workbench model. Some parameters are also fetched individually, as can be seen.

The reason why there is a lot of for-loops covering a rather small range each, instead of only one for-loop covering the whole range is because of the fact that a parameter that once is created in ANSYS Workbench and then deleted will result in that certain parameter number to not be used again. The next time a parameter is created it will automatically give it the number above the previously deleted one. Since the making of the ANSYS main files have been an iterated processes the parameters have been set sporadically.

Finally, Excel is closed since all parameters now have been collected for all chosen analyzes.

Setting parameters in ANSYS Workbench for bolt pre-tension analysis Now a for-loop over the flange set interval starts once again. This one will go through the whole rest of the script. The first thing that happens is that the chosen Workbench main model is re-opened in case the iteration step is 2 or higher, since this would mean that a new flange set analysis is taking place and a new set of parameters is needed.

The same for-loop pattern with the same ranges, as earlier when the parameters were collected from Herkules, now emerges but here the parameters are now to be put into the parameter library in ANSYS instead. Initially the system information of the ANSYS main file is retrieved by using the `GetSystem` command. This opens up the possibility to use the `GetDesignPoint`, `GetParameter` and `SetParameter` commands to modify the retrieved system. The new set of collected parameters can now be put into the parameter library in ANSYS which will replace the old parameters. All of these so-called 'parameter for-loops' should have the same chronological order, as the ones that were used when collecting parameters from Herkules.

Fetching gasket type Here the system is retrieved once again and the parameter that decides what gasket type to be used used is fetched. It has two purposes. To create a text string that is used to create individual file names and also to set the correct gasket criteria used later in the stepping up process.

Saving analysis as new file The newly created flange type based workbench file is saved with an individual which makes it possible to open this file in ANSYS Workbench for later evaluation, if the need would arise after the analysis.

Preparing for simulation in ANSYS Workbench for bolt pre-tension analysis The system of the new individual flange analysis file is now opened. The first step here is to go to into the engineering data container and make the flange material data open for modification. The commands `GetMaterial`, `GetProperty`, `GetPropertyData` and `SetSuppression` are now used to get into the *Bilinear* part of the $IsotropicHardening$ section of the material fetched with GetMaterial . The *Bilinear* part of the material is now unsuppressed with the `SetSuppression` command. This is to ensure that the selected parts in the model actually has a plastic material model. The geometry-, model-, and setup-containers are then refreshed which updates the new flange model with its new settings.

Running plastic simulation with stepped up bolt pre-tension An initial analysis is performed refreshing the *Solution* section of the project container in ANSYS Workbench. After this has been performed the output values, which will be compared to the criteria, are fetched from the analysis results. These are for the gasket minimum and maximum pressure as well as the plasticity in the radius of the flange. Then the force steps, depending on the amount of plasticity in the flange, is predefined with four different levels, as can be seen in the script. These will be used in the later stepping up process.

Stepping up bolt pre-tension A for-loop now starts with the predefined interval of number of maximum allowed times that the bolt pre-tension will be stepped up. To always get the full analysis of a flange set, one should give this a very high number in the beginning of the script.

Now another criteria is defined which was not able to be produced in the standard ANSYS workbench manner by using pre-processing functions. Instead the APDL *PostCommand*, which is at the bottom of the Workbench file that prints all results, produces a text file (.txt) containing values of the flange angle for each time step. This is then opened and read into this script by using the `readline` and `readlines` commands. An empty text file with the exact same name is created, since there otherwise would be an error while running the script stating that no such text file exists. After these specially produced flange angle output values have been loaded, they are compared to the flange angle criteria through if-statements. Thereafter if-statements follow for the gasket minimum and maximum pressure comparisons. If any of the criteria is violated the step up process of the for-loop will break. The final if-statement inside the for-loop checks the plasticity output value against the plasticity criterion.

If the plasticity value is below the criterion it enters the if-statement and the amount of increase of the bolt pre-tension is determined by the plasticity percentage. If the plasticity is below 0.5 % the new bolt pre-tension becomes the old plus an addition of 20 %. In other words a factor of 1.2 times the old bolt pre-tension. Below 0.75 % gives a factor of 1.1, below 0.9 % gives a factor of 1.05 and between 0.9 % and 1.0 % gives a factor of 1.01. When the bolt pre-tension has been determined the parameter is updated in the ANSYS analysis file. This update also concerns the `Run` parameter that steps up with an increment of 1 each time and defines in what folder the results from the *PostCommand* should placed. Finally the Workbench file is saved when the for-loop is finished.

Analysis continuation check This is a check to see if there at all should be a continuation of the analysis for the actual flange set. If `num == 0`, it means that one or more of the criteria were violated in the first run, which means that it never started a bolt pre-tension stepping up process and that no optimal bolt pre-tension has been established. If this is the case a message is printed in the command window informing that the initial guess for the bolt pre-tension was too high and that the analysis for the specific flange set is terminated.

The `continue` command will make the for-loop break and restart at the next increment for the specified for-loop range. Since `row` is updated this will result in the next flange set being analysed. However, if the initial guess of the bolt pre-tension is largely above a suitable one, the solving of the model could get convergence issues. If this would happen the `continue` command will not work since it would result in an error message occurring in ANSYS Workbench. This will also result in the script not being able to proceed and the script will have to be reset manually.

Running elastic simulation for optimal bolt pre-tension An elastic simulation is now being run for the highest allowed bolt pre-tension from the plastic analysis that did not violate any criteria. This is performed since the membrane and bending stress can not be accurately evaluated if all the material types in the model does not have an elastic material model.

The material models of the other parts in the model must not be of plastic nature since this could lower the impact on the bolts. If a a part of the flange starts to plasticize it can not pass on as much force or pressure compared to a non-plasticized part and will therefore possibly not reveal the worst case scenario for the bolts.

The current Workbench file is saved as a new file for the elastic analysis. The second last design point is copied to the place of the first design point and the rest of the design points is then deleted. The elastic analysis for the optimal bolt pre-tension will only be run once. Also, parameter `P130` is set to 0, since this will give another input to the post-processing APDL command. This will change a text string from `Pla` to `Ela` which is placed in the end of the file name so that it will be saved as a new file.

Next the *Engineering Data* section is opened and the flange material model is modified such that the *Isotropic Hardening* properties are suppressed. This will result in the material model not having any plastic properties. In fact this is the only thing differing from its elastic counterpart inside the ANSYS material library. The analysis is then run and the Workbench file is saved when the run is finished.

Initiating external moment analyzes with differing bolt pre-tensions The analyzes of the maximum external moment is now initiated. There will be four different analyzes where the external moment is stepped up until any of the same criteria used for the bolt pre-tension analysis is violated and `BPfac` is a factor multiplied with the optimal bolt pre-tension. Its start value is 0.8 and will be stepped up with an increment of 0.05 until `BPfac` has become 0.95. This is performed to test what maximum external moment a certain flange set can withstand before failing. Since it is not trivial to foresee how the impact of a certain bolt pre-tension magnitude will affect the maximum external moment, 4 different cases are tested.

Setting parameters in ANSYS Workbench for external moment analysis The external moment main model is opened. Then there is an if statement that assigns `BPfacText` a certain letter depending on which of the four cases that is run. This is used to give the flange angle text file, which includes the flange angle output, an individual name. This parameter is sent into the post-processing APDL command. Now the parameters are once again sent in to the ANSYS model. An addition of three extra parameters are also used, which can be seen in the script right before the next section begins. Finally the model is saved having a unique file name. From now on the script will include a very similar approach to the running of the stepped up bolt pre-tension. The main difference is that the external moment part has four different cases and the bolt pre-tension part has only one, as mentioned above.

B.2 Python script

```
# encoding: utf-8
SetScriptVersion(Version="14.5")
#
#
#       PYTHON SCRIPT
#
#
#   Created by: Joakim Ericson 2015-08-05
#   Modified by: Peter Ostrowskis 2016-05-23
#
# The script uses the following files:
# Herkules project file
# Herkules analysis files
# ANSYS main model files
#

################################################################################
#
#      Defining and opening ANSYS main model
#
#

## Flange material
mtrl=1    # 1= SA-105-B7, 2= SA-182-F304, 3= SA-182-F316L, 4= SA-182-F316

if mtrl == 1:
fmtrl="SA-105-B7"
PostCommParam=1
MtrlSupprString="Outer Metal Ring - A105 - LA"
elif mtrl == 2:
fmtrl="SA-182-F304"
PostCommParam=2
MtrlSupprString=""
elif mtrl == 3:
fmtrl="SA-182-F316L"
PostCommParam=3
MtrlSupprString=""
#elif mtrl == 4:
# fmtrl="SA-182-F316"
# PostCommParam=4
# MtrlSupprString=""


## Gasket type
gsk=3 # 1= Grafex, 2= Novapress, 3= Flexitallic

if gsk == 1:
fgsk="Grafex"
elif gsk == 2:
fgsk="Novapress"
elif gsk == 3:
fgsk="Flexitallic"

print("Opening WB Basic Model")
Open(FilePath="C:\HERK_our_mod_geom/"+fmtrl+"_"+fgsk+"_BP.wbpj")
print("WB Basic Model Opened")
################################################################################
#
#       Criteria, Loads and Output
#
#

#    Criteria

#GrafMinClosCrit1=  # depending on pressure level
#GrafMinClosCrit2=
#GrafMinClosCrit3=
#GrafMinClosCrit4=
#GrafMinClosCrit5=
#GrafMinClosCrit6=

#GrafMaxClosCrit1=  # dependening on temperature
#GrafMaxClosCrit2=

#NovaMinClosCrit1=   # depending on pressure level
#NovaMinClosCrit2=
#NovaMinClosCrit3=
#NovaMinClosCrit4=
#NovaMinClosCrit5=
#NovaMinClosCrit6=

#NovaMaxPresCrit1=  # dependening on temperature
#NovaMaxPresCrit2=

FlexMinPresCrit= 69  # [Mpa]
FlexMaxPresCrit= 220    # dependening on temperature

FlangeAngleCriterion1=1.0
FlangeAngleCriterion2=1.0

PlasticityCriterion=0.01

# Load Parameters # Load Parameters to be stepped up to find opt value against criteria

LoadParameter="P106"      # Bolt pretension
MomParameter="P170" # External moment


# Output Parameters

OutputParameter="P156" # Plasticity at radius. Value to be checked against

#OutputParameter3= "P161" # Gasket min closure ls1  ls1=load step 1 etc.
#OutputParameter4= "P162" # Gasket max closure ls1
#OutputParameter5= "P163" # Gasket min closure ls2
#OutputParameter6= "P163" # Gasket max closure ls2
#OutputParameter7= "P163" # Gasket min closure ls3
#OutputParameter8= "P163" # Gasket max closure ls3

###########################################################################
#
# Pre-defining Parameter Matrices
#
#

MaxNumberOfLoadCases=1    # Number of steps for the load parameter to be stepped up


FirstFlangeRow=112    # Interval of what rows should used in the Herkules project file.
LastFlangeRow=112    ## These are the same as individual flange sets


NumberOfParameterRows=LastFlangeRow+1-FirstFlangeRow


NumberOfParameterColumns=136

PValue = [[0 for x in range(NumberOfParameterColumns)] for x in range(NumberOfParameterRows)]
PValueName=[0 for x in range(NumberOfParameterRows)]
NumBolts=[0 for x in range(NumberOfParameterRows)]

PValueNameEX=[0 for x in range(NumberOfParameterRows)]


#########################################################################
#
```

```
# Collecting Parameters From Herkules
#
#

row=0
## Loop to collect flange sets from excel
for rows in range(FirstFlangeRow,LastFlangeRow+1):

## Open Excel
from System.IO import Directory, Path
import clr
clr.AddReference("Microsoft.Office.Interop.Excel")
import Microsoft.Office.Interop.Excel as Excel


## Open Herkules project file
ex1 = Excel.ApplicationClass()
ex1.Visible = False

fileMainProject="C:\HERK_our_mod_geom\_Sommarjobb_H2_PostP\H2_PostProcessing/Projectfile_"+fmtrl+"_"+fgsk+".xls"

workbook1 = ex1.Workbooks.Open(fileMainProject)
worksheet1=workbook1.ActiveSheet

# Hämtar namnet på simuleringen
FileAnalysis=worksheet1.Cells(rows,3).Value()
print("File name")
print(FileAnalysis)

fileAnalysis="C:\HERK_our_mod_geom\_Sommarjobb_H2_PostP\H2_PostProcessing\INDIE"+"\\"+FileAnalysis

#workbook.Close()
ex1.DisplayAlerts=False
ex1.Quit() # Stänger Herkules projektfil

## Open analysis project file
ex2 = Excel.ApplicationClass()
ex2.Visible = False
workbook2 = ex2.Workbooks.Open(fileAnalysis)
worksheet2=workbook2.ActiveSheet

NumBolts[row]=worksheet2.Cells(175,7).Value2.ToString()
#FlangNam[row]=worksheet2.Cells(176,7).Value2.ToString()


## Loop to collect values from excel
col=0
for i in range(1,9):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+83,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(10,37):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+82,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(40,84):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+79,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1
```

```
i=87
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(163,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(89,94):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+75,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

i=97
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(169,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(99,104):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+71,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1


# Anger antalet bultar
i=104
Ps=""
Ps="P"+str(i)
PValueNameEX[row]=(worksheet2.Cells(175,7).Value())
print(Ps)
print(str(PValueNameEX[row]))


# Anger namnet på simuleringen
i=105
Ps=""
Ps="P"+str(i)
PValueName[row]=(worksheet2.Cells(176,7).Value())
print(Ps)
print(str(PValueName[row]))

for i in range(106,119):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+71,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1


i=125
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(190,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(128,139):
Ps=""
```

```
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+63,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(140,145):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+62,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(146,151):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+61,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(152,154):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+60,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

i=155
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(214,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

for i in range(157,160):
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(i+58,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

# Defines Pressure Class

i=163
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(218,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1

print("Here comes the BOOM = EM-parameter")

# Retrieves the External Moment Parameter

i=170
Ps=""
Ps="P"+str(i)
PValue[row][col]=worksheet2.Cells(221,7).Value()
print(Ps)
print(str(PValue[row][col]))
col=col+1
row=row+1

# Stänger analysfilen
```

```
ex2.ActiveWorkbook.Close(SaveChanges=0)
ex2.DisplayAlerts=False
ex2.Quit()

print("Excel End")


###############################################################################
# Killing Excel.exe process
import os

os.system('taskkill /f /im Excel.exe')

print("Process killed")


###############################################################################
#
# Setting Parameters in ANSYS Workbench For Bolt Pretension Analysis
#

row=0 # Säkerställer att row och rows är återställda till 0
rows=0
for rows in range(FirstFlangeRow,LastFlangeRow+1):

## Open project if row > 0
if row > 0:
print("Opens WB Basic Model for the " +str(row)+ " time in the Flange Model for loop")
Open(FilePath="C:\HERK_our_mod_geom/"+fmtrl+"_"+fgsk+"_BP.wbpj")
print("WB Basic Model Opened")
else:
print("No New WB Basic Model Opened since row < 0")

## Define parameters from PValue-matrix to Workbench-parameters
it=0

system1 = GetSystem(Name="SYS")
for i in range(1,9):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(10,37):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(40,84):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1
```

```
i=87
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(89,94):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=97
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(99,104):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

# Anger antalet bultar
i=104
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValueNameEX[row]))
print(Ps)
print(str(PValueNameEX[row]))

# Anger namnet på simuleringen
i=105
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValueName[row]))
print(Ps)
print(str(PValueName[row]))

for i in range(106,119):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
```

```
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=125
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(128,139):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(140,145):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(146,151):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(152,154):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=155
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(157,160):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=163
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=168
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PostCommParam))
print(Ps)
print(str(PostCommParam))


###########################################################################################
#
# Fetching Gasket Type
#
#
# Fetching Gasket Parameter for File Name Creation. Also for later gasket criteria
# definiton

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
outputParam1 = Parameters.GetParameter(Name="P129")    # (senare parameter)
Gask = outputParam1.Value.Value

if Gask == 1:
Gasket = "Grafex"
# ytterligare if sats som kollar PRESSURE CLASS så att rätt textsträng och kriterie används

print("Using Grafex as Gasket")

elif Gask == 2:
Gasket = "NovaPress"


print("Using NovaPress as Gasket")

elif Gask == 3:
Gasket = "Flexitallic"
GasketMinPresCrit = FlexMinPresCrit
GasketMaxPresCrit = FlexMaxPresCrit
print("Using Flexitallic as Gasket")

###########################################################################################
#
# Saving Analysis As New File
```

```
#
#

PName=PValueName[row]
NumberOfBolts=NumBolts[row]


filPlatsPlast="C:\HERK_our_mod_geom\Saved_Plastic_Analyses_BP/"+fmtrl+"_"+fgsk+\
"/"+str(PName)+"_"+fmtrl+"_"+fgsk+"_BP_Plast.wbpj"
Save(
FilePath=filPlatsPlast,
Overwrite=True)

# row=row+1
# (Only used when ending script under this row and only checking the geometry parameter insertions in ANSYS)

###########################################################################################
#
# Preparing For Simulation In ANSYS Workbench For Bolt Pretension Analysis
#

## Define system name

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")

## Material properties
engineeringData1 = system1.GetContainer(ComponentName="Engineering Data")
matl1 = engineeringData1.GetMaterial(Name=MtrlSupprString)
matlProp1 = matl1.GetProperty(Name="Isotropic Hardening")
matlPropData1 = matlProp1.GetPropertyData(
Name="Isotropic Hardening",
Qualifiers={"Definition": "Bilinear"})
matlPropData1.SetSuppression(Suppressed=False)

print("System Data Retrieved")

## Geometry settings

system1 = GetSystem(Name="SYS")
geometry1 = system1.GetContainer(ComponentName="Geometry")
geometry1.Refresh()

print("Geometry Data Retrieved")

## Model settings
system1 = GetSystem(Name="SYS")
model1 = system1.GetContainer(ComponentName="Model")
model1.Refresh()

print("Model Data Retrieved")

## Setup settings
setup1 = system1.GetComponent(Name="Setup")
setup1.Refresh()

##########################################################################
#
#
# Running Plastic Simulation With Stepped Up Bolt Pretension
#
#

## Running the first simulation

system1 = GetSystem(Name="SYS")
solution1 = system1.GetComponent(Name="Solution")
solution1.Refresh()
#solution1.Update(AllDependencies=True)
print("Background 0 Running")
```

```
backgroundSession0 = UpdateAllDesignPoints(DesignPoints=[designPoint0])
print("Background 0 Finished")

# Gasket Min Pressure Value Fetching

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
outputParam5 = Parameters.GetParameter(Name="P161")
GasketMinPresValue = outputParam5.Value.Value

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
outputParam6 = Parameters.GetParameter(Name="P162")
GasketMaxPresValue = outputParam6.Value.Value

## Fetch the plasticity output value for

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
outputParam = Parameters.GetParameter(Name=OutputParameter)
outputValue = outputParam.Value.Value

## Create force steps

LoadParam = Parameters.GetParameter(Name=LoadParameter)
LoadValue=LoadParam.Value.Value
forceStep20=LoadValue*0.2 # Skapar force step som är 20% av första förspänningen
forceStep10=LoadValue*0.1 # Skapar force step som är 10% av första förspänningen
forceStep5=LoadValue*0.05 # Skapar force step som är 5% av första förspänningen
forceStep1=LoadValue*0.01 # Skapar force step som är 1% av första förspänningen
FirstLoadValue=LoadValue

## Run
num=0
Run=str(num+1)
print("Run_" + str(Run) +" Saved " )

###########################################################################################
#
# Stepping Up Bolt Pretension
#
#

## Loop increase load until failure

for numLoad in range(1,MaxNumberOfLoadCases+1):
print("Criteria For loop started")

# Fetching flange angle criteria from text-file

import os
try:
directory="C:/Herkules_Results/"+str(PName)+"_"+fmtrl+"_"+fgsk+"/Run_"+str(Run)
os.makedirs(directory)
except OSError:
pass

# Opens the text file containing the flange angles

FlangeAngleFile="C:/Herkules_Results/"+str(PName)+"_"+fmtrl+"_"+fgsk+"/Run_"+str(Run)+"/Fl_ang_lc2_Pla.txt"

with open(FlangeAngleFile, "a") as f:

# Reads flange angle at flange wing

f = open(FlangeAngleFile, 'r')
line1 = f.readline()
s = line1.split()
FlangeAngleValue1 = float(s[1].strip().replace(',','.')) #value for flange angle
```

```
print("Flange Angle at Gasket")
print(FlangeAngleValue1)

# Reads flange angle at upper gasket edge     (row 2 in FlangeAngle result file)

line2 = f.readlines()
s2 = line2[0].split()
FlangeAngleValue2 = float(s2[1].strip().replace(',','.'))

print("Flange Angle at Flange")
print(FlangeAngleValue2)

f.close()

if FlangeAngleCriterion1 < FlangeAngleValue1:
print("Flange Angle Criterion For Gasket Failed")
break


elif FlangeAngleCriterion2 < FlangeAngleValue2:
print("Flange Angle Criterion For Flange Failed")
break

#      elif GasketMinPresCrit < GasketMinPresValue:
#    print("Gasket Min Pressure Criterion Failed")
# break

#      elif GasketMaxPresCrit < GasketMaxPresValue:
#    print("Gasket Max Pressure Criterion Failed")
# break

elif PlasticityCriterion > outputValue:
system1 = GetSystem(Name="SYS")
print("PlasticityCriteria - if loop started")

num=num+1

Run=str(num+1)

if outputValue < 0.005:
EMLoadValue=LoadValue
LoadValue=LoadValue+forceStep20
print("RADPlastValue "+str(outputValue)+ " < 0.005 and")
print("New BoltPretension of "+str(LoadValue)+ " will be used")
print("where initial BoltPretension was "+str(FirstLoadValue))

elif outputValue >= 0.005 and outputValue < 0.0075:
EMLoadValue=LoadValue
LoadValue=LoadValue+forceStep10

print("0.005 < RADPlastValue "+str(outputValue)+ " < 0.0075 and")
print("New BoltPretension of "+str(LoadValue)+ " will be used")
print("where initial BoltPretension was "+str(FirstLoadValue))

elif outputValue >= 0.0075 and outputValue < 0.009:
EMLoadValue=LoadValue
LoadValue=LoadValue+forceStep5

print("0.0075 < RADPlastValue "+str(outputValue)+ " < 0.009 and")
print("New BoltPretension of "+str(LoadValue)+ " will be used")
print("where initial BoltPretension was "+str(FirstLoadValue))
elif outputValue >=0.0090:
EMLoadValue=LoadValue
LoadValue=LoadValue+forceStep1

print("0.009 <= RADPlastValue "+str(outputValue)+ " and")
print("New BoltPretension of "+str(LoadValue)+ " will be used")
print("where initial BoltPretension was "+str(FirstLoadValue))

designPoint = designPoint0.Duplicate()
```

```
designPoint.CopyParameterExpressions(ToDesignPoint=designPoint0)
designPoint= Parameters.GetDesignPoint(Name=str(num))
parameter6 = Parameters.GetParameter(Name=LoadParameter)
designPoint.SetParameterExpression(
Parameter=parameter6,
Expression=str(LoadValue))

designPoint= Parameters.GetDesignPoint(Name=str(num))
parameter7 = Parameters.GetParameter(Name="P132")
designPoint.SetParameterExpression(
Parameter=parameter7,
Expression=Run)

print("Run_" + str(Run)+" starting")

print("Background "+str(numLoad)+" running")
backgroundSession1 = UpdateAllDesignPoints(DesignPoints=[designPoint])
print("Background "+str(numLoad)+" finished")
param = Parameters.GetParameter(OutputParameter)
outputValue=designPoint.GetParameterValue(param).Value
print(outputValue)

OldRunNr=num+1

else:
print("One or more of the criteria for the BP model is not fulfilled.")
print("The optimal bolt pretension is "+str(EMLoadValue))
break
# For-loop över antal lastfall slutar

## Plastic simulation finished
print("Flange model")
print(str(PName)+" "+str(Gasket)+" Plastic for Bolt Pretension")
print("Finished")

Save(Overwrite=True)
basresultat=0

######################################################################
#
# Continuation Check
#
#

    # Prints message and restarts for new flange set if the initial Bolt pretension
    # is too high

if num == 0:

print("The initial condition of the BoltPretension was too high and")
print("one or more of the criteria was not met in the first run.")
print("No Elastic or External Moment analyses are performed.")
print("Use a lower initial Boltpretension value for Flange set:")
print(str(Gasket))
print(str(PName))

row=row+1

continue

######################################################################
#
# Running Elastic Simulation For Optimal Bolt Pretension
#
#

## Save as elastic model for bolt pretension

filPlatsElast="C:\HERK_our_mod_geom\Saved_Elastic_Analyses_BP/"+fmtrl+"_"+fgsk+\
                                "/"+str(PName)+"_"+fmtrl+"_"+fgsk+"_BP_Elast.wbpj"
```

```
Save(
FilePath=filPlatsElast,
Overwrite=True)

## Copying designpoint to current
system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
designPoint1 = Parameters.GetDesignPoint(Name=str(num-1))
designPoint1.CopyParameterExpressions(ToDesignPoint=designPoint0)

## Deleting the rest of the designpoints
for numDel in range(1,num+1):
designPointDelete = Parameters.GetDesignPoint(Name=str(numDel))
designPointDelete.Delete()
print("Delete" + str(numDel))

## Preparing elastic simulation
parameter7 = Parameters.GetParameter(Name="P130")
designPoint0.SetParameterExpression(
Parameter=parameter7,
Expression="0")

## Suppressing Isotropic Hardening, making it an elastic material model
engineeringData1 = system1.GetContainer(ComponentName="Engineering Data")
matl1 = engineeringData1.GetMaterial(Name=MtrlSupprString)
matlProp1 = matl1.GetProperty(Name="Isotropic Hardening")
matlPropData1 = matlProp1.GetPropertyData(
Name="Isotropic Hardening",
Qualifiers={"Definition": "Bilinear"})
matlPropData1.SetSuppression(Suppressed=True)

## Running elastic simulation
system1 = GetSystem(Name="SYS")
solution1 = system1.GetComponent(Name="Solution")
solution1.Refresh()
#solution1.Update(AllDependencies=True)
print("Background 0 Elastic running")
backgroundSession0 = UpdateAllDesignPoints(DesignPoints=[designPoint0])
print("Background 0 Elastic finished")

print("Flange model")
print(str(PName)+" "+str(Gasket)+" Elastic")
print("Finished")
Save(Overwrite=True)

################################################################################
#
# Initiating External Moment Analyses With Differing Bolt Pretensions
#
#

## For-loop for the 4 different EM analyses

BPfac=0.8
BPfacParam = 1

for j in range(1,5):

################################################################################
#
# Setting Parameters in ANSYS Workbench For External Moment Analysis
#
#

print("Opening WB Basic Model for External moment")
Open(FilePath="C:\HERK_our_mod_geom/"+fmtrl+"_"+fgsk+"_EM.wbpj")
print("WB Basic Model for External moment Opened")

if BPfac == 0.8:
BPfacText = "A"
```

```
elif BPfac == 0.85:
BPfacText = "B"
elif BPfac == 0.9:
BPfacText = "C"
elif BPfac == 0.95:
BPfacText = "D"

## Define parameters from PValue-matrix to Workbench-paramters
it=0
# Loopar från P1 till P74 och hämtar geometrisk värde från Excel till WB-parameter
system1 = GetSystem(Name="SYS")

for i in range(1,9):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(10,37):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(40,84):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=87
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(89,94):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=97
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
```

```
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(99,104):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

# Anger antalet bultar
i=104
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValueNameEX[row]))
print(Ps)
print(str(PValueNameEX[row]))

# Anger namnet på simuleringen
i=105
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValueName[row]))
print(Ps)
print(str(PValueName[row]))

for i in range(106,119):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=125
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1


for i in range(128,139):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(140,145):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(146,151):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(152,154):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=155
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

for i in range(157,160):
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

i=163
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1
```

```
## The external moment parameter  "P170"

i=170
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PValue[row][it]))
print(Ps)
print(str(PValue[row][it]))
it=it+1

# Material Parameter for PostCommand  "P172"

i=172
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(PostCommParam))
print(Ps)
print(str(PostCommParam))

# New Parameter definition for multiple EM analyses  "P173"

i=173
Ps="P"+str(i)
designPoint0 = Parameters.GetDesignPoint(Name="0")
parameter6 = Parameters.GetParameter(Name=str(Ps))
designPoint0.SetParameterExpression(
Parameter=parameter6,
Expression= str(BPfacParam))
print(Ps)
print(str(BPfac))


## Saving Model As New File

filPlatsPlast_EM = "C:\HERK_our_mod_geom\Saved_Plastic_Analyses_EM/"+fmtrl+"_"+fgsk+\
                                    "/"+str(PName)+"_"+fmtrl+"_"+fgsk+"_EM_Plast_BP_"+BPfacText+".wbpj"

Save(
FilePath=filPlatsPlast_EM,
Overwrite=True)

####################################################################################
#
# Preparing External Moment Analyses With Differing Bolt Pretensions
#
#

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")

## Material properties
engineeringData1 = system1.GetContainer(ComponentName="Engineering Data")
matl1 = engineeringData1.GetMaterial(Name=MtrlSupprString)
matlProp1 = matl1.GetProperty(Name="Isotropic Hardening")
matlPropData1 = matlProp1.GetPropertyData(
Name="Isotropic Hardening",
Qualifiers={"Definition": "Bilinear"})
matlPropData1.SetSuppression(Suppressed=False)

print("System Data Retrieved For EM")

## Geometry settings
```

```
system1 = GetSystem(Name="SYS")
geometry1 = system1.GetContainer(ComponentName="Geometry")
geometry1.Refresh()

print("Geometry Data Retrieved For EM")

## Model settings
system1 = GetSystem(Name="SYS")
model1 = system1.GetContainer(ComponentName="Model")
model1.Refresh()

print("Model Data Retrieved For EM")

## Setup settings
setup1 = system1.GetComponent(Name="Setup")
setup1.Refresh()

##########################################################################
#
#
# Running Plastic Simulation With Stepped Up External Moment
#
#

## Sets the Bolt Pretension to 0.8*EMLoadValue

BPfacLoadValue=EMLoadValue*BPfac

system1 = GetSystem(Name="SYS")
designPoint= Parameters.GetDesignPoint(Name="0")
parameter7 = Parameters.GetParameter(Name="P106")
designPoint.SetParameterExpression(
Parameter=parameter7,
Expression=str(BPfacLoadValue))

## First external moment run

system1 = GetSystem(Name="SYS")
solution1 = system1.GetComponent(Name="Solution")
solution1.Refresh()
#solution1.Update(AllDependencies=True)     # Kör första simuleringen
print("Background 0 For EM Model Running")
backgroundSession0 = UpdateAllDesignPoints(DesignPoints=[designPoint0])
print("Background 0 For EM Model Finished")

# Gasket Min Pressure Value Fetching

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
outputParam5 = Parameters.GetParameter(Name="P161")
GasketMinPresValue = outputParam5.Value.Value

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
outputParam6 = Parameters.GetParameter(Name="P162")
GasketMaxPresValue = outputParam6.Value.Value

## Fetch the plasticity output value for

system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
outputParam = Parameters.GetParameter(Name=OutputParameter)
outputValue = outputParam.Value.Value

## Create External Moment steps

MomParam = Parameters.GetParameter(Name=MomParameter)
MomValue=MomParam.Value.Value
momStep20=MomValue*0.2
momStep10=MomValue*0.1
```

```
momStep5=MomValue*0.05
momStep1=MomValue*0.01
FirstMomValue=MomValue

## Run
num=0
Run=str(num+1)
print("Run_" + str(Run) +" for External Moment Saved " )


for numLoad in range(1,MaxNumberOfLoadCases+1):
print("Criteria For loop started For External Moment")


import os
try:
directory="C:/Herkules_Results/"+str(PName)+"_"+fmtrl+"_"+fgsk+"/Run_"+str(Run)
os.makedirs(directory)
except OSError:
pass

FlangeAngleFile="C:/Herkules_Results/"+str(PName)+"_"+fmtrl+"_"+fgsk+\
                                    "/Run_"+str(Run)+"/Fl_ang_lc3_Pla_"+BPfacText+".txt"

with open(FlangeAngleFile, "a") as f:

f = open(FlangeAngleFile, 'r')
line1 = f.readline()
s = line1.split()
FlangeAngleValue1 = float(s[1].strip().replace(',','.'))

print("Flange Angle at Gasket")
print(FlangeAngleValue1)

# Bending Stress     (row 2 in FlangeAngle result file)

line2 = f.readlines()
s2 = line2[0].split()
FlangeAngleValue2 = float(s2[1].strip().replace(',','.'))

print("Flange Angle at Flange")
print(FlangeAngleValue2)

f.close()

if FlangeAngleCriterion1 < FlangeAngleValue1:
print("Flange Angle Criterion For Gasket Failed")
break


elif FlangeAngleCriterion2 < FlangeAngleValue2:
print("Flange Angle Criterion For Flange Failed")
break


 #      elif GasketMinPresCrit < GasketMinPresValue:
 #    print("Gasket Min Pressure Criterion Failed")
 # break

 #      elif GasketMaxPresCrit < GasketMaxPresValue:
 #    print("Gasket Max Pressure Criterion Failed")
 # break


elif PlasticityCriterion >= outputValue:
system1 = GetSystem(Name="SYS")
print("PlasticityCriteria - if loop started")

num=num+1
```

```
Run=str(num+1)

if outputValue < 0.005:
MomValue=MomValue+momStep20

print("RADPlastValue "+str(outputValue)+ " < 0.005 and")
print("New External Moment of "+str(MomValue)+ " will be used")
print("where initial External Moment was "+str(FirstMomValue))

elif outputValue >= 0.005 and outputValue < 0.0075:
MomValue=MomValue+momStep10

print("0.005 < RADPlastValue "+str(outputValue)+ " < 0.0075 and")
print("New External Moment of "+str(MomValue)+ " will be used")
print("where initial External Moment was "+str(FirstMomValue))

elif outputValue >= 0.0075 and outputValue < 0.009:
MomValue=MomValue+momStep5

print("0.0075 < RADPlastValue "+str(outputValue)+ " < 0.009 and")
print("New External Moment of "+str(MomValue)+ " will be used")
print("where initial External Moment was "+str(FirstMomValue))

elif outputValue >=0.0090:
MomValue=MomValue+momStep1

print("0.009 <= RADPlastValue "+str(outputValue)+ " and")
print("New External Moment of "+str(MomValue)+ " will be used")
print("where initial External Moment was "+str(FirstMomValue))



designPoint = designPoint0.Duplicate()
designPoint.CopyParameterExpressions(ToDesignPoint=designPoint0)
designPoint= Parameters.GetDesignPoint(Name=str(num))
parameter6 = Parameters.GetParameter(Name=MomParameter)
designPoint.SetParameterExpression(
Parameter=parameter6,
Expression=str(MomValue))

designPoint= Parameters.GetDesignPoint(Name=str(num))
parameter7 = Parameters.GetParameter(Name="P132")
designPoint.SetParameterExpression(
Parameter=parameter7,
Expression=Run)

print("Run_" + str(Run)+" starting for External Moment")

print("Background "+str(numLoad)+" running")
backgroundSession1 = UpdateAllDesignPoints(DesignPoints=[designPoint])
print("Background "+str(numLoad)+" klar")
param = Parameters.GetParameter(OutputParameter)
outputValue=designPoint.GetParameterValue(param).Value
print(outputValue)

else:
print("One or more of the criteria for the EM model is not fulfilled")
break
# For-loop över antal lastfall slutar

## Plastic simulation finished
print("Flange model")
print(str(PName)+" "+str(Gasket)+" Plastic For External Moment")
print("Finished")

Save(Overwrite=True)
basresultat=0


######################################################################
```

```
#
# Running Elastic Simulation For Optimal External Moment
#
#

## Save as elastic model for bolt pretension
filPlatsElast_EM = "C:\HERK_our_mod_geom\Saved_Plastic_Analyses_EM/"+fmtrl+"_"+fgsk+\
                  "//"+str(PName)+"_"+fmtrl+"_"+fgsk+"_EM_Elast_BP_"+BPfacText+".wbpj"
Save(
FilePath=filPlatsElast_EM,
Overwrite=True)

## Copying designpoint to current
system1 = GetSystem(Name="SYS")
designPoint0 = Parameters.GetDesignPoint(Name="0")
designPoint1 = Parameters.GetDesignPoint(Name=str(num-1))
designPoint1.CopyParameterExpressions(ToDesignPoint=designPoint0)


## Deleting the rest of the designpoints
for numDel in range(1,num+1):
designPointDelete = Parameters.GetDesignPoint(Name=str(numDel))
designPointDelete.Delete()
print("Delete" + str(numDel))

## Preparing elastic simulation
parameter7 = Parameters.GetParameter(Name="P130")
designPoint0.SetParameterExpression(
Parameter=parameter7,
Expression="0")

## Suppressing Isotropic Hardening, making it an elastic material model
engineeringData1 = system1.GetContainer(ComponentName="Engineering Data")
matl1 = engineeringData1.GetMaterial(Name=MtrlSupprString)
matlProp1 = matl1.GetProperty(Name="Isotropic Hardening")
matlPropData1 = matlProp1.GetPropertyData(
Name="Isotropic Hardening",
Qualifiers={"Definition": "Bilinear"})
matlPropData1.SetSuppression(Suppressed=True)

## Running elastic simulation
system1 = GetSystem(Name="SYS")
solution1 = system1.GetComponent(Name="Solution")
solution1.Refresh()
#solution1.Update(AllDependencies=True)
print("Background 0 Elastic for External Moment running")
backgroundSession0 = UpdateAllDesignPoints(DesignPoints=[designPoint0])
print("Background 0 Elastic for External Moment finished")

print("Flange model")
print(str(PName)+" "+str(Gasket)+" Elastic For External Moment")
print("Finished")
Save(Overwrite=True)

## The Bolt Pretension factor and parameter is stepped for the next analysis

BPfac=BPfac+0.05
BPfacParam = BPfacParam+1

row=row+1

print("All analyses are finished")
```

C

Appendix 3 - APDL

C.1 APDL pre-processing description

APDL commands for loads and boundary conditions

In order to vary the loads when using the automated Python script, they needed to be assigned parameters, or arguments, such as ARG1, ARG2, and so forth. Arguments were applied in every command controlling the loads and in the following section the APDL-commands behind the load-parametrization will be explained. The APDL scripts from the load commands are presented in C.2, which might be beneficial to study while reading the explanations. The script sections are implemented as commands in the same order as they are presented in the script.

Bolt pre-tension load The initial command controlling the bolt pre-tension was very complex, but after discovering that the bolts could be inserted as one unit into the Workbench function, the command could be simplified significantly. Since the Bolt pre-tension command is the first command in the ANSYS Workbench's outline tree the APDL-command `\prep7` have to be used such that the all commands inserted in Workbench are implemented in the ANSYS pre-processor. Thereafter the mirrorplane are selected, since this plane was used when creating the original bolt pre-tension. After selecting the node in origin of the mirrorplane, using `boltN=NODE(0,0,0)`, a new force defined by ARG1 is applied to the bolt pre-tension elements defined by that node. These few rows of code have thereby inserted a new bolt pre-tension and after doing so the global coordinate system have to be reselected again with `csys,0`.

Pressure load The second command in ANSYS outline tree control the pressure applied on the internal surfaces of the flange. When updating this load, first the named selection where the pressure shall be applied is selected, using the APDL-command `cmsel,,PressureN`. These selected surfaces will be used later. But first the values of the arguments will be inserted into the load-matrix where the magnitude of the pressure is defined for the various time steps. This is done with `_loadvari2443(1,1,1)=arg1`, where the name of the load-matrix (`_loadvari2443`) was defined by ANSYS. All four arguments are inserted into the matrix. Thereafter the pressure are applied as a surface force with the command `sf,all,pres,%_loadvari2443%`.

Temperature load In a similar manner the temperature is applied, except this load is applied on the volumes of all bodies. The first APDL-command, `cmsel,,TemperatureNS`, selects the elements included in the named selection `TemperatureNS`. In order to apply the temperature load as a *Body Force load* the nodes related to the elements are selected as well. And after inserting the new argument values into the temperature load matrix the *Body Force load* are applied to the selected nodes using `bf,all,temp,%_loadvari2445%`.

External force load For the external force the implementation of the new load values was somewhat more complicated. To ensure that the correct force is applied to the flange the original force has to be removed. Initially, the named selection was selected, containing the nodes related to the surface where to the force is applied. The surface load created on this surface by ANSYS were then deleted by `sfedele,all,all,all`. Thereafter the nodes (which the force was to be applied on) were counted and the number of nodes was stores as in the scalar `nnodes` after using the command `*get,nnodes,node,,count`. The argument giving a value of the new force to be applied was then divided by the number of nodes on the surface, before being inserted into the load-matrix, `_loadvari2447y(1,1,1)=arg1/nnodes`. Thereafter the external force was applied as a nodal force with `F,all,FY,%_loadvari2447y%`.

Extra boundary conditions Extra boundary conditions needed to be added as commands in order to implement the thermal expansion correctly. This command selected the named selection containing the lower surface of the Bottom Pipe and then selected three nodes from the ones related to this surface. The three nodes that were selected were located on the outer surface of the pipe. Two of them were located on the x-axis, on symmetry plane, and the third one was located exactly in-between those, on the z-axis. The two nodes located on the x-axis were locked in z-direction, which only enabled them to deform in the x-direction. And the third node was only allowed to deform in the z-direction. The nodes present on the surface in the named selection of the surface where the force was to be applied was

C.2 APDL pre-processing scripts

```
BOLT PRE-TENSION:
---------------------------------------------------------------------------------------------------------
/prep7

csys,99 ! mirrorplane nummer
boltN=NODE(0,0,0)
f,boltN,fx,arg1

csys,0
---------------------------------------------------------------------------------------------------------

PRESSURE:
---------------------------------------------------------------------------------------------------------
cmsel,,PressureNS !Selecting the surface where the pressure is applied

_loadvari2443(1,1,1) = arg1
_loadvari2443(2,1,1) = arg2
_loadvari2443(3,1,1) = arg3
_loadvari2443(4,1,1) = arg4

sf,all,pres,%_loadvari2443%

nsel,all
esel,all
---------------------------------------------------------------------------------------------------------

TEMPERATURE:
---------------------------------------------------------------------------------------------------------
cmsel,,TemperatureNS
nsle

_loadvari2445(1,1,1) = arg1
_loadvari2445(2,1,1) = arg2
_loadvari2445(3,1,1) = arg3
_loadvari2445(4,1,1) = arg4

bf,all,temp,%_loadvari2445%

nsel,all
esel,all
---------------------------------------------------------------------------------------------------------

EXTERNAL FORCE:
---------------------------------------------------------------------------------------------------------
cmsel,,ForceMomentNS !Selecting the surface where the force and moment is applied
esln,s,1
sfedele,all,all,all
*get,nnodes,node,,count !Counting the number of selected nodes and saving the value as a scalar, nnodes

_loadvari2447y(1,1,1) = arg1/nnodes
_loadvari2447y(2,1,1) = arg2/nnodes
_loadvari2447y(3,1,1) = arg3/nnodes
_loadvari2447y(4,1,1) = arg4/nnodes

F,all,FY,%_loadvari2447y%

nsel,all
esel,all
---------------------------------------------------------------------------------------------------------

EXTRA BOUNDARY CONDITIONS:
```

```
----------------------------------------------------------------------------------------------------
cmsel,,FixtsupportNS !Selecting the first node and locking it in z-direction
nsel,r,loc,z,-0.01,0.01
nsel,r,loc,x,0,-100000
*SET,ndnr,node(-1000000,0,0)
nsel,r,node,,ndnr
d,all,uz,0

cmsel,,FixtsupportNS !Selecting the second node and locking it in x-direction
nsel,r,loc,x,-0.01,0.01
nsel,r,loc,z,0,100000
*SET,ndnr,node(0,0,1000000)
nsel,r,node,,ndnr
d,all,ux,0

cmsel,,FixtsupportNS !Selecting the last node and locking it in z-direction
nsel,r,loc,z,-0.01,0.01
nsel,r,loc,x,0,100000
*SET,ndnr,node(1000000,0,0)
nsel,r,node,,ndnr
d,all,uz,0

nsel,all
esel,all
----------------------------------------------------------------------------------------------------
```

C.3 APDL post-processing description

To be able to produce figures and result output values an APDL Command is used at the bottom of the outline-tree under *Solution* inside the ANSYS main files. This command is exactly the same for all different ANSYS bolt pre-tension main files as well as for all ANSYS external moment main files, respectively. It is very little difference between the command files for the two different main files as well. Therefore the APDL command for the external moment main model will be used, where the differences between the two scripts will be stated. The content of the following subsections will explain and refer to the marked sections inside the post-processing APDL script in Appendix C.4 being titled the same.

Input parameters First of all eight parameters are predefined in the parameter library of the ANSYS Workbench file which are fetched and used in this script, while running the Python script. It is clearly stated in the beginning of the script what each parameter controls. `ARG8` does not exist in the bolt pre-tension command since this parameter helps creating individual names for all four external moment cases. The `halfnumbo` parameter is `ARG6` divided by 2 which represents the number of bolts present in each generated ANSYS workbench model.

Number of load steps is set to 3 in the external moment model and to 2 in the bolt pre-tension model. Thereafter is a number of different if-statements which creates text strings whose appearance will depend on the values of the input parameters. This is in order to give individual names to both result files and the folder in which the result files will be put. Further down the folders are created from the file path which in turn is individually designed from parameter values and parameter defined text strings.

Plotting preparations Here the colour scheme inside ANSYS APDL is modified such that the background turns from black to white, as well as adjusting other colours to go along with that, in order to produce more suitable figures for reports and presentation overall. Then follows a macro which pre-defines the way to construct a figure. It is created by the `*create` function and will be fetched each time a figure is produced in this script by using the */input* command.

Later in this script both figures and values will be generated through for-loops. This calls for pre-dimensioning of appropriately sized matrices that will be used during this process. The matrices are being pre-dimensioned depending on number of different values to be in the matrix and the number of load steps. The first of the four is a 3-dimensional tensor where the third dimension corresponds to the number of bolts being present in a certain model. This will consist of membrane and bending stresses of the bolts and will be obtained for each bolt. The other three matrices represent values for the flange angle measured at two places, the plasticity arising at the radius and the minimum and maximum pressure of the gasket.

Creating geometry groups for flange angles In this part of the script specific nodes are defined and saved as geometry groups in order to later define the paths used when obtaining the flange angles. This is performed in a general manner by selecting predefined *NamedSelections* created in the ANSYS main models. These *NamedSelections* are groups of nodes where the unwanted nodes are unselected such that the single desired one is left. This is done four times which will make two different paths possible and each node is saved as an individual component by using the `cm` command.

Generating results for each time step Now a for-loop is stated which will generate result figures and result output data for the number load steps chosen. As can be seen in the script the do-loop has a range of 3 to the number of load steps, where the number of load steps is set to 3. So it actually only does one iteration with *i* equal to 3. The reason is that when the external moment analysis is run the only interesting load step is actually the last one. The post-process command for the bolt pre-tension model contains the same for-loop but the range of `i` is set to from 1 to 2 instead. The `set` command next is what sets the `i` to be time dependent. When `i` is equal to 1 it will make the results show for time 1 second. Finally a dummy figure is produced for the other figures to be created properly in APDL. The reason for this is unknown and merely accepted.

Generating bolt stress results for each bolt Initially the unit for angles is set to degrees. Then a parameter, `rot_ang`, is defined to be 90 degrees divided by the number of bolts for the actual flange model. Now another do-loop over the number of bolts is created. The first thing inside the loop is a new local coordinate system being created and at the same time set to be active. Relative to the global coordinate system it is displaced by 2.25 mm in the negative y-direction which places the origin in the middle of the gasket instead of being vertically aligned with upper edge of the gasket. The reason for this is that there are nodes in the modelled bolts existing in this exact horisontal plane which becomes easy to select.

For the models having Grafex or Novapress instead of Flexitallic this negative y-displacement is instead 0.75 mm, since their thickness is 1.5 mm. The local coordinate system is then rotated in the negative `rot_ang` which results in the new x-axis cutting exactly in the middle of each bolt during a whole for-loop run. The node furthest away and the one closest to the origin are then selected as nodes for each bolt. These nodes are then made to define a path through each bolt for which the membrane and bending stress can be obtained. Diagrams regarding this as well as values placed in matrix, `CB_TB_mb,` are then generated for each bolt.

Generating flange angle results The flange angle is calculated by retrieving the x-location and the relative y-displacement for both points that resembles the path. These lengths constitutes the length and height of a right angle and the angle can be calculated using the arctangent for the quotient of the height divided by length. This is done for both flange angles and put into the corresponding predefined matrix, `vink`.

Generating plasticity results There will be two figures with different views produced in order to get a clear view of the plasticity behaviour at the radius. The range of the color scheme is set from 0 to 0.01 where 0.01 corresponds to a plasticity value of 1 %. There should never be a figure generated where the plasticity has gone beyond 1 %, since the Python script should stop before that could arise. Also a plasticity plot as well as result output is generated for the whole flange. This is mostly to be able to compare the plasticity in the radius compared to anywhere in the flange and will not be used as a criterion.

Generating Gasket Pressure Results This is where the plot and results for the minimum and maximum gasket pressure are created. The Named Selection being selected corresponds to the lower surface of the gasket. In order to get the results for the actual pressure that the gasket experiences, the contact elements of this surface has to be selected. The figure is then saved and the minimum and maximum values are then placed in the predefined matrix, `minmaxpress`.

Plotting various figures A few various plots of the total deformation, the equivalent von Mises stresses and the equivalent internal stresses are performed to get a picture of what is happening to the model. The deformation plots has a slightly exaggerated deformation scale in order to give the observer a better impression of how the different loads affect the flange joint. The other two equivalent stress plots could serve as something to compare to the yield limit of the flange material. This is a section which could be developed in case of future work arising.

Fetching macros for generating text files Finally all the result values being generated through this script will be transferred into text files (.txt) by the use of macros. These macros are text files which are placed in a separate folder and fetched whenever they are to be used. Different macros are fetched depending on what ANSYS main model is run, since results are generated for time step 1 and 2 in the bolt pre-tension analyzes while results for time step 3 is generated in the external moment analyzes.

Inside these macros there is pre-defined syntax where a row of suitable names made of text strings is pre-dimensioned. The result values are then fetched and put in a row to the right of row of result names. The structure of having a row of names to the left and then a row of corresponding names to the right with at least one space between is so that Herkules is able to retrieve the values correctly.

C.4 APDL post-processing script

```
!##############################################################################################
!       POSTCOMMAND SCRIPT
!
! Author: Torkel Davidsson
! Controlled by: David Fors
! Modified by: Peter Ostrowskis
!
! Datum: 2016-05-26
!
! DESCRIPTION:  Creates result plots and tabular data for a flange model in Ansys Workbench
!
!
!
!
!
!================================= Input Parameters =================================
!
!
! ARG1 = Pressure and flange dimension (ex 1500012)
! ARG2 = 1 (Grafex), 2 (Novapress), 3 (Flexitallic)
! ARG3 = 1 (Plastic), 2 (Elastic)
! ARG4 = Boltpretension (ex 58000)
! ARG5 = Run (1)
! ARG6 = Number of bolts (ex 4)
! ARG7 = Material Number (ex 1)
!  ARG8 = Letter A,B,C to D (representing bolt pretension factor 0.8, 0.85, 0.9 to 0.95)


! For testing in APDL

!arg1=1500012
!arg2=3
!arg3=1
!arg4=58000
!arg5=1
!arg6=4
!arg7=1
!arg8=1


! Half of the number of bolts in a complete flange set
halfnumbo=arg6/2

! Number of load steps
antal_lastfall=3



*if,arg2,eq,1,then
gasket='Grafex'
*elseif,arg2,eq,2,then
gasket='Novapress'
*elseif,arg2,eq,3,then
gasket='Flexitallic'
*endif


*if,arg3,eq,1,then
fileEnd='Pla'
*else
fileEnd='Ela'
```

```
*endif


*if,arg7,eq,1,then
material='SA-105-B7'
*elseif,arg7,eq,2,then
material='SA-182-F304'
*elseif,arg7,eq,3,then
material='SA-182-F316L'
*elseif,arg7,eq,4,then
material='SA-182-F316'
*endif


*if,arg8,eq,1,then
bpfac='A'
*elseif,arg8,eq,2,then
bpfac='B'
*elseif,arg8,eq,3,then
bpfac='C'
*elseif,arg8,eq,4,then
bpfac='D'
*endif


! Creates filepath for result output

*dim,fil_plats,string,90,1
fil_plats(1,1)='C:\Herkules_Results\%arg1%.0_%material%_%gasket%\Run_%arg5%\'


! Creates folders from filepath

/syp,md ,fil_plats(1,1)
*get,jobnam,active,,jobnam



!=============================== Plotting Preparations ================================

!! reverse color

/RGB,INDEX,100,100,100, 0
/RGB,INDEX, 80, 80, 80,13
/RGB,INDEX, 60, 60, 60,14
/RGB,INDEX, 0, 0, 0,15


!! Figure generating macro

*create,bild_tmp,inp
  !/replot
/SHOW,PNG
PNGR,COMP,1,-1
PNGR,ORIENT,HORIZ
PNGR,COLOR,2
PNGR,TMOD,1
/REPLOT
/SHOW,CLOSE
/DEVICE,VECTOR,0
*get,int,active,,int
*if,int,eq,0,then
/sho,file
*endif
/rename, %jobnam%000,'png',,temp,'png'
*end


!! Predimensioning bolt stress matrix
```

```
*dim,CB_TB_mb,array,4,halfnumbo,antal_lastfall


!! Predimensioning flange angle matrix
*dim,vink,array,2,antal_lastfall

!! Predimensioning plasticity value matrix
*dim,PlasticRADRF,array,2,antal_lastfall

!! Predimensioning min/max pressure matrix
*dim,minmaxpress,array,2,antal_lastfall




!=================== Creating Geometry Groups For Flange Angles ======================


! Two defined nodes for the gasket angle

csys,0
alls
cmsel,s,gasketNS
nsle
nsel,r,loc,y,-0.1,0.1
nsel,r,loc,z,-0.1,0.1
nsel,r,loc,x,0,1000000
ndnr=node(1000000,0,0)
nsel,r,node,,ndnr
cm,gasketuP1,node


alls
cmsel,s,gasketNS
nsle
nsel,r,loc,y,-0.1,0.1
nsel,r,loc,z,-0.1,0.1
nsel,r,loc,x,0,1000000
ndnr=node(0,0,0)
nsel,r,node,,ndnr
cm,gasketuP2,node


! Two defined nodes for the flange wing angle

alls
cmsel,s,toplf
nsle
nsel,r,loc,x,0,1000000
nsel,r,loc,z,-0.1,0.1
ndnr=node(1000000,0,0)
nsel,r,node,,ndnr
cm,flangeup1,node


alls
cmsel,s,toplf
nsle
nsel,r,loc,x,0,1000000
nsel,r,loc,z,-0.1,0.1
ndnr=node(0,0,0)
nsel,r,node,,ndnr
cm,flangeup2,node



!==================== Generating Results For Each Time Step  =======================

!i=1
*do,i,3,antal_lastfall
```

```
set,,,,,i ! anger lastfall
alls

! Creates a dummy figure for the other figures to work
/AUTO,1
PLNSOL,s,eqv,0,1.0
/input,bild_tmp,inp
/DELETE, temp,'png'


!============ Generating Bolt Stress Results For Each Bolt ===================



!! Defining degrees as angle units
pi=2*asin(1)
*afun,deg
alls
set

rot_ang=90/halfnumbo

*do,l,1,halfnumbo

local,20,cart,0,-2.25,0,0,0,-rot_ang      !

! Skapar geometrigrupp (två ytterpunkter i CENTRUM av bulten)
! Creates geometry group with one node on "each side" of the bolt

esel,s,ename,,179
nsle
nsel,r,loc,y,-0.001,0.001
nsel,r,loc,z,-0,001,0.001
nsel,r,loc,x,0,1000000

ndnr=node(10000,0,0)
nsel,r,node,,ndnr

CM,outer_centre_point,NODE,1

allsel

esel,s,ename,,179
nsle
nsel,r,loc,y,-0.001,0.001
nsel,r,loc,z,-0,001,0.001
nsel,r,loc,x,0.1,1000000

ndnr=node(0,0,0)
nsel,r,node,,ndnr

CM,inner_centre_point,NODE,1
alls
cmsel,s,outer_centre_point
cmsel,a,inner_centre_point
cm,allnd1,node,

! Creating Path

PATH,BCR,2,,5
alls
cmsel,s,allnd1
ndnr=NODE(0,0,0)

PPATH,1,ndnr

ndnr=NODE(1000000,0,0)
PPATH,2,ndnr
```

```
PLSECT,s,int
/TITLE,Membrane and Membrane+Bending stresses in CENTRE of bolt nr.%l% in lc %i%, BP*%bpfac%.
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%bolt_stress_c_b_%fileEnd%_lc%i%_bolt_%l%_BP_%bpfac%,png
PRSECT

! Fills the predefined matrices with results

alls
*GET,CB_TB_mb(1,l,i), SECTION,MEMBRANE, INSIDE, s, int
*GET,CB_TB_mb(2,l,i), SECTION,BENDING, INSIDE, s, int
*GET,CB_TB_mb(3,l,i), SECTION,SUM, INSIDE, s, int
*GET,CB_TB_mb(4,l,i), SECTION,TOTAL, INSIDE, s, int


rot_ang=rot_ang+2*rot_ang

*enddo




!====================== Generating Flange Angles Results  =======================



! Angle 1 - gasket upper edge

alls
cmsel,s,gasketuP1
nodeNr=ndnext(0)
*get,rotationp1_uy,node,nodeNr,u,y
*get,rotationp1_x,node,nodeNr,loc,x
cmsel,all
cmsel,s,gasketup2
nodeNr=ndnext(0)
*get,rotationp2_uy,node,nodeNr,u,y
*get,rotationp2_x,node,nodeNr,loc,x
langd=rotationp1_x-rotationp2_x
hojd=rotationp1_uy-rotationp2_uy
vink(1,i)=atan(hojd/langd)
alls



! Angle 2 - upper flange wing lower edge

alls
cmsel,s,flangeuP1
nodeNr=ndnext(0)
*get,rotationp1_uy,node,nodeNr,u,y
*get,rotationp1_x,node,nodeNr,loc,x
cmsel,all
cmsel,s,flangeup2
nodeNr=ndnext(0)
*get,rotationp2_uy,node,nodeNr,u,y
*get,rotationp2_x,node,nodeNr,loc,x
langd=rotationp1_x-rotationp2_x
hojd=rotationp1_uy-rotationp2_uy
vink(2,i)=atan(hojd/langd)
alls



!======================= Generating Plasticity Results  =======================

! At Radius view 1

allsel,all
```

```
plesol,s,int

alls
esel,s,mat,,9,10
nsle
cmsel,r,PlasticRad
esln,r

nsle
/AUTO,1
/VIEW,1,1,1,1
/auto,1
/contour,,,0,,0.01
/replot
PLNSOL,EPPl,eqv,0,1.0
/DSCALE,1,1
!/TITLE,Dp%dp% Plastic strain on radius, load case %i%, view 1
/TITLE,Plastic strain on radius, load case %i% view 1, BP*%bpfac%
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%PlastRAD_view1_%fileEnd%_lc%i%_BP_%bpfac%,png


! At Radius view 2

allsel,all
plesol,s,int

alls
cmsel,s,PlasticRAD

alls
esel,s,mat,,9,10
nsle
cmsel,r,PlasticRad
esln,r

nsle
/AUTO,1
/VIEW,  1,-0,0,-1
/ANG,   1,0
/DIST,1,2,1
/auto,1
/contour,,,0,,0.01
/replot
PLNSOL,EPPl,eqv,0,1.0
/DSCALE,1,1
!/TITLE,Dp%dp% Plastic strain on radius, load case %i%, view 2
/TITLE,Plastic strain on radius, load case %i% view 2, BP*%bpfac%
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%PlastRAD_view2_%fileEnd%_lc%i%_BP_%bpfac%,png


! Extracting the plasticity value at radius

*get,PlasticRADRF(1,i),PLNSOL,0,max


! Plots the plasticity for whole flange view 1

allsel,all
plesol,s,int

alls
cmsel,s,TemperatureNS
esln,r

nsle
/AUTO,1
/VIEW,1,1,1,1
/auto,1
```

```
/contour,,,0,,0.01
/replot
PLNSOL,EPPl,eqv,0,1.0
/DSCALE,1,1
!/TITLE,Dp%dp% Plastic strain on raised face, load case %i%, view 1
/TITLE,Plastic strain on raised face, load case %i% view 1, BP*%bpfac%
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%PlastRF_view1_%fileEnd%_lc%i%_BP_%bpfac%,png


! Plots the plasticity for whole flange view 2

allsel,all
plesol,s,int

alls
cmsel,s,TemperatureNS
esln,r

/AUTO,1
/VIEW,  1,-0,0,-1
/ANG,   1,0
/DIST,1,2,1
/auto,1
/contour,,,0,,0.01
/replot
PLNSOL,EPPl,eqv,0,1.0
/DSCALE,1,1
!/TITLE,Dp%dp% Plastic strain on raised face, load case %i%, view 2
/TITLE,Plastic strain on raised face, load case %i%, view 2, BP*%bpfac%
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%PlastStrain2_%fileEnd%_lc%i%_BP_%bpfac%,png


! Extracting the plasticity value in whole flange

*get,PlasticRADRF(2,i),PLNSOL,0,max


!==================== Generating Gasket Pressure Results  =====================


alls
/replot
cmsel,s,BotBonded_G_RF
esln
esel,r,ename,,174
nsle
NdNr=NODE(0,-1000000,0)
*get,pos_gas,node,NdNr,loc,y
nsel,r,LOC,y,pos_gas-0.1,pos_gas+0.1
esln,,1
esel,r,ename,,174

/AUTO,1
/VIEW,1,0,1,0
/ANG,1,-180
/auto,1
/contour,,,0,,,
/replot
PLNSOL,CONT,pres,0,1.0
/DSCALE,1,1
!/TITLE,Dp%dp% Contact pressure, load case %i%
/TITLE,Contact pressure, load case %i%, BP*%bpfac%
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%Contact_pressure_%fileEnd%_lc%i%_BP_%bpfac%,png
! Extraherar max och min kontakttryck i packning
*get,minmaxpress(1,i),PLNSOL,0,min
*get,minmaxpress(2,i),PLNSOL,0,max
```

```
!======================== Plotting Various Figures  ========================


! Plotting total deformation
alls
/AUTO,1
/VIEW,  1,-0,0,-1
/ANG,   1,0
! /DIST,1,2,1
PLNSOL, U,SUM, 0,1.0
/DSCALE
!/TITLE,Dp%dp% Total deformation, load case %i%
/TITLE,Total deformation, load case %i%, BP*%bpfac%
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%Total_deformation_%fileEnd%_lc%i%_BP_%bpfac%,png

! Plotting equivalent von Mises stresses
PLNSOL, s,eqv, 0,1.0
/DSCALE,1,1
!/TITLE,Dp%dp% Von-Mises stress, load case %i%
/TITLE,Von-Mises stress, load case %i%, BP*%bpfac%
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%Von-Mises_stress_%fileEnd%_lc%i%_BP_%bpfac%,png

! Plotting equivalent internal stresses
PLNSOL, s,int, 0,1.0
/DSCALE,1,1
!/TITLE,Dp%dp% Eq. internal stresses, load case %i%
/TITLE,Eq. internal stresses load case %i%, BP*%bpfac%
/input,bild_tmp,inp
/COPY,temp,png,,%fil_plats(1,1)%int_stress_%fileEnd%_lc%i%_BP_%bpfac%,png

*enddo





!================ Using Macros For Result Output In Text-Files  ====================


! Macro filepath

*dim,mac_filepath,string,90,1
mac_filepath(1,1)='C:\HERK_our_mod_geom\_Sommarjobb_H2_PostP\Skriptfiler\Makro'



! Macro for flange angles

/cwd,mac_filepath(1,1)
*use,flange_ang_ls3_mac.mac



! Macro for plasticity

/cwd,mac_filepath(1,1)
*use,plast_RAD_RF_ls3_mac.mac



! Macro for gasket pressure

/cwd,mac_filepath(1,1)
*use,gask_minmaxP_ls3_mac.mac



! Macros for bolt stress depending on number of bolts
```

```
*if,halfnumbo,eq,2,then

/cwd,mac_filepath(1,1)
*use,nb2_lc3_mac.mac

*elseif,halfnumbo,eq,4,then

        /cwd,mac_filepath(1,1)
*use,nb4_lc3_mac.mac

*elseif,halfnumbo,eq,6,then

/cwd,mac_filepath(1,1)
*use,nb6_lc3_mac.mac

*elseif,halfnumbo,eq,8,then

        /cwd,mac_filepath(1,1)
*use,nb8_lc3_mac.mac

*elseif,halfnumbo,eq,10,then

        /cwd,mac_filepath(1,1)
*use,nbX_lc3_mac.mac

*endif
```