Automated and Generic Finite Element Analysis for Industrial Robot Design

Bhanoday Reddy
Rainel González Brioso

Linköping University Department of Management and Engineering
Machine Design Division
SE-581 83 Linköping, Sweden

LIU-IEI-TEK-A--11/01221--SE
Dated: 14/10/2011

Abstract

In a competitive market it has become necessary to deliver products in a fast and efficient way. The mechanical design process needs to be modified in a way to be able to deliver reliable and fast products. One way of modifying the process is by reducing the amount of manual work by automating the repetitive tasks using scripts built in the computer tools.

This thesis work aims to develop a methodology where tedious and repetitive design processes are automated. The design of an industrial robot lower arm will be used as a proof of concept. The modeling is done in the CAD software SolidWorks and is evaluated structurally in the FEM software ANSYS.

The FE process in ANSYS is automated by the use of the programming languages Python and JavaScript. Furthermore a user interface is created using Microsoft Excel with Visual Basic. To allow automatic FEM, a parametric CAD model of a Robot lower arm is constructed. To validate its properties, the arm's mass properties are compared with the real non-parametric model of the Industrial Robot IRB6640. It is shown that the parametric CAD model can be obtained with high accuracy. It has also been possible to validate the automated framework and the parametric design by comparing the Maximum Von Mises Stress of both arm models.

Performing a Design of Experiments it was possible to obtain the functional relation between the Mesh Element Size and the FE results. The validation tests conducted further strengthen the hypothesis that tedious and repetitive design processes can be automated in order to reduce the work load of engineers.

Acknowledgement

The master thesis work presented here is carried out in the division of Machine Design at Linköping University. We would like to express our sincere gratitude to the invaluable support and guidance given by Mehdi Tarkian, Xiaolong Feng and Johan Ölvander. The vision and research guidance by our supervisor Mr. Mehdi Tarkian has helped completing this thesis work with great satisfaction. We would like to thank Mr. Xiaolong Feng for his excellent feedback and guidance which reflected in the results of this thesis work and facilitated our understanding of industrial working principles. In addition we would like to thank our colleagues and members of the Division of Machine Design in regards to this thesis work.

Linkoping, November 2011,
Rainel González Brioso.
Bhanoday Reddy Vemula.

Abbreviation

CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAX	Computer Aided Technologies
FEM	Finite Element Method
FEA	Finite Element Analysis
FE	Finite Element
CFD	Computational Fluid Dynamics
GDO	Goal Driven Optimization
DOE	Design of Experiment
VB	Visual Basic
GUI	Graphical User Interface
XML	Extensible Markup Language
API	Application Programming Interface
PLM	Product Lifecycle Management

Table of Contents

Automated and Generic Finite Element Analysis for Industrial Robot Design i
Abstract ii
Acknowledgement iii
Abbreviation iv
Chapter 1 – Introduction 1
1.1 Background 1
1.2 Problem Statement 1
1.3 Objectives 2
1.4 Limitations 2
Chapter 2 – Theory 4
2.1 Industrial Robots 4
2.1.1 Definition 4
2.1.2 Design process of industrial robots 4
2.2 Design automation 6
2.2.1 Programming languages 7
2.3 Computer Aided Design (CAD) 8
2.3.1 Parametric modeling 8
2.4 Finite Element Method (FEM) 8
2.4.1 FEM working principle 8
2.4.2 FEM in structural analysis 9
2.4.3 FEM steps to automate 9
2.5 Mesh evaluation 12
2.5.1 Design of experiments (DOE) 12
Chapter 3 - Method 13
3.1 Parametric CAD models design 13
3.1.1 Approach 1 14
3.1.2 Approach 2 14
3.2 FEM Automation 15
3.2.1 Developing the Python Script 19
3.2.2 Developing the Java Scripts 20
3.3 Graphical User Interface (GUI) 24
3.4 Mesh evaluation 26
Chapter 4 – Results 28
4.1 Parametric CAD models validation 28
4.1.1 Approach 1 28

4.1.2 Approach 2 29
4.2 FE-model validation 31
4.2.1 FEA validation results: Approach 1 32
4.2.2 FEA validation results: Approach 2 34
4.3 FEM automation validation 35
4.4 Mesh evaluation 36
4.4.1 Mesh Evaluation for FEM simulation of the Parametric CAD model generated through approach 1 36
4.4.2 Mesh Evaluation for FEM simulation of the Parametric CAD model generated through approach 2 36
Chapter 5 – Conclusions 37
Future improvements of the presented framework could be: 37
References 39
Appendix 1 41
Appendix 2 45
Appendix 3 58
Appendix 4 60
Appendix 5 75

Chapter 1 – Introduction

To be competitive in a changing market it is necessary to deliver reliable products in the shortest period of time possible. Before the advancement of personal computers, only few institutions were able to perform Finite Element Analysis, making the design process extensive and exclusive in the automobile and aeronautic industries. Nowadays the use of this tool has become a routine in different areas of engineering, as stated by Turkiyyah & Fenves (1996) **[1]**. "FE methods have become the standard techniques for evaluating the physical performance of structural systems in various engineering applications"

In the concept realization phase of the mechanical design process, it is necessary to evaluate if the model can resist loads applied to it. In order to analyze the structure, static and dynamic calculations are performed. The time required to evaluate complicated structures is extensive, even though computational tools are utilized.

In order to perform FEA it is not sufficient to just run the simulation in any FE available software, it is more important to understand the method behind the analysis. There are certain parameters which control the accuracy of the FEA, such as: model simplification, mesh size, element type and accuracy percentage. By understanding the impact of the parameters mentioned above, it is possible to run the simulation efficiently.

The main goal of this thesis is to automate the FEA process in ANSYS Workbench and prove the advantages of having a parametric design of a model. This thesis also aims to create a generic framework which can import CAD models automatically and evaluate mesh size in order to establish fast analysis without the results being compromised.

1.1 Background

Design engineers have to perform various designs of experiments on the product and validate them in an environment which has to keep in mind the complex customer needs. The design variations and validation requires the user to continuously interact with the design tools which can be avoided by automation. According to La Rocca & van Tooren (2007) **[4]** "Process automation is a way to go in order to relieve the operators from the continuous interaction with their software tools".

Tarkian et al. **[2]** and Nezhadali **[3]** have done previous work in the area; however they mostly concentrated in the general automated design process of products. The focus of this work is how to automate the FE process without compromising the quality of the analysis. Furthermore, in this work, a dedicated and high end FE software is utilized, which requires substantial work in order to automate.

1.2 Problem Statement

During industrial robot design, engineers have to manually evaluate various tools such as CAD and CAE tools. This process takes considerable amount of time and effort. Furthermore, the process of FEM simulations such as meshing and post processing is very iterative and time consuming. In this work, an alternative way to perform FEA will be presented. The main objective of this approach is to relief engineers from time consuming and iterative work.

1.3 Objectives

After formulating the problem in the previous section (**1.2 Problem Statement***)* the following objectives are laid in order to achieve the best solutions possible.

1. **Creation of parametric structure.** To be able to analyze in a fast way, different geometries of the same model, it is necessary to be able to modify its dimension. Hence there is a need for a parametric design. However, the mass properties of this model should be similar to the real model in order to obtain similar results when performing a structural analysis.

2. **FE Process automation.** In order to dimension the structure it is necessary to know the stress distribution, this operation is time consuming and it depends on the complexity of the design. When similar models are evaluated, it is required to create a FE-model for each design, increasing the overall time of the design process. The main objective of this thesis is to automate the Finite Element Analysis.

3. **Creation of a Graphical User Interface (GUI).** A GUI is required to control the different stages of the design process, by integrating design tools. A framework which facilitates the user to modify the different parameters which have an impact in the FE-simulation has to be established.

4. **Methods comparison.** An important objective is the comparison of the automatic methodology with the manual, in order to establish the benefits from the proposed automatic methodology.

5. **Mesh evaluation.** One of the most important objectives of this thesis is to minimize the Finite Element simulation time. An optimal mesh element size has to be estimated by performing Design of Experiments which can result in the reduction of simulation time without compromising the results.

6. **Validation**
   - The parametric CAD model needs to be validated by comparing with respect to the non parametric CAD model from ABB. The mass properties are the main criteria for the comparison.

   - Results from the FEM simulations performed on the parametric and non parametric models need to be compared to validate further the parametric model.

1.4 Limitations

In engineering there always exist limitations and unknowns. Below the limitation of this thesis work is described.

- **Documentation about ANSYS Workbench scripting:** there is no real documentation available which talks about how to perform programming in ANSYS Workbench. The necessary information was obtained by the support through the ANSYS Customer Portal on Internet.

Chapter 2 – Theory

This chapter is dedicated for grouping all theories used throughout the thesis work. It starts by describing Industrial Robot which is the product selected for validating the results. Furthermore the actual tools utilized for creating the framework are defined in the order that they are implemented.

2.1 Industrial Robots

In this section the Industrial Robot and its design process is described. It is important for the reader to understand what is being tested during this work and the reason why this model was selected.

2.1.1 Definition

According to the International Organization for Standardization the definition of Industrial Robot is the following: "automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes" **[5]**. Tarkian et al. **[6]** have defined an Industrial Robot as complex mechatronic system which integrates several areas of knowledge. It has a strong dependence between geometric, dynamic performance, structural strength, functionality and cost.

Depending on the working space and application, Industrial Robots can have different configurations, e. g. Cylindrical Coordinate Robots, Scara Robots, Articulated Arm Robots, etc. The work done on this thesis work is concentrated on Articulated Arm Robots. Articulated Arm Robots are the one which uses rotary joints to access the working environment forming what is known as a Kinematic Chain.

The components of an Industrial Robots are the following: 1. Mechanical Unit which would be the one in movement; it is composed of many different mechanical parts such as: bearings, shafts, links, actuators, etc. 2. Power Supply: depending on the application it can be pneumatic, hydraulic, and electrical or a combination of the previous mentioned. 3. Control System: this component of the Industrial Robot takes care of the communication between the different devices (sensors, drivers, encoders, etc.), the sequencing on how the motor should rotate in order to produce a path and the memory function.

2.1.2 Design process of industrial robots

The design process of an industrial robot starts by identifying the requirements of the customers. The requirements mostly are: application of the robot, working space, reach, accuracy, repeatability, resolution, degree of freedom and payload. The application varies on the industry, some examples of applications are the following: painting, assembling, positioning, inspection, machining, etc. The working space is the volume where the robot can position the end effectors. Reach is defined as the longest length the robot can position the end effectors in the working space; it is measured from the first axis of rotation until the last one. The accuracy is measure by how close the robot gets to the desired position.

The repeatability of the robot is the ability to repeat certain movements and to come back to previous established positions. The working space is divided into small distances, the resolution is how divided the working space is, this will determine all the points which the robot can be positioned. One important requirement of the robot is the degree of freedom,

this will determine the working space of the robot and the possible independent movements the structure can do. The load the robot can carry is called payload.

As stated in section **2.1.1 Definition,** Industrial Robots are very complex machines which require many different analyses to obtained the optimal design. After knowing the requirements the following design process is follow:

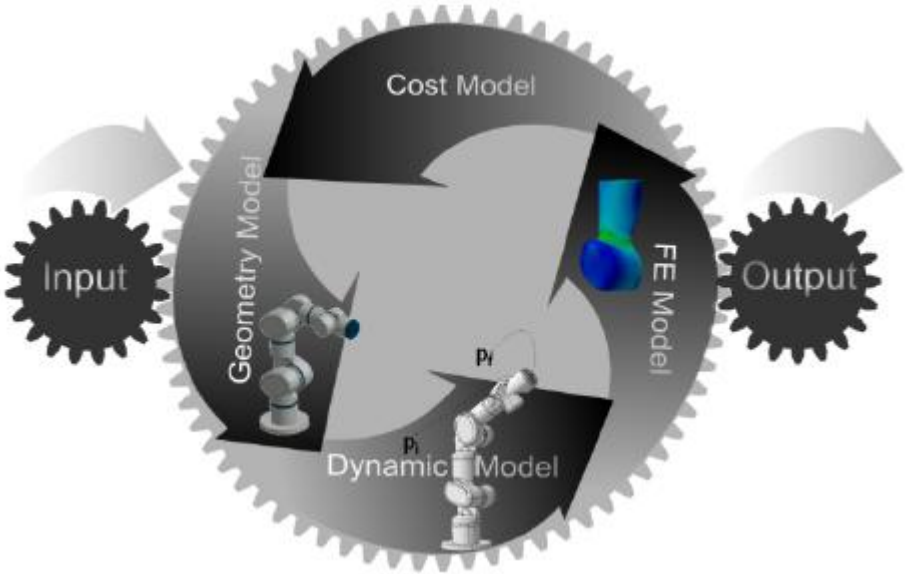


Figure 1 - Industrial Robot Design Process [6]

Error! No bookmark name given. presents an overall design process for Industrial Robots. It starts with the inputs, which are the requirements of the customers and also inputs impose by the designer to start constructing the model. As soon the input are in the process, the next step should be modeling the geometry of the model. This part of the process might require a large amount of time, because it depends on the creativity of the designer. The mass properties and the geometry of the model are very important for the later stages of the cycle.

It is very common to start the construction of the geometry by utilizing previous constructed geometries and make minor changes in order to reduce the modeling time, however if the concepts is completely new this procedure cannot be done. After the geometry is finalized the next step is to identify through dynamic simulations which actuators are suitable. Here the mass properties of the geometry are used as inputs. From the dynamic model the highest load conditions in each joint will be calculated and used as input for the FEM Model.

In the FEM model the structure is evaluated to determine if it can handle the load conditions calculated before in the dynamic model. It is important to determine how stiff the structure is. The geometry of the model is used for this block. This part of the process will determine the changes needed to be done in the structure to be able to have a stiff structure.

The final block of the inner wheel is the cost estimation; here with the weight of the structure and the price of the actuators the cost can be estimated. The main part of the overall cost come from the actuators, reason why is very important to select the most

suitable. As seen in the inner wheel of **Figure 1** the design process does not stop on the cost estimation, it continues repeating the same procedure until the termination criteria are met in each block of the design process. Finally the output is obtained and a prototype can be built to validate the outputs obtained from the design process.

2.2 Design automation

The process of using programming to perform automated design work is called design automation. To evaluate the feasibility of the product during the design process the design variables have to be experimented which require changes in the geometry and evaluation of parameters.

The design process of an Industrial Robot mentioned above (**2.1.2 Design process of industrial robots**) shows how different tools are integrated in order to create a final product. Many repetitive tasks are involved in the use of those different tools, e. g. obtaining the mass properties from the CAD software every time the geometry is changed to be use in the dynamic model. The communication between the different stages of the design process can be automated, allowing the user to concentrate more on the design concept. The main objectives of this methodology are to avoid the repetitive tasks, facilitating the interaction between the different stages of the design process and accelerating the process of creating new product concept.

Knowledge Based Engineering (KBE) is a technology which facilitates the implementation of design automation. It is based on retaining the knowledge gained from the different stages of the design process to be reuse where the information is needed. La Rocca et al. **[7]** proposed a Multi Model Generator, which uses KBE for generating automatically aircrafts geometries. La Rocca et al. **[4]** also proposed an automatic way for setting the wing geometries previously generated in order to perform different analysis where a mesh model is required, such as: structural analysis, CFD, etc.

Tarkian et al. **[6]** proposed the use of design automation methodology for the Multidisciplinary Design Optimization of Modular Industrial Robots. Through the creation of a library with different components which integrates the robots, different configurations were automatically created and evaluated in order to determine the optimal designs. The selection criteria for the best models were based on dynamic and structural analysis performance. In **Figure 2** the approach taken by Tarkian et al. in **[6]** is presented.

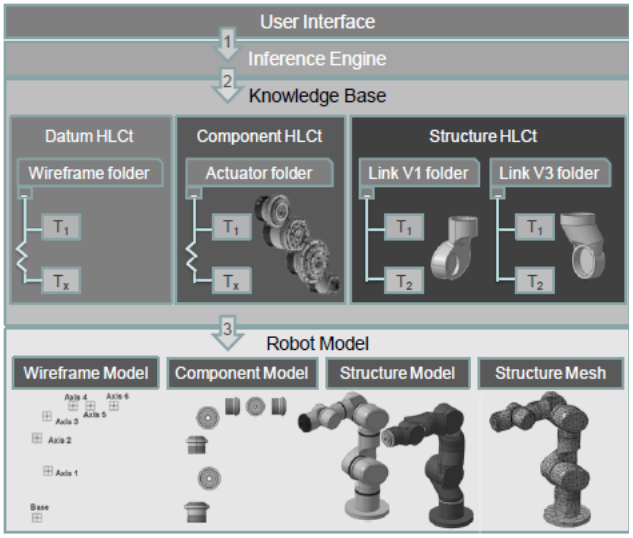


Figure 2 - Relation between components library with CAD model [6]

In this thesis work this methodology is used for integrating CAD with FEM. The users would have the opportunity of creating structural analysis for different concepts of a model in an automated way from a Graphical User Interface in Microsoft Excel. Design automation is also used to modify the parameters of the geometries of the models. Extensive researches have been conducted in this area in the Department of Machine Design at Linköping's University **[8]**.

2.2.1 Programming languages

In this current section brief descriptions are given of the different programming languages which are used for achieving design automation.

Python. This programming language is used in automating the FE process, especially in ANSYS Workbench. Python is an object oriented and interpreted language which can be used for many purposes. It has a powerful high level of data structures, dynamic, and clear and intuitive syntax. See more in **[9].**

JavaScript. JavaScript is used for the two different modules that compose ANSYS workbench for performing structural analysis, which are: Design Modeler for the importation of the geometry and Mechanical for performing the FE analysis. JavaScript as Python is an object-based scripting and interpreted language. It uses similar syntax from C and supports different data types, such as String, Number, Boolean, Object and Array. It is mainly used for web page design. See more in **[10].**

Visual Basic. For the creation of the Graphical User Interface, Microsoft Excel is used; this software offers the user the capability of creating Macros using the programming language Visual Basic. Visual Basic is a high level programming language which means Beginners' All-purpose Symbolic Instruction Code. The coding is very similar to the English language, making easy the learning of programming. It is an object oriented language mainly used for Windows and mobiles devices. See more in **[11]**.

2.3 Computer Aided Design (CAD)

The advent of CAD software's replaced the manual drawings performed by the drafters with 2D or 3D graphical representation of physical objects. In industrial design and product design CAD software's are used to model 3D (solid, Surface, Sheet metal) components or 2D drawings of physical components. These 3D models are extensively used in the engineering design process like FEM analysis to evaluate and analyze various concepts.

Advanced CAD software's have the capability to perform analysis of different aspects of the product such as FEM, Manufacturing, etc and have the features which can make it possible to integrate with other CAX software's.

2.3.1 Parametric modeling

The features and geometry's of the CAD model are assigned with specific parameters which when edited will result in the modification of the CAD model as desired by the user. This technique of modeling is very convenient in the concepts phase of product development due to agility of changing the geometry for obtaining different geometric properties of the model. According to Myung & Han (2001) **[12]** "The parametric modeling technique is useful when the geometric model should be changed frequently during the design process". Complex Non parametric CAD models may take considerable time for the user to modify the design variants which can be a drawback in the design process.

2.4 Finite Element Method (FEM)

The finite element method (FEM) is a numerical technique use for finding approximate solutions of partial differential and integral equations. The method works by assuming a continuous function for the solution and obtaining the parameters that govern this function which minimizes the error in the solution. In this section a general explanation of the working principle of this method is explained, the application of FEM for structural analysis is described briefly and finally the steps for performing structural analysis are described

2.4.1 FEM working principle

The FEM is a numerical technique which intends to find approximate solutions by expressing an unknown function in terms of know functions. The governing equations and the boundary conditions must be satisfied by the approximate solution. The use of trial function is used in order to obtain the approximate solution, the principle consist of the following steps:

- Start with an approximate solution for the unknown
- Apply a criterion in order to known if the solution is optimized
- Estimate how accurate the approximate solution is

Depending on the governing equations two methods are used in order to optimize the approximate solution:

1. Methods of Weighted Residuals (MWR) for differential equations, these methods try to minimize the error in the accuracy of the approximate solution.

2. Ritz Variational Method (RVM) for integral equations, the main objective of this method is to minimize a physical quantity, such as energy.

2.4.2 FEM in structural analysis

The FEM method is used in structural analysis to calculate the stiffness matrix which is the measure of how the body resists to deformation under certain load condition. Knowing the stiffness matrix the displacement of the body can be calculated to obtain the stress values on the model.

The method starts by calculating the E-matrix, this matrix depends on the material properties (Young's Modulus and Poisson ratio) of the model and on the type of analysis in consideration (Plane Stress or Plane Strain). After this matrix is known it is proceed to divide the model into many sub-models, also called elements. Each element is composed of nodes, the number of nodes depends on the type of element e. g. 2D Quadrature = 4 nodes, 2D Serendipity = 8 nodes, 3D Tetra10 = 10 nodes, etc. Each node has information on how the boundary conditions are affecting it.

The method would work individually in each element to later combine the results of each element. Using approximation techniques described in the previous section (**2.4.1 FEM working principle**) and the E-matrix, the stiffness matrix is obtained at each element and later assembled resulting in the global stiffness matrix which would be used to obtain the displacement at each node of the model. Knowing the displacement at each node, it is progressed to obtain the distribution of stresses in each element to finally be assembled on the same fashion as the stiffness matrix. See more in **[13]**, **[14]** and **[15]**.

When using commercial FEM tools, the algorithm explained above is not visible for the user, and are done automatically when performing the simulation. However the setting up of the simulation must be done manually. These processes are time consuming as well and could maybe be automated. In the next section, the steps to automate are being described.

2.4.3 FEM steps to automate

In this section, the steps taken to perform a structural analysis in ANSYS are explained. It is necessary to identify the tedious and time consuming steps and try to automate them to reduce the FE simulation time and to avoid the constant interaction of the user with the FE tool. Following the list of steps are presented.

1. **Geometry.** The first step to take in order to perform the analysis is to define the geometry to be evaluated. This geometry is normally done in CAD software and later imported into a dedicated FE-program.

   SolidWorks can be combined with the dedicated FE-tool ANSYS Workbench where it is possible to create the Name Selection; this mean to assign names to surfaces, lines or points from the geometry that can be used later on in the analysis. Manual importation of CAD models requires the user to browse between different folders or opening SolidWorks and pressing the Workbench button as it can be seen in **Figure 3**. This procedure requires automation to avoid spending time in the tasks explained above. **Figure 3** represent the menus in SolidWorks that are related to ANSYS.

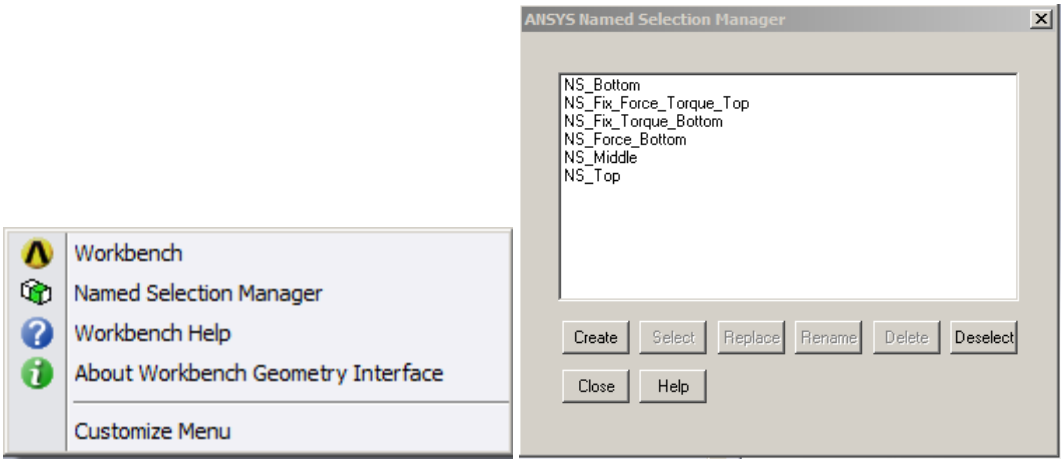
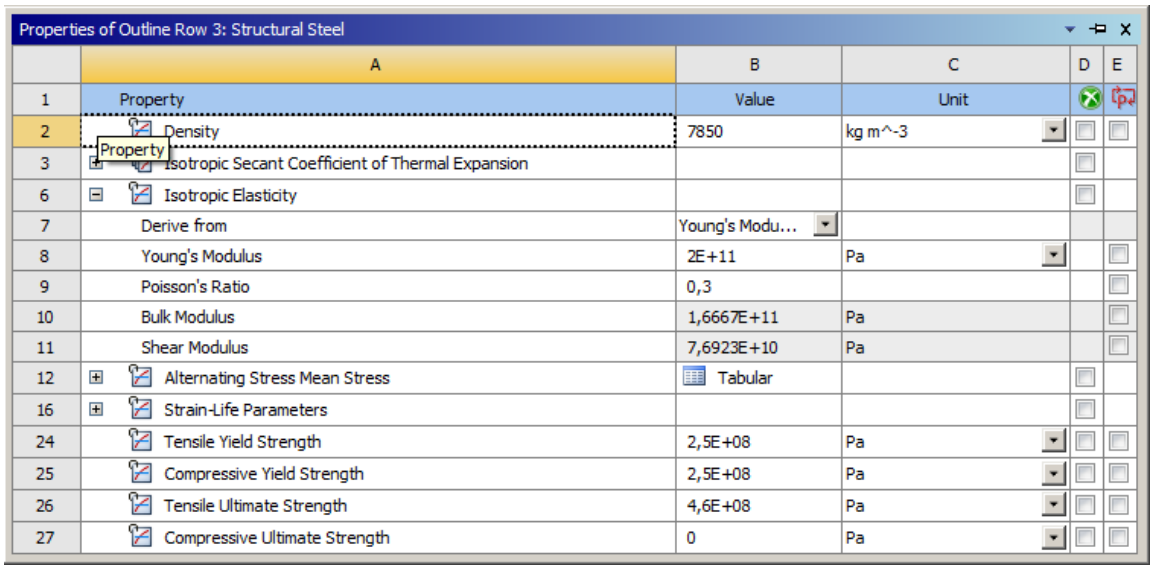


Figure 3 - ANSYS Menu and Name Selection Manager in SolidWorks

2. **Material.** After having the geometry defined, the next step is to assign a material to this geometry. Depending on the type of analysis some properties have more importance than others. For a structural analysis the Young's modulus and the Poisson ratio are the most important. The importance of automating this step is to avoid the need of manually selecting the required material from a long list located in ANSYS, especially when the user knows beforehand the name of the material. **Figure 4** shows how the material properties can be changed in ANSYS.



Properties of Outline Row 3: Structural Steel

	A	B	C	D	E
1	Property	Value	Unit		
2	Density	7850	kg m^-3		
3	Isotropic Secant Coefficient of Thermal Expansion				
6	Isotropic Elasticity				
7	Derive from	Young's Modu...			
8	Young's Modulus	2E+11	Pa		
9	Poisson's Ratio	0,3			
10	Bulk Modulus	1,6667E+11	Pa		
11	Shear Modulus	7,6923E+10	Pa		
12	Alternating Stress Mean Stress	Tabular			
16	Strain-Life Parameters				
24	Tensile Yield Strength	2,5E+08	Pa		
25	Compressive Yield Strength	2,5E+08	Pa		
26	Tensile Ultimate Strength	4,6E+08	Pa		
27	Compressive Ultimate Strength	0	Pa		

Figure 4 - Material mechanical properties

The following steps are time consuming and iterative and require constant supervision from the user, hence the reason to automate them.

3. **Meshing.** One of the most relevant steps in the Finite Element Analysis is the meshing. The speed and the accuracy of the results have a direct connection in how this part is done. The higher the numbers of nodes are the higher the accuracy of the

results, however the speed of the simulation decreases. **Figure 5** shows how the mesh looks in ANSYS Mechanical.

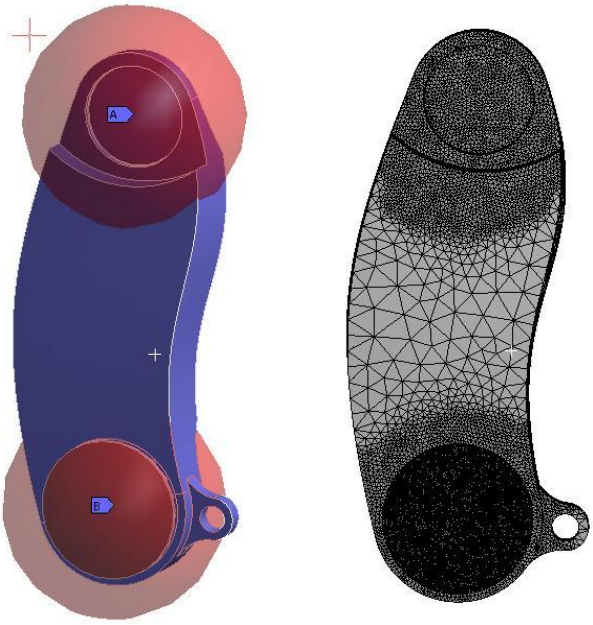


Figure 5 - Mesh procedure in ANSYS Mechanical

4. **Preprocessing.** After meshing the structure, the Boundary Conditions have to be applied in the model. For obtaining the stress the algorithm first calculates the displacements, hence the necessity to fix the model. Furthermore, after fixing the model the load conditions that influence the structure are given as inputs to the analysis. In **Figure 6** it is possible to observe how these boundary conditions are placed in the structure.

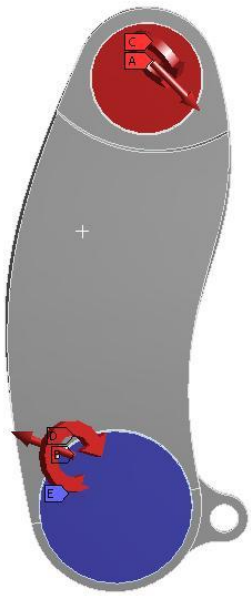


Figure 6 - Boundary conditions

5. **Postprocessing.** The final step is to run the simulations, but before it has to be specified which results are required by the user. In order to determine if the model

can resist the loads applied to it, it is necessary to know, e. g. the Maximum Von Mises stress and the displacement. Knowing these results the user can compare with the data from the material used and applying the safety factor it can be determined if the structure is stiff enough. Another use is being able to extract the results automatically for the possibility to optimize the structure.

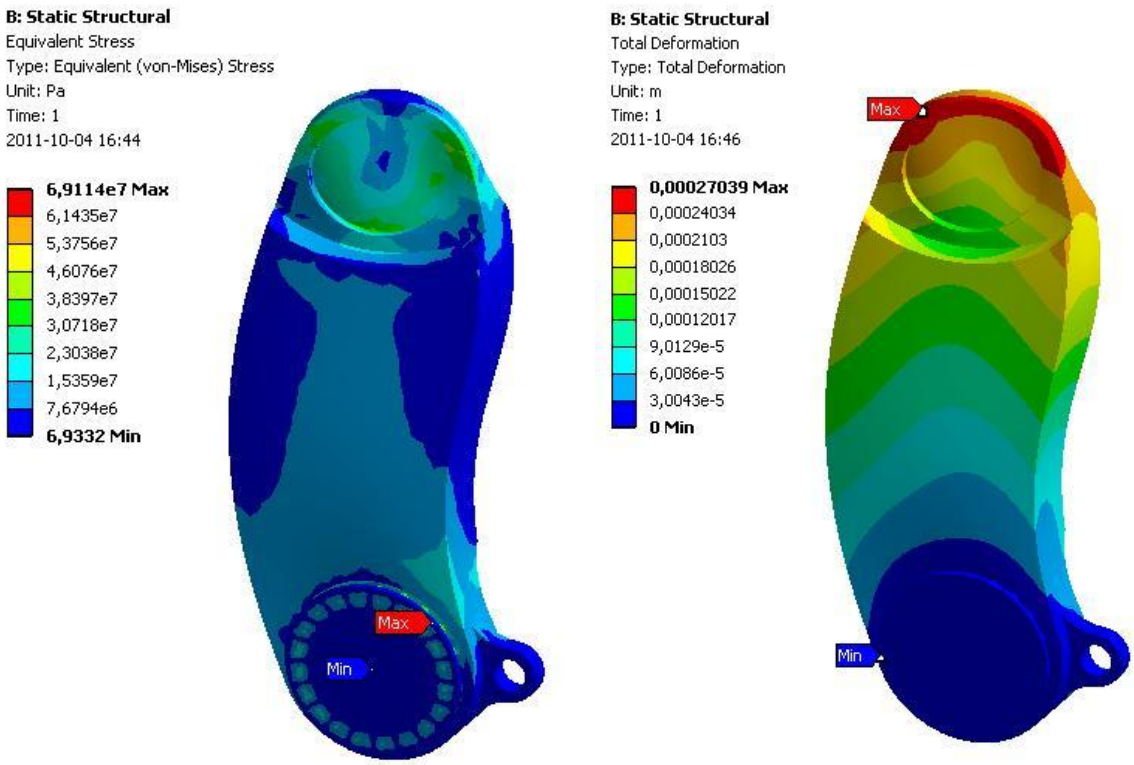


Figure 7 - Postprocessing, Von Mises Stress and Displacement

2.5 Mesh evaluation

As explained in section **2.4 Finite Element Method (FEM)**, the mesh element size is inverse proportional to simulation time and accuracy in the results. In order to obtain reliable results in a shorter simulation time an optimal mesh element size should be achieved.

2.5.1 Design of experiments (DOE)

Design of experiment is a technique that gathers design data which can be later examined through statistical analysis to determine the variation between the data points. According to Franceschini and Macchietto (2008) **[16]** "It aims at obtaining the maximum information, in a statistical sense, for use in parameter estimation and model validation".

The DOE cell is used to preview or generate and solve the DOE Matrix. The following action can be performed in the DOE cell **[17]**.

1. Select the DOE cell and the properties can be changed.

2. The input parameters can be assigned as a design variable and their lower bounds and upper bounds can be selected.

3. The values of the output variables can be accessed through DOE cell.

4. The DOE results can be viewed in the tabular and graphical forms.

Chapter 3 - Method

In this chapter, the steps taken during the analysis of this thesis in order to tackle the design problem are described. The first part of the chapter described the framework formulated which combined the different process of the design cycle. In **Figure 8** the various activities performed in this thesis is shown in a chronological order.

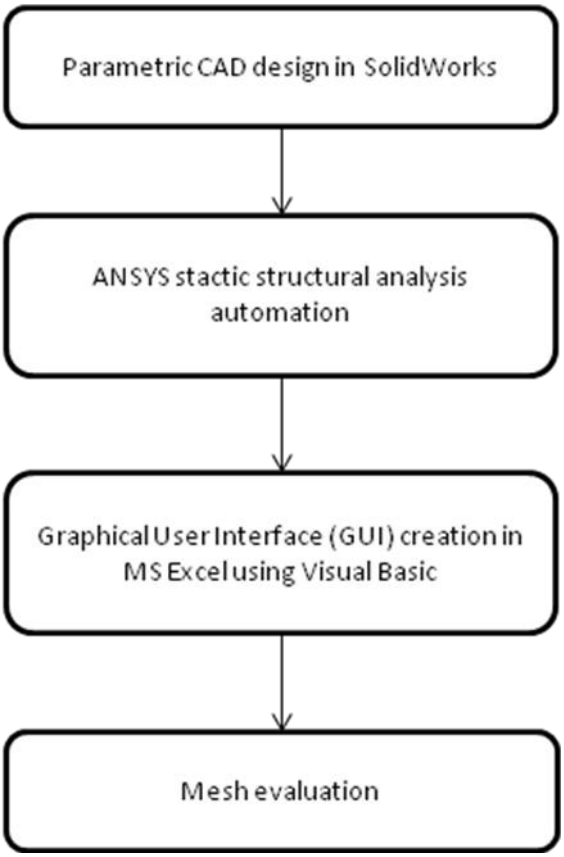


Figure 8 – Various activities performed during the thesis project

3.1 Parametric CAD models design

A generic and automated Design framework requires a parametric CAD model which can be remodeled automatically and send the CAD geometry to the FEM software. In an effort to establish the viability of the automated methodology there is a great need to validate the results as close as possible to the results obtained from the existing non automated methodology.

The results are based upon the CAD model, a high accuracy of the model increase the possibility of validating, which is the only way to check the feasibility of the project. The challenging part is making the structure parametrically which can help the user to change the design variables, such as the thickness and length. Two approaches were taken in order to construct parametric models; both described in below sub-sections.

Similar for both approaches, the model is simplified by removing some features such as screws and bolts which do not inflict high noticeable stresses to reduce the simulation time. This is done by doing some experiments using both the original model and the simplified model. The simplification of the model has given similar results in terms of the Stress values.

The Surfaces of the CAD models are named with the Named selection manager (**Appendix 1**) and the coordinate systems (**Appendix 1**) are added in a way to facilitate the automation process (**Appendix 2**).

3.1.1 Approach 1

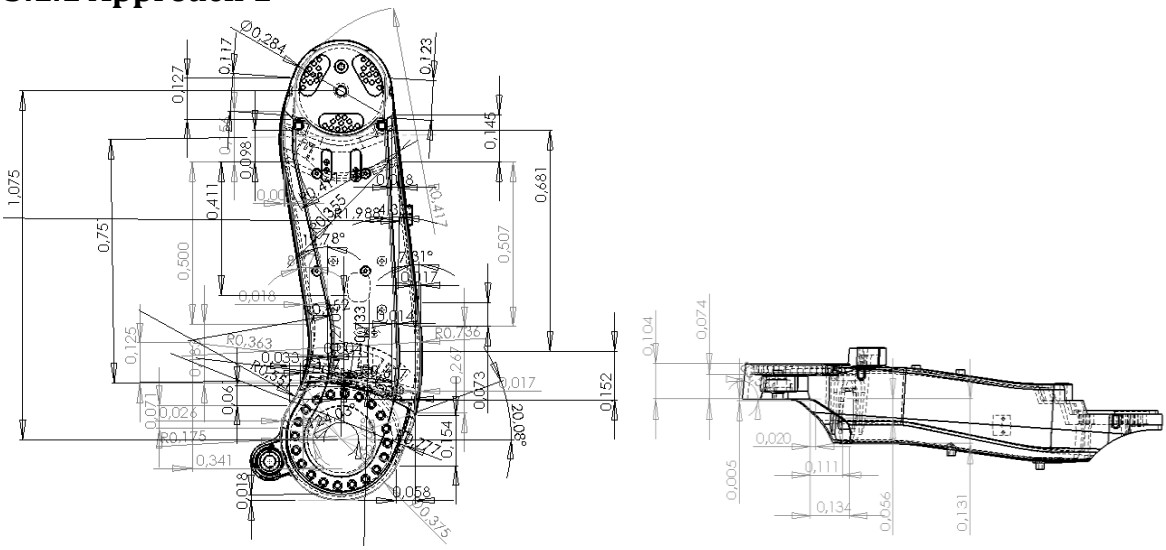


Figure 9 - 2D Drawings [18]

The main idea of this approach is to build a new parametric model as close as possible to the ABB model in terms of mass properties. So during the process of CAD modeling the mass properties are constantly reviewed and the modeling method is constantly modified until the satisfactory percentage of accuracy is achieved.

To have a good guide for creating the geometry a 2D Engineering drawing (**Figure 9**) has been done of ABB IRB 6640 lower arm **[18]**. Dimensions of the layout are plotted and with this reference the parametric CAD model in SolidWorks is built.

3.1.2 Approach 2

The second approach is done be modifying a nonparametric CAD model from ABB and assign parameters to the dimension which are needed to be controlled. These dimensions are linked to the rest of the dimensions of the model to make the entire model parametric. The parameters which are used in this CAD model are shown in **Figure 10**.

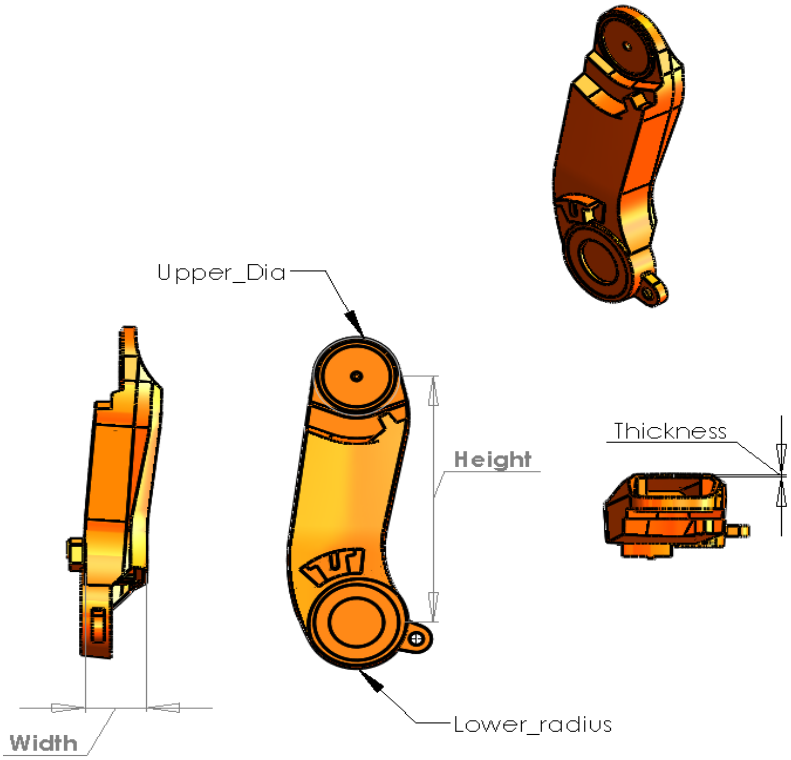


Figure 10 – Drawing with global parameters

3.2 FEM Automation

Automation of the entire simulation process in the ANSYS workbench is performed through Journaling and Scripting. As mentioned in **1.4 Limitations** there is no explicit documentation regarding the scripting in ANSYS Workbench, the following references were used to be able to accomplish the results, see **[19]**, **[20]**, **[21]**, and **[22].**

The manual simulation process in ANSYS workbench requires the user to use the graphical user interface which activates the XML string and finally the activated string calls the in-built java scripts which perform the desired action as it represented in **Figure 11**. The java scripts come in the ANSYS package. The automating task has to be accomplished by automatically calling the java scripts without the need of the ANSYS Graphical User Interface. This sets the basis for the automation in the ANSYS workbench.

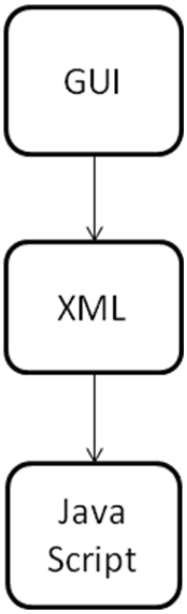


Figure 11 - ANSYS working diagram

As mentioned above the scripting in ANSYS Workbench is possible through journaling. The actions performed on the workbench can be recorded with the help of the function record journal. The action will be recorded in the form of the python script. However, the recorded journal will not record the actions performed outside the workbench. For instance when the user edits the Design Modeler and generates the geometry and/or adds parameters to control the geometry from Workbench, the action will not be recorded in the journal because those are performed outside the workbench.

It can be concluded that it is possible to record only the skeleton of the workable script through recording the journal. More work need to be done in developing the scripts for the different modules in the ANSYS workbench.

The script needs to be developed for ANSYS Design Modeler and ANSYS Mechanical in the form of macros. The macros can be run from these modules to automate the simulation process (*macro* - a script that accesses the Mechanical Application Programming Interface (API)).

Activities that are performed in each of these modules are listed below:

- **Design Modeler:** generating the geometry and assigning the parameters for the CAD geometry.

- **ANSYS Mechanical:** Meshing, Preprocessing and Post processing. All these three actions are performed in the Mechanical module through a single macro script will automate the actions performed in this module.

The macro files cannot be recorded in the mechanical or Design Modeler module. The Macro tool allows the user to execute custom functionality that is not included in a standard Mechanical application menu entry via its **Run Macro** feature. The following **Figure 12**

presents the dependency between the different components and their programming languages for a structural analysis in ANSYS Workbench.

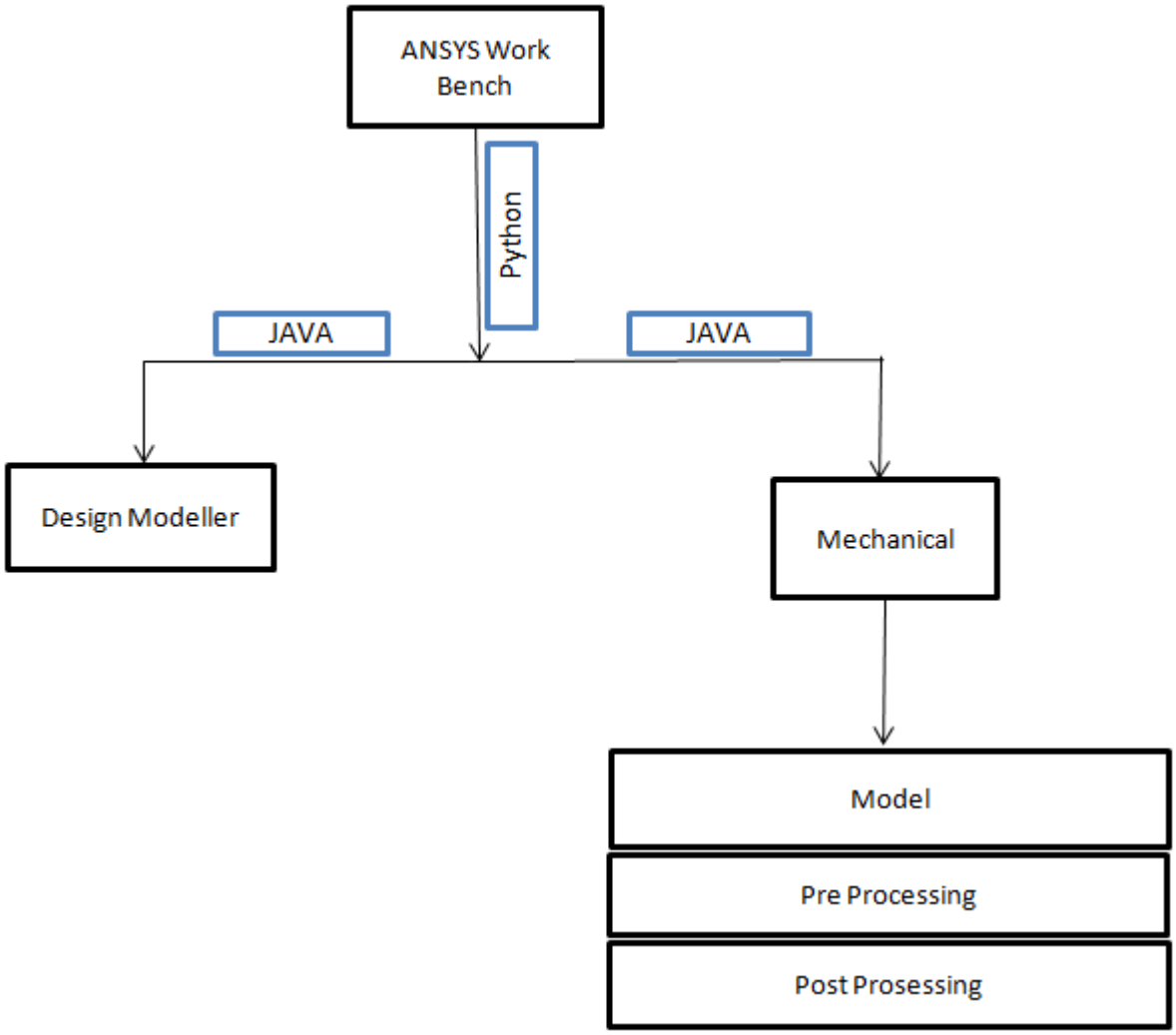


Figure 12 - Programming languages dependency chart

The next **Figure 13** shows the activities being performed by the algorithms developed with Python and JavaScript for ANSYS Workbench and its different modules which are explained in detail in the next sections. Note that the types operations differ depending if the geometry has been imported or not. If the geometry is in place then only parameters are modified and the geometry doesn't need to undergo the pre-processing stages.

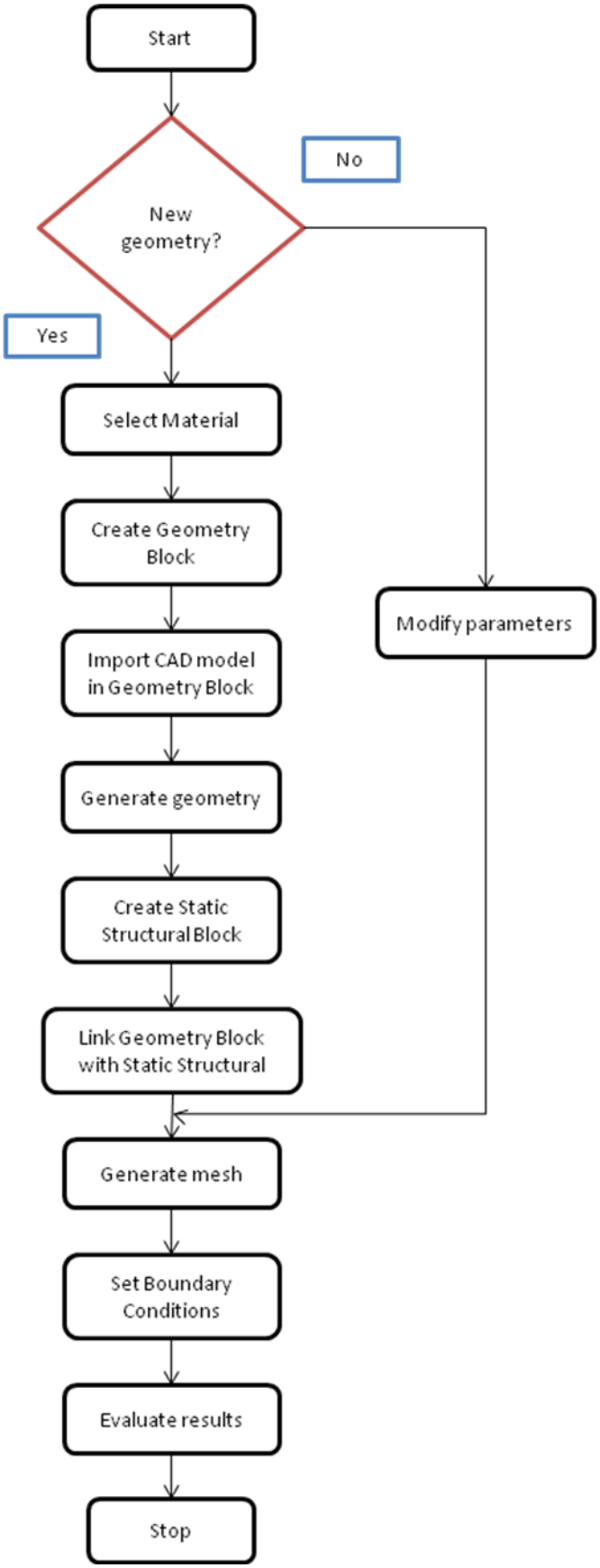


Figure 13 - FEM automation flow chart

3.2.1 Developing the Python Script

As mentioned in the previous section, Python script is used to automate the operations which occur in ANSYS Workbench. It is possible to record the activities through Journal which are based on this programming language. In **Figure 13** it can be observed the different task that the Python Script will performed. In this section, examples are being presented of how the script was written to perform the automation of the structural analysis in ANSYS.

- **Select Material.** In the following script, the material is changed from Structural Steel which is the default in ANSYS Workbench into Aluminum Alloy. The user can now change this script to choose any other material, just changing the name Aluminum Alloy to the desired material name, e. g. Carbon Steal.

```
#Accessing the Engineering data and modifying the Material
engineeringData1 = system1.GetContainer(ComponentName="Engineering Data")
matl1 = engineeringData1.GetMaterial(Name="Structural Steel")          #Default material
matl1.Delete()
favorites1 = EngData.LoadFavoriteItems()
library2 = EngData.OpenLibrary(
                Name="General Materials",
                Source="General_Materials.xml")
matl2 = engineeringData1.ImportMaterial(
                Name="Aluminum Alloy",                                 #New material
                Source="General_Materials.xml")
```

- **Geometry Block creation and Geometry importation.** To import the geometry from SolidWorks, it is required to have a place where this can be done. ANSYS Workbench has a Geometry Block where the geometry can be imported from different CAD software. The following script shows how the Block is created automatically and how the properties can be changed. At the end, the last three lines perform the geometry import procedure automatically.

```
template1 = GetTemplate(TemplateName="Geometry")                       #Geometry Block selection
                                                                       from menu
system1 = template1.CreateSystem()                                     #Block creation
geometry1 = system1.GetContainer(ComponentName="Geometry")

                                                                       #Geometry properties
geometryProperties1 = geometry1.GetGeometryProperties()
geometryProperties1.GeometryImportParameters = True
geometryProperties1.GeometryImportParametersFilter = ""
geometryProperties1.GeometryImportNamedSelections = True
geometryProperties1.GeometryImportNamedSelectionsFilter = ""
geometryProperties1.GeometryImportCoordinateSystems = True
geometryProperties1.GeometryImportSmartUpdate = True

geometry1.SetFile(
  FilePath="Z:/Framework/SW_Parts/L66_V1/L66_V1.SLDASM",               #Path from the CAD part or
                                                                       assembly
  PlugInName="SolidWorks[4605]")
```

- **Creation of Static Structural Block and linkage with Geometry Block.** The block that facilitates the structural analysis in ANSYS Workbench is called Structural Block; there is where the mesh, preprocessing and postprocessing are performed. To establish the integration between SolidWorks and ANSYS there needs to be a link between the Geometry Block and the Structural Block. The following code lines show how this procedure is done automatically.

```
template2 = GetTemplate(
                TemplateName="Static Structural",
                Solver="ANSYS")
system2 = template2.CreateSystem(                    #Block creation
  Position="Right",                                  #Positioned at right from
                                                     Geometry Block
  RelativeTo=system1)
component1 = system2.GetComponent(Name="Geometry")   #Link between blocks
component2 = system1.GetComponent(Name="Geometry")
component1.ReplaceWithShare(
  TargetSystem=system2,
  ComponentToShare=component2,
  SourceSystem=system1)
```

- **Send Command function.** To be able to run the Java Scripts from Python, there is a function which sends java commands to the different modules (Design Modeler and Mechanical). In the following scripts, it can be seen how this procedure works.

Design Modeler

```
geometry1.Edit()
geometry1.SendCommand(Command = """var Geometry_path =      #Send Command function
"Z:/Framework/DM_Script/DM_Script.js";                      #Path of the Java Script
ag.guiScript.agRunScript(Geometry_path);""")                #JavaScript function which
                                                            runs macro
geometry1.Exit()
```

Mechanical

```
model1.Edit()
model1.SendCommand(Command="WB.AppletList.Applet(\"DSApplet\").App.Script.doToolsRunMacr
o(\"Z:/Framework/ME_Script/ME_Script.js\")")
                                                            #Line with Send Command
                                                            function, JavaScript function
                                                            and with the script path
model1.Exit()
```

In order to obtain the above scripts, manually performed actions were recorded through journaling. The rest of the operations were obtained in a similar fashion, see more in details in **Appendix 2**.

3.2.2 Developing the Java Scripts

The process of debugging which is explained briefly in the **Appendix 2** allows the user to access the local commands which call in the desired Java functions situated in the Locals.

Furthermore the script can be developed through the commands from the Locals according to the requirements.

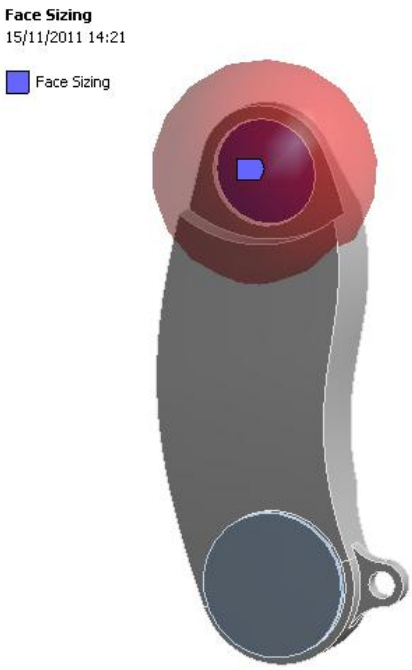
The alternative way to find the commands that can call the Java scripts directly is to search in the XML files provided in the ANSYS installation folder. The commands can be found through manual search in the file (dstoolbar.xml).

The Java scripts which are used as the macros can be run from the Python script which is written in a journal. These scripts when encoded with the journal which was described as a skeleton file can be used for automating the entire simulation process. The Python script and the Java scripts used in the thesis work are fully listed in the **Appendix 2.**

A few examples are explained below of how the Java scripts look.

- **Mesh sized sphere of influence on the top**

```
DS.Script.doInsertMeshSize(1)                          #Insert Mesh size
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "ANSYS_Persist_Key_1" ;           #Location of sphere
ListView.ActivateItem("Type");
ListView.ItemValue = "Sphere of Influence" ;           #Type of sizing
ListView.ActivateItem("Sphere Center");
ListView.ItemValue = "L66_V1_CS_Top";
ListView.ActivateItem("Sphere Radius");
ListView.ItemValue = "0,25"                            #Sphere radius
ListView.SelectedItem.IsChecked="true"                 #Select as parameter
ListView.ActivateItem("Element Size");
ListView.ItemValue = "0,01"                            #Mesh element size
ListView.SelectedItem.IsChecked="true"                 #Select as parameter
```

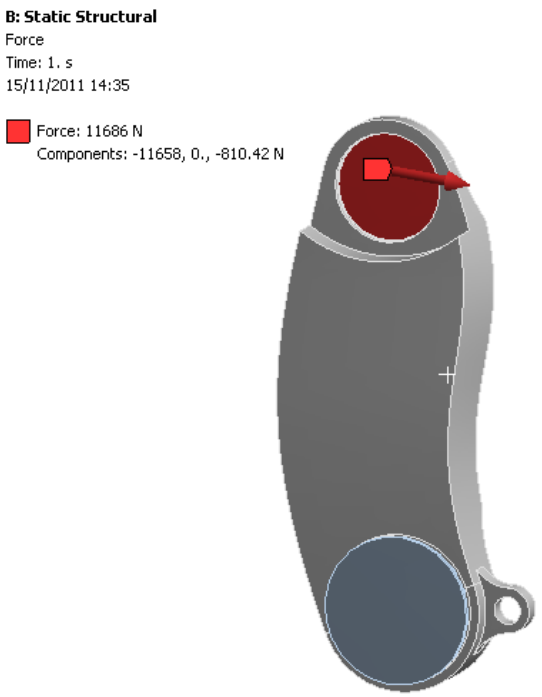


Definition	
Suppressed	No
Type	Sphere of Influence
Sphere Center	L66_V2_CS_Top
P Sphere Radius	250. mm
P Element Size	10. mm

Figure 14 - Parameters and sphere of influence on the top

- **Force on the top**

```
DS.Script.doInsertEnvironmentForce(1)                    #Insert Force
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "NS_Ftop" ;                         #Location of Force
ListView.ActivateItem("Define By");
ListView.ItemValue = "Components" ;                      #Force definition
ListView.ActivateItem("X Component");
ListView.ItemValue = "-10000"                            #Force in X
ListView.SelectedItem.IsChecked="true"                   #Select as parameter
ListView.ActivateItem("Y Component");
ListView.ItemValue = "0"                                 #Force in Y
ListView.SelectedItem.IsChecked="true"                   #Select as parameter
ListView.ActivateItem("Z Component");
ListView.ItemValue = "-810,42"                           #Force in Z
ListView.SelectedItem.IsChecked="true"                   #Select as parameter
```



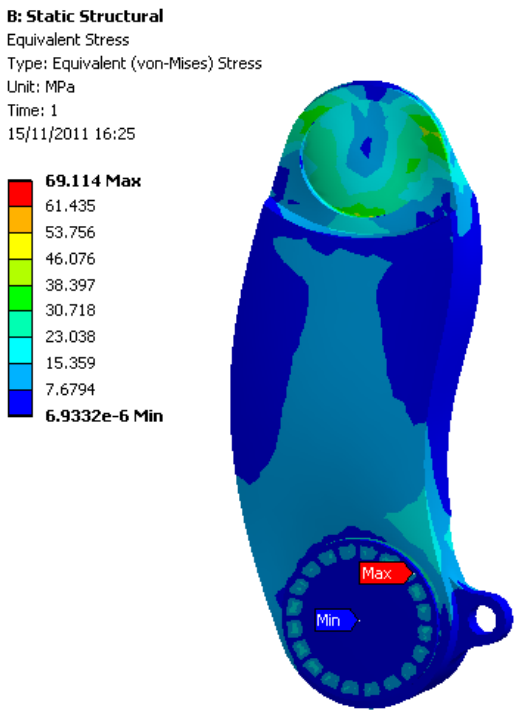
Definition

Type	Force
Define By	Components
Coordinate System	Global Coordinate System
P X Component	-11658 N (ramped)
P Y Component	0. N (ramped)
P Z Component	-810.42 N (ramped)
Suppressed	No

Figure 15 - Parameters and force on the top

- **Equivalent Von Mises Stress**

```
DS.Script.doInsertSolutionEquivalentStress(1)          #Insert Eqv Von Mises Stress
ListView.ActivateItem("Maximum");
ListView.SelectedItem.IsChecked="true"                 #Select as parameter
```



Results

Minimum	6.9332e-006 MPa
P Maximum	69.114 MPa
Minimum Occurs On	flange-1
Maximum Occurs On	flange-1

Figure 16 - Parameters and stress results

3.3 Graphical User Interface (GUI)

In the design process the communication between different tools involved needs to be precise and accurate; manual data transfer from one tool to the other might lead to data entry errors. To avoid this, a framework is created in Microsoft Excel using Visual Basic. According to Tarkian (2009) **[8]** "In order to implement design automation and optimization, a manual approach is not feasible".

The following flow chart (**Figure 17**) presents how the Graphical User Interface is used from the user point of view in order to perform an automatic simulation.

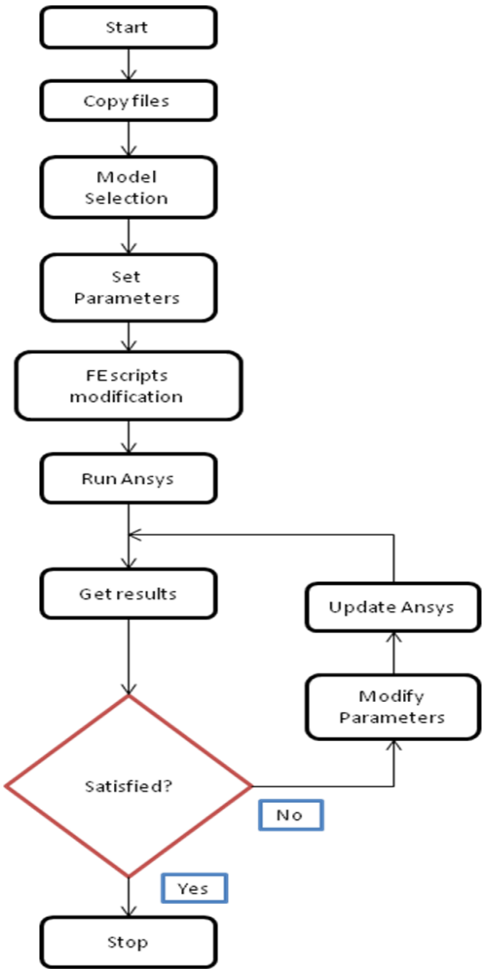


Figure 17 - MS Excel GUI flowchart

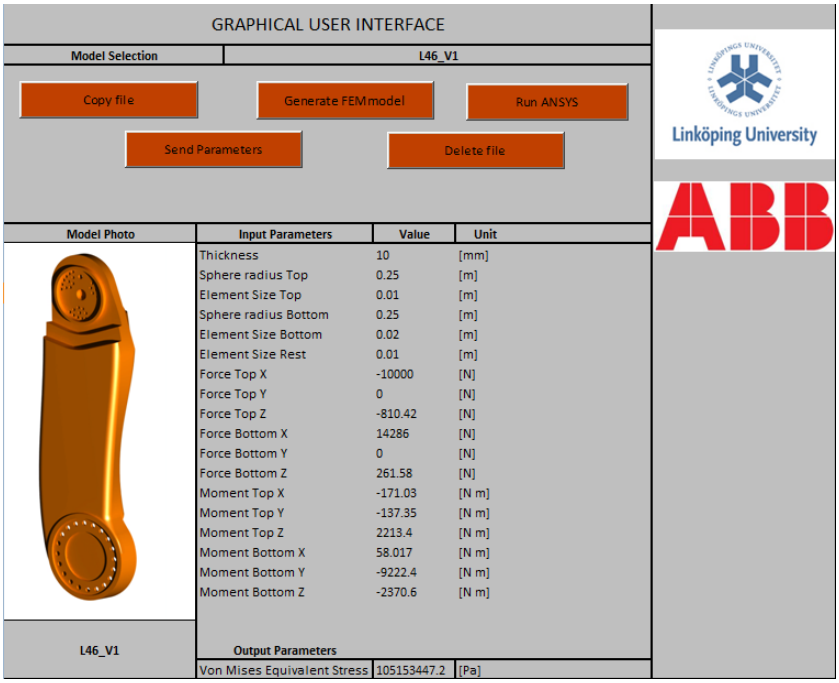


Figure 18 - MS Excel Graphical User Interface

Using Visual Basic it was possible to create the above framework. The following list explains how the above framework works.

1. **Copy files**. The first step is to press this button, the function of it is to copy and paste files needed to open ANSYS automatically and run the script which automates the FEM process explained in the theory section (**2.4.3 FEM steps**). See **Appendix 4** for more detail explanation.

2. **Model selection.** One of the requirements of design automation is to be able to perform the same tasks in different models. The GUI allows the user to select different models to analysis them automatically.

3. **Set parameters.** Knowing the model to analyze the users can now set the values of the parameters. The parameters which the users can modify are: thickness of the structure, mesh element size and the boundary conditions (forces, moments).

4. **FE scripts modification.** In order to make the framework generic, it was necessary to be able to modify automatically all the scripts involved in the automation process. The task of changing the scripts is performed by pressing the button “Generate FEM model”. The Visual Basic scripts of that button searches all the automation scripts and changes the lines where the name of the model to evaluate is written. See **Appendix 4** for more detail explanation.

5. **Run ANSYS.** After completing the four previous steps, the user can now perform the automatic simulation in ANSYS. This is done by pressing the button “Run ANSYS” in the GUI. This task is possible by calling a batch file from Visual Basic which tells the system to open ANSYS and at the same time to run the script which was copy in step 1 (Copy files). See **Appendix 4** for more detail explanation.

6. **Get results.** When ANSYS stops it returns the output parameter into the Excel framework. The following two steps are optional for the user.

7. **Modify parameters.** If the user is not satisfied with results obtained in step 6 the framework offers the users the capability of changing the parameters.

8. **Update ANSYS.** After modifying the parameters the user needs to press the button "Send Parameters" and ANSYS will open again, however this time it will open the required document which the users wish to reanalyze. ANSYS will run a script which changes the input parameters and rerun the simulation. The new results would be later updated in the Excel sheet. This process can be repeated until the user is satisfied. See **Appendix 4** for more detail explanation.

3.4 Mesh evaluation

In this part of the project, Design Exploration tool from ANSYS Workbench is used to estimate the approximate optimal mesh element size. The process can be seen in the **Figure** 19. The FEM simulation process is setup by assigning mesh size as an input parameter and Max Von Mises Stress and Total Deformation as the output parameters. The Mesh Evaluation is initiated by performing the DOE to find the functional relation between the mesh element size and the output values and further evaluated for the optimal mesh which can result in reduction of the simulation time without compromising the accuracy of the results.

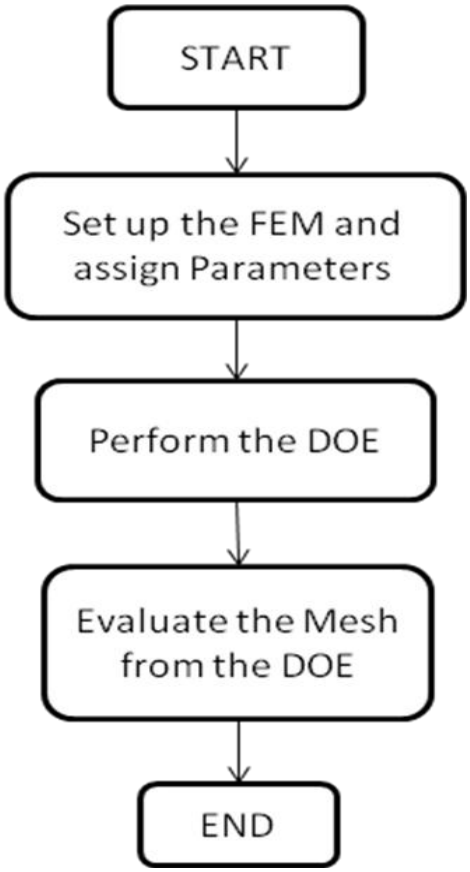


Figure 19 - Mesh evaluation flowchart

FEM Set Up

The FEM simulation is setup as explained in **2.4.3 FEM steps**. The mesh element size is assigned as an input parameter and the variables from the postprocessor are assigned as output parameters.

Design of experiments (DOE)

A custom type method is used to set up the DOE. 16 experiments are performed by giving a range of [1mm -5mm] for the input parameter and the corresponding output parameters are obtained which are further used to establish the functional relation between the input and output parameters as explained in the **2.5.1 Design of experiments (DOE)**.

Mesh Evaluation

The output values at the minimum mesh size are approximated as the accurate value. To realize the objective, which is to minimize the simulation time, an optimal mesh size can be evaluated from the DOE curve, knowing the percentage of accuracy achievable by selecting the different mesh sizes, as can be seen in Figure 20**.** Stress value obtained at 1mm mesh size is approximated as the accurate value, the accuracy of 85 % is estimated at the mesh size of 2mm, 62.5 % accurate at the mesh size of 4 mm and so on.

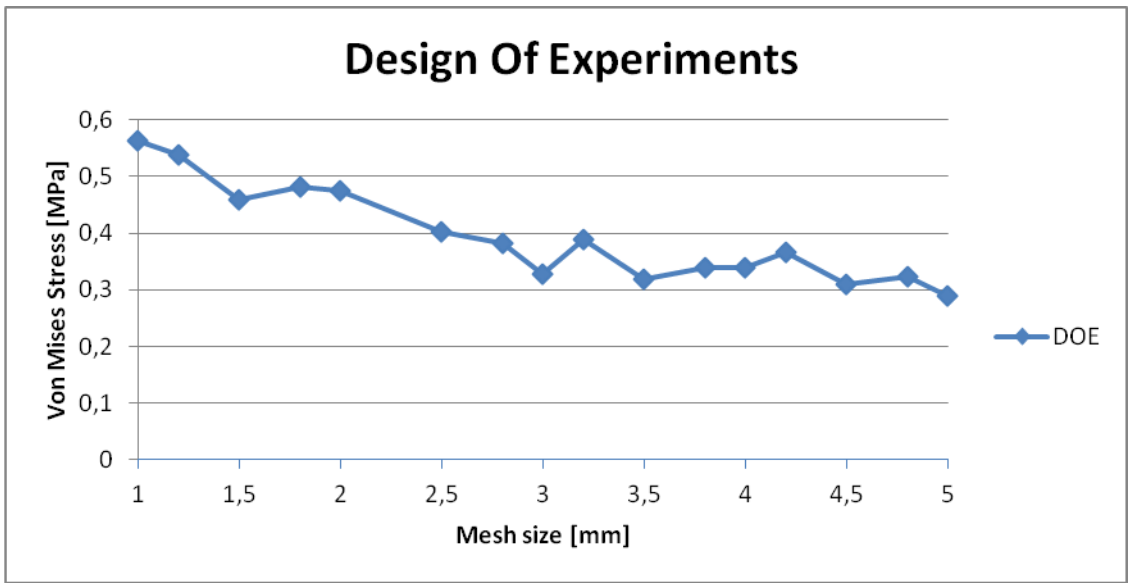


Figure 20 – DOE example

Chapter 4 – Results

This chapter is intended for presenting the results obtained after learning the theories and applying the method described in the two previous chapters. It starts with the validation of the parametric model, next with the validation of the FEM automation process, followed by the mesh evaluation and finalizing with response surfaces obtained for different load cases.

4.1 Parametric CAD models validation

In this section the results obtained using the different approaches explained above (**3.1 Parametric CAD models)** are presented.

4.1.1 Approach 1

As shown in the table an accuracy of 96% is achieved through approach 1 with respect to the ABB CAD model in terms of Mass, Volume and Surface area. However the Parametric CAD model is only 63% accurate in terms of Center of Mass and Moment of Inertia. The model can facilitate the parametric change of Height, Thickness and Width of the lower arm. The time taken to develop this approach was 2 weeks. The model from ABB was simplified in order to fairly compare both models as it can be seen in **Figure 21**.

Thickness		8 [mm]
Properties		**% Error**
Mass [kg]		2,99
Volume [m3]		0,00
Surface area [m2]		3,98
Center of Mass [m]	X	100,00
	Y	12,50
	Z	50,00
Moment of Inertia at Center of Mass [kg*m2]	Lxx	14,41
	Lyy	21,88
	Lzz	13,40
Moment of Inertia at the output coordinate system [kg*m2]	Ixx	10,20
	Ixy	52,04
	Ixz	62,50
	Iyx	52,04
	Iyy	24,67
	Iyz	86,99
	Izx	62,50
	Izy	86,99
	Izz	10,19
Average		37,07

Table 1 - Mass properties comparison approach 1

The following **Figure 21** shows the parametric model obtained using approach 1 on the right and the original model done by ABB AB on the left.

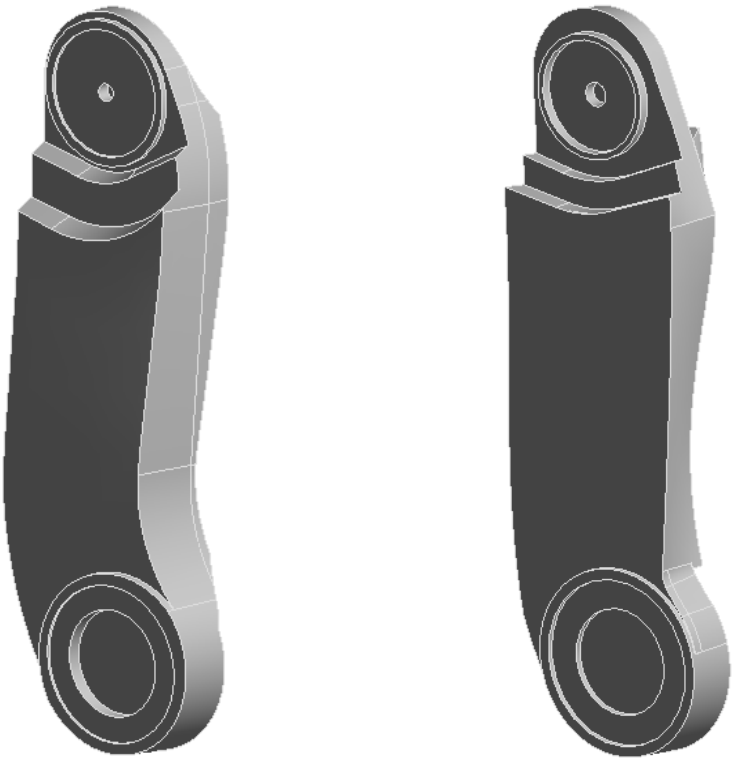


Figure 21 - To the left ABB Model and to right parametric model (Approach 1)

4.1.2 Approach 2

The parametric CAD model through approach 2 achieved more than 97% accuracy in terms of all mass properties with respect to the Non parametric CAD model from ABB after the models are simplified for FEA purpose. The model can facilitate the parametric change of Height, Thickness, Width, Lower_Dia and Upper_Dia of the Lower arm. In this approach the time of development was 3 days, since it was only modifications of the already existing CAD model from ABB. In the following Error! Reference source not found. and **Figure 22** both models are compared.

Thickness		8 [mm]
Properties		**% Error**
Mass [kg]		0.51
Volume [m3]		0
Surface area [m2]		1.06
Center of Mass [m]	X	1.81
	Y	3.81
	Z	3.55
Moment of Inertia at Center	Lxx	0.01
	Lyy	0.7

of Mass [kg*m2]	Lzz	0.06
Moment of Inertia at the output coordinate system [kg*m2]	Ixx	4.16
	Ixy	1.71
	Ixz	7.42
	Iyx	1.71
	Iyy	1.75
	Iyz	4.97
	Izx	7.42
	Izy	4.97
	Izz	4.03
Average		2.76

Table 2 - Mass properties comparison approach 2

The following **Figure 22** shows the parametric model obtained using approach 2 on the right and the original model done by ABB on the left.

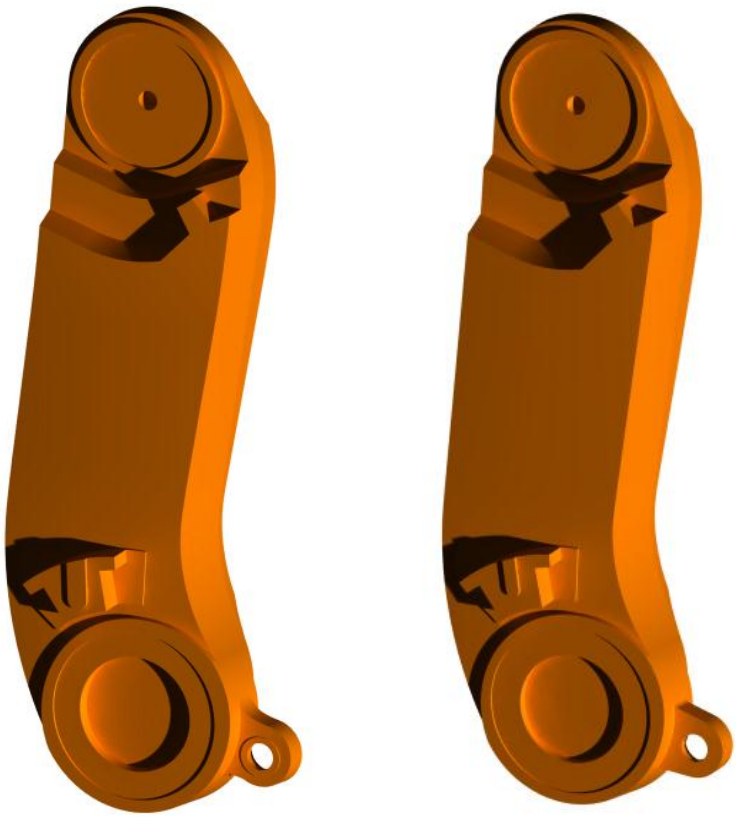


Figure 22 - To the left ABB Model and to right parametric model (Approach 2)

The next **Figure 23** presents different configurations obtained changing the parameters described in section **3.1.2 Approach 2.**

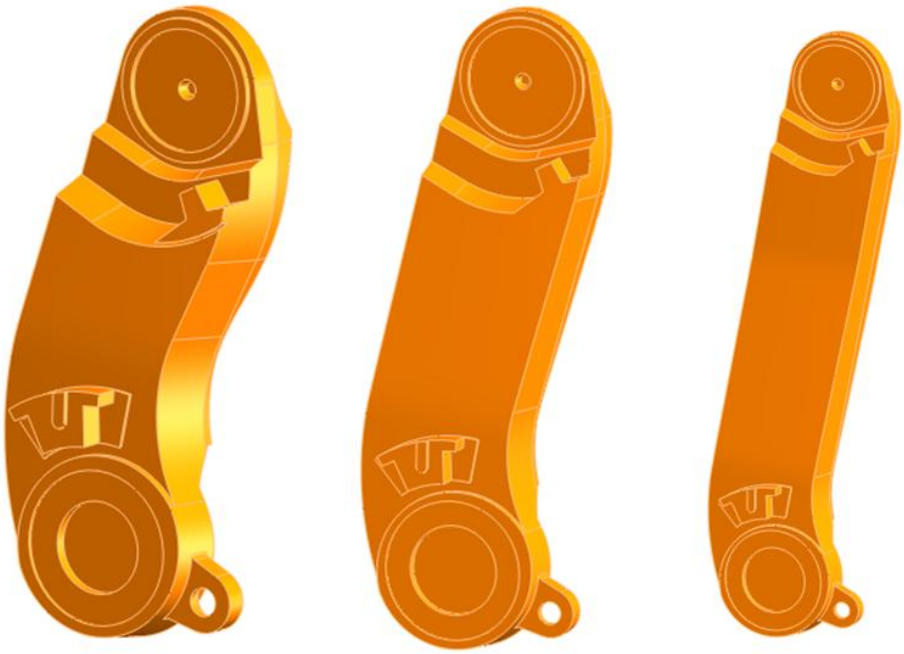


Figure 23 - Examples of configurations

Since in the parametric design details have been avoided, it is considered unnecessary to have such a level of details to perform the structural analysis.

4.2 FE-model validation

To validate our parametric models of the lower arm it is not enough to obtain comparable mass properties between both models, it is also necessary to perform FEA to estimate if they show similar behavior under the same load conditions. In the next two sections, both approaches of the parametric arm are being compared with the non-parametric ABB model. The criteria for comparing the models were the values from the Maximum Von Mises Stress and The Maximum Total Deformation when the Mesh Element Size is decreased from 5mm to 1mm.

Table 3 presents the Geometry and Boundary Conditions utilized in the comparison.

Geometry Conditions	**Value**	**Unit**
Thickness	8	[mm]
Boundary Conditions	**Value**	**Unit**
Force Top X	1000	[N]

Table 3 - Geometry and boundary conditions

Figure 24 shows an example of how the Boundary Conditions are implemented in the models.

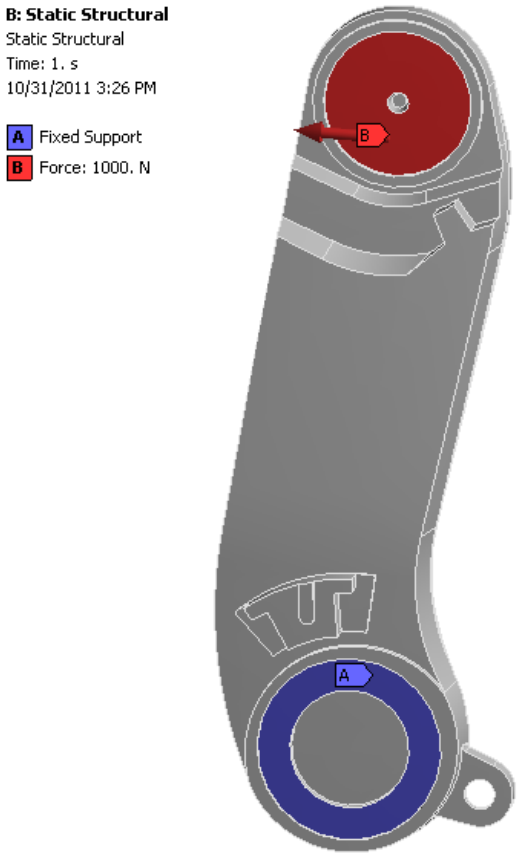


Figure 24 – Boundary Conditions

4.2.1 FEA validation results: Approach 1

In this section the non-parametric model from ABB is compared with the geometry obtained in the first approach described in section **3.1.1 Approach 1.** The following **Table 4** presents the error obtained in the parametric model in terms of the stress and deformation values at in respect to the ABB simplified model.

No.	Mesh Element Size	Stress %error	Deformation %error
1	5.00	277.66	4.94
2	4.70	260.17	4.95
3	4.50	240.76	4.91
4	4.30	270.96	4.97
5	4.00	154.94	4.97
6	3.80	199.53	4.96
7	3.50	261.26	4.92
8	3.40	219.15	4.94
9	3.20	229.71	4.98
10	3.00	283.30	4.97
11	2.90	211.48	4.97
12	2.70	204.54	4.99
13	2.50	255.69	5.01

14	2.40	235.06	5.01
15	2.30	215.78	5.00
16	2.00	202.30	5.01
17	1.80	187.48	5.01
18	1.50	189.69	4.96
19	1.30	160.60	4.93
Average		224.21	4.97

Table 4 - DOE for comparison FEA in ABB and Parametric models (Approach 1)

In **Figure 25** it can be observed how much both models differs in terms of the Maximum Von Mises Stress, being the average error of 224.21%, see **Table 4** above.

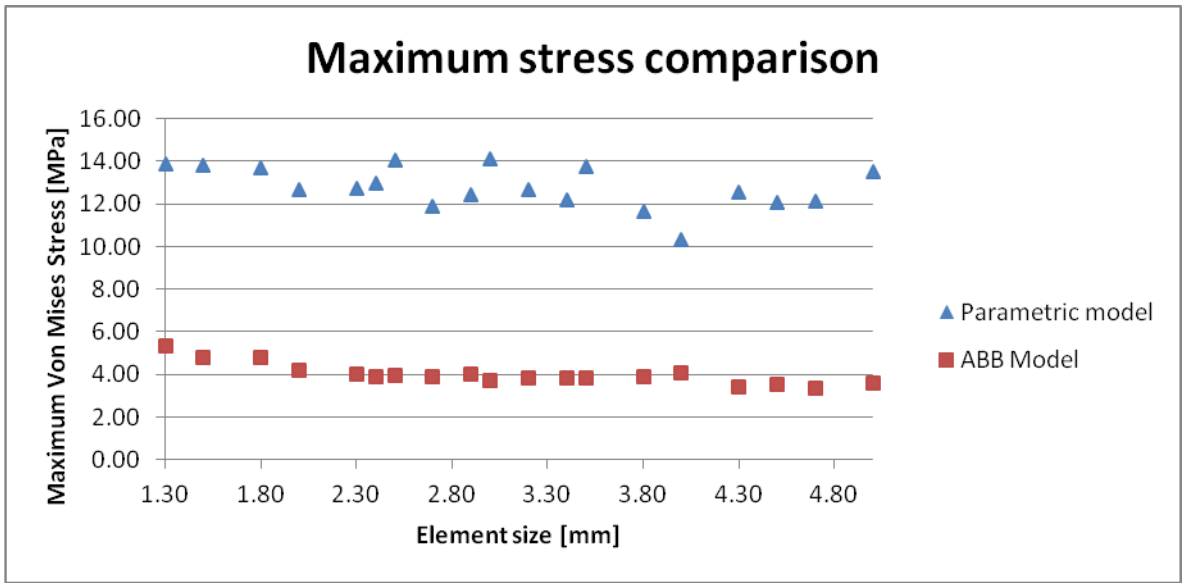


Figure 25 - Maximum Stress Comparison (Approach 1)

On the other hand the Total Deformation values are very close to each other with a average error of 4.97% (**Table 4**). **Figure 26** shows how the Total Displacement varies in both models.

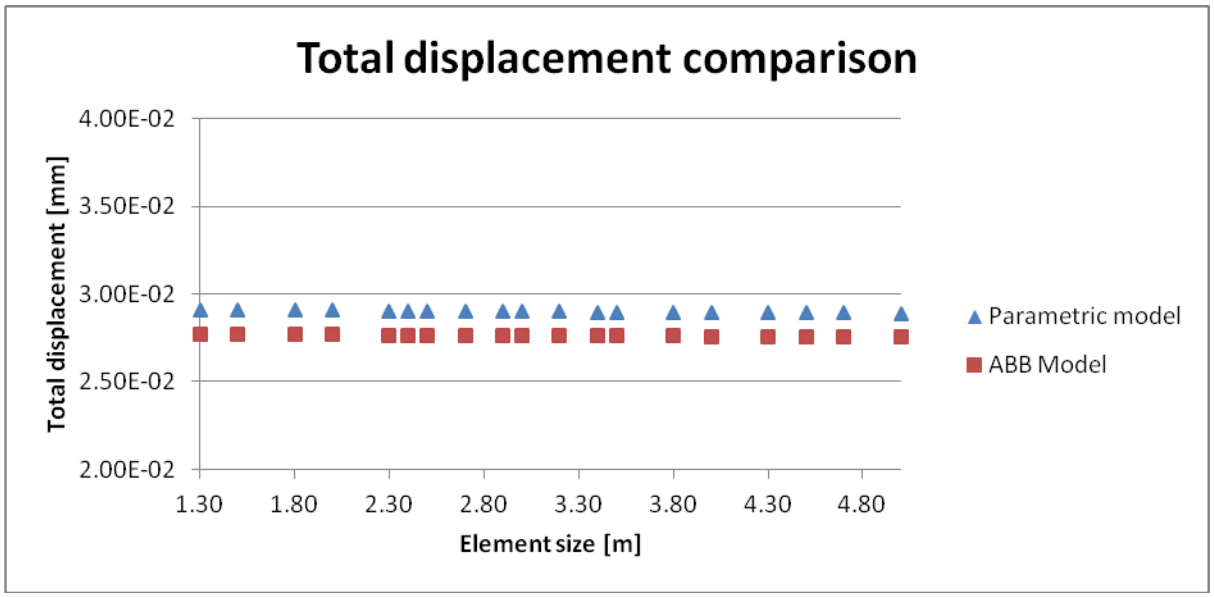


Figure 26 - Total displacement comparison (Approach 1)

4.2.2 FEA validation results: Approach 2

For this section the non-parametric model from ABB is compared with the geometry obtained in the second approach described in section **3.1.2 Approach 2.** The following **Table 5** shows the percentage error obtained from comparing the values from the Maximum Von Mises Stress and the Maximum Total Deformation in both models.

No.	Mesh Element Size	Stress %error	Deformation %error
1	5.00	1.02	0.28
2	4.70	0.50	0.30
3	4.50	3.32	0.30
4	4.30	11.28	0.30
5	4.00	5.94	0.27
6	3.80	3.64	0.29
7	3.50	0.25	0.29
8	3.40	3.14	0.29
9	3.20	2.65	0.29
10	3.00	2.06	0.26
11	2.90	9.31	0.25
12	2.70	8.80	0.27
13	2.50	10.32	0.25
14	2.40	8.61	0.25
15	2.30	5.35	0.24
16	2.00	5.82	0.26
17	1.80	13.32	0.26
18	1.50	5.00	0.26
19	1.30	6.19	0.25
Average		5.61	0.27

Table 5 - DOE for comparison FEA in ABB and Parametric models (Approach 2)

The following **Figure 27** and **Figure 28** represent **Table 5** results clearer.

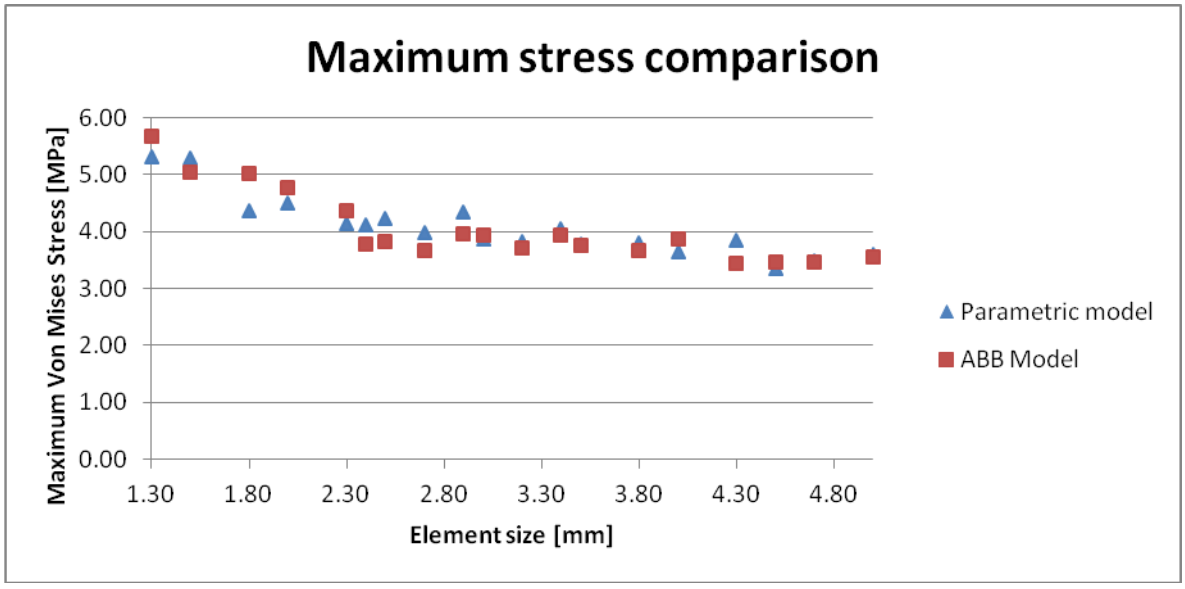


Figure 27 - Maximum Stress Comparison (Approach 2)

It can be viewed in **Figure 27** the similarities in both models with respect to the stress values. The average difference was 5.61% as showed in **Table 5**.

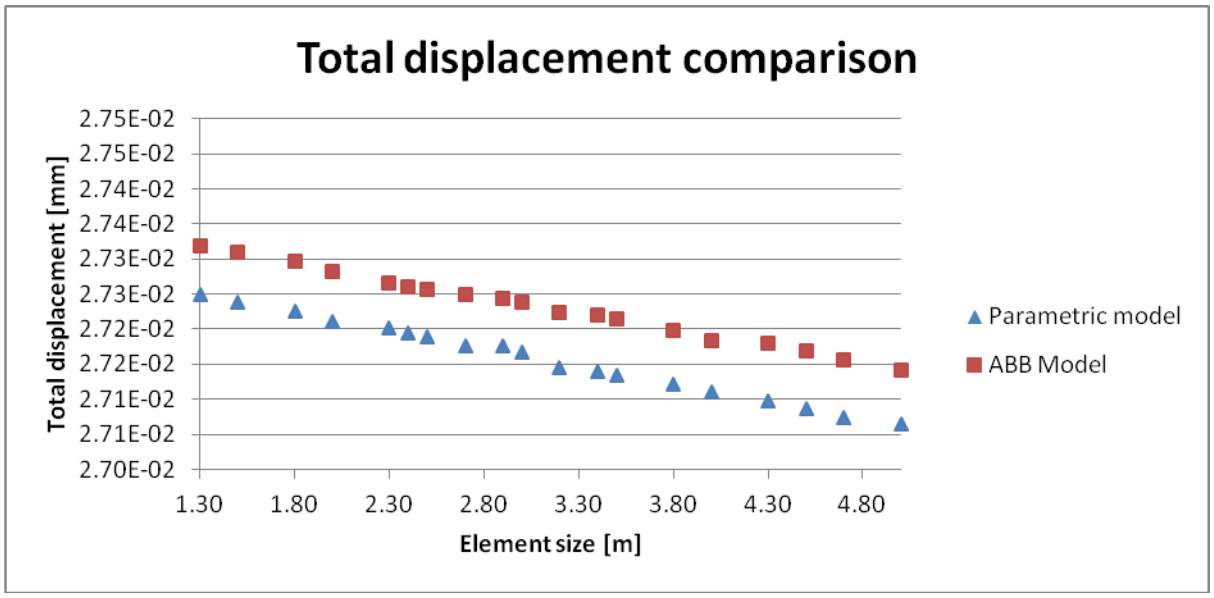


Figure 28 - Total displacement comparison (Approach 2)

In **Figure 28** it can be observed the difference in the displacements values remain constant, the average was 0.27% as it can be seen in **Table 5**.

4.3 FEM automation validation

To validate the automation process described in the previous chapter, a comparison between automatic and manual simulations were performed. The simulations were done in different computer systems, in order to demonstrate the dependency between RAM capacity and Processors velocity with simulation time. The following **Table 6** shows the results obtained on this experiment.

System	RAM Capacity	Processors Velocity	Simulation Type	Simulation Time [s]
1	2 GB 800 MHz	2 X 2.2 GHz	Manual	510
			Automatic	240
2	16 GB 1333 MHz	8 X 3.2 GHz	Manual	360
			Automatic	144

Table 6 - Manual and automatic simulation comparison

As it can be analyzed in **Table 6** the automatic simulation in the first system is around 2.13 faster than the manual simulation. In the second system however, this ratio increase to 2.5. If both systems are compared the second one is ~1.42 faster with respect to manual simulations and ~1.67 faster for automatic simulation. Besides the reduction of time, the second system offers the user the capability of selecting a smaller Mesh Element Size.

4.4 Mesh evaluation

As explained in the **3.4 Mesh evaluation,** the mesh evaluation is performed in the FEA of the two parametric arms and are briefed in this section.

4.4.1 Mesh Evaluation for FEM simulation of the Parametric CAD model generated through approach 1.

The plot obtained from the DOE conducted can be seen in **Figure 25.** Stress values obtained at least mesh size which is 1.30 mm is approximated as the accurate value. The user can estimate an accuracy of 98 % at 2 mm mesh size, 85% accuracy at 3 mm mesh size, 71 % accuracy at 4 mm mesh size and so on.

4.4.2 Mesh Evaluation for FEM simulation of the Parametric CAD model generated through approach 2.

The plot obtained from the DOE conducted can be seen in **Figure 27**. Stress values obtained at least mesh size which is 1.30 mm is approximated as the accurate value. The user can estimate an accuracy of 86 % at 2 mm mesh size, 73% accuracy at 3 mm mesh size, 67 % accuracy at 4 mm mesh size and so on.

Chapter 5 – Conclusions

The most important objective of this Master Thesis is to try to minimize repetitive work at industry performing automatic FEA. To automate the FEA process in ANSYS was not straight forward since the documentation about automation with Python and JavaScript was thin.

After establishing how to automate the process, it was necessary to create a generic framework, where new arm models could be analyzed automatically. By using Microsoft Excel and Visual Basic, it was possible to create this framework. Changes on an ANSYS Workbench project file previously created were also possible through the automatic framework.

Two parametric CAD models of an industrial robot lower arm are constructed. These two are created by using two different modeling approaches and they are validated with the original model from ABB. The models are compared using the automated framework and it can be concluded that the mass properties are sufficiently close to each other.
The parametric model obtained with approach 1 did not give comparative Von Mises Stress results with the original CAD model from ABB. Approach 2 on the other hand has given satisfactory results in the comparison. This test further strengthens the hypothesis that parametric CAD models together with automated FEA are an efficient alternative to non-parametric CAD and manual FE processes.

Mesh evaluations have been performed in both the parametric CAD models and the functional relation between Mesh Element Size and the outputs has been estimated. Using this estimation it is possible to choose between the simulation time and the desired percentage of accuracy in the results.

Future improvements of the presented framework could be:

- Creation of Named Selections automatically in ANSYS Workbench was not possible. These were done manually in SolidWorks.
- Editing the parameters through the automated framework was not possible without reopening the ANSYS Workbench file which was currently opened. The working file needed to be saved before closing and reopened through the framework to edit the parameters for the simulation.
- Some of the files need to be saved in the ANSYS Installation folder for the functioning of the framework. It would be recommended to save these files in another user defined folder.
- Check the feasibility of automating in other commercial FE software such as ABACUS and NASTRAN and compare the merits and demerits with automated framework involving ANSYS.
- Integration of other CAD software for instance CATIA, Unigraphics and Pro E with ANSYS.
- Perform multi analysis systems (mode analysis, fatigue analysis, etc.) using the same geometry and Mesh model in the automated generic framework.

Possible future applications based on the presented framework could be FEM analysis on a parametric conceptual aircraft design **[23]**. CFD analysis and structural analysis can be performed in ANSYS through this framework.

Moreover, the presented framework could, with some modifications, be directly applied to products such as wind turbines. For instance the blades of a wind turbine can be parameterized in terms of thickness and length and used to perform FEM analysis.

References

[1] Turkiyyah, G. M. & Fenves, S. J. 1996. “Knowledge-based assistance for finite-element modeling”, Intelligent Systems, vol. 11, no. 3, pp. 23–32.

[2] Tarkian, M., Ölvander, J., Feng, X. & Pettersson, M. 2011, “Design Automation of Modular Industrial Robots”, paper presented to ASME - IDETC/CIE 2009, San Diego, California, USA, August 30th to September 1st.

[3] Nezhadali, V. 2011. “Multi-objective optimization of industrial robots”, Thesis No. 1112, Department of Management and Engineering, Linköping University.

[4] La Rocca, G. & Van Tooren, M.J.L. 2007, "A knowledge based engineering approach to support automatic generation of FE models in aircraft design", Collection of Technical Papers - 45th AIAA Aerospace Sciences Meeting, pp. 11724.

[5] ISO Standard 8373:1994, Manipulating Industrial Robots – Vocabulary

[6] Tarkian, M., Persson, J., Ölvander, J. & Feng, X. 2011, “Multidisciplinary Design Optimization of Modular Industrial Robots”, paper presented to ASME - IDETC/CIE 2011, Washington, DC, USA, 28th – 31st August.

[7] La Rocca, G., van Tooren, M.J.L. 2009, “Knowledge-Based Engineering Approach to Support Aircraft Multidisciplinary Design and Optimization”, Journal of Aircraft, Vol. 46, No. 6, pp. 1875-1885.

[8] Tarkian, M. 2009, “Design Reuse and Automation: On High Level CAD Modeling for Multidisciplinary Design and Optimization”, Thesis No. 1419, Department of Management and Engineering, Linköping University.

[9] Python for Unix/C Programmers Copyright 1993 Guido van Rossum 1 –Ss Um An, Guido Van Rossum — 1993 — Proc. of the NLUUG najaarsconferentie. Dutch UNIX users group

[10] http://msdn.microsoft.com/en-us/library/6974wx4d%28v=VS.94%29.aspx

[11] http://msdn.microsoft.com/en-us/library/2x7h1hfk.aspx

[12] Myung, S. & Han, S. 2001, "Knowledge-based parametric design of mechanical products based on configuration design method", Expert Systems with Applications, vol. 21, no. 2, pp. 99-107.

[13] Rao, H. S. 2007, “Finite Element Methods vs. Classical Methods”. Daryaganj, Delhi: New Age International.

[14] Lanczos, C. 1970, “The Variational Principles of Mechanics”, 4th Ed., General Publishing Co., Canada.

[15] Dym, C. L. & Shames, I. H. 1973, “Solid Mechanics: A Variational Approach”, McGraw-Hill.

[16] Franceschini, G. & Macchietto, S. 2008, "Model-based design of experiments for parameter precision: State of the art", Chemical Engineering Science, vol. 63, no. 19, pp. 4846-4872.

[17] ANSYS, Inc., ANSYS DesignXplorer [pdf] Available at: < http://www1.ansys.com/customer/content/documentation/130/wb_dx.pdf>.

[18] http://www.abb.com/product/seitp327/26e3882ff473f5b2c125736a002f451a.aspx

[19] http://www.kxcad.net/ansys/ANSYS/workbench/dmscriptapi.html

[20] http://www.padtinc.com/blog/post/2010/11/10/ANSYS-Mechanical-Scripting-HOWTO-Part-1.aspx

[21] https://www1.ansys.com/customer/default.asp

[22] ANSYS, Inc., 2011. ANSYS Workbench Scripting Guide [pdf] Available at: < www1.ansys.com/customer/content/documentation/130/wb2_js.pdf>.

[23] Raghu Chaitanya, M. V. 2009, “Model Based Aircraft Control System Design and Simulation”. Thesis No. 630, Department of Management and Engineering, Linköping University.

Appendix 1

Creation of Named Selection and Coordinate systems in SolidWorks

The named selections and Coordinate systems help the script to recognize the faces of the CAD model and apply the mesh and load conditions. It is necessary to have the same named selections which are mentioned in the java script. All the CAD models in the pool which are to be tested for the feature of generic automation should have the same named selections to make it easily recognizable by the java script which automates the Mechanical module in ANSYS Workbench.

Procedure to create the Named Selection:

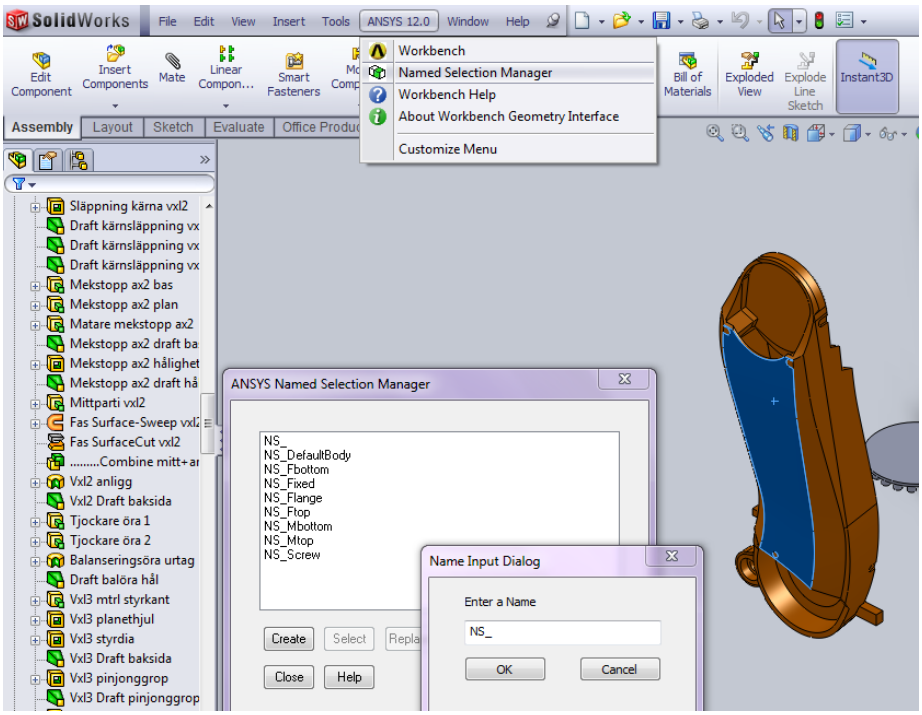


Figure 29 - Name selection creation procedure

Named Selection are created in Named Selection Manager in the Menu ANSYS 12.0 shown in the **Figure 53**.The surface that needs to be named is selected and the Create button in the named selection manager is clicked to get the Name input Dialog where the name of the surface can be specified. After entering the name click ok to exit the dialog. More than one surface can also be named at the same time using this option is SolidWorks.

Creation of the Coordinate Systems.

The Sizing option is used to mesh in the Mechanical module. The coordinate systems are used as the sphere center for the sphere of influence. The Sphere Radius is given which specifies the volume where the specified Element size is applied in the Model. This feature enables the use of variable mesh in the mesh model. Denser mesh is given in the spheres where it was estimated to have higher stress concentrations.

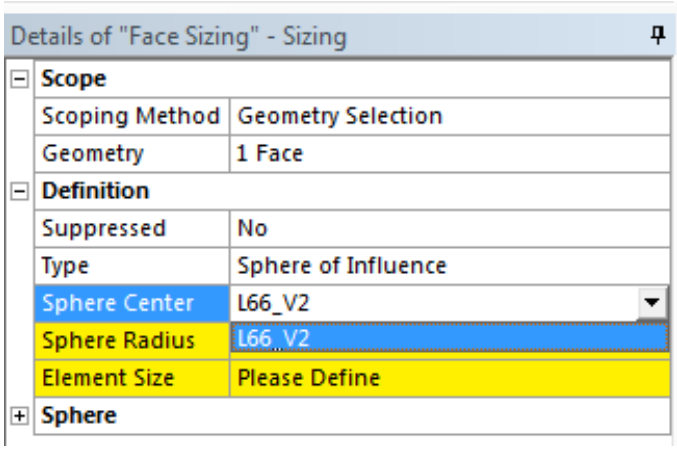


Figure 30 - Sphere mesh sizing

Procedure to assign Coordinate systems in SolidWorks

The Coordinate systems can be assigned from the Reference geometry tool bar. Coordinate systems created are named accordingly and used in the script to perform the automated mesh in the mechanical module.

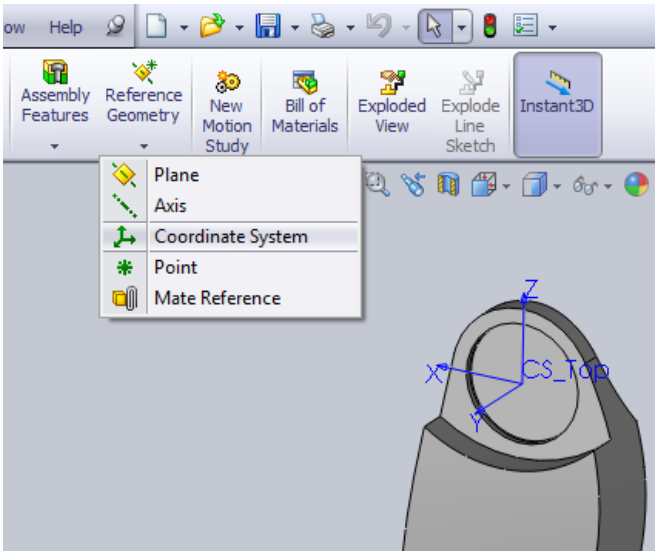


Figure 31 - Coordinate system creation

Integration of SolidWorks with ANSYS Workbench:

<u>Importing the CAD Model in ANSYS Workbench:</u>

- Insert a Geometry cell in the ANSYS Workbench.
- Right Click on the Geometry and select Import geometry.
- Browse to the location where the file is stored and press ok.

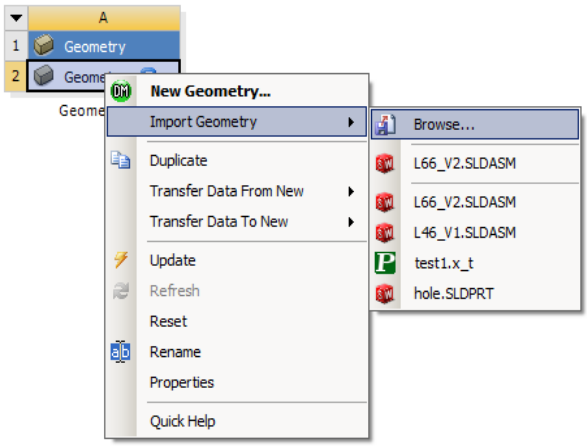


Figure 32 - Importing geometry in ANSYS

<u>Linking the geometry to the Static structural cell in ANSYS Workbench:</u>

- Place a static structural cell to the right of the Geometry cell.
- Click on the geometry in the geometry cell and drag it to the geometry in the static structural cell to form a link. This forms a link between the SolidWorks and ANSYS.

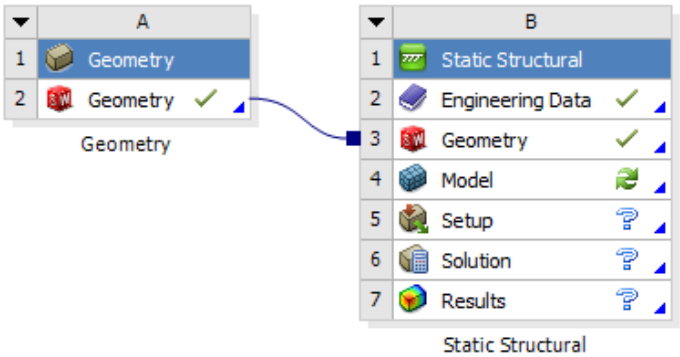
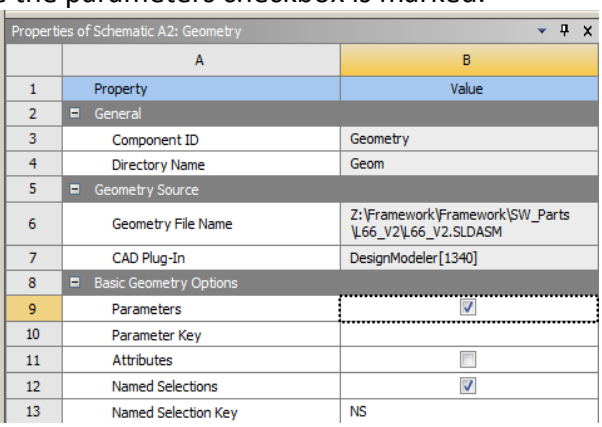


Figure 33 - Connecting geometry with static structural block

<u>Design Loop between the ANSYS and SolidWorks:</u>

- Click on the geometry in the geometry cell. In the properties of schematic as shown in the make sure the parameters checkbox is marked.



Properties of Schematic A2: Geometry

	A	B
1	Property	Value
2	General	
3	Component ID	Geometry
4	Directory Name	Geom
5	Geometry Source	
6	Geometry File Name	Z:\Framework\Framework\SW_Parts \L66_V2\L66_V2.SLDASM
7	CAD Plug-In	DesignModeler[1340]
8	Basic Geometry Options	
9	Parameters	☑
10	Parameter Key	
11	Attributes	☐
12	Named Selections	☑
13	Named Selection Key	NS

Figure 34 - Geometry properties

- Right click on the geometry in the geometry cell and click edit geometry. The Design Modeler module will be opened. Generate button when pressed attaches the model from the SolidWorks. Click on the geometry parameters in the Design Modeler as shown in the **Figure 35**.

D1@Extrude14@L66_V2.Part	40
D1@Fillet11@L66_V2.Part	2
D1@Fillet12@L66_V2.Part	2
D1@Extrude17@L66_V2.Part	0,0001
P D1@Thickness@L66_V2.Part	12,1555...
D1@plane_cylinder_holder@L66_V2.Part	40

Figure 35 - Geometry parameter selection

- The thickness parameter is shown in the outline. The parameter value can be edited by double clicking on the corresponding value column shown in **Figure 36**.

Outline of Schematic B8: Parameters

	A	B	C
1	ID	Parameter Name	Value
2	Input Parameters		
3	Geometry (A1)		
4	P1	D1@Thickness@L66_V2.Part	12,156
*	New input parameter	New name	New expression
6	Output Parameters		
*	New output parameter		New expression
8	Charts		

Figure 36 - Outline of parameters

- When the parameter value is edited and the value is changed the CAD model is modifi**ed in SolidWorks and regenerated in the geometry ce**ll. The modified geometry will be reattached to the geometry in the static structural cell. This forms a design loop between the SolidWorks and ANSYS Workbench.

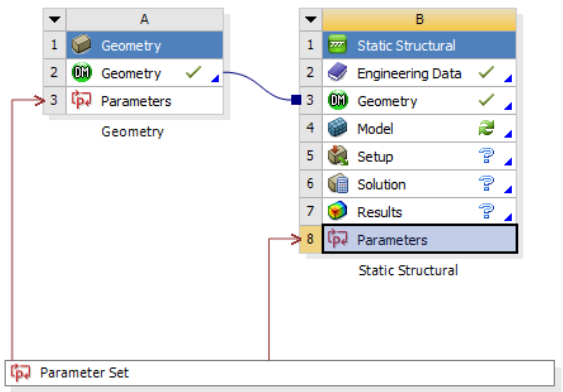


Figure 37 - Representation of parameters

Appendix 2
Automation of structural analysis

Journal script using python programming language:

- Start recording the journal

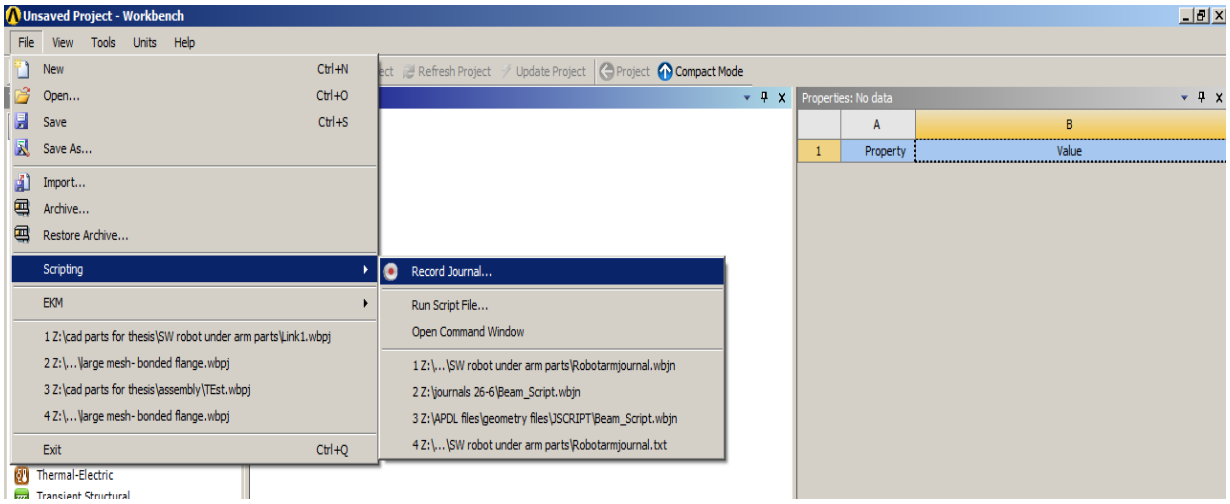


Figure 38 - Menu for recording journal in ANSYS Workbench

- Create a Geometry Set from the Component Systems and import the geometry file

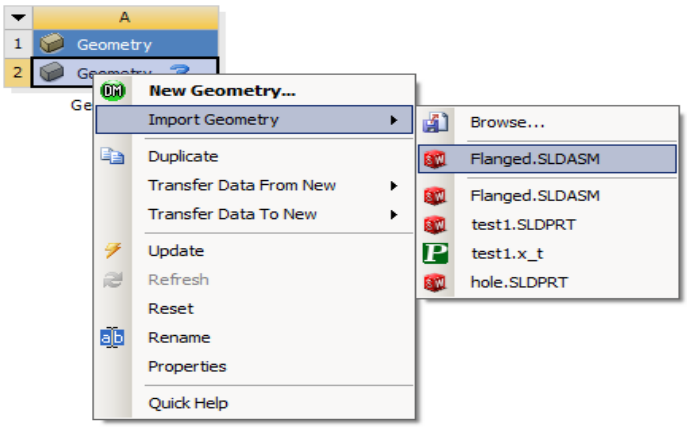


Figure 39 - Importing geometry

- Select from the Geometry properties what needs to be imported

Properties of Schematic A2: Geometry

	A	B
1	Property	Value
2	General	
3	Component ID	Geometry
4	Directory Name	Geom
5	Geometry Source	
6	Geometry File Name	Z:\cad parts for thesis\SW robot under arm parts\Link1_files\dp0\Geom\DM\Geom.agdb
7	CAD Plug-In	SolidWorks[4604]
8	Basic Geometry Options	
9	Attributes	☐
10	Named Selections	☑
11	Named Selection Key	
12	Material Properties	☐
13	Advanced Geometry Options	
14	Analysis Type	3D
15	Use Associativity	☑
16	Import Coordinate Systems	☑
17	Import Work Points	☐
18	Reader Mode Saves Updated File	☐
19	Import Using Instances	☑
20	Smart CAD Update	☑
21	Enclosure and Symmetry Processing	☑

Figure 40 - Table with geometry properties

- Create a Static structural template and link the geometry

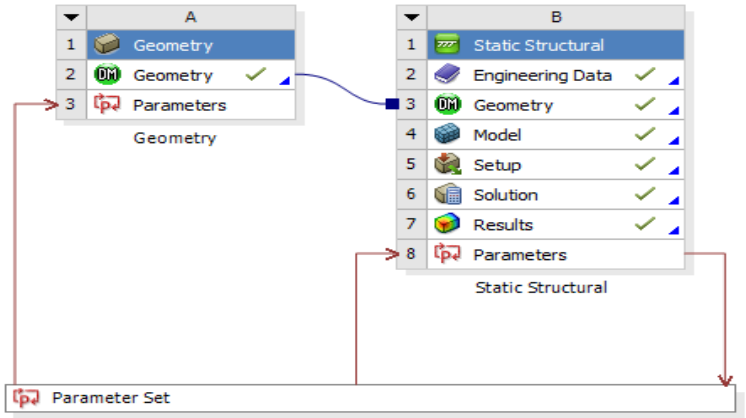


Figure 41 - Project schematic

- After performing the previous activities the recording must be stop

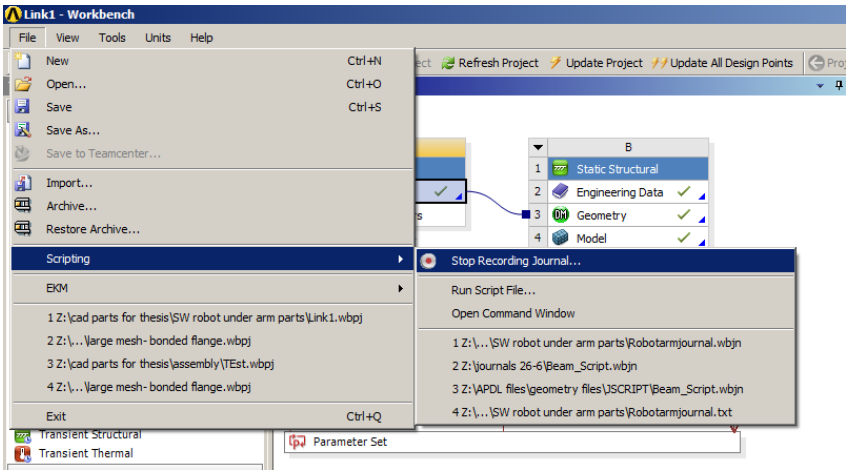


Figure 42 - Stoping the journal

- Journal script used in the Thesis Work

```
SetScriptVersion(Version="13.0")
import os

Path = "Z:/Framework/SW_Parts/L66_V1/L66_V1.SLDASM"

#Creating a new geometric

template1 = GetTemplate(TemplateName="Geometry")
system1 = template1.CreateSystem()
geometry1 = system1.GetContainer(ComponentName="Geometry")
geometryProperties1 = geometry1.GetGeometryProperties()
geometryProperties1.GeometryImportParameters = True
geometryProperties1.GeometryImportParametersFilter = ""
geometryProperties1.GeometryImportNamedSelections = True
geometryProperties1.GeometryImportNamedSelectionsFilter = ""
geometryProperties1.GeometryImportCoordinateSystems = True
geometryProperties1.GeometryImportSmartUpdate = True

#Geometry gath

geometry1.SetFile(
    FilePath=Path,
    PlugInName="SolidWorks[4605]")

 #Accesing the geometry and importing the Solidwork file

geometry1.Edit()
geometry1.SendCommand(Command = """var Geometry_path = "Z:/Framework/DM_Script/DM_Script.js";
ag.guiScript.agRunScript(Geometry_path);""")
```

```
geometry1.Exit()

#End of Importing Geometry From SolidWorks

                #Creating a new static structural project and Connection with previous geometry

template2 = GetTemplate(
                TemplateName="Static Structural",
                Solver="ANSYS")
system2 = template2.CreateSystem(
   Position="Right",
   RelativeTo=system1)
component1 = system2.GetComponent(Name="Geometry")
component2 = system1.GetComponent(Name="Geometry")
component1.ReplaceWithShare(
   TargetSystem=system2,
   ComponentToShare=component2,
   SourceSystem=system1)


#Material Selection

#Accesing the Engineering data and modifying the Material
#engineeringData1 = system1.GetContainer(ComponentName="Engineering Data")
#matl1 = engineeringData1.GetMaterial(Name="Structural Steel")
#matl1.Delete()
#favorites1 = EngData.LoadFavoriteItems()
#library2 = EngData.OpenLibrary(
                #Name="General Materials",
                #Source="General_Materials.xml")
#matl2 = engineeringData1.ImportMaterial(
                #Name="Aluminum Alloy",
                #Source="General_Materials.xml")

#End of Material Selection

with Transaction():
   component3 = system2.GetComponent(Name="Model")
   component3.Refresh()
   model1 = system2.GetContainer(ComponentName="Model")
   model1.Edit()

model1.SendCommand(Command="WB.AppletList.Applet(\"DSApplet\").App.Script.doTools
RunMacro(\"Z:/Framework/ME_Script/ME_Script.js\")")
model1.Exit()
Update()
```

```
#MECHANICAL APDL COMPARISON

designPoint1 = Parameters.GetDesignPoint(Name="0")
UpdateAllDesignPoints(DesignPoints=[designPoint1])

template1 = GetTemplate(TemplateName="Mechanical APDL")

system1 = GetSystem(Name="SYS")
system2 = template1.CreateSystem(
   Position="Right",
   RelativeTo=system1)
component1 = system1.GetComponent(Name="Model")
component2 = system2.GetComponent(Name="Setup")
component1.TransferData(TargetComponent=component2)

system3 = template1.CreateSystem(
   Position="Below",
   RelativeTo=system2)
component3 = system1.GetComponent(Name="Setup")
component4 = system3.GetComponent(Name="Setup")
component3.TransferData(TargetComponent=component4)


system5 = template1.CreateSystem(
   Position="Below",
   RelativeTo=system3)
component5 = system1.GetComponent(Name="Solution")
component6 = system5.GetComponent(Name="Setup")
component5.TransferData(TargetComponent=component6)

system6 = GetSystem(Name="Geom")
system7 = template1.CreateSystem(
   Position="Below",
   RelativeTo=system1)
component7 = system6.GetComponent(Name="Geometry")
component8 = system7.GetComponent(Name="Setup")
component7.TransferData(TargetComponent=component8)


designPoint1 = Parameters.GetDesignPoint(Name="0")
UpdateAllDesignPoints(DesignPoints=[designPoint1])
Refresh()
Update()
#Saving Project

Save(
```

FilePath="C:/Program Files/ANSYS Inc/v130/Framework/bin/Win32/L66_V1.wbpj")

Debugging Java Scripts using Microsoft Visual Studio 2010

- The first step is to go to the Control Panel of the computer and in "Internet Options the Advanced Tab" make sure that "Disable Script Debugging (Other)" is unchecked. See figure below:

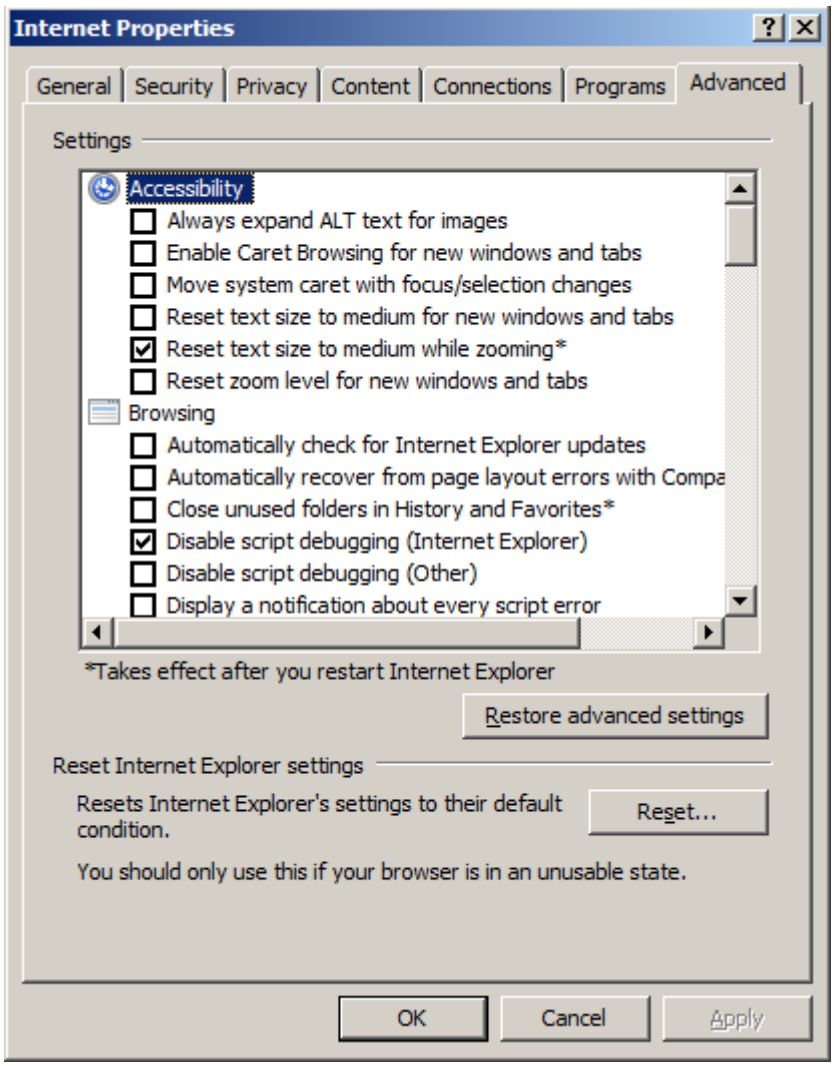


Figure 43 - Internet properties

- After unchecking the box explained in the previous section, the following registry setting needs to be adjust, HKEY_CURRENT_USER\Software\Microsoft\Windows Script\Settings\JITDebug = 1

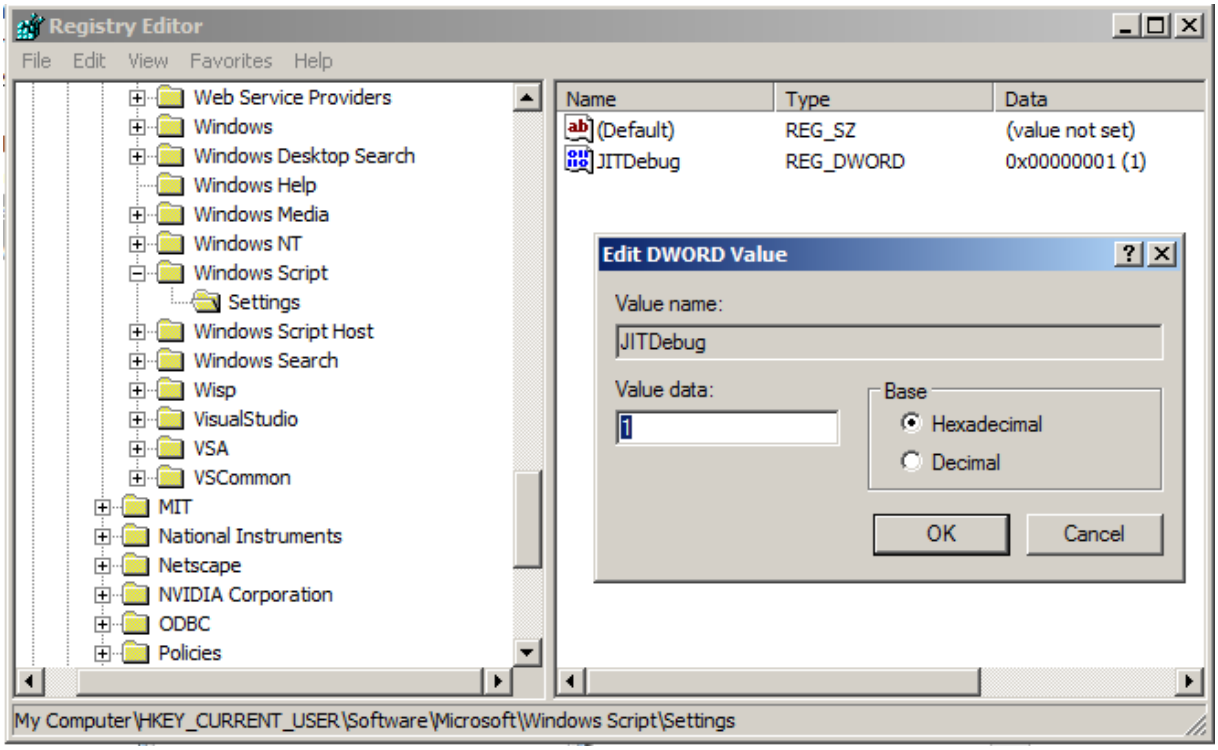


Figure 44 - Registry editor

- The next step is to attach ANSYS Designmodeler with Microsoft Visual Studio

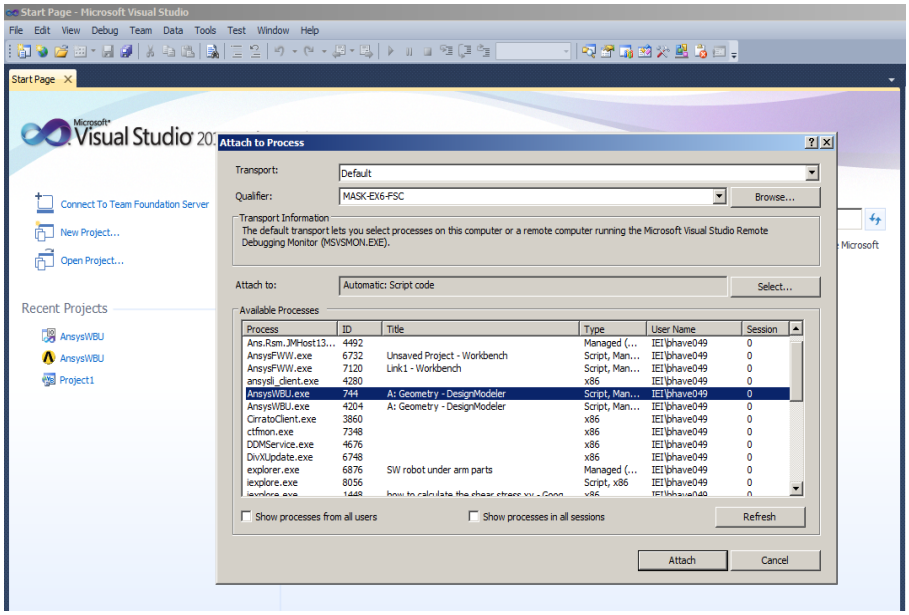


Figure 45 - Attachment of ANSYS into MS Visual Studio 2010

- To start debugging, it is necessary to create a Java Script with a error, for example:

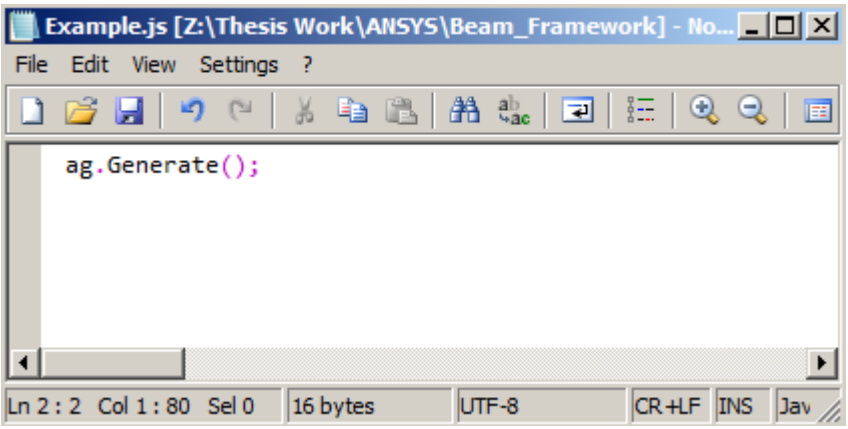


Figure 46 - Java script example

- After creating the script then use the future in ANSYS DesignModeler to Run Script.

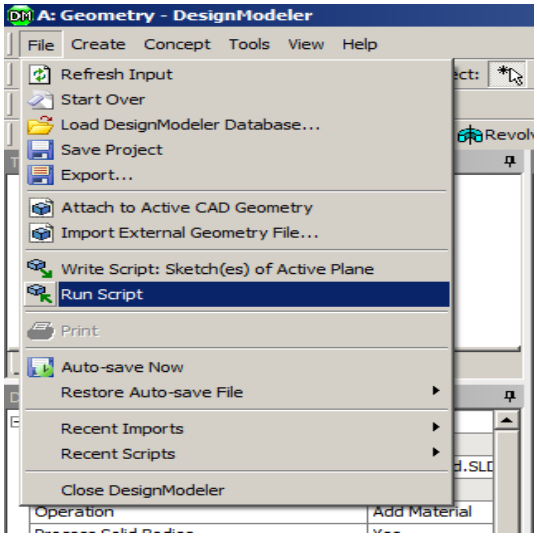


Figure 47 - Menu of DesignModeler

- Since the previous script has some errors, it will ask the user to debug using MS Visual Studio 2010 and the following error will prompt.

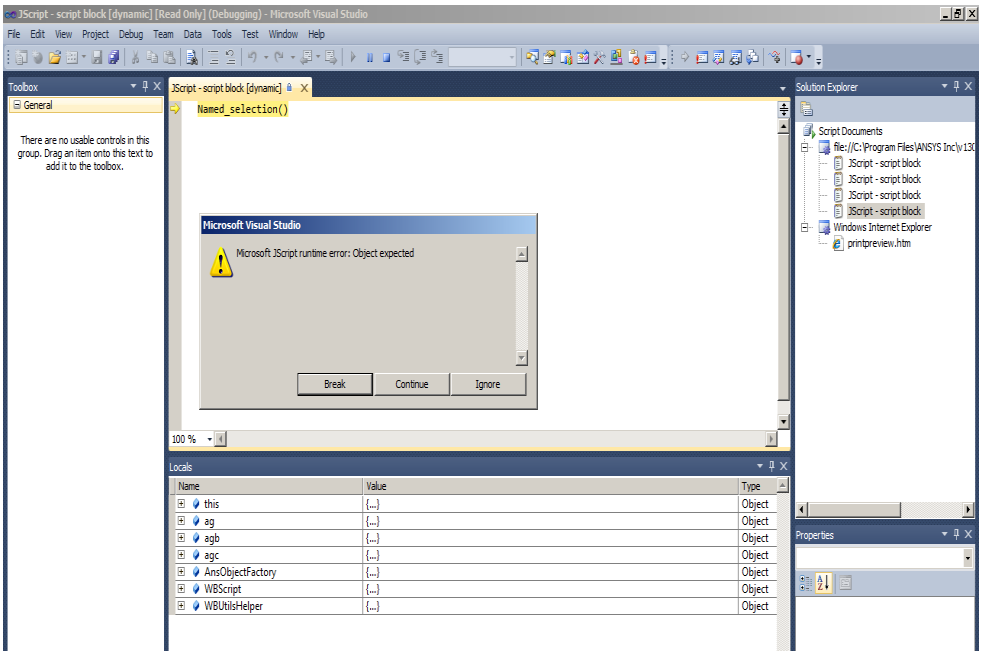


Figure 48 - Error prompt in MS Visual Studio 2010

Push Break on the error message and start searching the Locals of the program to learn the different functions built in ANSYS Workbench.

Java script for ANSYS Designmodeler

In this line it is only necessary to select the Thickness parameter from the imported geometry from SolidWorks. If more parameters are needed, it must only be included in the following line.

```
ag.m.ParameterMgr.FullTextADP="Attach1:D1@Thickness@L66_V1.Part = 10";
```

Java script for ANSYS Mechanical

```
//Accesing Mesh Control Properties

var Mesh_Mod = DS.Tree.FirstActiveBranch.MeshControlGroup;
DS.Script.SelectItems(""+Mesh_Mod.ID);

//Insert the face sizing for default faces in the body Sphere of influence at the top

DS.Script.doInsertMeshSize(1)
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "ANSYS_Persist_Key_1" ;
ListView.ActivateItem("Type");
ListView.ItemValue = "Sphere of Influence" ;
ListView.ActivateItem("Sphere Center");
ListView.ItemValue = "L66_V1_CS_Top";
```

```
ListView.ActivateItem("Sphere Radius");
ListView.ItemValue = "0.25"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Element Size");
ListView.ItemValue = "0.01"
ListView.SelectedItem.IsChecked="true"

//Insert the face sizing for default faces in the body Sphere of influence at the Bottom
DS.Script.doInsertMeshSize(1)
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "ANSYS_Persist_Key_1" ;
ListView.ActivateItem("Type");
ListView.ItemValue = "Sphere of Influence" ;
ListView.ActivateItem("Sphere Center");
ListView.ItemValue = "L66_V1_CS_Bottom";
ListView.ActivateItem("Sphere Radius");
ListView.ItemValue = "0.25"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Element Size");
ListView.ItemValue = "0.02"
ListView.SelectedItem.IsChecked="true"

//Insert the face sizing for default faces on the rest of the body

DS.Script.doInsertMeshSize(1)
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "ANSYS_Persist_Key_2" ;
ListView.ActivateItem("Type");
ListView.ItemValue = "Element Size" ;
ListView.ActivateItem("Element Size");
ListView.ItemValue = "0.01"
ListView.SelectedItem.IsChecked="true"

//Generate Mesh
DS.Script.doModelGenerateMesh(1)




//Accesing the boundary conditions properties

var Env = DS.Tree.FirstActiveBranch.Environment;
```

```
DS.Script.SelectItems(""+Env.ID);

//Creating force at the top of the structure

DS.Script.doInsertEnvironmentForce(1)
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "NS_Ftop" ;
ListView.ActivateItem("Define By");
ListView.ItemValue = "Components" ;
ListView.ActivateItem("X Component");
ListView.ItemValue = "-10000"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Y Component");
ListView.ItemValue = "0"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Z Component");
ListView.ItemValue = "-810.42"
ListView.SelectedItem.IsChecked="true"

//Creating force at the bottom of the structure

DS.Script.doInsertEnvironmentForce(1)
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "NS_Fbottom" ;
ListView.ActivateItem("Define By");
ListView.ItemValue = "Components" ;
ListView.ActivateItem("X Component");
ListView.ItemValue = "14286"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Y Component");
ListView.ItemValue = "0"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Z Component");
ListView.ItemValue = "261.58"
ListView.SelectedItem.IsChecked="true"



//Creating Moment at the top of the structure

DS.Script.doInsertEnvironmentFreeMoment(1)
ListView.ActivateItem("Scoping Method");
```

```
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "NS_Mtop" ;
ListView.ActivateItem("Define By");
ListView.ItemValue = "Components" ;
ListView.ActivateItem("X Component");
ListView.ItemValue = "-171.03"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Y Component");
ListView.ItemValue = "-137.35"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Z Component");
ListView.ItemValue = "2213.4"
ListView.SelectedItem.IsChecked="true"


//Creating Moment at the bottom of the structure

DS.Script.doInsertEnvironmentFreeMoment(1)
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "NS_Mbottom" ;
ListView.ActivateItem("Define By");
ListView.ItemValue = "Components" ;
ListView.ActivateItem("X Component");
ListView.ItemValue = "58.017"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Y Component");
ListView.ItemValue = "-9222.4"
ListView.SelectedItem.IsChecked="true"
ListView.ActivateItem("Z Component");
ListView.ItemValue = "-2370.6"
ListView.SelectedItem.IsChecked="true"

//insert the fixed support

DS.Script.doInsertEnvironmentFixedDisplacement(1)
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection" ;
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "NS_Fixed" ;
//Solution

var Sol_eqv = DS.Tree.FirstActiveBranch.Environment.AnswerSet;
DS.Script.SelectItems(""+Sol_eqv.ID);
```

```
//Inserting Maximum Von Mises Equivalent Stress

DS.Script.doInsertSolutionEquivalentStress(1)
ListView.ActivateItem("Maximum");
ListView.SelectedItem.IsChecked="true"

//Calculation start

DS.Script.doCalculateResults(1)
DS.Script.doSolveDefaultHandler(1)
```

Appendix 3
Goal Driven Optimization

Design of Experiments (DOE):

- Click on the Goal driven optimization in the design exploration tool box and drag it on to the parameter set. This can be seen in the **Figure 49.**

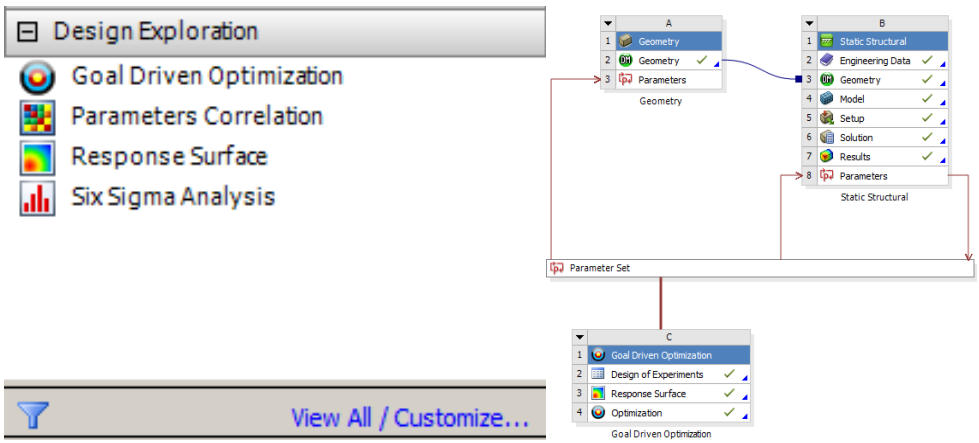


Figure 49 - Goal Driven Optimization Block

- Click on the Design of experiments in Goal driven optimization cell and select the input parameters which are the design variables. In this case the thickness parameter is selected as the design variable.

16	P54 - Moment Z Component	☐
17	P55 - Moment 2 Y Component	☐
18	P56 - Moment 2 Z Component	☐
19	P58 - Body Sizing Element Size	☐
20	P59 - Body Sizing 3 Element Size	☐
21	P60 - Body Sizing 2 Element Size	☐
22	Geometry (A1)	
23	P44 - D1@Thickness@L66_V2.Part	☑

Figure 50 - Selection of Design Variables

- The range of the input variables can be specified by clicking on the input parameter and the lower bound and Upper bound can be edited in the properties of outline.

Properties of Outline A23: P44

	A	B
1	Property	Value
2	General	
3	Units	
4	Type	Design Variable
5	Classification	Continuous
6	Values	
7	Lower Bound	8
8	Upper Bound	12,375
9	Initial Value	8

Figure 51 - Setting the Upper and Lower Bound

- The DOE type and the total number of samples can be specified in the properties after clicking on the DOE in the Goal driven Optimization cell. The DOE can be previewed from the DOE in GDO cell.

Properties of Schematic C2: Design of Experiments

	A	B
1	Property	Value
2	General	
3	Component ID	Design of Experiment
4	Directory Name	GDO
5	Design Points	
6	Preserve Design Points After DX Run	☐
7	Design of Experiments	
8	Design of Experiments Type	Custom + Sampling
9	Total Number of Samples	7

Figure 52 - DOE settings

- The DOE sequence can be controlled by selecting the ascending, Descending or sort settings options in the table of schematic in DOE.

Table of Schematic C2: Design of Experiments

	A	B	C
1	Name	P44 - D1@Thickness@L66_V2.Part	P19 - Equivalent Stress Maximum (P
2	3	8,0007	
3	7	8,8755	
4	4	9,7502	
5	10	10,188	
6	8	10,625	
7	1	11,5	3,2969E+08
8	9	12,375	2,9809E+08
*	New Design Point		

Ascending
Descending
Sort Settings...
Cancel sorting

Figure 53 - Sort settings

Response surface in Goal Driven Optimization:

- The response surface type and the number of verification points can be specified prom the properties as shown in the **Figure 54.**

Properties: No data

	A	B
1	Property	Value
2	Design Points	
3	Preserve Design Points After DX Run	☐
4	Meta Model	
5	Response Surface Type	Standard Response Surface - Fu...
6	Refinement	
7	Refinement Type	Manual
8	Verification Points	
9	Generate Verification Points	☑
10	Number of Verification Points	5

Figure 54 - Response surface settings

Optimization in Goal Driven Optimization:

- Clicking on the Optimization in the GDO cell enables the user to specify the optimization method and Number of samples.

Properties of Outline A13: Sensitivities

	A	B
1	Property	Value
2	Chart	
3	Display Parameter Full Name	☑
4	Mode	Bar
5	Design Points	
6	Preserve Design Points After DX Run	☐
7	Optimization	
8	Optimization Method	Screening
9	Number of Samples	1000
10	Constraint Handling (GDO)	As Hard Constraints
11	Size of Generated Sample Set	1000
12	Verify Candidate Points	☐

Figure 55 - Optimization settings

- The design Objective can be specified by selecting the suitable option as shown in the fig which in this case is to minimize and the importance of the Objective can be specified. The constraint can be given to the output parameter which in this case is to have the induced maximum stress parameter less than or equal to 4,6E 08. Update the project and the results will be display.

Schematic C4: Optimization

A	B	C
	P44 - D1@Thickness@L66_V2.Part	P19 - Equivalent Stress Maximum (Pa)
Optimization Study		
Objective	Minimize	Values <= Target
Target Value		4,6E+08
Importance	Higher	Higher

Figure 56 - Setting the objective and constraints

Appendix 4
Framework automation

As mentioned in section **3.3 Graphical User Interface (GUI),** the program selected for controlling the different tools was Microsoft Excel. Excel used the programming language Visual Basic. In the following sections it will be explained how this automation process was done.

1. **Open framework**

- **Figure 57** shows the framework created in Microsoft Excel with the use of ActiveX Controls and Visual Basic

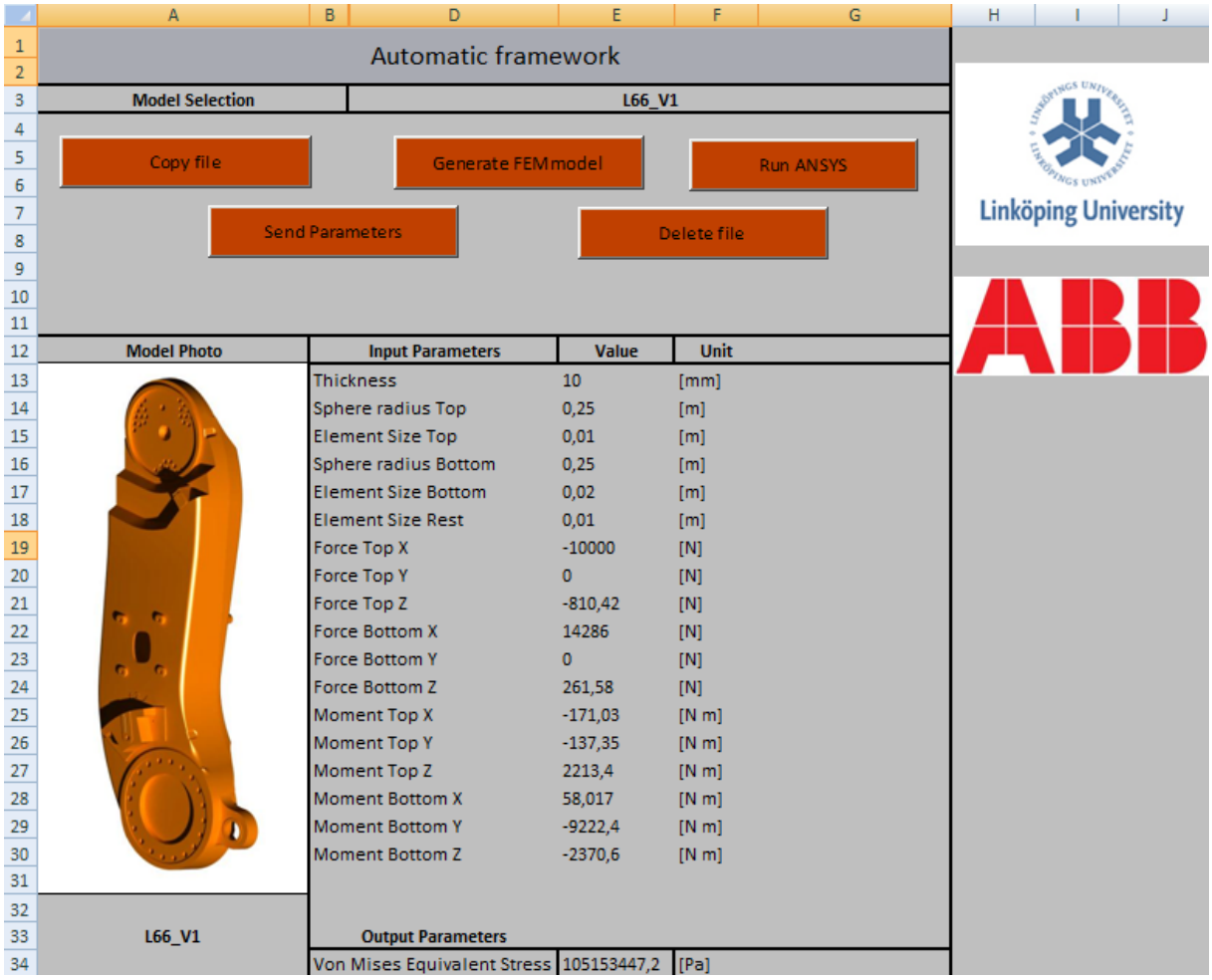


Figure 57 – Framework

Structure of folder

- In **Figure 58** the structure of folders is presented, the framework is located in the folder called Framework and inside each of the folders with Script at end the different scripts are located.

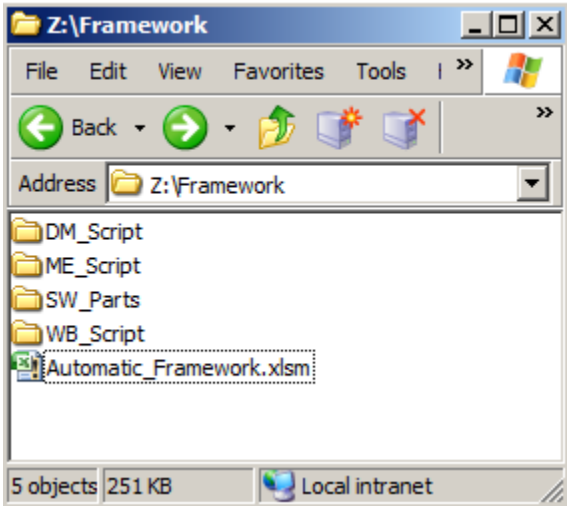


Figure 58 – Folders structure

- In the folder SW_Parts the different models are located; in the next **Figure 59** this can be observed.

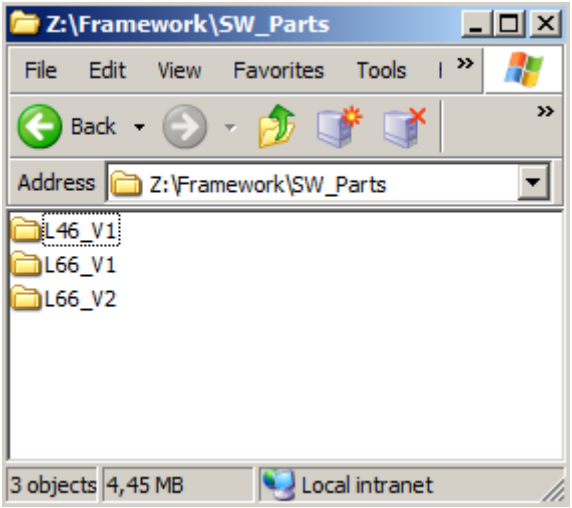


Figure 59 – SolidWorks parts

2. **Copy and pasting files**

- **The first step was to copy some files need for runni**ng automatically ANSYS from Excel. The path is: C:\Program Files\ANSYS Inc\v130\Framework\bin\Win32. By pressing the button Copy File in the framework this copy and pasting operations were done.

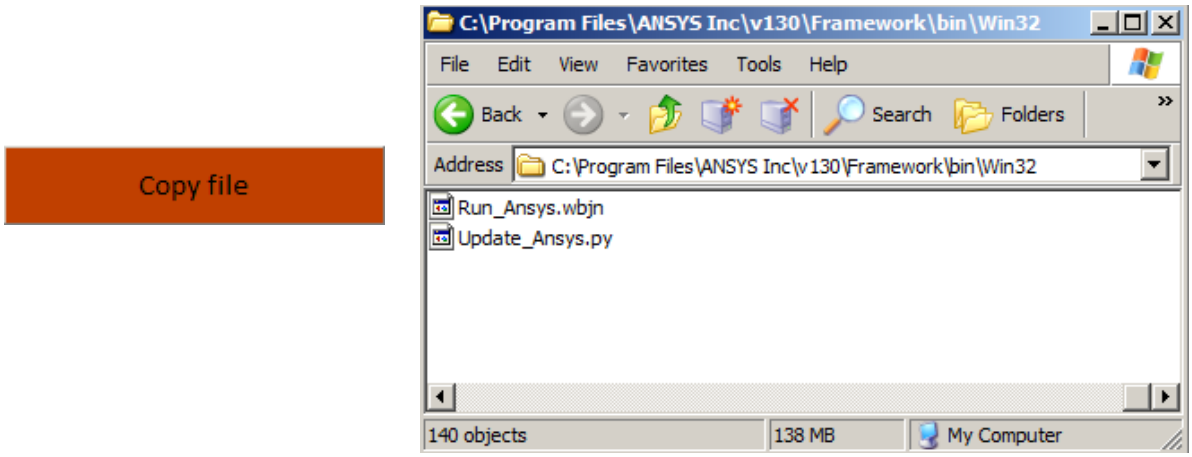


Figure 60 – Copy and pasting operation

- The VB code for performing the previous operations:

```
Sub Copy_file()
Working_dir = ThisWorkbook.Path
WB_framework_path = Replace(Working_dir, "\", "/")
WB_file_path = Working_dir & "\WB_Script\Run_Ansys.wbjn"
Line_Path = "RunScript(FilePath=" & Chr(34) & WB_framework_path &
"/WB_Script/WB_RobotArm.wbjn" & Chr(34) & ")"

Const ForReading = 1
Const ForWriting = 2
```

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile(WB_file_path, ForReading)

iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = 3 Then
strNewContents = strNewContents & Line_Path & vbCrLf
Else
strNewContents = strNewContents & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(WB_file_path, ForWriting)
objFile.Write strNewContents
objFile.Close

Dim fs As Object
Dim oldPath As String, newPath As String
oldPath = Working_dir & "\WB_Script" 'Folder file is located in
newPath = "C:\Program Files\ANSYS Inc\v130\Framework\bin\Win32" 'Folder to copy file to
Set fs = CreateObject("Scripting.FileSystemObject")
fs.CopyFile oldPath & "\" & "Run_Ansys.wbjn", newPath & "\" & "Run_Ansys.wbjn" 'This file
was an .xls file
fs.CopyFile oldPath & "\" & "Update_Ansys.py", newPath & "\" & "Update_Ansys.py" 'This
file was an .xls file
Set fs = Nothing

End Sub
```

3. **Generating FEM model**

- After copying the files, the model is selected in the Cell D3 and the button Generate FEM model needs to be press so all the scripts change according to the model selected. Another feature of this button is that generate automatically the picture of the model to be analyze.

Figure 61 – Button to Generate FEM Model

- The VB code for performing the previous operations:

```
Sub Change_path()

Working_dir = ThisWorkbook.Path 'Path of working directory
Model_name = Worksheets("Framework").Range("Model").Value 'Model Name
```

```
Thick_value = Worksheets("Framework").Range("E13").Value 'Thickness value

WB_framework_path = Replace(Working_dir, "\", "/")
Line_Save_Path = Replace("C:\Program Files\ANSYS Inc\v130\Framework\bin\Win32", "\",
"/")

Line_Path = "Path = " & Chr(34) & WB_framework_path & "/SW_Parts/" & Model_name &
"/" & Model_name & ".SLDASM" & Chr(34)
Line_DM = "geometry1.SendCommand(Command = " & Chr(34) & Chr(34) & Chr(34) & "var
Geometry_path = " & Chr(34) & WB_framework_path & "/DM_Script/DM_Script.js" &
Chr(34) & ";"
Line_ME = "    model1.SendCommand(Command=" & Chr(34) & "WB.AppletList.Applet(\" &
Chr(34) & "DSApplet\" & Chr(34) & ").App.Script.doToolsRunMacro(\" & Chr(34) &
WB_framework_path & "/ME_Script/ME_Script.js\" & Chr(34) & ")" & Chr(34) & ")"
'Line_Save = "FilePath=" & Chr(34) & WB_framework_path & "/" & Model_name & ".wbpj"
& Chr(34) & ")"
Line_Save = "FilePath=" & Chr(34) & Line_Save_Path & "/" & Model_name & ".wbpj" &
Chr(34) & ")"
Line_Thickness = "ag.m.ParameterMgr.FullTextADP=" & Chr(34) &
"Attach1:D1@Thickness@" & Model_name & ".Part = " & CStr(Thick_value) & Chr(34) & ";"
Line_CS_Top = "ListView.ItemValue = " & Chr(34) & Model_name & "_CS_Top" & Chr(34) &
";"
Line_CS_Bottom = "ListView.ItemValue = " & Chr(34) & Model_name & "_CS_Bottom" &
Chr(34) & ";"

Const ForReading = 1
Const ForWriting = 2

'Block to change Python script A-------------------------------------------------------
'Block to change the path of the part in python----------------1-----------------

Set objFSO = CreateObject("Scripting.FileSystemObject")
WB_file_path = Working_dir & "\WB_Script\WB_RobotArm.wbjn"
Set objFile = objFSO.OpenTextFile(WB_file_path, ForReading)

iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = 5 Then
strNewContents = strNewContents & Line_Path & vbCrLf
Else
strNewContents = strNewContents & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(WB_file_path, ForWriting)
```

```
objFile.Write strNewContents
objFile.Close

'-------------------------------End of block 1-----------------------------------

'Block to change the path of the DM in python----------------2-----------------
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile(WB_file_path, ForReading)

iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = 32 Then
strNewContents_5 = strNewContents_5 & Line_DM & vbCrLf
Else
strNewContents_5 = strNewContents_5 & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(WB_file_path, ForWriting)
objFile.Write strNewContents_5
objFile.Close

'-------------------------------End of block 2-----------------------------------

'Block to change the path of the ME in python----------------3-----------------
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile(WB_file_path, ForReading)

iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = 75 Then
strNewContents_6 = strNewContents_6 & Line_ME & vbCrLf
Else
strNewContents_6 = strNewContents_6 & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(WB_file_path, ForWriting)
objFile.Write strNewContents_6
objFile.Close
'-------------------------------End of block 3-----------------------------------

'Block to change the path for saving the analysis in python-----------4--------
```

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile(WB_file_path, ForReading)

iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = 129 Then
strNewContents_7 = strNewContents_7 & Line_Save & vbCrLf
Else
strNewContents_7 = strNewContents_7 & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(WB_file_path, ForWriting)
objFile.Write strNewContents_7
objFile.Close

'-------------------------------End of block 4-----------------------------------
'-------------------------------End of block A-----------------------------------------------------

'--------------------------------Block B-----------------------------------------------
'Block to change Jscript for Designmodeler

Set objFSO = CreateObject("Scripting.FileSystemObject")
DM_file_path = Working_dir & "\DM_Script\DM_Script.js"
Set objFile = objFSO.OpenTextFile(DM_file_path, ForReading)

iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = 1 Then
strNewContents_1 = strNewContents_1 & Line_Thickness & vbCrLf
Else
strNewContents_1 = strNewContents_1 & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(DM_file_path, ForWriting)
objFile.Write strNewContents_1
objFile.Close


'Block to change the Jscript for Mechanical 1

Set objFSO = CreateObject("Scripting.FileSystemObject")
```

```
ME_file_path = Working_dir & "\ME_Script\ME_Script.js"
Set objFile = objFSO.OpenTextFile(ME_file_path, ForReading)

iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = 22 Then
strNewContents_2 = strNewContents_2 & Line_CS_Top & vbCrLf
Else
strNewContents_2 = strNewContents_2 & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(ME_file_path, ForWriting)
objFile.Write strNewContents_2
objFile.Close


'Block to change the Jscript for Mechanical 2

Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile(ME_file_path, ForReading)

iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = 45 Then
strNewContents_3 = strNewContents_3 & Line_CS_Bottom & vbCrLf
Else
strNewContents_3 = strNewContents_3 & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(ME_file_path, ForWriting)
objFile.Write strNewContents_3
objFile.Close

'Block to put the values of the parameters

Dim Line_index(0 To 16) As String
Line_index(0) = 25
Line_index(1) = 28
Line_index(2) = 48
Line_index(3) = 51
Line_index(4) = 66
```

```
Line_index(5) = 96
Line_index(6) = 100
Line_index(7) = 104
Line_index(8) = 123
Line_index(9) = 127
Line_index(10) = 131
Line_index(11) = 149
Line_index(12) = 153
Line_index(13) = 157
Line_index(14) = 176
Line_index(15) = 180
Line_index(16) = 184

For i = 0 To 11
Component = "ListView.ItemValue = " & Chr(34) &
CStr(Worksheets("Framework").Range("E" & i + 14).Value) & Chr(34)
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile(ME_file_path, ForReading)
strNewContents_4 = ""
iLineNumber = 0
Do Until objFile.AtEndOfStream
strLine = objFile.Readline
iLineNumber = iLineNumber + 1
If iLineNumber = Line_index(i) Then
strNewContents_4 = strNewContents_4 & Component & vbCrLf
Else
strNewContents_4 = strNewContents_4 & strLine & vbCrLf
End If
Loop
objFile.Close
Set objFile = objFSO.OpenTextFile(ME_file_path, ForWriting)
objFile.Write strNewContents_4
objFile.Close
Next
'-----------------------------End of Block B-------------------------------------------------------
End Sub
```

- VB codes to run the change picture in framework:

```
Sub change_picture()
Model_name = Worksheets("Framework").Range("Model").Value 'Model Name
Working_dir = ThisWorkbook.Path
Picture_path = Working_dir & "\SW_Parts\" & Model_name & "\" & Model_name & ".jpg"
InsertPictureInRange Picture_path, _
   Range("A13:B31")
Worksheets("Framework").Range("A32").Value = Model_name
End Sub
Sub InsertPictureInRange(PictureFileName, TargetCells As Range)
```

```
' inserts a picture and resizes it to fit the TargetCells range
Dim p As Object, t As Double, l As Double, w As Double, h As Double
   If TypeName(ActiveSheet) <> "Worksheet" Then Exit Sub
   If Dir(PictureFileName) = "" Then Exit Sub
   ' import picture
   Set p = ActiveSheet.Pictures.Insert(PictureFileName)
   ' determine positions
   With TargetCells
      t = .Top
      l = .Left
      w = .Offset(0, .Columns.Count).Left - .Left
      h = .Offset(.Rows.Count, 0).Top - .Top
   End With
   ' position picture
   With p
      .Top = t
      .Left = l
      .Width = w
      .Height = h
   End With
   Set p = Nothing
End Sub
```

4. **Running ANSYS Workbench**

- After generating the FEM Model the script for running ANSYS can be run. To be able to run ANSYS it was necessary to create a batch file, which contains the following lines shown in **Figure 62.**

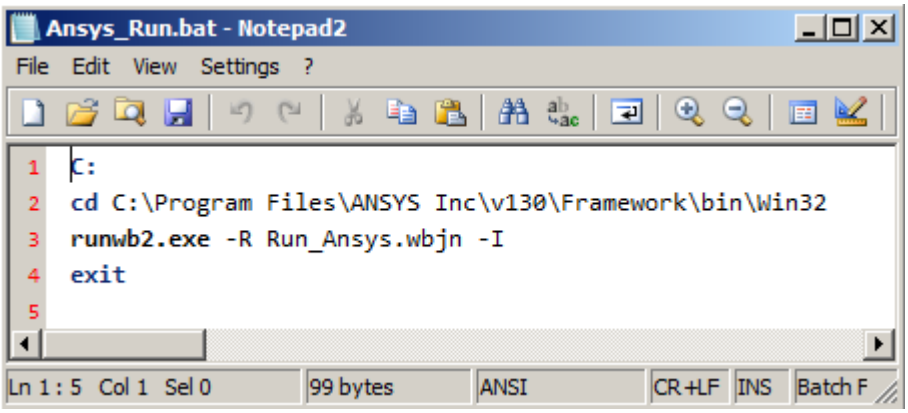


Figure 62 – Batch file to run ANSYS

This batch files, tells the system to open ANSYS workbench and run the python script Run_Ansys.wbjn and this script open the Journal that create the whole analysis automatically, see **Appendix 2**.

1. In the **Figure 63** the python script is shown.

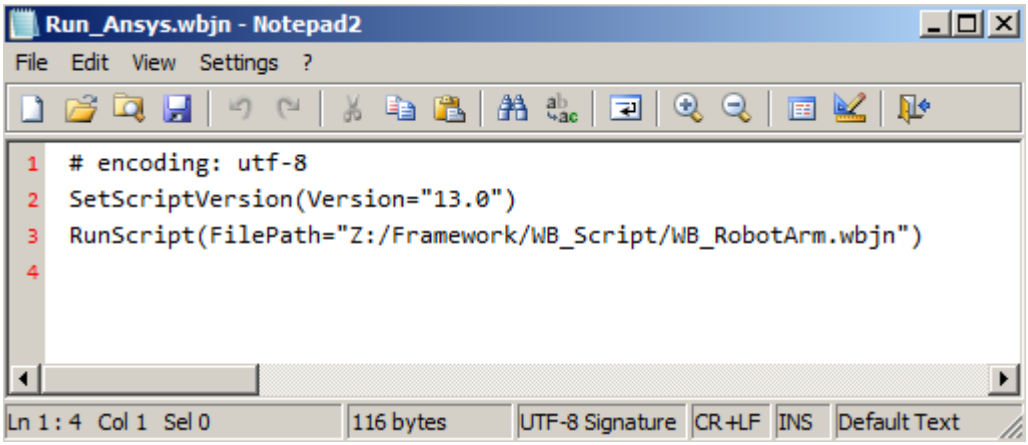


Figure 63 – Python code to run Journal inside ANSYS Workbench

- The Button that triggers this script for running the batch file can be seen in can be seen in **Figure 64.**

Figure 64 – Button to run ANSYS Workbench

- VB codes to run the batch file Ansys_Run.bat:

```
Sub Run_Ansys()
Working_dir = ThisWorkbook.Path
Bat_path = Chr(34) & Working_dir & "\WB_Script\Ansys_Run.bat" & Chr(34)
Shell Bat_path
End Sub
```

5. Updating ANSYS Workbench

- If changes need to be done in the analysis after being created the Send Parameters button take cares of that. It works similar to the button Run ANSYS but in this case it opens the analysis already created. The following figure shows the batch file used.

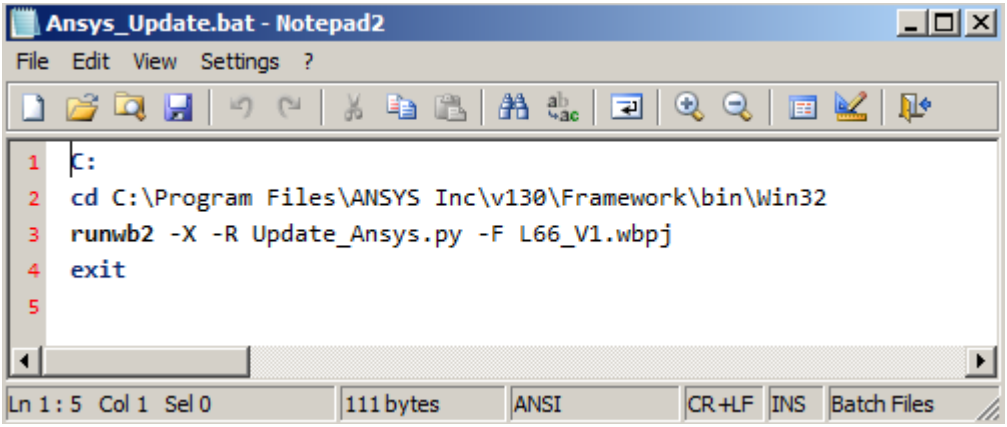


Figure 65 – Batch file to update ANSYS

- The Button that triggers this script for running the batch file for updating ANSYS can be seen in can be seen in **Figure 66**.

Figure 66 – Button to update ANSYS Workbench

- The batch file Ansys_Update.bat tell the system to open the corresponding analysis which depends on the selected model to run the following python script.

```
# IronPython imports to enable Excel interoperation,
import clr
clr.AddReference("Microsoft.Office.Interop.Excel")
import Microsoft.Office.Interop.Excel as Excel
from System.Runtime.InteropServices import Marshal

# import system things needed below
from System.IO import Directory, Path
from System import DateTime

# use the ANSYS function GetProjectDirectory to figure out what directory you are in
# and set that to the current directory
projDir = GetProjectDirectory()
Directory.SetCurrentDirectory(projDir)

# Open up a log file to put useful information in
logFile = open("Link.log","w")

# Put a header in the log file
logFile.write("==================================================\n")
logFile.write("Link Run Log File\n")
logFile.write("==================================================\n")
logFile.write("Start time: " + DateTime.Now.ToString('yyyy-mm-dd hh:mm:ss') + "\n")
logFile.write("Proj Dir: %s\n\n" % projDir)

# Use the Excel GetActiveObject funtion to get the object for the excel session
ex = Marshal.GetActiveObject("Excel.Application")
# Make Excel visible
ex.Visible = True

# Define the active workbook and worksheet
wb = ex.ActiveWorkbook
ws = wb.ActiveSheet

# In Excel: Grab values for the cells that we want data from (input cells)

Thickness = ws.Range["E13"](1,1).Value2
```

```
S_R_Top = ws.Range["E14"](1,1).Value2
Element_Size_Top = ws.Range["E15"](1,1).Value2
S_R_Bottom = ws.Range["E16"](1,1).Value2
Element_Size_Bottom = ws.Range["E17"](1,1).Value2
Element_Size_Rest = ws.Range["E18"](1,1).Value2
Force_Top_X = ws.Range["E19"](1,1).Value2
Force_Top_Y = ws.Range["E20"](1,1).Value2
Force_Top_Z = ws.Range["E21"](1,1).Value2
Force_Bottom_X = ws.Range["E22"](1,1).Value2
Force_Bottom_Y = ws.Range["E23"](1,1).Value2
Force_Bottom_Z = ws.Range["E24"](1,1).Value2
Moment_Top_X = ws.Range["E25"](1,1).Value2
Moment_Top_Y = ws.Range["E26"](1,1).Value2
Moment_Top_Z = ws.Range["E27"](1,1).Value2
Moment_Bottom_X = ws.Range["E28"](1,1).Value2
Moment_Bottom_Y = ws.Range["E29"](1,1).Value2
Moment_Bottom_Z = ws.Range["E30"](1,1).Value2

# In Excel: See if the user wants to save the project after the update
saveit = ws.Range["E31"](1,1).Value2

# In Workbench: Grab the parameter objects for the input values

Thickness_param = Parameters.GetParameter(Name="P1")
S_R_Top_param = Parameters.GetParameter(Name="P2")
Element_Size_Top_param = Parameters.GetParameter(Name="P3")
S_R_Bottom_param = Parameters.GetParameter(Name="P4")
Element_Size_Bottom_param = Parameters.GetParameter(Name="P5")
Element_Size_Rest_param = Parameters.GetParameter(Name="P6")
Force_Top_X_param = Parameters.GetParameter(Name="P7")
Force_Top_Y_param = Parameters.GetParameter(Name="P8")
Force_Top_Z_param = Parameters.GetParameter(Name="P9")
Force_Bottom_X_param = Parameters.GetParameter(Name="P10")
Force_Bottom_Y_param = Parameters.GetParameter(Name="P11")
Force_Bottom_Z_param = Parameters.GetParameter(Name="P12")
Moment_Top_X_param = Parameters.GetParameter(Name="P13")
Moment_Top_Y_param = Parameters.GetParameter(Name="P14")
Moment_Top_Z_param = Parameters.GetParameter(Name="P15")
Moment_Bottom_X_param = Parameters.GetParameter(Name="P16")
Moment_Bottom_Y_param = Parameters.GetParameter(Name="P17")
Moment_Bottom_Z_param = Parameters.GetParameter(Name="P18")

# In Workbench: Get the object for the deflection parameter vlue
Von_Mises_Param= Parameters.GetParameter(Name="P19")


#In Workbench: Set the value of the input parameters in Workbench using the values
```

```
#   we got from Excel

Thickness_param.Expression = Thickness.ToString()
S_R_Top_param.Expression = S_R_Top.ToString()
Element_Size_Top_param.Expression = Element_Size_Top.ToString()
S_R_Bottom_param.Expression = S_R_Bottom.ToString()
Element_Size_Bottom_param.Expression = Element_Size_Bottom.ToString()
Element_Size_Rest_param.Expression = Element_Size_Rest.ToString()
Force_Top_X_param.Expression = Force_Top_X.ToString()
Force_Top_Y_param.Expression = Force_Top_Y.ToString()
Force_Top_Z_param.Expression = Force_Top_Z.ToString()
Force_Bottom_X_param.Expression = Force_Bottom_X.ToString()
Force_Bottom_Y_param.Expression = Force_Bottom_Y.ToString()
Force_Bottom_Z_param.Expression = Force_Bottom_Z.ToString()
Moment_Top_X_param.Expression = Moment_Top_X.ToString()
Moment_Top_Y_param.Expression = Moment_Top_Y.ToString()
Moment_Top_Z_param.Expression = Moment_Top_Z.ToString()
Moment_Bottom_X_param.Expression = Moment_Bottom_X.ToString()
Moment_Bottom_Y_param.Expression = Moment_Bottom_Y.ToString()
Moment_Bottom_Z_param.Expression = Moment_Bottom_Z.ToString()

# Set the output values to "Calculating..." since they no longer match the input values
ws.Range["E34"](1,1).Value2 = "Calculating..."

# Now let Workbench go to town and update the systems using the new parameter values
logFile.write("Updating Project\n")

Update()

# If asked for, save the project

Save()

# Assign the value of the Excel deflection cell output deflection from Workbench
ws.Range["E34"](1,1).Value2 = Von_Mises_Param.Value.Value

# Now go through the value of each natural frequency in Workbench and
#   set the corresponding cell in Excel
#    This could be made more general or at least more concise by using a do loop
#    Also note that instead of getting the objects, then the values the two steps are
#    combined for these values

#ws.Range["Mode_1"](1,1).Value2 = Parameters.GetParameter(Name="P6").Value.Value
# Done!  Close the log file and move on

logFile.write("End time: " + DateTime.Now.ToString('yyyy-mm-dd hh:mm:ss') + "\n")
logFile.close()
```

- VB codes to run the batch file Ansys_Update.bat:

```
Sub Update_Ansys()
Working_dir = ThisWorkbook.Path
Bat_path = Chr(34) & Working_dir & "\WB_Script\Ansys_Update.bat" & Chr(34)
Shell Bat_path
End Sub
```

6. **Deleting files**

- After performing all the task mention in the previous points, the user can decided to delete the files copied before in step 2. The button that perform this action is the following:

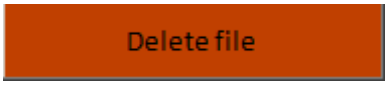


Figure 67 – Button to delete files

- VB code to delete files

```
Sub Delete_file()

Dim fs As Object
Dim oldPath As String, newPath As String
Working_dir = ThisWorkbook.Path
'newPath = Working_dir & "\WB_Script" 'Folder file is located in
oldPath = "C:\Program Files\ANSYS Inc\v130\Framework\bin\Win32" 'Folder to copy file
to
Set fs = CreateObject("Scripting.FileSystemObject")
fs.DeleteFile oldPath & "\" & "Run_Ansys.wbjn" 'This file was an .xls file
fs.DeleteFile oldPath & "\" & "Update_Ansys.py"
Set fs = Nothing

End Sub
```

Appendix 5

Different configurations achieved with approach 2 of Parametric CAD Model

Using the approach 2 for the construction of a parametric lower arm 26 different configurations was achieved changing the parameters and are presented in **Table 7**.

Tested configurations of the parametric arm					
Configurations	**Height [mm]**	**Thickness [mm]**	**Upper_dia [mm]**	**Lower_Dia [mm]**	**Width [mm]**
1	800	8	285	350	225
2	900	8	315	380	225
3	1000	10	315	380	215
4	1100	12	355	420	225
5	1200	8	335	400	215
6	1075	8	315	380	225
7	1300	12	450	500	255
8	1450	8	450	500	255
9	1450	10	350	500	225
10	1500	8	450	500	225
11	1550	8	450	520	270
12	1550	8	420	480	270
13	1550	8	420	540	270
14	1550	8	420	580	270
15	1550	8	400	580	270
16	1550	8	380	580	270
17	1550	8	360	580	270
18	1070	8	360	580	270
19	1070	8	330	580	270
20	1070	8	450	380	270
21	1075	15	315	380	225
22	1075	4	315	380	225
23	1550	8	315	380	225

24	1075	8	315	440	225
25	1075	8	315	340	225
26	1550	8	450	520	300

Table 7 - Lower arm configurations