Master of Science in Mechanical Engineering
June 2026

Simulation-Driven Design Automation for Mechanical Positioners

Development of a Metamodel-Based Configuration Tool

Kevin Samuelsson
Isak Andersson

Faculty of Engineering, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering. The thesis is equivalent to 20 weeks of full-time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Authors:
Kevin Samuelsson
E-mail: kevsam2002@gmail.com

Isak Andersson
E-mail: anderssonisak2001@gmail.com

University advisor:
Assistant Professor Ansel Berghuvud
Department of Mechanical Engineering

Faculty of Engineering
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Engineering design work often involve time consuming simulations to check that structural properties are satisfactory under load. In order to streamline the process of early design work, this work will present a configuration tool which with use of a metamodel intends to provide a way of getting a design proposal without running simulations. The configuration tool will be implemented on a positioner from Yaskawa. This work will use a positioner as a case study in building a configuration tool based on a parametric CAD model that uses metamodel data to circumvent simulations in early design stages.

The challenge of properly exploring a design space when gathering data for a metamodel is performed by using the sampling method Latin Hypercube Sampling (LHS). In addition, to reduce complexity of metamodel creation reduction of amount of design parameters is performed with usage of a parameter sensitivity analysis method called the Morris method. By usage of 500 generated samples (LHS), input data for metamodeling was created. The process involves creating a simulation automation loop, in Autodesk Inventor, using iLogic rules.

Using the results from simulation automation, a configuration tool is created within the Autodesk Inventor environment. The configuration tool takes user specified inputs, such as height and load capacity, and runs optimization on the metamodel to find a design proposal for the given inputs. Resulting proposed design is showed within the Inventor environment to the user. Overall outcome of the performed work provides a case-study in integration of metamodel in design work, final metamodel used in configuration tool has a relative error of 24%. The configuration tool provides a design proposal which can be used to further develop the design and is intended as an early approximation of what a final design can look like.

Keywords: Design Automation, Surrogate Model, Optimization, FEM, Engineering Tool

Sammanfattning

Ingenjörsarbete inom design innefattar ofta tidskrävande simuleringar för att kontrollera strukturella egenskaper. För att effektivisera design processen i tidiga konstruktionsfaser presenterar detta arbete ett konfigurationsverktyg som med hjälp av en metamodell är avsett att erbjuda ett sätt att ta fram ett designförslag utan att genomföra simuleringar. Konfigurationsverktyget kommer att implementeras på en lägeställare från Yaskawa. Detta arbete använder en lägeställare som fallstudie för att utveckla ett konfigurationsverktyg baserat på en parametrisk CAD-modell, där metamodell-data används för att kringgå simuleringar i tidiga designfaser.

Utmaningen att på ett bra sätt utforska designrymden vid insamling av data till en metamodell hanteras genom användning av samplingsmetoden Latin Hypercube Sampling (LHS). För att minska komplexiteten vid framtagning av metamodellen reduceras dessutom antalet designparametrar med hjälp av en känslighetsanalysmetod kallad Morris-metoden. Genom användning av 500 genererade datapunkter skapades indata för metamodellering. Processen omfattar skapandet av en automatiserad simuleringsloop i Autodesk Inventor med hjälp av iLogic-kod.

Med hjälp av resultaten från den automatiserade simuleringsprocessen skapas ett konfigurationsverktyg i Autodesk Inventor. Konfigurationsverktyget tar användarspecificerad indata, såsom höjd och lastkapacitet, och utför optimering på metamodellen för att hitta ett designförslag för de givna indata. Det resulterande designförslaget visas för användaren i Inventor.

Det övergripande resultatet av det genomförda arbetet är en fallstudie av integrering av metamodeller i konstruktionsarbete. Den slutliga metamodellen som används i konfigurationsverktyget har ett relativt fel på 24 %. Konfigurationsverktyget tillhandahåller ett designförslag som kan användas för vidareutveckling av konstruktionen och är avsett som en tidig approximation av hur en slutlig design kan se ut.

Nyckelord: Designautomatisering, Surrogatmodell, Optimering, FEM, Ingenjörsverktyg

Acknowledgments

We want to thank Ansel Berghuvud who has been the supervisor for the project. Ansel has provided help and guidance through the work in structure, report writing and simulations. We also want to thank the supervisors and engineers from Yaskawa who has been instrumental in providing insights and advice in design and component choice. Further, the access to positioner models and getting the opportunity to see positioner in the real world at Yaskawa has been of great importance for us during the work.

Contents

Abstract i

Sammanfattning ii

Acknowledgments iii

1 Introduction 2
- 1.1 Background . . . 2
- 1.2 Scope and Objectives . . . 3
- 1.3 Delimitations . . . 3
- 1.4 Research Question . . . 4
- 1.5 Outline . . . 4

2 Theory 5
- 2.1 Bearing load carrying capacity and life . . . 5
- 2.2 Drive torque and drive power of screw drive . . . 6
- 2.3 Critical speed of screw shaft . . . 6
- 2.4 Topology optimization . . . 7
- 2.5 Morris method . . . 7
- 2.6 Design of Experiments (Latin Hypercube) . . . 8
- 2.7 Metamodels . . . 9
  - 2.7.1 Response surfaces . . . 9
  - 2.7.2 Gaussian Process Regression . . . 9
  - 2.7.3 Neural networks . . . 10
  - 2.7.4 Decision Tree Ensembles . . . 11
  - 2.7.5 Support Vector Regression . . . 11
- 2.8 Optimization Algorithms . . . 12
  - 2.8.1 Gradient based optimization . . . 12
  - 2.8.2 Genetic Algorithm NSGA-II . . . 13
  - 2.8.3 Pattern search . . . 14

3 Related Work 15
- 3.1 Research on positioners . . . 15
- 3.2 Multi Criteria Optimization . . . 16
- 3.3 Optimization using metamodels . . . 16
- 3.4 Knowledge Based Engineering (KBE) . . . 17
- 3.5 Design Automation . . . 18
- 3.6 Research gap . . . 18

4 Method **20**
4.1 Research design . . . 20
4.1.1 Measures of success . . . 21
4.1.2 Choice of design and components for drivetrain . . . 22
4.1.3 Topology Optimization . . . 22
4.1.4 Modeling . . . 25
4.1.5 Simulation Set Up . . . 25
4.1.6 Sensitivity analysis . . . 26
4.1.7 Optimization algorithm for drivetrain components . . . 27
4.1.8 Rail calculations . . . 29
4.1.9 Design of Experiments . . . 31
4.1.10 Simulation . . . 31
4.1.11 Metamodel . . . 32
4.1.12 Configurator . . . 34
4.2 Verification . . . 35
4.2.1 Rail Model simplification . . . 35
4.2.2 FEM stress . . . 35
4.2.3 Mesh size . . . 36
4.2.4 Simplified test of the Morris method . . . 36

5 Results and Analysis **39**
5.1 Choice of design and components for drivetrain . . . 39
5.2 Topology optimization . . . 40
5.3 Parametric CAD model . . . 41
5.4 Sensitivity analysis . . . 43
5.5 Metamodel . . . 45
5.6 Optimization algorithms . . . 48
5.7 Final design proposals from the configurator . . . 49

6 Discussion **51**
6.1 Design . . . 51
6.2 Verification . . . 52
6.3 Choice of variables for metamodel . . . 53
6.4 Metamodel . . . 54
6.5 Resulting proposal . . . 54
6.6 Generalized method . . . 57

7 Conclusions and Future Work **58**
7.1 Conclusion . . . 58
7.2 Future Work . . . 59

References **60**

A Supplemental Information **65**
A.1 Tables . . . 65
A.2 Figures . . . 66
A.3 Matlab code . . . 70

A.4 iLogic code 86

List of Figures

2.1 Generic example of Gaussian Process Regression, visualized 10
2.2 Neural network, reproduced from [1] 10
2.3 Random forest, reproduced from [2] 11
2.4 Support Vector Regression, reproduced from [3] 12
2.5 Visualization of a generic Pareto front 14

3.1 Illustration of a single axis positioner from Yaskawa. [4]. 15

4.1 Flowchart over the research design used in this study. 20
4.2 Detailed view of the "Design and develop a configurator"-step. 21
4.3 Assembly for the Topology Optimization of the carriage bracket. . . . 23
4.4 Mesh of the assembly for the Topology Optimization of the carriage bracket. 24
4.5 Faces used for constraint and applied loads in simulation. 26
4.6 FEM setup for rail calculations. 30
4.7 Morris sensitivity test on rectangular bar. 37
4.8 Simplified model for Morris test 38

5.1 Resulting structure for the carriage bracket with target volume fraction 10%. 40
5.2 Resulting structure for the carriage bracket with target volume fraction 40%. 40
5.3 Design of Carriage Mount with inspiration from the Topology Optimization. 41
5.4 Positioner assembly in Autodesk Inventor with part numbers attached to respective part. See table 5.1 for part descriptions. 43
5.5 Sensitivity analysis of the complete model showing parameter influence on stress. 44
5.6 Comparison of metamodel predictions (GPR and Neural Network). . 46
5.7 Comparison of metamodel predictions (LS Boost and SVR). 47
5.8 Configurator input 49
5.9 Configurator output 50

A.1 Morris sensitivity test on simplified model. 66
A.2 Sensitivity analysis of the complete model showing parameter influence on displacement. 67
A.3 Sensitivity analysis of the complete model showing parameter influence on stress, zoomed 68

A.4 Sensitivity analysis of the complete model showing parameter influence on displacement, zoomed. 68
A.5 Positioner assembly in Autodesk Inventor with part numbers attached to respective part. See table 5.1 for part descriptions. 69
A.6 Positioner assembly in Autodesk Inventor with part numbers attached to respective part. See table 5.1 for part descriptions. 69

List of Tables

4.1 Inputs for the configurator 34
4.2 FEM stress comparison table 36

5.1 Table of parts used in assembly. First column part number, second column part name and third column amount of given part that is used in assembly. 42
5.2 Decisions made for which parameters was to be included in the meta-model. 45
5.3 Mean relative errors (MAPE) of predicted stress and mass for the metamodels. 48
5.4 Comparison of 3 tested optimization algorithms using LS Boost for stress and SVR for mass. 48
5.5 Comparison of 3 tested optimization algorithms using SVR. 48

A.1 Boundaries of the parametric model 65

List of Algorithms

1 Matlab code: Morris screening. 70
2 Matlab code: GA_optimization. 74
3 Matlab code: Objective of component optimization. 75
4 Matlab code: Non linear constraints of component optimization. . . . 76
5 Matlab code: Latin Hypercube sapmpling. 78
6 Matlab code: Metamodels. 79
7 Matlab code: Optimization for configurator. 84
8 iLogic code: Run simulations main code. 86
9 iLogic code: Change size. 89
10 iLogic code: Code necessary for simulations. 97
11 iLogic code: Run simulation. 98
12 iLogic code: Read stress and displacement. 100
13 iLogic code: Configurator main code. 102
14 iLogic code: Run Matlab. 102
15 iLogic code: Display solution summary. 103

Chapter 1

Introduction

1.1 Background

In manufacturing or other industrial processes, handling and orientation of large and heavy workpieces are often necessary. For such tasks, a so called positioner can be used. This project involves work based on a positioner assembly, which is produced and assembled by Yaskawa. Yaskawa is a company which provides automation and production processes and equipment with emphasis on robotics and mechantronics, with products and solutions ranging from assembly lines, welding solutions, packaging and more. A positioner of the kind in this project is used to hold larger structures, typically car bodies or other larger structures intended to be for example welded during holding in positioner. Positioners from Yaskawa currently range in load capacities and sizes, with current load capacity up to 10 tonnes. The positioner holds the workpiece and can rotate it, in order to access different sections of the workpiece without having to put it down for reorientation.

Current positioner, so called HSB positioner from Yaskawa does not come standard with height adjustability but can be custom built upon request. When custom order of height adjustable HSB is made, a new design or at least not standardized design is created by Yaskawa, which has led to many differing designs of what in principle can be standardized. With many design alternatives for height adjustable positioners, maintenance work and general understanding of the engineering solution can become tedious and unsustainable. Since the company now has to accommodate and build many different versions of positioners, instead of having one standardized solution.

This is a common problem in industries since having too high product variety can decrease the productivity and quality. Higher variety can increase the manufacturing cost and make the assembly process more complex. This is some of the reasons why it is good to standardize the products, but there are also disadvantages with to low product variety which is that market shares could be lost if the standard products does not fit the needs of the customers [5]. Therefor this is a about keeping a balance between keeping the cost down while still having large enough product range to satisfy as many customers as possible.

Further, if existing way of working with height-adjustable positioners withstands, the risk of doing redundant work is high. If there is no standard of what the design should look like or consist of, then the engineers working on design have "free reign" to do any design that works, which might include something which has been designed before. The standardization of HSB positioners with height-adjustability is the core

of what this work intends to explore. In essence, standardization will involve limiting choices to the vertical movement components for example screw-movement and bearing solution.

This standardization work can be costly and time-consuming, and since standard products will be built in high volume there is a value in trying to optimize the products as much as possible. To help with these aspects, design automation will be used. Usual areas where design automation is useful is: to reduce cost, cut lead time, improve product performance and the possibility to adapt products to different costumer specifications [6]. The first three aspects here is highly relevant when designing new standard products. But the possibility to quickly adapt the products for special request is also valuable for when the standard products is not enough for the customers requests.

Design automation will be used in form of a configuration tool that will be used by engineers in the early stages of design and to streamline the process of choosing components and design parameters without having to run any FEM-simulations. By reducing the need for FEM-simulations, it is possible to faster produce new design concepts since the FEM-simulations of this kind usually have long solving times. In general the thesis will to provide a study in application of optimization, metamodeling and creation of a configuration tool with a specific case study on a positioner assembly.

1.2 Scope and Objectives

The aim of this thesis is to create a design automation tool which implements additional features on an already existing design with the goal to optimize the structure. This can be divided into two parts, the first is to design what the further development is to be, and in this case is performed by using off-the-shelf products and put these together in an optimized way. The second part is to create an automation tool for the design with the new developments included, which should handle different configurations of components and design choices. Lastly, the design should be optimized with respect to weight and cost while still meeting the structural requirements of strength and stiffness.

1.3 Delimitations

In order to properly perform above said aim, the project will be limited to only cover one product line: HSB positioners which has one axis of rotation and one lifting axis. However, the process of developing a configuration tool which uses a metamodel as a backbone will still be relevant to other similar products. Additionally the analysis is limited to focus solely on structural stress in the critical components of the positioner, which is the vertical movement system. Further, the load case is assumed to be of similar nature between the different load capacities of positioner, with a vertical downward force on the fixture and a moment on the fixture disc, but with different magnitudes corresponding to the load capacity. The positioner floor mounting and other accessories than the core positioner are also left out of analysis.

1.4 Research Question

- How can a configuration tool be created to provide a design proposal in early stages of design work on a positioner?

1.5 Outline

The thesis is divided into 7 chapters which will be explained briefly below.

Chapter 1 contains the introduction of the thesis which includes background information about the company and product line which will be used for the case study. Thereafter, scope, objectives, limitations and research question are stated. With the introductory information and objective of the project stated, following chapter contains the necessary theory.

Chapter 2 explains the theory needed for the thesis, it begins with several formulas for the dimensioning of the drivetrain. After that comes brief explanations of the methods used, ranging from Topology optimization, Morris method, Latin Hypercube, metamodels and optimization algorithms. In order to justify our thesis academically the next chapter presents a review of related works that are applicable and related to this thesis scope.

Chapter 3 is a review of the works related to this study. They are divided into 5 groups: research on positioners, multi criteria optimization, metamodels, Knowledge Based Engineering and Design Automation. Finally a research gap is defined based on the review of related works. Following the related work review, the thesis will go through and present the methods used throughout the thesis.

Chapter 4 includes all the steps of the method used in this thesis. It begins with the research design and a summary of all steps which are thereafter explained in detail. The end of this chapter goes through the verifications performed together with results that gives an indication to what degree the later results can be trusted. With the method and surrounding verification work, the next step in the thesis is to present the results.

Chapter 5 presents the results from all parts of this study together with analysis that clarifies how the results can be interpreted. Following results and subsequent analysis, the results are discussed.

Chapter 6 discusses the meaning of the results both in detail and in a bigger picture. From the discussion, some conclusions of the performed work are to be drawn, along with answering the research question.

Chapter 7 contains the main conclusions that can be drawn from the work and ends with possibilities of future works that could be explored which extends beyond the scope of this thesis.

Chapter 2

Theory

2.1 Bearing load carrying capacity and life

The load capacity is divided into two parts: static and dynamic load carrying capacity. The static load capacity is limited by plastic deformations of the bearing which mostly affects the noise level of the bearing during operation. The static load safety factor S_0 is calculated with from the basic static load rating C_0 of the bearing and the equivalent static bearing load P_0 (Equation 2.1) [7]. Recommended static safety factors for ball and roller bearing varies greatly for different applications. Schaefler recommends safety factors S_0 ranging from 1 to 3 depending on application [7], but none of these includes overhead arrangements. Rexroth recommends static safety factor $S_0 \geq 12$ for overhead arrangements and applications representing a high hazard potential [8].

$$S_0 = \frac{C_0}{P_0} \tag{2.1}$$

In cases where there is both axial and radial loads, an equivalent static bearing load P_0 is calculated from the radial and axial loads F_r and F_a together with the radial and axial load factors X_0 and Y_0 (Equation 2.2) [7].

$$P_0 = X_0 \cdot F_r + Y_0 \cdot F_a \tag{2.2}$$

The operating life of the bearing measured in hours can be calculated from (Equation 2.3) [7] where L_{10h} is the basic rating life in operating hours, that is reached by 90% of identical bearings before the first indications of material fatigue appear. C is the basic dynamic load rating for the bearing, P is the equivalent dynamic bearing load, p is the life exponent which is 3 for ball bearings and n is the operating speed.

$$L_{10h} = \frac{10^6}{60 \cdot n} \cdot \left(\frac{C}{P}\right)^p \tag{2.3}$$

If the speed is varying, it is necessary to calculate the equivalent speed. For variation in steps this is calculated using (Equation 2.4) [7] where q_1 is the percentage of time for the corresponding speed n_1.

$$n = \frac{q_1 n_1 + q_2 n_2 + ... + q_z n_z}{100} \tag{2.4}$$

2.2 Drive torque and drive power of screw drive

The drive torque (Nm) required to convert rotary motion into linear motion is calculated from (Equation 2.5) [9] where F_e is the external load (N), P_h is the lead (mm) and η_s is the efficiency which is approximately 0.8 for planetary screw drive.

$$M_{ta} = \frac{F_e \cdot P_h}{2000 \cdot \pi \cdot \eta_s} \tag{2.5}$$

The drive power (kW) required is calculated with (Equation 2.6) [9] where n is the speed (rpm) and η_g is the efficiency of the gearbox. In addition to this it is important to consider the bearing frictional torque from the seals, dynamic drag torque resulting from preloading and check that the drive torque is less than the permissible torque for the end machining.

$$P_a = \frac{F_e \cdot n \cdot P_h}{60 \cdot 10^6 \cdot \eta_g} \tag{2.6}$$

2.3 Critical speed of screw shaft

A roller screw has a natural frequency in bending mode which is depending on the position of the nut and the size of the screw. It is important to not excite its natural frequency when rotating the screw and to avoid this the rotational speed needs to be kept below the critical speed of the screw shaft. If this speed is exceeded it could lead to radial deflection of the beam that could permanently bend the screw shaft. This critical speed n_{cr} (rpm) is calculated with (Equation 2.7) [9]

$$n_{cr} = 49 \cdot 10^6 \frac{f_1 d_2}{l_{cr}^2} \tag{2.7}$$

where f_1 is a mounting factor which is 0.9 for fixed-free mounting type, d_2 is the root diameter (mm) of the screw and l_{cr} is the free length (mm) which in fixed-free mounting is the distance between the free screw end and the highest position of the nut. As a general rule a safety factor of 0.8 should be used after calculating the critical speed because of potential inaccuracies in the mounting of the screw assembly.

The permissible rotational speed is also limited by the rollers inside the nut. For planetary roller screws this permissible speed n_p (rpm) is calculated with (Equation 2.8) [9]

$$n_p \leq \frac{160 \cdot 10^3}{d_0} \tag{2.8}$$

where d_0 is the nominal diameter (mm).

2.4 Topology optimization

Efficient material usage is important for mechanical design where it is desirable to minimize mass while structural properties are kept. Topology optimization (TO) is a optimization method which can be implemented to provide a way to minimize mass while maintaining structural properties. The essence of topology optimization is to let a structural criteria, for example stress, displacement or mass dictate how the geometry should be formed. Several different variants of implementing topology optimization exists, as mentioned by Tang et al. [10], where the so called element-based method is applicable and used to optimize geometry in the work performed in this thesis. The element based method uses FEA where a preexisting geometry is meshed into nodes and elements. Calculations using FEM are then performed to asses if the constraint is satisfied, where constraint can be for example stress, displacement or forces (in nodes or elements). Preexisting geometry is typically an already existing part or model which is to be optimized or a generic shape which covers the total allowed volume for the design. Through usage of the so called solid isotropic material with penalization (SIMP) [11] method the material density for nodes are continually updated to minimize mass while adhering to the set constraint. Topology optimization will involve multiple iterations of calculations and nodal density adjustment until convergence with set objective and constraint. Resulting optimized model will consist of nodes and elements which corresponds to a minima in mass while satisfying the constraint.

Software used to implement topology optimization in this thesis, is Autodesk Inventor Nastran. The described methodology above, using FEM and SIMP, is the methodology which Autodesk Nastran uses to perform topology optimization [12].

2.5 Morris method

The Morris screening method is a method for evaluating the impact of different parameters in a design, which is helpful for deciding what parameters are important to include when optimizing a design. This method is made for exploring the design space in a efficient way with a relatively small number of simulations. This method works by traveling a trajectory through the design space by only changing each parameter once for each trajectory. The number of simulations needed is $r \cdot (k+1)$ where r is number of trajectories and k is the number of design variables. The trajectory is randomized each time where the trajectory design matrix is given by b^* (Equation 2.9) [13].

$$b^* = (J_k x^* + \frac{\Delta}{2} \times ((2 \times b - J_k) \times d^* + J_k) \times p^* \tag{2.9}$$

where

- J_k is a $(k+1) \times k$ matrix of 1s.
- x^* is a random starting point row vector.
- Δ is the step size.

- b is a strictly lower triangular matrix of 1s, with dimensions $(k+1) \times k$.
- d^* is a k-dimensional diagonal matrix which each element is either +1 or -1 with equal probability. This determines whether a parameter value will decrease or increase.
- p^* is $k \times k$ random permutation matrix in which each row contains one element equal to 1, all other are 0 and no two columns have 1s in the same position. This matrix determines in which order the parameters are changed.

For each step an elementary effect is then calculated which is used to derive sensitivity information about the parameter. The elementary effect is calculated with (Equation 2.10) [14].

$$EE_k = \frac{Y(X_1, ..., X_k + \Delta, ..., X_d) - Y(X_1, ..., X_k, ..., X_d)}{\Delta} \tag{2.10}$$

This is used to calculate the following sensitivity indices (Equation 2.11-2.12).

$$\mu_k^* = \frac{1}{r} \sum_{i=1}^{r} |EE_k^r| \tag{2.11}$$

$$\sigma_k = \sqrt{\frac{1}{r} \sum_{i=1}^{r} (EE_k^r - \mu_k)^2} \tag{2.12}$$

where μ_k is μ_k^* without the absolute value.

The Morris method can be made even more efficient by adding another method for choosing trajectories that was proposed by Campolongo et al [13]. This idea is to select the trajectories to maximize their spread in the input space, which is done by generating a large number (M∼500-1000) of morris trajectories and then select the subset of r (e.q. $r = 10$) with the highest spread. The spread is based on the following definition on distance d_{ml} between two trajectories m and l (Equation 2.13)

$$d_{ml} = \sum_{i=1}^{k+1} \sum_{j=1}^{k+1} \sqrt{\sum_{z=1}^{k} [X_i^m(z) - X_j^l(z)]^2} \tag{2.13}$$

where $X_i^m(z)$ is the z-th coordinate of the i-th point of the m-th Morris trajectory.

2.6 Design of Experiments (Latin Hypercube)

When dealing with a design space consisting of many parameters and variations, it is of interest how the design space can be explored in an efficient way. The design space is limited by upper and lower boundaries for each respective parameter and Latin Hypercube sampling (LHS) is one widely used method which provides a methodology to distribute sampling of the design space. In essence, Latin Hypercube sampling method divides the design space into non-overlapping segments of samples in a pseudo-randomized manor and provides a good foundation for usage in surrogate (or meta) models. For further and more in depth reading on LHS see McKay et al. [15].

2.7 Metamodels

A metamodel can be described as "a model of the model" which has the purpose of making faster approximations of simulations that would otherwise have long running times. The metamodel is supposed to take a vector of design variables X as input and then compute a vector of responses Y as output. A metamodel is mathematically described as follows: If the true function for the simulation is

$$y = f(x)$$

then the metamodel is

$$\hat{y} = g(x) \implies y = \hat{y} + \epsilon$$

where ϵ represents both the error of the approximation and random errors. There exist many different types of metamodels, some of the most common types are: response surfaces, neural networks, inductive learning and kriging [1]. The types of metamodels considered in this thesis is briefly explained below.

2.7.1 Response surfaces

Response surfaces are usually low order polynomials that are fitted to the existing data by using least squares analysis. Tools for making response surfaces are well established and easy to use. This type of metamodel is appropriate for applications with few variables (less than 10) and is best suited for problems with random errors [1].

2.7.2 Gaussian Process Regression

Gaussian Process Regression (GPR) is a method for creating a metamodel that also gives a measure for uncertainty. To understand how the method works, start by imagining an infinite amount of smooth functions in a diagram. Then apply the training data points and remove the functions that does not go through (or close to) the data points. That could look like the blue area in (Figure 2.1) where the red points are the training data. GPR will then use the mean of this blue area as function to predict the values in between, while the blue area shows the uncertainty. In other words the GPR uses gaussian distribution as a function of mean and standard deviation to predict values. It is also possible to control how smooth the functions need to be from covariance which is controlled by which kernel function is used [16] [17].

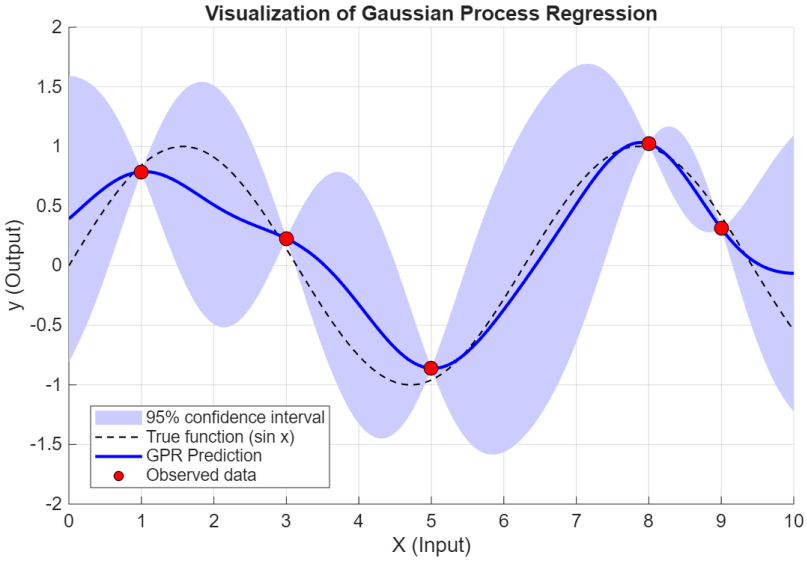


Figure 2.1: Generic example of Gaussian Process Regression, visualized

2.7.3 Neural networks

Neural networks are inspired of the biological brains structure and consists of multiple neurons in layers. Each neuron performs non-linear transformations on the inputs and sends the output to the next layer. Each neuron has weights that controls the strength of each input and a bias value that shifts the threshold for when the neuron should activate to send the information forward. These weights and biases are determined by training the model on a set of training data points [18]. A visual representation of a neural network is shown in (Figure 2.2).

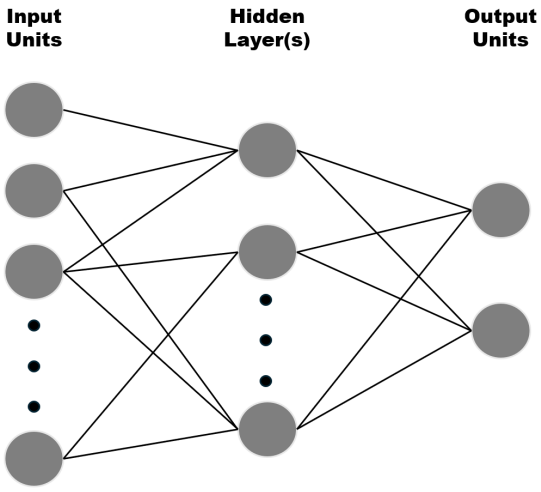


Figure 2.2: Neural network, reproduced from [1]

2.7.4 Decision Tree Ensembles

Combining multiple simple decision trees can make powerful learning algorithms. Decision trees are made up of nodes where each node asks a simple question to decide which path it should take. Multiple decision trees can then be combined into an ensemble of decision trees. There are mainly two types of ensembles: bagging and boosting. In bagging all trees are trained independently on a random subset of the training data, a popular algorithm of this type is Random Forest (Figure 2.3). In boosting each tree is trained sequentially, where each tree is focusing on the observations that the previous trees misclassified [19].

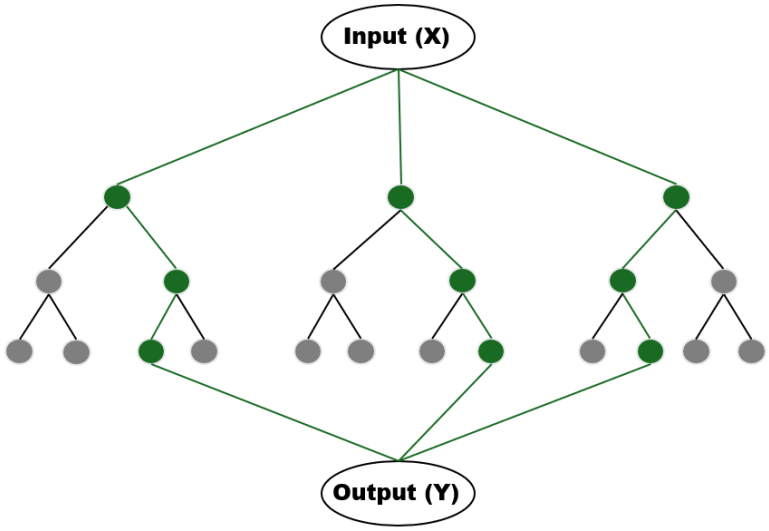


Figure 2.3: Random forest, reproduced from [2]

2.7.5 Support Vector Regression

Support Vector Regression (SVR) is a variant of Support Vector Machines (SVM), SVM is popular to use for classification problems but with a few small changes it can be turned into SVR to use for regression. SVR works by trying to find a function f(x) that deviates from y_n no more than a value ϵ for each training point, while having the function as flat as possible [3].

In cases where it is not possible to find this function because of points that lies outside of ϵ a slack variable ξ can be used which is a soft margin to allow errors up to the value of ξ, see (Figure 2.4). In nonlinear cases it is possible to transform the data points with a nonlinear kernel function, which is usually gaussian or polynomial [3].

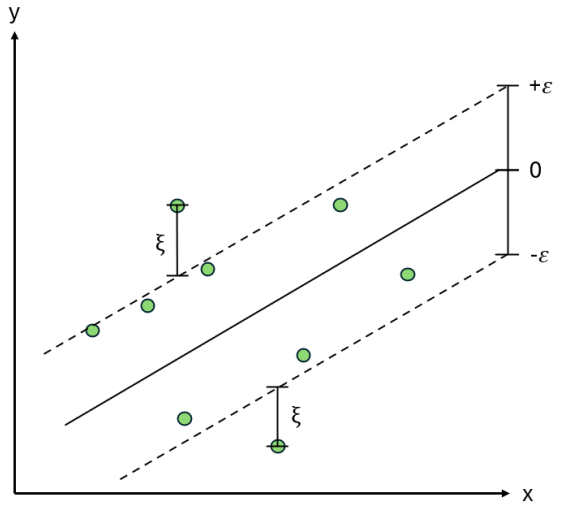


Figure 2.4: Support Vector Regression, reproduced from [3]

2.8 Optimization Algorithms

Optimization in general can be described in following way, for minimization, if we let f be the function to optimize and x be inputs to said function.

$$min(f(x)) \tag{2.14}$$

Engineering optimization problems often fall within the realm of constrained problems, with variables, x, that are limited by upper and lower boundaries. Further, nonlinear inequality (g_i) and equality constraints (h_j) may be introduced to ensure that the result from optimization satisfies for example stress thresholds, mass or displacement.

$$g_i(x) \leq 0, \quad i = 1, ..., m \tag{2.15}$$
$$h_j(x) = 0, \quad j = 1, ..., m \tag{2.16}$$

This framework of function f, inputs x and constraints g_i and h_j is broadly descriptive of how optimization is performed and is foundational for each of the following optimization algorithms to be presented [20].

2.8.1 Gradient based optimization

Gradient based optimization is a methodology for optimization with the core being based on the gradient, or the slope of the given function that is to be optimized. [20] For a continuous and sufficiently smooth objective function, gradient based optimization provides one way to find a local optima via the usage of the slope of the function. Gradient based optimization uses gradient information to determine in which direction to guide the search toward an optimum, for minimization the algorithm typically follows in a descent direction of which the gradient is negative and converges toward a local minima.

2.8.2 Genetic Algorithm NSGA-II

Evolutionary Optimization is a commonly used procedure for solving different types of optimization problems, both single and multi-objective optimization. Evolutionary Optimization can be used for a wide variety of problems and differs from classical optimization methodologies in a few ways that can be of an advantage for certain types of problems. One difference is that it does not use gradient information but instead uses direct search procedures. Another difference is that Evolutionary Optimization uses more than one solution each iteration (a population approach). The advantages of this is that it can efficiently search the whole solution space at the same time and provide multiple optimal solutions. Evolutionary Optimization uses stochastic operators which uses higher probabilities for more desirable solutions the deterministic operators with predetermined and fixed rules used in classical optimization methods. This is an advantage for solving complex problems with a global perspective [21].

The first step of Evolutionary Optimization is to create a population of solutions, this is usually created randomly within the specified lower and upper boundaries for each variable. Thereafter an iterative process begins that uses the current population to create a new population by using 4 main operators: selection, crossover, mutation and elite-preservation. The selection operator chooses which solutions are allowed to reproduce and pass their genetic information to the next population by giving the better solutions higher probability to fill the mating pool. Then there are two variation operators: crossover and mutation which are used to create a modified population. The crossover operator picks two or more parent solutions randomly from the mating pool and exchanges information from the parents to create one or more solutions for the next generation. Each child solution can then be mutated where each variable has a probability p_m of being mutated. This probability is usually set so that on average one variable per solution is mutated. The elitism operator makes sure that the algorithm is monotonically non-degrading by combining the old population with the new and chooses to keep better solutions from the combined population. This iterative process continues until a predetermined termination criteria is met [21].

Some problems have multiple objectives that should be minimized or maximized and with multiple objectives there will also be multiple optimal solutions. The solutions of this kind of problem can be plotted as points on the objective space where a non-dominated front (also called Pareto front) can be drawn from a set of these points (see Figure 2.5). The non-dominated points have one specific property which is that non of these points can improve in one objective without degrading at least one other objective [21].

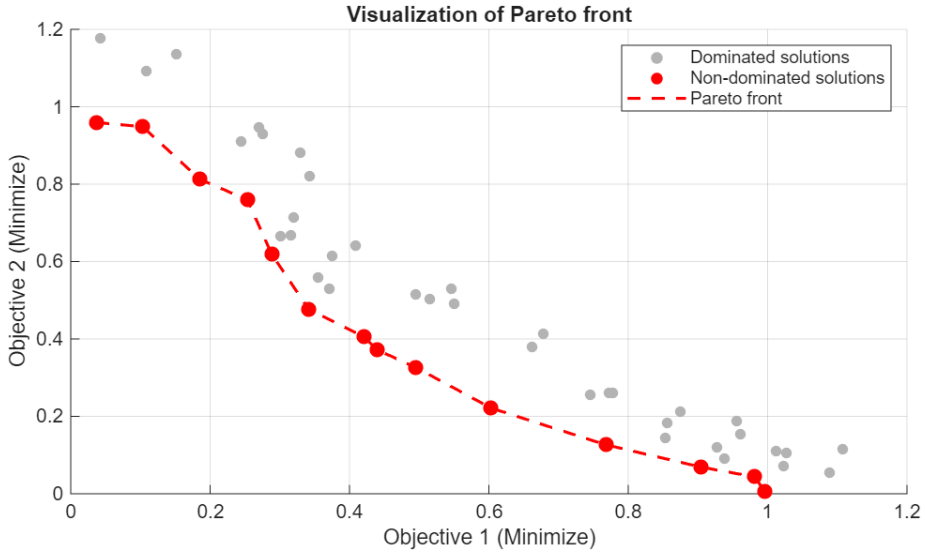


Figure 2.5: Visualization of a generic Pareto front

One popular variant of the Evolutionary Multi-objective Optimization procedures is Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II). NSGA-II tries to find multiple Pareto-optimal solutions in a multi-objective optimization problem and has the three following features [21]:

- Elitism approach
- Diversity preservation (Crowding distance)
- Emphasizes non-dominated solutions

2.8.3 Pattern search

Pattern search is a non-derivative based optimization method. In comparison to gradient based optimization, more function evaluations is needed but is less likely to get stuck in local optimum. Pattern search works on non-continuous functions since it doesn't use any derivatives when finding optimum, instead comparing function evaluations. Usage of pattern search in this work is the MATLAB function *patternsearch*, for further and more in depth reading on pattern search see C. Audet and J. E. Dennis paper "Analysis of generalized pattern searches" [22].

Chapter 3

Related Work

3.1 Research on positioners

A positioner is a type of machine that is used to rotate and move a workpiece into a wanted position so that a robot can reach all necessary places of the workpiece. These are often used for welding and a common example of a positioner with a single rotation axis called HSB from Yaskawa is shown in (Figure 3.1).

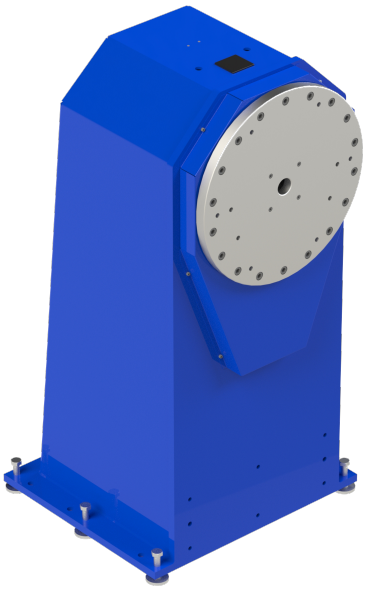


Figure 3.1: Illustration of a single axis positioner from Yaskawa. [4].

Welding Positioners are not mentioned much in research and literature, but is mostly based on manufacturers own procedures as is mentioned in [23] where they trace the creative process of developing and implementing an L-type positioner. This included both simulations of reach when cooperating with robots and strength analysis using FEA.

Another type of positioner was studied by Lei et al. [24] (machine translated from Chinese to English), the type of positioner used was similar to one considered in this thesis except that it lacks the z-axis. This paper includes selection and calculations on the transmission system and usage of FEM to optimize the structural design.

3.2 Multi Criteria Optimization

Structural multi criteria optimization is also relevant to this project, where intensions are to optimize for multiple aspects (weight, speed, cost). Existing work is focused on specific categories of structures or products with ranging applicability to this project. A topological optimization of multiple assembly aeronautical shelves were researched by Wang, J. et al. [25], where modular structural components are optimized with consideration to different objectives ranging from cost to ease of transport.

Similar research regarding multi criteria optimization is the area of multidisciplinary design optimization (MDO), which intends to support and enable engineering work to be performed with modularity, design automation and knowledge based engineering. As presented by M. Tarkian, who developed templates that are meant to be the backbone when designing modular structures or assemblies which are evaluated automatically [26]. Further M. Tarkian implements metamodels in the evaluations, which is something we wish to accomplish, and thus his paper is of great relevance to the work to be performed.

Research of similar optimization methods have also been implemented for infrastructure, where Zawidzki M. and Jankowski L. [27] lays forward an approach for multi criteria optimization with objectives to optimize for being the geometrical and structural properties. Focus of Zawidzki and Jankowskis research [27] was on a modular reconfigurable ramp system used in pedestrian traffic environments.

3.3 Optimization using metamodels

Modeling and optimizing complex systems are an ongoing research area where metamodels are commonly used because of the time consuming simulations. For these systems it can be necessary to perform multi-objective or multi-disciplinary optimization which are areas that Persson investigated in [28]. Specifically metamodels was used on a geometrical model of a robot link, this includes Kriging, Neural Networks, Radial Basis functions, Gaussian Processes and polynomial metamodels. Variables used was a continiuos thickness and a discrete actuator. The result was that Kriging, Neural Networks and Radial Basis functions were most accurate, with anisotropic Kriging as the top candidate.

Other areas where metamodels have been used are optimization of electric motors [29] where 50 FEAs where performed using design points created by Optimal Latin Hypercube consisting of 5 design variables. 11 metamodels were created for each output, these where variants of Radial Basis functions, Polynomial Response surfaces, Kriging, Ensamble of Decision trees and Neural Network. The results were different depending on which output was used. Metamodels that performed best on an output were Radial Basis functions, Kriging and Neural Network.

Another case were metamodels have been used is the optimization of wind turbines [30] where 8 design variables was used. Two metamodels were tested: Random Forest and Neural Network, where Random Forest performed better. The optimization was solved with NSGA-II to create a Pareto front.

Structural optimization based on metamodeling was also investigated by Park et al. [31] where polynomial response surface and Radial Basis functions was used.

Their training points was also chosen by Latin Hypercube which seems to be the most popular method for choosing training points. This was tested on 3 different cases and the optimization was solved with Matlabs *fmincon* function. All this was put together in a seamless CAD-CAE integration framework which cuold be used as a tool for structural engineering designers. They claimed that response surfaces were more suitable when less than 10 variables was used and Radial basis function was better with more variables.

3.4 Knowledge Based Engineering (KBE)

Knowledge based engineering (KBE) is a well populated field of research, with implementation in a wide range of different applications and topics. The core of knowledge based engineering is to capture knowledge gathered during design and automate repetitive engineering tasks. The process of KBE is completely in line with what this report is set out to explore and implement, with both design work and design automation which aims to streamline and build knowledge for design work. To provide an outline of sources for different implementations or studies of KBE, three different research projects will be presented below, which covers the basics, implementation and a more system based approach of KBE.

Firstly, a broad overview of the state of the field of KBE is performed by Verhagen et al. [32] where the various applicable areas and challenges for successful implementation of KBE is stated and discussed. They state that the need for KBE largely comes from the wish to 'mass customize', or that companies in a competitive business area tends to move toward custom solutions for individual customers. Thus the need for quick iterative design is of interest, which KBE can provide. The bulk of the authors work consists of reviewing 50 papers in the field of KBE and identifying shortcomings and proposing solutions to the identified shortcomings. All in all the research shows areas in implementation of KBE which can be improved and provides concrete answers to how improvements can be made.

Additionally work on automation of mechanical design, Pastor et al. [33] set out to encapsulate the whole process of optimizing for different objectives in design. Their work is performed on a robot manipulator as a case study, where the design optimization involves multiple segments of design where they state some examples: kinematics design, mechanical topology and drive selection. The proposed method was to use a genetic optimization algorithm for the kinematic structure and iterative design automation setup for the CAD design.

Another, system based approach of KBE is performed by Chapman and Pinfold [34] where the process of broadening the scope of design for engineers working on automotive bodies. The performed work takes knowledge from existing process, technical literature and expert engineers and builds a KBE application. The application performs optimization with the objective to reduce weight of the so called body-in-white (BIW) of a car. As stated by the authors, they develop a design analysis response tool (DART) which is intended to be used as a helping tool in design or analysis work on the BIW. Resulting DART and surrounding method uses both computer aided finite element analysis, manual design inputs and optimization to provide a quicker way of performing analysis or design iterations for the engineers.

3.5 Design Automation

Design automation can be used to reduce cost, cut lead time and to improve the product performance. Tasks that can be suitable for design automation are according to Cedefeldt et al. [6]:

- Selecting components according to given rules.
- Using design tables for variant design.
- Using design tables with additional preplanned changes in topology for variant design.
- Using a rule base to combine a set of given components to meet desired product performance.
- Optimizing and improving existing function or products to meet new conditions and demands.

One way to automate the design optimization process is to make a configuration tool where a user can define some input parameters from which the configuration tool then automatically creates a design proposal that follows these requests.

One example is where Cinar et al. [35] creates a configuration tool that automatically creates optimized gear pump housings. They did this by integrating the SOLIDWORKS API with a custom C# desktop application. The optimizations were performed by using SOLIDWORKS integrated "Design study" tool, with the objective of minimizing mass while maintaining mechanical integrity. Another configuration tool was created using Autodesk Inventor and the iLogic module by Grkovic et al. [36], which was used to design strongrooms based on costumer requests. This tool could also automatically create the technical documentation needed like 3D models, 2D drawings and Bill of Materials.

3.6 Research gap

From the review of related works there has been many studies on multi criteria optimization, focusing on combining multiple components or products. There has also been many studies focused on optimizing geometrical and structural properties, both with or without the help of metamodels. However, the combination of optimizing off-the-shelf products and geometrical properties is not as well explored.

Regarding the research on positioners there has been few mentions of this area in research and most of the design process is based on manufacturers own procedures. There where not much focus on optimization algorithms or metamodeling in the literature about positioners, thus there is an opportunity to add value to this by performing a case study in application of optimization techniques and metamodeling. Since the design process of a positioner includes both choosing off-the-shelf product and structural dimensioning it would be suitable to combine the optimization of both these areas which is not yet that extensively explored.

Additionally the research shows that it is necessary to test multiple metamodels on each new use case to see which is best since there is no way beforehand to know which model is most suitable for this particular use. Therefore it is beneficial to use a method where it is easy to test and compare multiple metamodels.

Lastly configuration tools have been proven to be useful for a range of different problems, so a tool like this could be valuable as a way of streamlining the early design process of positioners, which is the goal of this study. Further, to incorporate metamodeling and optimization in a configuration tool it will require additions to the existing research for building a configuration tool, and is something that is intended to be explored in this thesis.

Chapter 4

Method

4.1 Research design

The research method intended to use is the Design Science Research (DSR) [37], which can be broken down into 6 steps (Figure 4.1). The first step is to define the problem that is intended to be solved. Then the objectives/needs of the solution will be defined to clarify what the solution should accomplish. During the first and second step, information from the company about design and decision making is of importance and is intended to be performed with meetings at the companies facilities. This method of communication with the company is maintained during the entire thesis. General information in the related fields, positioners, optimization, metamodeling and configuration tools is also gathered during this phase, and is formulated into the related works chapter of this thesis (Chapter 3), papers and studies was searched for using Google Scholar and the Summon - BTH database.

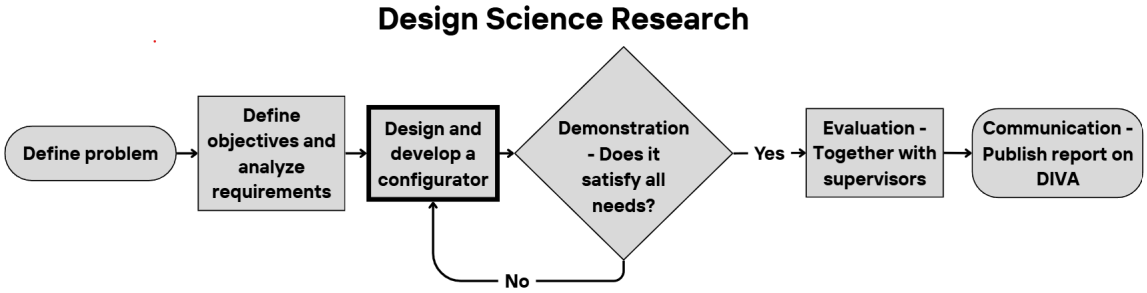


Figure 4.1: Flowchart over the research design used in this study.

The third step is to design and develop an artifact which is the configuration tool, a graphical view of this step is shown in (Figure 4.2). This is the main step of the research consisting of multiple steps where a range of methods are combined to develop this artifact. The first step is to choose what type of linear movement system that should be used by comparing different systems for the needs of the positioner. Next method that will be used is topology optimization to optimize where mass is placed in some parts of the main structure, the software used for this is Inventor Nastran. This information will be used to create a parametric CAD model of the structure. The dimensions of this model should then be optimized for different height and load conditions. This optimization would require a large number of FEM simulations, so to make the optimization process faster, a metamodel will be created. The number of simulations needed for this metamodel grows with the number of

inputs variables used, therefore it is important to limit the amount of design variables. A sensitivity analysis is therefore performed to find out which parameters has the biggest influence on the output and the amount of design variables can then be limited based on this information. The metamodel will take inputs from dimensions and applied load, outputs will be weight and max stress. The final step is to create a configuration tool which uses this optimization process to automatically create an optimized structure. Parallel to the structure optimization another optimization algorithm will be used to choose the components for the drivetrain. This algorithm is used first as input for the simulations to the metamodel and then finally also integrated in the configuration tool. The accuracy of the metamodel will be measured and if it is below an acceptable level, the simulations for the metamodel will need to be performed again but with modified settings to get a more accurate result.

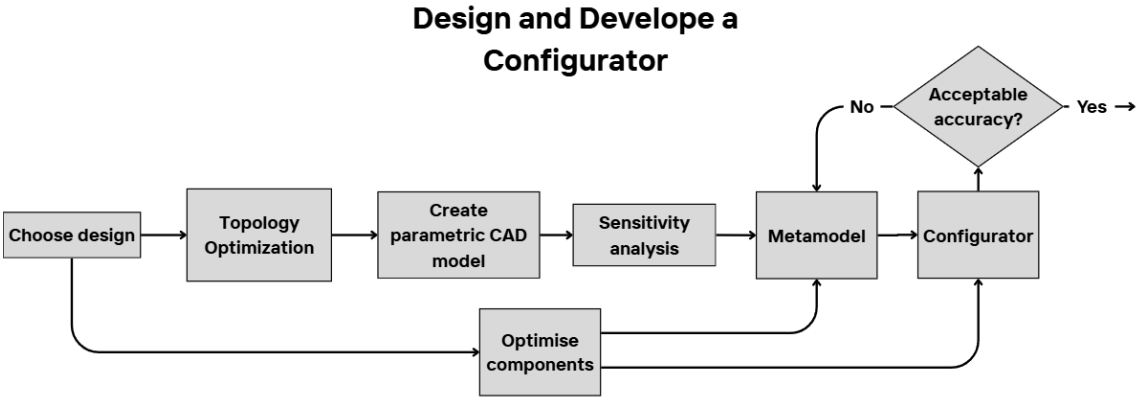


Figure 4.2: Detailed view of the "Design and develop a configurator"-step.

The fourth step in DSR is to demonstrate and validate that the artifact can be used to solve the problem. This will be done by reviewing if the structure and system satisfies all objectives that was set in the second step by doing additional FEM simulations on the resulting structure and compare the result with the used input. Next step is evaluation of the solution, which will be done together with the supervisors at the company to evaluate if the solution seems reasonable and what the use cases for the tool could be. Last step is to communicate the result by publishing the report on "DIVA".

4.1.1 Measures of success

In order to determine if the performed work is deemed successful, some success criteria are necessary. The success criteria can be divided into two categories, the first is to see if the output from the configuration tool could be used as it is without further simulations or adjustments to general design. This would require high accuracy since the potential error would have to be added to the safety factor, which increases the material cost. It is hard to set a specific value for this, but 10% in relative error could be argued to still be worth to add to the safety factor for small volume products in comparison with the the development time saved.

The other category is to see if the output can be used as a starting point for the design process with need of further FEM simulations and tweaking of the dimensions.

This is a more realistic goal for this type of problem and will be the focus of this work. For this criteria it is even more difficult to set a specific value for success since the dimensions will be changed later anyway. What is more important for this is that the proportions between dimensions are close to an optima to ensure a good starting point. Although this could be difficult to verify in this work, some form of evaluation could be made together with experienced supervisors at the company to see if the output looks reasonable.

In addition to the two categories mentioned above, the configuration tool should provide a result in a reasonable time-frame. Where the user saves time by using the configuration tool instead of running simulations. To put a metric on the time saved, the configuration tool should provide a result in less time than a simulation run would take. Further, the time spent using the configuration tool should not be inhibiting the workflow of the user, where the user can return to working in as short of a time as possible and preferably within five minutes.

4.1.2 Choice of design and components for drivetrain

Essential to the design of positioner assembly is what type of drivetrain to use. Since the drivetrain will dictate the limitations and design of the positioner. In order to find possible and feasible options that are available to use, two primary ways of selection are used. Firstly the current solutions used in products at Yaskawa are studied, rack and pinion solutions and ball or planetary screw motion. By studying and discussing current solutions at Yaskawa, many learnings was gathered around mounting options, pitfalls to avoid in design and availability of solutions from suppliers. Lastly, the current market of positioners were studied, solutions using hydraulics are widely used in the positioner market and was also taken into account when selecting drivetrain. Criteria for selection are following, load capacity (static and dynamic load capacity), cost and stroke length. Design is intended to handle a load range of 1 to 20 tons, thus the product line or solution must satisfy the load range. Further, in order to keep the design proposal scalable and leaving the possibility of using similar drivetrains for other products than positioners, the complexity of mounting the drivetrain is preferable to keep as simple as possible. Minimizing cost is always of interest. Finally, the stroke length, in essence the supported distance of travel was to be kept at the same length as current solutions at Yaskawa.

4.1.3 Topology Optimization

Topology optimization (TO) is used in this thesis as a guiding tool to get insights in where material should be placed for maximum stiffness while minimizing mass. TO often generates complex shapes that are expensive to manufacture which is a drawback since cheap manufacturing is an important requirement of the positioner. Therefore the shapes from the TO will not be used directly but rather as inspiration when creating the simple manufacturable parts. The software used for the TO is Inventor Nastran from Autodesk, this software is used because Autodesk offers possibilities for most of the work intended to be performed in this thesis. Inventor can be used for the CAD work, iLogic can be used for design automation and Nastran

can be used for FEA including TO with advanced enough options for the intended use.

TO was used on three segments of the structure: the main body, the support for the upper bearing and the bracket between the screw nut and the carriage. The first step for each segment was to decide what other parts are important to include in the analysis to get the be able to set the constraints and forces and that they acted on the specific part in a realistic way while removing unnecessary parts to decrease the computation time. The bracket for the carriage will be used as example where the assembly for the analysis is shown in (Figure 4.3). The bracket is supposed to connect the ball screw nut (black) and the carriage (grey), the design region for the bracket is shown in blue.

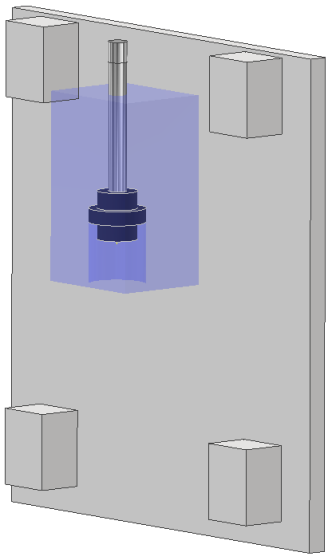


Figure 4.3: Assembly for the Topology Optimization of the carriage bracket.

Next step was to setup the FEA of this assembly in Nastran where it was meshed in two separate solids shown in (Figure 4.4). The design region in blue has a smaller mesh size to get a smother solution. The material chosen is regular construction steel S 235 with Young's modulus 210 GPa. The upper part of the ball screw is fully constrained where it sits in the bearing and the four connection points for the runner blocks are constrained in all DOFs except for vertical translation since the carriage is free to travel this direction on the rails. A load of 4 tons is applied to the back of the carriage as this is the max load for this particular positioner. There should also be a moment on the carriage from the rotating axis but since this moment is supposed to be absorbed by the runner blocks, this moment was neglected in this study. The surface contacts was then assigned where welded or bolted surfaces were set to bonded, touching surfaces set to separation and the contact between the design region and screw was deleted since there should be no contact there. To see that the model is behaving as intended, a regular static study was performed and then the topology optimization was set up.

This was done in the parameters tab where the first parameter chosen was the *TOPTGEN* parameter. With this parameter it is possible to choose objective and design constraints. For this part the most important constraint is the displacement

since the ball screw requires high tolerances for the position of the bearing and nut. Ideally the *TOPTGEN* parameter should be set to *VFDISP* since this setting minimizes mass with max displacement as constraint [38]. But *VFDISP* showed to be unreliable for some setups and often resulted in a structure where all mass was removed and the constraints clearly not met. So the next best parameter to use was *COMPVF* since this setting minimizes compliance (inverse of stiffness) with a volume fraction as constraint. In other words *COMPVF* makes the user choose how much of the volume that should be removed while maximizing the stiffness of the structure.

Another constraint used was the *TOPTMANCONSTR*, with this constraint it is possible to set a manufacturing constraint, for example that it should be possible to manufacture the structure by using milling or additive manufacturing. For *TOPTMANCONSTR*, the symmetry setting was chosen around the plane that splits the assembly into symmetric halves. This makes sure the resulting structure is symmetric. To get a good result from this it was necessary to increase the symmetry tolerance which is done in the *TOPTELEMSYMTOL* parameter which was set to 0.9 [38]. To better visualize the result, a stl file was saved by setting the parameters *OUTGEOMFILETYPE* to 1 and *TRSLTOPTDATA* to ON. The optimization was then performed multiple times with different target volume fraction set in the *TOPTDESIGNCONSTR* starting at 0.5 and decreasing until the max displacement target was reached. For example, setting *TOPTDESIGNCONSTR* to 0.2 means that 80% of the volume is removed and the objective is to maximize stiffness.

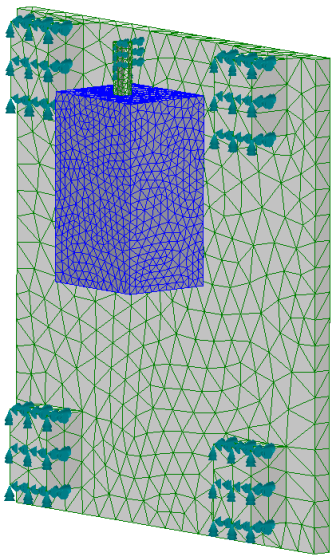


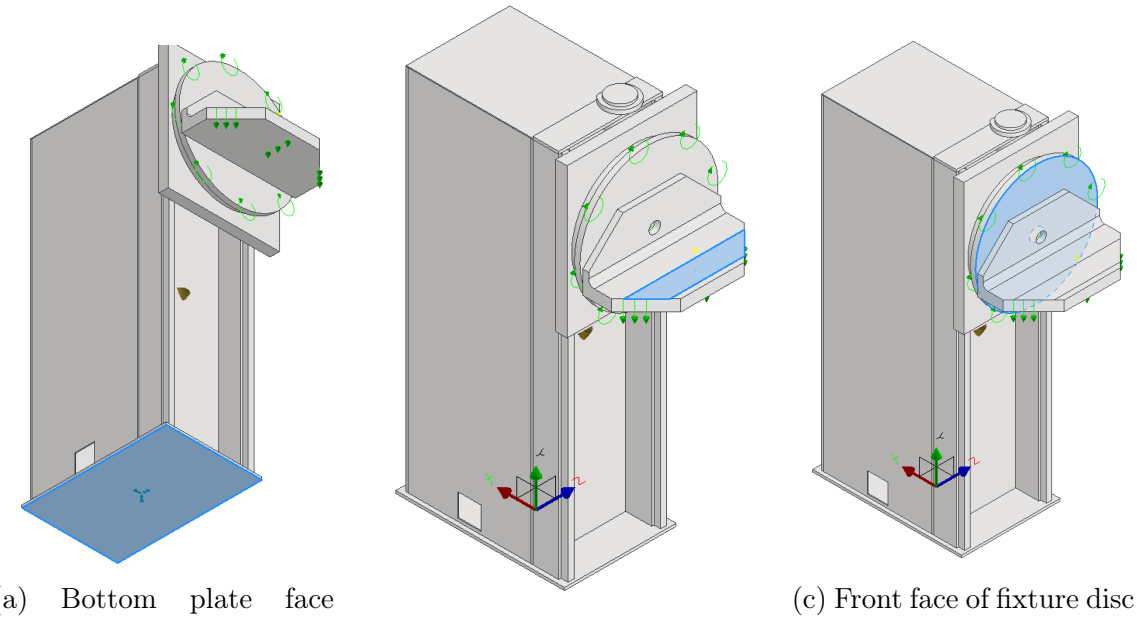
Figure 4.4: Mesh of the assembly for the Topology Optimization of the carriage bracket.

4.1.4 Modeling

During the design process considerations to previous HSB designs, input from engineers at Yaskawa, topology optimization and own ideas where taken into account. By continuously saving data and inputs for each step taken in modeling the design and future work on design can be informed on previous endeavors and reasoning behind why and what decisions were made. For the previous design of HSB positioner, the decision making and current information regarding design choices lies in the CAD model and with the engineers at the company. The topology optimization results are stored as files and respective inputs and outputs are stored partly in this report and as data sheets (excel). The design went through several iterations, with the early iterations having similar appearance to existing HSB designs. Continued design work is based on results from topology optimization, where the affected design areas are adjusted so that the design matches the topology optimization result. Implementation of simulation automation revealed issues with the design, since the intention is to let dimensions be changed externally from excel document, the design does in the state after topology optimization, not handle this properly. To clarify, problems arose when handling a varying amount of cutouts from side-plates, chamfer edge plate (top plate) and amount of bulkheads. In order to simplify simulation automation, the side plate cut outs are limited to only one hole, the top chamfer plate was removed and amount of bulkheads limited to one. With the simulation automation adjusted design, one additional redesign is done by introducing rail offset plates, which serve the purpose of moving the rails forward toward the carriage instead of having carriage extensions for the runnerblocks to be mounted on. The final design for usage in simulation automation is the last one mentioned here.

4.1.5 Simulation Set Up

With use of iLogic rules within the positioner assembly file the automation of simulation runs are set up. Firstly the assembly is set up for simulation in Nastran, with constraint on the underside of the bottom plate (Figure 4.5a), a force on the L-bracket top face (Figure 4.5b) and a moment on front face of the fixture disk (Figure 4.5c). Gravitational acceleration was also applied in the simulations, with magnitude of 9.82 ms^{-2}. Constraint on the bottom plate was to prohibit translation and rotation in all directions. Created iLogic rules are presented in Appendix A.4, for the simulation loop five rules in total were used (Algorithms 8, 9, 10, 11 and 12).



(a) Bottom plate face (marked in blue) on which the constraint was placed in simulations.

(b) Top face of L-Bracket (marked in blue) on which the force was placed in simulations.

(c) Front face of fixture disc (marked in blue) on which the moment was placed in the simulations.

Figure 4.5: Faces used for constraint and applied loads in simulation.

All parts are idealized as solid bodies with material of S235 steel. In order to circumvent stress singularities between critical load bearing components, contacts between planet screw and bearing and contacts between planet screw and nut respectively, the planet screw, bearing and nut are simplified to one part with fillets placed on connecting edges. Further, two mesh controls are placed on the fillet faces and the screw in order to get greater accuracy of stress, mesh sizes chosen via verification simulations (see Chapter 4.2.3). All contacts between parts were set to bonded, except the contacts between rails and runnerblocks which were set to sliding with no separation, in order to resemble the real world application where the runnerblock slide along the rail and thus does not hold any load in vertical direction. First step of design automation step is to perform sensitivity analysis on the design parameters, which in essence is to find what parameters are of importance to the assembly. To give one example, finding if and to what extent the height impacts the stresses and displacement of the assembly.

4.1.6 Sensitivity analysis

The constructed parametric model had 19 variables that can be changed which is a lot of variables when trying to make a metamodel. To decrease the number of variables a sensitivity analysis was made on the variables. The method used for the sensitivity analysis was the Morris screening method where a few trajectories in the design space are explored and a measure for the sensitivity of each variable was plotted in a diagram.

The reason for choosing the Morris method was that it can give a measure for

interactions between variables while requiring a relatively few amount of simulations. Other computationally cost effective methods like the One-At-a-Time (OAT) method cannot estimate interactions between variables. Other methods that can estimate interactions, like Sobol or other Monte Carlo based methods require more samples to be calculated than Morris [39].

The Morris method was implemented in Matlab (Algorithm 1), beginning with a function for creating trajectory design matrices (Equation 2.9). The number of variables k was set to 19, number of levels p was set to 4 and the step size Δ was set to $2/5$ since this results in a matrix with values between 0 and 1.

Ideally trajectories with maximum spread should be chosen, where spread is defined from (Equation 2.13). Setting $M = 500$ and $r = 10$ resulted in a combinatorial problem since the number of possible combinations r from M where order does not matter is around $2.46 \cdot 10^{20}$ which is way more than Matlab could handle. To get around this problem and still get large spread, a random trajectory was chosen first and the next trajectories were then chosen iteratively where the trajectory with the largest minimum distance from any previously selected trajectories was chosen each time until 10 trajectories has been selected.

The trajectories were then scaled between the lower and upper boundary (Table A.1) of each variable and finally the scaled trajectories were saved in a Excel sheet so that the parametric model could read the values as input for the simulations. These simulations are then run in inventor Nastran through iLogic and the result was printed in a Excel sheet where the max stress and deflection then can be read back to Matlab. The elementary effect for stress or deflection can then be calculated (Equation 2.10) by using the unscaled trajectories and the sensitivity indices are then calculated (Equation 2.11,2.12) and plotted. With the plotted parameter sensitivities, decisions can be made regarding which parameters to keep as design variables in final simulation set up.

4.1.7 Optimization algorithm for drivetrain components

The process of choosing components for the drivetrain can be made into an optimization problem with some objectives that should be minimized subjected to a set of constraints. This problem is shown mathematically in (Equation 4.1).

How this was solved and what the notations mean will be explained in this chapter.

$$\begin{aligned} \text{Minimize } & f(x) = [m_{tot}, -v_{lift}] \\ \text{subject to } & g_1(x) \equiv S_{0B} \geq S_{0min} \\ & g_2(x) \equiv S_{0N} \geq S_{0min} \\ & g_3(x) \equiv L_{10hB} \geq L_{10hmin} \\ & g_4(x) \equiv L_{10hN} \geq L_{10hmin} \\ & g_5(x) \equiv M_{tot} \leq M_{allowed} \\ & g_6(x) \equiv M_{tot} \geq M_{ta} \\ & g_7(x) \equiv l_{screw} \leq l_{max} \\ & lb \leq [x_1, x_2, x_3, x_4] \leq ub \end{aligned} \tag{4.1}$$

The choice of drivetrain components was to be made automatically and to get the optimal combination of components an optimization algorithm was made in Matlab called "GA_Optimization" (Algorithm 2). This algorithm is based on the *gamultiobj* function [40] from the Global Optimization Toolbox. This function uses a controlled, elitist genetic algorithm which is a variant of NSGA-II. *gamultiobj* needs an objective-function (Algorithm 3) that returns the objectives of the optimization. The two objectives chosen for this optimization was to maximize the lift speed while minimizing the mass of the components. The maximum rotational speed is determined by calculating 4 different speed limiting factors: critical speed and permissible speed from (Equation 2.7,2.8), the possible speed from the motors drive power (Equation 2.6) and the rated speed from the motor. The lowest of these 4 speeds is then used to calculate the lift speed v_{lift} (mm/min) (Equation 4.2)

$$v_{lift} = n_{min} \cdot p \tag{4.2}$$

where p is the lead (mm). The other objective is the combined mass m_{tot} (kg) of the bearing, nut, screw, motor and gearbox.

There are also 7 nonlinear constraints $g_1 - g_7$, the first two constraint g_1 and g_2 says that the static safety factor (Equation 2.1) for the bearing and nut should be above the minimum static safety factor $S_{0min} = 5$. g_3 and g_4 says that the lifetime (Equation 2.3) for the bearing and nut should be above the minimum life $L_{hmin} = 10^5$ hours. The calculations for lifetime is based on 10% duty cycle at 150 rpm (Equation 2.4) i.e. $n_1 = 0, q_1 = 90, n_2 = 150, q_2 = 10$.

g_5 says that the total torque from the motor and gearbox M_{tot} should be below the maximum allowed torque for the gearbox $M_{allowed}$. g_6 says that M_{tot} should be more than the drive torque M_{ta} needed for the planetary screw (Equation 2.5). g_7 makes sure that the length of the screw l_{screw} is less than the maximum screw length available of that diameter l_{max}. All these constraints are collected in a function called *nonlcon* (Algorithm 4) and used as input for *gamultiobj*.

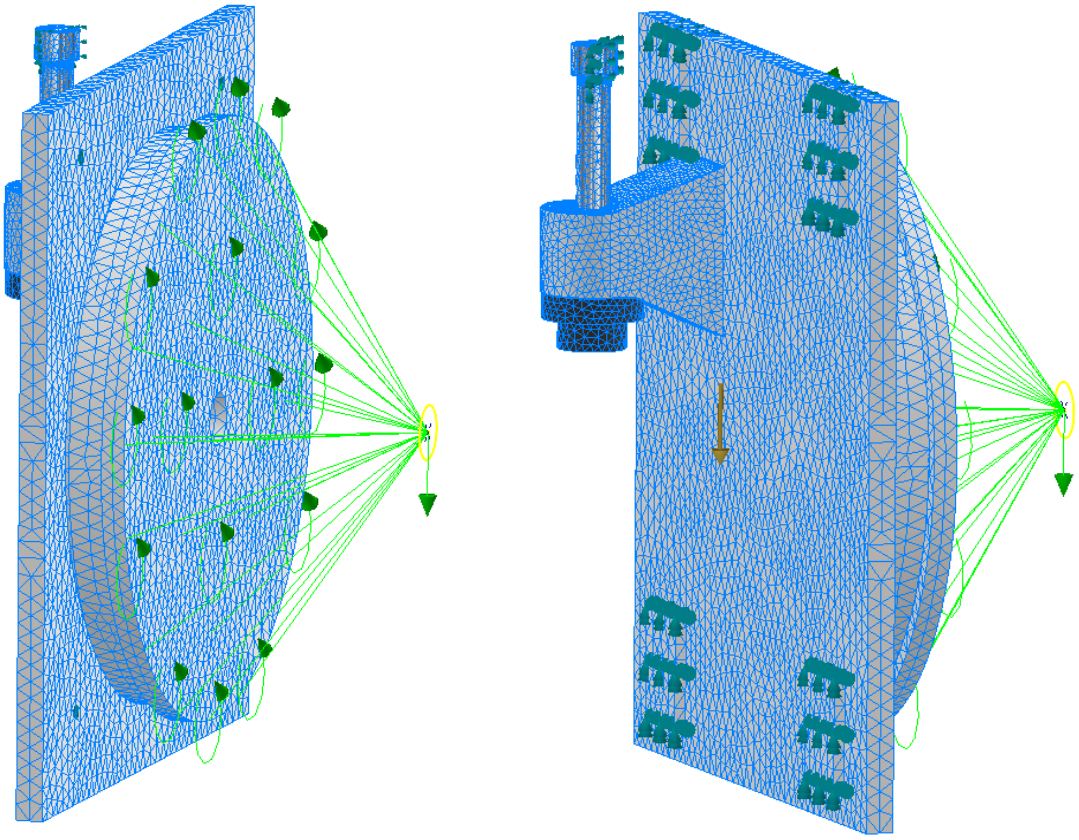
Finally the boundaries are set so that the lower boundary is 1 $lb = [1, 1, 1, 1]$ and the upper boundary ub is the number of available variants for each component. These boundaries are used as input for *gamultiobj* together with an integer constraint that says that all variables should be integers. The options used for *gamultiobj* are that the population size is set to 300 which seems to be high enough for it to give the same result every time while still being fast.

gamultiobj then returns a Pareto front consisting of a number of optimized points with different weights to speed and mass. The "best" point now needs to be chosen automatically since the components are supposed to be used as input to create the metamodel where a few hundred points will be needed. The following method was used to choose points automatically: first all points are normalized between 0 and 1. Then the point chosen is the point closest to the "ideal" point [0,0] [41]. This should give an optimal point that is balanced between the two objectives. It was later found valuable to be able to regulate where the "ideal" point is. This was made possible by adding a user input for "desired speed" which is then used to move the upper boundary for speed. The upper boundary for speed was set to twice of the desired speed since this was found to result in the algorithm choosing a point roughly at the desired speed.

4.1.8 Rail calculations

The rail system used is from Bosch Rexroth which offers ball rail systems or roller rail system. Ball rail systems was chosen since it gives high enough accuracy and load bearing capacity for the application. Rexroth also provides a calculation program for linear guides and screw drives called Linear Motion Designer (LMD) [42]. This program was used for the rail system dimensioning since it calculates static safety factor, dynamic safety factor and lifetime among other things. This program needs the reaction forces on the runner blocks as input and to get these a FEM simulation is needed for each load case. To keep the simulation number down it was decided to dimension the rails for a set of different load cases. Nine load cases was chosen: [1000, 2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000] kg that covers the intended load interval.

The setup for the FEM simulation is shown in (Figure 4.6) consisting of the carriage with a fixture disc and the planetary screw with mounting on the backside. It has two constraints, the top part of the screw is fully fixed and the mounting locations for the runner blocks are fixed in all DOFs except for vertical movement. It has 3 loads: gravity, a remote force simulating the load 400 mm out from the fixture disc, and a torque on the fixture disc. The load is applied 400 mm out from the fixture disc because the load is usually placed freely on a fixture that extends a distance out from the disc which is roughly 400 mm. The torque on the fixture disc need to be high enough to rotate the load with a small offset. This offset is usually between 100 and 200 mm and is varying depending on motor and gearing. The exact torque for these simulations was based on an existing positioner which had a torque of 1658 Nm per ton of load. This is corresponding to a max offset of 169 mm.



(a) FEM setup for rail calculations, front.

(b) FEM setup for rail calculations, back.

Figure 4.6: FEM setup for rail calculations.

The sizes of the screw, nut and mounting was decided from the Matlab drivetrain choice algorithm and the sizes of the fixture disc and slewing bearing was chosen from existing solutions divided into weight classes.

All contact surfaces were bonded and the structure was then meshed with a mesh size small enough to get at least 2 elements along the thickness and the simulations were performed. The reaction forces could then be collected from the constraint on the runner block surfaces and then be used as input in LMD.

The first thing to declare in LMD is the positioning of the runner block, this was initially set to 4 runner block in a 1x1 m square in vertical rotation. Next part is to set up the dynamics for the rail, this was set to a 2 m stroke with 10% duty cycle with 60 s per stroke and 540 s break. Then the forces are declared and there were no option for torques so the torques were converted into remote forces with 1 m offset. The specific ball rail system was then chosen and no preload used, this then displays a selection of different sizes with the corresponding safety factors and lifetime etc. The smallest size that fulfilled the wanted safety factors was chosen and this procedure was then repeated for all load cases.

For the higher load cases even the largest size was not enough to get meet the safety factors. This was solved by either increase the distance between the runner blocks or by adding another pair of runner blocks if the safety factor was too low.

4.1.9 Design of Experiments

To be able to explore the design space efficiently Design-Of-Experiments (DOE) was used. In the CAD model there are many variables and each simulation is slow which means it is important to choose a DOE that explores the design space efficiently with as few simulations as possible. A DOE suitable for this is Optimal Latin Hypercube since this method spreads the points evenly by maximizing the minimum distance between the points [43].

To create this Optimal Latin Hypercube design, Matlab (Algorithm 5) was used with the built in *lhsdesign* function from the Statistics and Machine Learning Toolbox. The variables for this design consisted of dimensions chosen from the sensitivity analysis and the applied load. The lower boundaries for the height and width was set to the lowest possible for the model so that all parts could fit together and the upper boundary for height was set to the max height needed from the company. All other boundaries was set based on FEM simulations by varying each variable one at a time for the minimum and maximum load and analyzing the stress to get reasonable boundaries for each variable.

lhsdesign was then used to create a design matrix which was written to an Excel file. The load and height column from this matrix was also used as input to the GA_Optimization function (Algorithm 2) which in turn outputs the drivetrain components suitable for each point in the design matrix. The bearing and nut/screw number is written to the same Excel file so that the suitable components can be used for each FEM simulation.

4.1.10 Simulation

With the simulations set up and proper care for sampling of design space (LHS) performed, the next step was to run simulations and store data for usage in creation of metamodel. To begin, a set of 100 sample points using LHS was created and then run with mesh settings in accordance to pre-verification of mesh size (see Chapter 4.2.3), with coarse meshing and no specific mesh controls. The 100 initial simulation runs were not deemed satisfactory with stress and displacement data that fluctuated drastically when running verification simulations. For more in depth description of mesh verification, see verification chapter 4.2.3. Proceeding after mesh verification, a larger sample set of 500 samples (LHS) was used in following simulation run. Of the 500 samples, the first 100 runs were performed again in order to verify consistency of simulations. From verification of consistency, changes in mesh settings were made, specifically changes in mesh control on screw surface with reduction in mesh size, see Chapter 4.2.3 for specification. With the new mesh control, the 500 simulations were run again and after verification of consistency (similar to previous runs) the results were deemed satisfactory for use in metamodels. Data storage was done by reading output file from each simulation run, maximum stress and displacement in respective direction, which was stored in an excel sheet.

4.1.11 Metamodel

The goal is to optimize the positioner with respect to mass while keeping sufficient strength to being able to operate with the intended load for the target life time. Minimizing the mass requires a lot of FEM simulations to calculate the stress and deflection to see if the constraints are fulfilled. To perform all these FEM simulations for each optimization would be very time consuming so to speed up this process a metamodel was created to be a part of the procedure. The metamodel is a mathematical model that can predict some of the positioners properties, the metamodel should be able to predict the maximum stress, deflection or mass of the positioner with some chosen variables as input.

The metamodel needs to be trained on data (training set) from FEM simulations and to validate the accuracy and precision of the model, the output needs to be compared with independent reference data (validation set). These sets of data points where received from the Design of Experiments Matlab code (Algorithm 5) using Latin Hypercube and the automated FEM simulations where performed using Inventor Nastran. A total of 500 simulations were performed, 450 of these points were selected as training set and the other 50 points as validations set. This strategy of removing k points for validation and train the metamodel on the rest is called the leave-k-out strategy [28]. The value of k was chosen from the recommendation to set $k = 0.1N$ from Meckesheimer et al. [44]. The order of what points are selected should not matter since Latin Hypercube gives the points in random order. The data set consists of input points with corresponding output (maximum stress, deflection and mass). These training sets where then used to train four metamodels of different types.

There are many different types of metamodels that could be used to create representations of the true model. Some common examples are polynomial, Kriging/Gaussian Process Regression (GPR) and Neural Network. For this thesis it is important that the metamodel can handle a relatively large number of variables. The polynomial metamodels was ruled out since these would require a large number of terms with these many variables and the number of data points to fit the curve would grow to large. There is also a risk of overfitting when using few points, which would result in bad predictions between training points. The GPR category however is suitable since it can handle many variables and requires a relatively low amount of data points compared to Neural network. In the Statistics and Machine Learning Toolbox there is a function *fitrgp* which uses GPR to create metamodels. Different hyperparameters was tuned to minimize the error from the validation set, this includes kernel functions and basis functions. Another important setting was to standardize the data which greatly improved the results. The final settings used is shown in (Algorithm 6).

Neural network could also be useful for creating the metamodel, the possible downside is that it usually requires a large data set to give reliable predictions. The Matlab function used for this is *fitrnet* where layer sizes and lambda were the main hyperparameters used in tuning.

Other interesting possibilities that could be suitable are Ensemble of decision trees and Support Vector Regression (SVR). These are also common regression methods that work in different ways than the other mentioned methods. To create an ensamble of decision trees, the Matlab function *fitrensemble* was used where both Least Squares Boosting (LSBoost) and Bagging (Bag) was tested together with other hyperparameters like learning cycles, learners and learn rate. The SVR metamodel was created with the Matlab function *fitrsvm* where kernel function, box constraint and epsilon were the main hyperparameters tuned.

These four methods where used to create metamodels using the training set (Algorithm 6). The accuracy and precision of the models are then evaluated by calculating the relative error, specifically Mean Absolute Percentage Error (MAPE) [45] $\bar{e}$ (Equation 4.3) where n is the number of reference points, $\hat{y}_i$ is the predicted value from the metamodel and y_i is the corresponding value from the FEM simulation.

$$\bar{e} = \frac{1}{n}\sum_{i=1}^{n} |\frac{\hat{y}_i - y_i}{y_i}| \tag{4.3}$$

When the metamodels has been trained and validated the best model was then used for the optimization (Equation 4.4). There are multiple algorithms that could be used to optimize the dimensions of the positioner. One common method is to use a gradient based algorithm to find the the minima of the solution space. The Optimization toolbox in Matlab contains a function called *fmincon* which uses the gradient to find the local minima of a multivariable function while fulfilling nonlinear constraints. This makes it a suitable candidate for solving the optimization problem. The downside of this algorithm is that it finds local minima based on a starting point given as input, so in order to find the global minima it is necessary to test multiple starting points and thereafter choose the best point.

$$\begin{aligned} &\text{Minimize } f(X) = m_{tot} \\ &\text{subject to } g_1(X) \equiv \hat{\sigma}_{max} \leq R_{el} \\ &lb \leq [X] \leq ub \end{aligned} \tag{4.4}$$

Another possible method is to use a Genetic Algorithm similar to the one used for choosing drivetrain components (Chapter 4.1.7) with the difference that this is a continuous single-criteria optimization instead of the discrete multi-criteria optimization for choosing components. The advantage of this algorithm is that it searches the whole solution space from start so it does not need a starting point. The downside is that it takes longer time to reach high precision.

Pattern Search is another method that could be useful for this optimization in case the gradients from the metamodels would prove to be unreliable. This method searches in a pattern around a chosen point without using gradients which means it is also necessary to test multiple starting points to get the global minima. The Global Optimization toolbox in Matlab contains a functions for this called *patternsearch*.

These three methods where compared by running them using the best metamodel and comparing speed, accuracy and precision. The best metamodel and optimization algorithm was then used for the configurator.

4.1.12 Configurator

With the metamodel and optimization algorithm it is now possible to receive a proposal of the dimensions and components that should be used for a particular load case. But to make this process more usable so that it can be used by others as well, this process needs to be combined into an easy to use tool. It was therefore decided to create a configurator in Inventor, where the user writes a few inputs and then the configurator automatically produces an optimized design of the positioner both with dimensions and used drivetrain components. The model in Inventor should automatically be updated and an output summary should be given with key information about the designs safety factors and life time.

The developed configurator (Algorithm 13-15) starts in Inventor by showing a form with 6 inputs that should be filled in by the user, showed in (Table 4.1). The Static safety factor and Life are only for dimensioning the bearing and nut based on the formulas in (Chapter 2). The Allowed max stress is related to the stress from the FEM simulations and is therefore a separate input from the Static safety factor. The purpose of the Desired speed input is to be able to regulate the size of the drivetrain, since a higher speed would need larger components. This input will only be an approximate target of the speed, the solution can end up both below or above the this target.

These inputs are then transferred to Matlab opened from Inventor and additionally the metamodel is loaded in form of a ".mat" file as well as the component information tables which is needed for the component optimization.

Table 4.1: Inputs for the configurator

Load (kg)
Height (mm)
Static Safety Factor
Life (h)
Allowed max stress (MPa)
Desired speed (mm/min)

The Matlab code (Algorithm 7) then starts by optimizing the dimensions from the variables in the metamodel by using the chosen optimization algorithm. After this the drivetrain components are optimized and information (Static safety factor, life time, mass and speed) from the solution is calculated. This is then written to an Excel sheet together with the dimensions and components. Inventor then reads all this information, updates the model automatically and displays a message box with information about the solution.

4.2 Verification

Verification of results, simplifications and simulations have been performed in multiple stages of the project. In essence, the verification work that has been performed serves to strengthen the consistency and reliability of the results in the respective areas of verification. Below, four main areas of verification will be presented: Simplification, FEM stress, Mesh size and Morris method verification.

4.2.1 Rail Model simplification

The rails and runner blocks where simplified to flat-bars with rectangular cross-section where the outer dimensions of the original runner blocks where kept the same in the simplification. Reasoning behind the simplification was to reduce complexity in simulations, if the rail (or runnerblock) 3d-model was for example imported it would maintain a geometry that would require a very fine mesh in order to successfully simulate. The height and length of the rail were kept as the original and the width was varied until the deflection of the simplified model was approximately the same as the original model, where deflection was received from a static study. Two arbitrary forces of 25 kN on each runnerblock was applied, one out from the rail and one to the side. The displacement was measured at two points of the edge with maximum displacement (8 and 9 μm) with a difference between simplified rail and original (3% and 1%).

The stress was higher at a few local points 40 MPa compared to 11 MPa, but these models are not used to determine stress in the rail and runnerblock, for this bosch's configurator is used. Overall the stresses are very similar.

4.2.2 FEM stress

To verify how accurate the stress is from the Nastran result file (.OUT file), a simple validation test was performed where analytic stress was compared to the max stress from the plot and the result file for 5 cases that are relevant for the real structure. The test where performed on rods 100 mm long and 10 mm in width or diameter. The rods where fixed in one end and a force applied in the other end. The force was 1000 N for tension and 100 N for bending as these forces resulted in realistic stresses but not above yield limit. The fillet rod had a starting diameter of 20 mm transitioning to 10 mm with a fillet radius of 0.5 mm. All rods were set to be made of steel S235.

The element size, for this verification of FEM stress, was set to 5 mm in Inventor since this results in at least 2 elements across the thickness, but since the meshing is curvature based it still results in smaller mesh around a smaller radius. The resulting stresses are shown in (Table 4.2) for the first four cases the max stress from the plot is overestimating the stress by around 2-20% and the max stress from the result file is overestimating the stress by around 10-45%. This trend was broken by the fillet test where the max stress from the plot was underestimating by 23% and the result file was overestimating by only 0,6%. This means it it safer to use the stress from the result file for two reasons, it overestimated in all cases which is safe since it only results in a stronger structure.

The other reason is that the largest stresses will probably be from stress concentrations around small fillets which is where the result file performed better.

Table 4.2: FEM stress comparison table

Case	Analytic stress (MPa)	Plot stress (MPa)	OUT stress (MPa)
Sq rod tension	10,0	11,0	13,0
Sq rod bending	60,1	61,1	71,9
Circ rod tension	12,7	15,6	18,5
Circ rod bending	101,9	107,9	113,6
Fillet tension	31,8	24,4	32,0

4.2.3 Mesh size

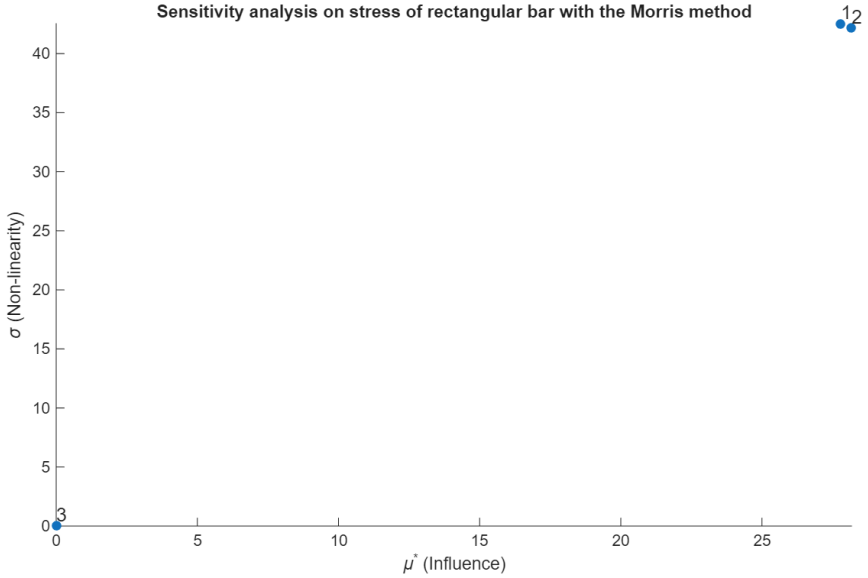
In order to select mesh size when running FEM simulations for the whole positioner assembly and finding a balance between time to run simulations and accuracy in solution. The mesh setup involved a general mesh and two mesh controls, one for the fillet faces on the screw, bearing and nut interfaces and one for the outside face of the screw. Verification was performed by letting each mesh have an arbitrary starting value, 50 mm for the general mesh and 25 mm for the screw mesh control. Then ten simulation runs were performed, with inputs from LHS, and the result were stored. This process was then performed with decreasing mesh size, with 5 mm steps, for the general mesh until any convergence in result could be noticed. The general mesh did not converge cleanly, resulting values (max stress and displacement) behaved erratically and thus the screw face mesh control was next to analyze. However, due to time of running simulations it was decided to not let the general mesh be smaller than 30 mm. With general mesh set, a similar simulation and verification loop to the general mesh was performed for each respective mesh control, but with a smaller starting value of 25 mm.

4.2.4 Simplified test of the Morris method

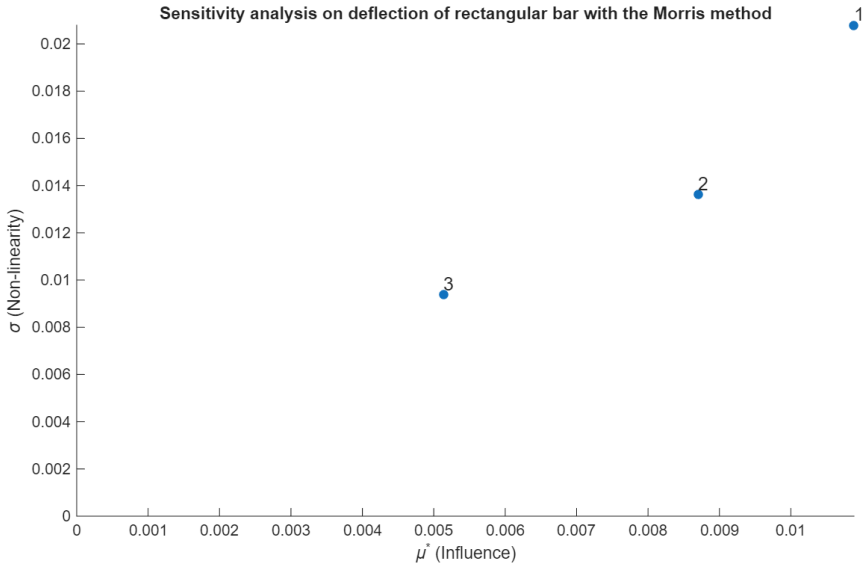
The Morris method was tested on two simplified cases to verify that the result is showing the intended information. The first case was tension on a rectangular bar with 3 variables: height, width and length. The boundaries was set between 10 and 100 mm for all variables and the bar was fixed with frictionless constraints that prevents movement normal to the surface on all but one side and a force of 10 kN was applied to the last side. These constraints was chosen to eliminate the stress concentrations that otherwise arise when setting a regular fixed constraint on one side.

The Morris test was performed with number of trajectories $r = 10$ and the result is shown in (Figure 4.7). The result for the stress was as expected in that length (variable number 3) had no influence on the stress while width and height had the same influence. Both are non-linear which is expected since the formula for stress is $\sigma = \frac{F}{bh}$ where b is the width and h is the height. The formula for elongation in tension is $\delta = \frac{F \cdot L}{A \cdot E}$ so it is easy to make the mistake in thinking the influence from

L should be linear, but since all 3 variables affect the deflection at the same time this makes L show as non-linear in the plot. The most surprising thing about the deflection plot is that it is a large difference in influence from height and width (point 1 and 2) since they theoretically should be the same. This could stem from a random difference in the trajectories so that height shows as more influential, so to test this hypothesis the test was repeated multiple times and the placement of the points for deflection was different every time, but the stress plot remained the same.



(a) Morris test of stress on bar.



(b) Morris test of deflection on bar.

Figure 4.7: Morris sensitivity test on rectangular bar.

The second case was on a simplified model of the positioner (Figure 4.8) with 6 variables: width, height, length, thickness of body plates, thickness of roof, number of holes in the side. The positioner was fixed in the bottom and a force was applied at the bottom of the screw. The expectations on this was that the thicknesses should have the biggest influence and that the number of holes would have the smallest influence.

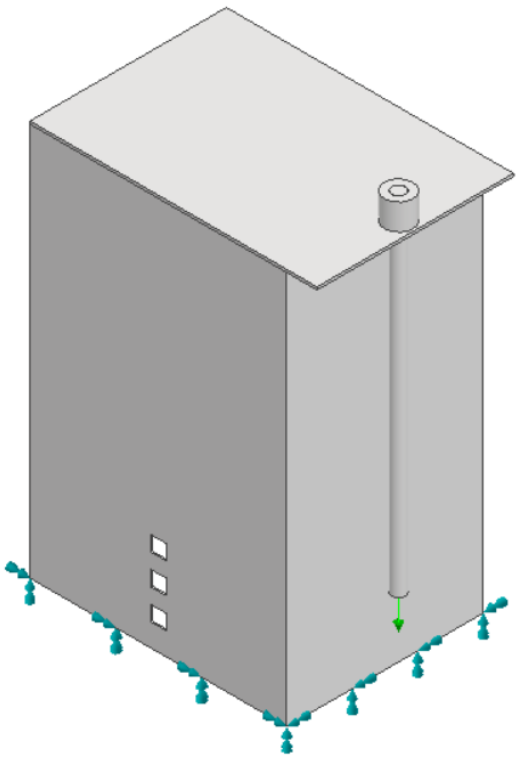


Figure 4.8: Simplified model for Morris test

The result is shown in (Appended Figure A.1) and as expected the thicknesses had the biggest influence on stress and number of holes the least influence. The results for deflection showed that the roof thickness had the biggest influence and the height in second place. This is also reasonable since the screw gets longer with increased height which increases the deflection. The number of holes also had the least influence on deflection as expected.

Chapter 5

Results and Analysis

5.1 Choice of design and components for drivetrain

Choice of drivetrain variant is based on current contemporary market solutions and specific criteria (Chapter 4.1.2). There were three candidates found in market that is suitable for the design of the drivetrain: Linear rack and pinion, Ball/Planetary screw and Hydraulics. The rack and pinion system was ruled out first since the found systems did not offer high enough lift capacity. The highest lift capacity found for linear rack and pinion system was "PLS 112" from Wittenstein with 112 kN lift capacity which is not enough for the goal of 20 tons. It could be possible to use double rack and pinions but this would be a large system that is difficult to fit to the carriage that already has limited space. Next system is the ball or planetary screw. The lift capacity of the ball screws found was also to low but the planetary screw had higher lift capacity for the same size with a range that covered the target load range of 1 to 20 tons. The hydraulic system could also meet load requirements and would probably be cheaper. But the downside is that there are not as many complete solutions to buy for this system so it would need a special design. Another big drawback of the hydraulic system is that it cannot be synchronically driven with other tools and robots, according to a supervisor at Yaskawa. For these reasons the planetary screw was chosen and next to decide was how to fit the screw. The screw can be mounted in a bearing at the top or at the bottom or both, all these mounting types have different advantages and disadvantages. When mounting only at the bottom there is a risk of buckling if the screw is too long. When mounting at the top there is no risk if buckling, but there is a risk of vibrations with high rotational speed. When mounting at both ends these risks are minimized, but the requirements on mounting tolerances increases and the cost increases. Calculations of critical speed was performed and it was decided that the critical speed was high enough for a single top mounting and therefore this mounting type was chosen with lower cost and easier mounting procedure.

The drivetrain consists of five main components: planetary screw, nut, bearing, gearbox and motor. The planetary screw and nut comes as a package and the requirements for this is that it needs high enough static and dynamic load rating, have long enough stroke length and have acceptable critical speed.

5.2 Topology optimization

The Topology Optimization yielded interesting designs for the components it was used on, however these designs were not suitable to use since they would require complex manufacturing techniques. One of the requirements of the positioner is cheap manufacturing which resulted in these designs not being used. Nevertheless this does not mean the Topology optimizations were a waste of time, there were still many lessons to be learned from the results that was considered in the design.

Some of the results from the example with the carriage mount is shown in (Figure 5.1-5.2) with two different volume fractions 10% and 40%. From these results it is clear that the TO tries to maximize the height and width of the mount, dividing the mount into 4 rods. Learnings from this that was not thought of before by the authors was that the mount should be resting on both upper surfaces, not just the lower surface where it is mounted. Another learning was that the mount should be extended below the screw as well, not just above.

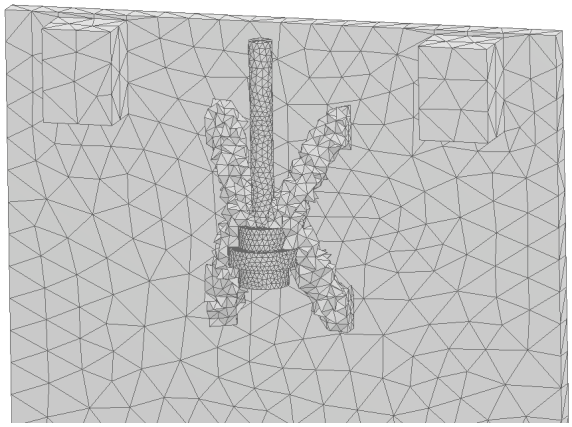


Figure 5.1: Resulting structure for the carriage bracket with target volume fraction 10%.

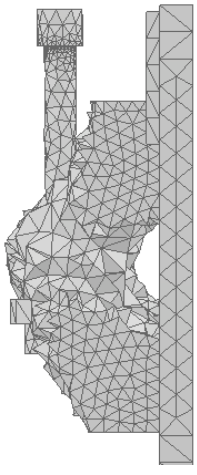


Figure 5.2: Resulting structure for the carriage bracket with target volume fraction 40%.

When taking these learning into account a parametric model was constructed (Figure 5.3). The size of the mount could be changed by using model states to fit all available planetary screw sizes.

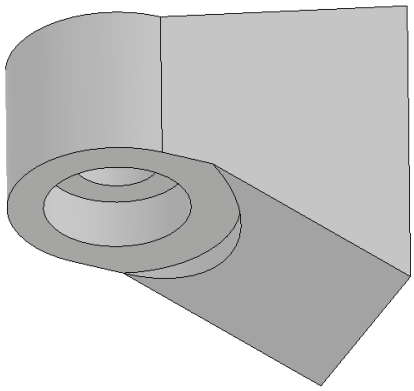


Figure 5.3: Design of Carriage Mount with inspiration from the Topology Optimization.

TO was also performed on the main body of the positioner, where a few learnings was made. The first is that much of the material should be at the front, this was implemented by adding an extra flat bar at the front just behind the mount for the bearing. It also showed that the roof should be rounded at the back, although this was later removed because it caused problems when automating the simulations. As for the bearing seat TO, it places all material in under the load bearing segment, the bearing. This was not unexpected but it served as a reminder to reduce material usage by substituting a large block of material to only needing triangular brackets underneath the bearing seat.

5.3 Parametric CAD model

Result from design work is a proposal which takes into account topology optimization results, simulation automation setup and additional design features proposed from Yaskawa. The proposed positioner assembly consists of 25 parts where 9 are simplified off-the shelf products and remaining 16 where created for the assembly. Below all parts are listed in (Table 5.1), with part number, part name and amount used in the assembly.

Table 5.1: Table of parts used in assembly. First column part number, second column part name and third column amount of given part that is used in assembly.

Part nr	Part name	Amount
1	Bottom Plate	1
2	Back Plate	1
3	Front Plate	1
4	Side plate	2
5	Rail offset bar	2
6	Bulkhead	1
7	Top rail cover plate	1
8	Top cover plate	1
9	Bracket	2
10	Carriage mount	1
11	Carriage	1
12	Front flat bar	1
13	L-Bracket	1
14	Rail	2
15	Runner block	4
16	Planet screw	1
17	Planet nut	1
18	Bearing	1
19	Fixture disk	1
20	Slew bearing	1

Figure below (Figure 5.4) shows a screenshot of the assembly in Autodesk Inventor, with attached numbers for each visible part. See appendix A, (Figure A.5 and A.6) for more detailed figures.

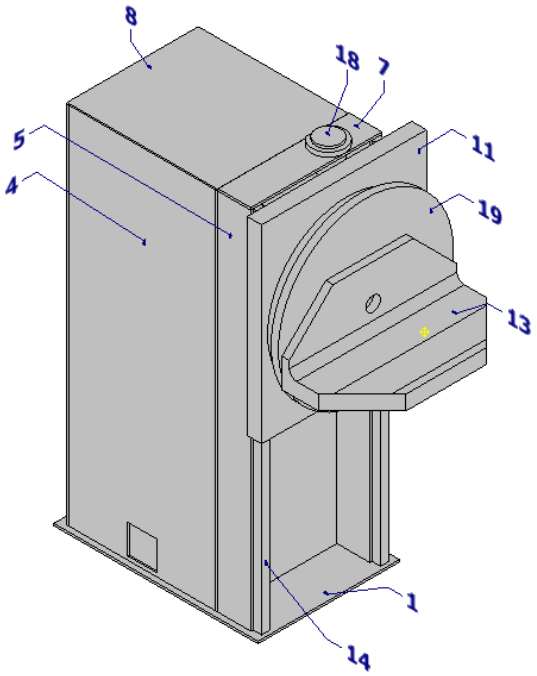


Figure 5.4: Positioner assembly in Autodesk Inventor with part numbers attached to respective part. See table 5.1 for part descriptions.

5.4 Sensitivity analysis

The sensitivity analysis on the complete model was performed using the Morris method and the result is shown in (Figure 5.5) for influence on stress and a similar result in appendix (Figure A.2) for displacement. Many of the points were close in a group so a zoomed figures are shown in (Figure A.3-A.4).

The plots show what influence each variable has on max stress and displacement, the x-axis show how much influence the variable has and the y-axis show how non-linear the influence is. From the plots it is clear that there are 4 outliers: Carriage thickness, width, bottom thickness and length.

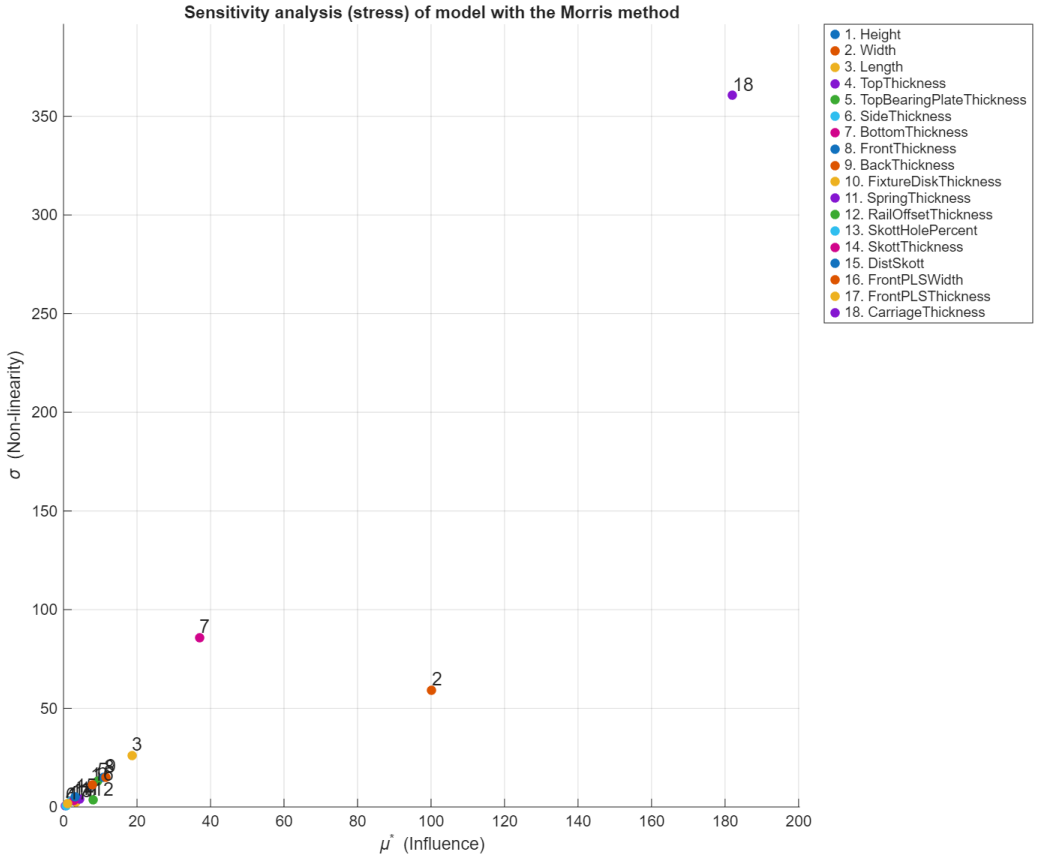


Figure 5.5: Sensitivity analysis of the complete model showing parameter influence on stress.

The plots were then analyzed to decide what variables that needed to be included in the metamodel. The main factors that decided if a variables was to be included or not was how big influence it has and does this influence seems reasonable. If the influence was low or the results did not seem credible then it was explored if the variable could be linked to another variable or if it could be set to a fixed value. The decisions made are shown in (Table 5.2) where "Yes" means that the variable is included as a free variable in the metamodel, "Partly" means that it is linked as a fraction of another variable and "No" means it is set to a fixed value. The status for the width was changed from "Yes" to "No" at a later stage since it was noticed that the optimization algorithm always minimized the width, so for the second round of simulation runs it was set to be as small as possible limited by the size of the the other components. More of the reasoning about some of the choices are included in (Chapter 6.3).

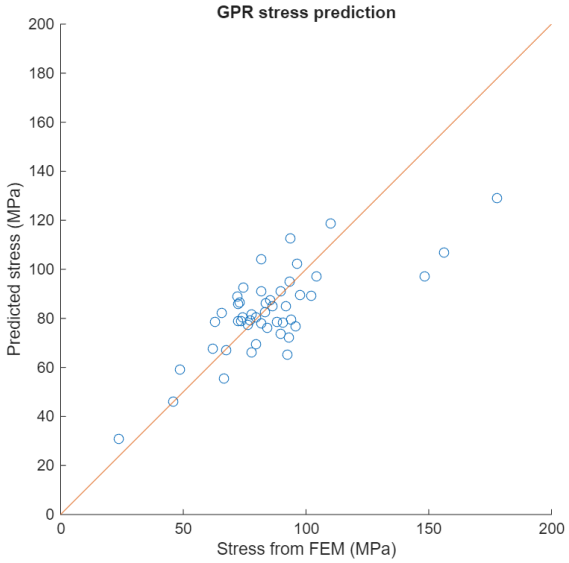
Table 5.2: Decisions made for which parameters was to be included in the metamodel.

Parameter	**Keep**	**Comment**	**Number**
Height	Yes		1
Width	No	Should be as small as possible.	2
Length	Yes		3
TopThickness	Partly	Low influence on both displacement and stress. Fraction of the back plate thickness.	4
TopBearingPlateThickness	Yes		5
SideThickness	Partly	Fraction of the back plate thickness.	6
BottomThickness	Partly	Not clear why it gets that high influence. Fraction of the top bearing plate thickness. Should later be dimensioned from the screw anchors.	7
FrontThickness	Yes		8
BackThickness	Yes		9
FixtureeDiskThickness	Partly	Fraction of the carriage thickness.	10
SpringThickness	Yes		11
RailOffsetThickness	Yes		12
SkottHolePercent	No	Set to fixed value.	13
SkottThickness	Partly	Fraction of the front plate thickness.	14
DistSkott	No	Set to the bottom height of the spring.	15
FrontPLSWidth	Yes		16
FrontPLSThickness	Partly	Fraction of the front plate thickness.	17
CarriageThickness	Yes		18

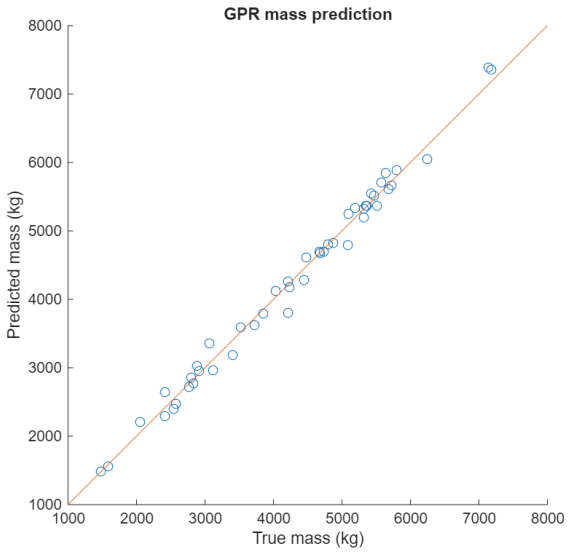
5.5 Metamodel

The accuracy of the metamodels are visualized in (Figure 5.6-5.7) where the predicted values are plotted against the reference values. The further away a point is from the line the bigger the error, so for an ideal metamodel that always predict the true values, all points should be on the line.

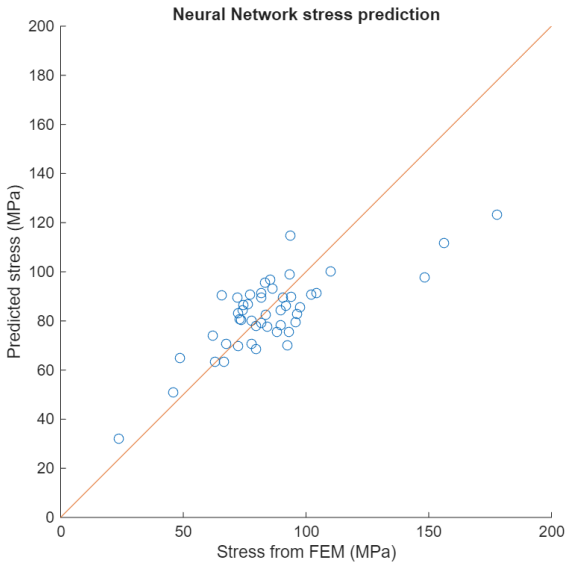
When comparing the mass and stress predictions it is clearly visible that the mass predictions have higher accuracy since the points overall are closer to the line. When comparing the different metamodels it looks like the LS boost is best at predicting stress, while for mass predictions it is more difficult determine which model is best. To distinguish which model is best for mass predictions, the mean relative error (MAPE) is compared in (Table 5.3).



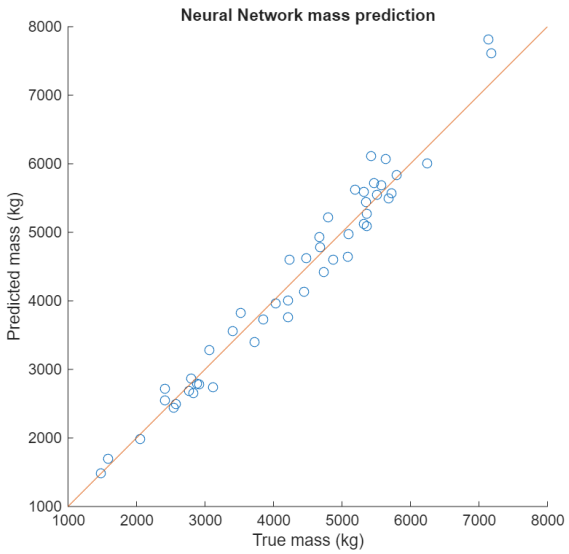
(a) GPR Stress prediction.



(b) GPR mass prediction.

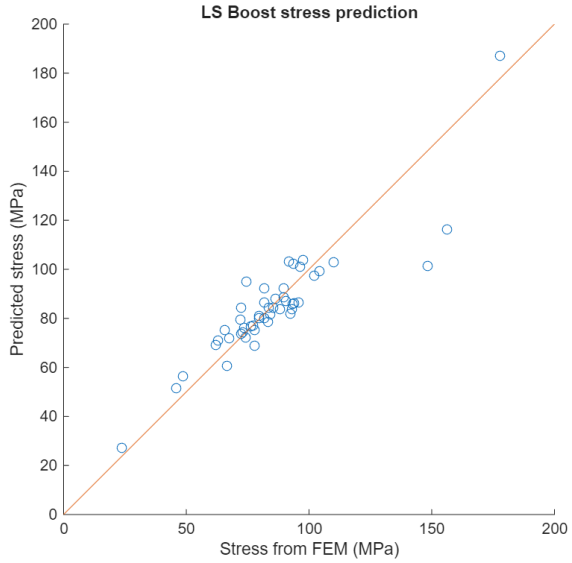


(c) Neural Network Stress prediction.

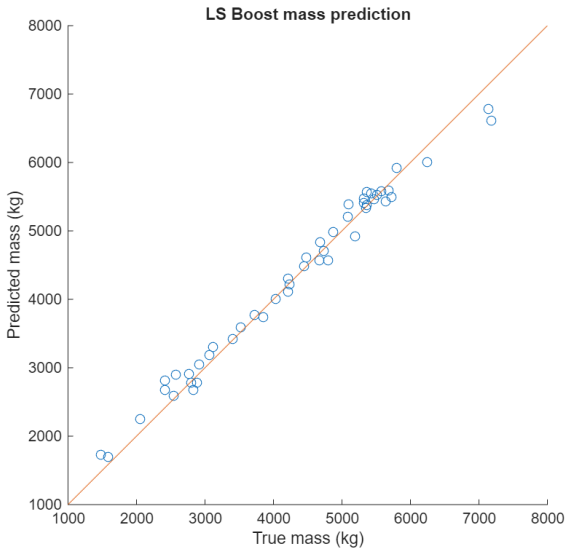


(d) Neural Network mass prediction.

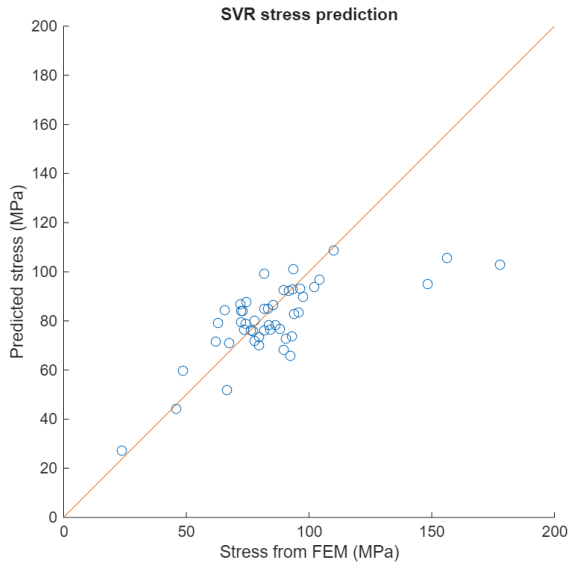
Figure 5.6: Comparison of metamodel predictions (GPR and Neural Network).



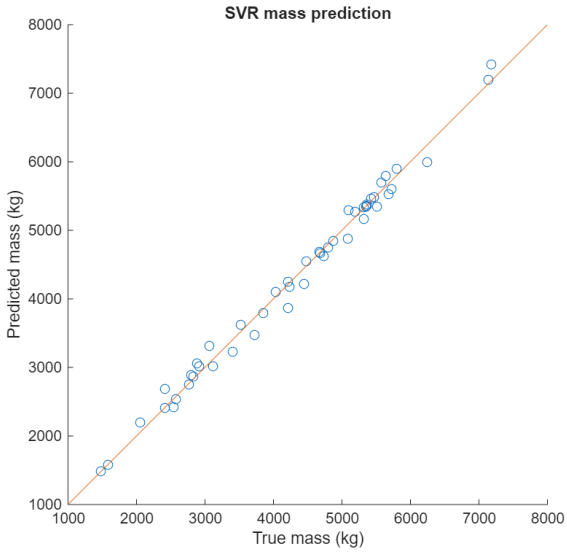
(a) LS Boost Stress prediction.



(b) LS Boost mass prediction.



(c) SVR Stress prediction.



(d) SVR mass prediction.

Figure 5.7: Comparison of metamodel predictions (LS Boost and SVR).

The comparison of mean relative error (Table 5.3) confirms that LS Boost is the most accurate metamodel for predicting stress of the positioner based on the training data. The mean relative stress error is 8% for LS Boost and around 13% for the other metamodels. For mass predictions, the Support Vector Regression has the lowest mean relative error at 2.68% followed by Gaussian Process Regression at 2.81%.

Table 5.3: Mean relative errors (MAPE) of predicted stress and mass for the metamodels.

metamodel	Stress prediction error (%)	Mass prediction error (%)
GPR	13.16	2.81
Neural Network	13.63	5.31
LS Boost	8.00	3.87
SVR	12.54	2.68

5.6 Optimization algorithms

A comparison of the Optimization algorithms was made, comparing how well the three algorithms solves the optimization problem of the two most accurate metamodels, LS Boost predicting stress and SVR predicting mass. The test was made by running each algorithm 10 times, measuring time and accuracy. The accuracy was measured by seeing how many times the algorithm find the value within 5% of a reference value. The reference value was chosen as the lowest value received from any of the algorithms. In (Table 5.4) the total running time of 10 runs is shown together with the hit rate (percentage of results within 5%).

Table 5.4: Comparison of 3 tested optimization algorithms using LS Boost for stress and SVR for mass.

Optimization algorithm	Running time (s)	Hit rate (%)
Fmincon	20	0
Pattern search	100	30
Genetic Algorithm	4134	0

Pattern search outperformed the other algorithms in the sense that it could find lower points more reliably than the other two algorithms. However, the superiority of Pattern search only applied when using LS Boost as metamodel, for any of the other metamodels, Fmincon was both faster and more reliable in finding the minimum.

Later when testing the configurator it was discovered that SVR might be a better choice for stress predictions as well. The reasoning for this is explained in (Chapter 5.7). Therefore another comparison of optimization algorithms is shown in (Table 5.5) where SVR is used for both mass and stress predictions.

Table 5.5: Comparison of 3 tested optimization algorithms using SVR.

Optimization algorithm	Running time (s)	Hit rate (%)
Fmincon	4	80
Pattern search	43	50
Genetic Algorithm	150	0

When only using SVR as metamodel it is clear that Fmincon is the best optimization algorithm of the three tested algorithms. Fmincon is both faster and finds an acceptable minima more often than the other algorithms.

5.7 Final design proposals from the configurator

When comparing the accuracy of metamodels (Table 5.3) the obvoius choice would be to use LS Boost as stress predictor and SVR (or GPR) as mass predictor, since they have the lowest mean relative error. This combination was tested first but it proved to perform worse than expected.

The first area where it underperformed was that it could not find valid solutions for some of the inputs where it was expected to be able to find solutions. This was especially noticed when setting a low *Allowed Max Stress* for example 50 MPa combined with a high *Load* above 10 tons.

The second thing noticed was that it could give some unexpected results for certain inputs that could not be explained. One example of this is that it suddenly decreased the length from 880 to 400 mm when the only input changed was to increase *Allowed Max Stress* from 80 to 90 MPa. This change in length did not follow any pattern and could not be logically explained.

There was a suspicion that these two problems could stem from the fact that LS Boost gives a discrete metamodel. To test this hypothesis LS Boost was replaced with the SVR metamodel so that SVR is used to predict both stress and mass. SVR gives a continuous metamodel which performed the second best with respect to mean relative errors. The results from the configurator now looked more like expected and when varying the inputs, the resulting structure also varied more accordingly. Another advantage of using SVR is that fmincon can be used which significantly increases the solving speed.

The developed configuration tool integrated in Inventor starts with form (Figure 5.8) where the user fills in the wanted inputs. The user then presses the "done" button which starts the optimization process through Matlab and after a short time (20 to 30 s) the optimized CAD model is updated and a summary of the solution is shown (Figure 5.9). The summary shows if the solution is valid, what components are used and specifications of the solution.

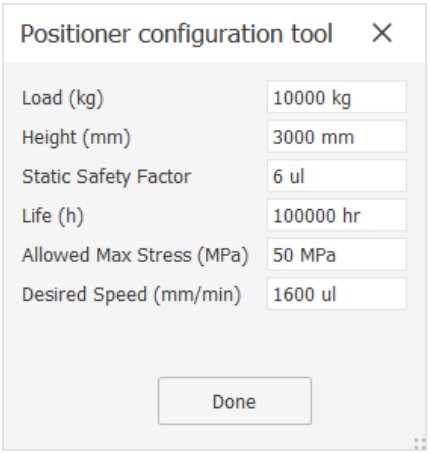


Figure 5.8: Configurator input

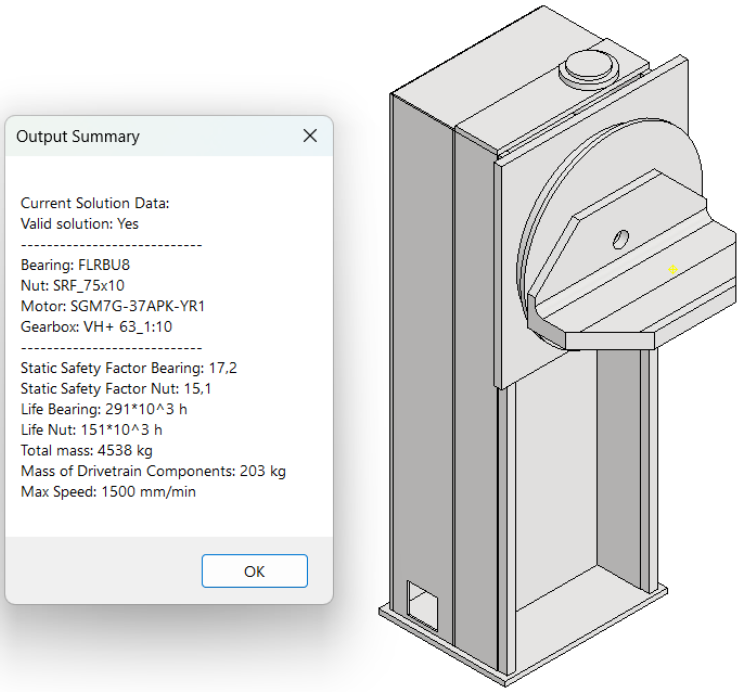


Figure 5.9: Configurator output

Finally the proposed solutions were used in FEM simulations to verify if the solutions matched the requirements from the input. This was tested on five separate sets of inputs arbitrary chosen within reasonable bounds. The resulting max stress from Inventor Nastran was between -23% to $+45\%$ in comparison to the input value, with a mean relative error (Equation 4.3) of 24%. The height of the solutions where always the same as the input height, the static safety factor and life times where always above the requirements from the input. The resulting max speed could end up both below or above the input, but it is possible to regulate the max speed by increasing (or decreasing) the desired max speed until the next optimal point is chosen.

Chapter 6

Discussion

6.1 Design

Selection of components to use in the drivetrain involved first finding what type of driving component to use, screw motion, hydraulic or rack and pinion. Since the design largely hinges on what kind of drivetrain was used, the selection of components was of significance. Criteria for selection were load capacity, cost, drive speed and stroke length. To gauge the options at hand, first thing to check was the in-house information and experience at Yaskawa which mostly involved drivetrain solutions using screw motion. However, to be able to asses if hydraulics or rack and pinon solution are to replace the current screw motion systems, comparisons using the presented criteria was done. The comparison work was done mostly to find the general use cases within the positioner market and to find if any objective measures can justify one drivetrain solution over another. By surveying competitor solutions it could be seen that most common solution for driving vertical movement was hydraulics or screw systems. Rack and pinion solutions were also found for positioners where the stroke length requirement was significantly smaller than is needed here. In addition to the market survey performed by us, the choice was strengthened by the comparison presented forward by J. Pustavhr et. al [46], which provides an in depth experimental comparison of different actuation systems. The comparison study brings up and focuses more on performance metrics which are outside the scope of this thesis to discuss, but general comments regarding draw-backs and benefits when choice between electrical and hydraulic are taken into account here. Primarily the benefit of electrical system in weight and overall size of complete solution.

Both the Yaskawa in-house experience and comparison with criteria resulted in that screw motion systems in general was best suited for the design. However, the criteria presented are not set in stone and is subjective from both Yaskawa and the authors of this report, thus it might very well be more suitable in other applications to use hydraulics or rack and pinion movement systems. In this case it can be argued that the benefits of being able to drive motion precisely in tandem with attached equipment provides reason enough to choose the screw system for motion. But in addition, the scalability of screw motion system which can accommodate the whole designated load capacity range both in speed and cost, the choice is argued to be well founded.

Topology optimization (TO) provides worthful insights in geometry of components subject to a predictable force or other load. In this work TO was used as a guidance in designing load bearing components, but it can be argued that it is un-

necessary and somewhat of an overcomplication of a problem which could be solved with simpler methods such as mechanical calculations or simple simulations. TO can however be of great use for other cases and thus it provides value to be kept in the thesis. For example, TO can provide more freedom in design when using non-traditional construction methods such as additive manufacturing where caution for machining or casting is non-existent.

6.2 Verification

Verification has been performed in multiple sections of this work, and it is of interest to discuss why the implemented verification methods work. Firstly, the FEM stress verification was performed in order to gain understanding and information how Inventor Nastran handles maximum stress outputs in both graphical and output file. By using a simplified case, analytical solutions was used as reference to compare the graphical and output file stresses. As mentioned in the verification (Chapter 4.2.2) section, it was deemed to be safer to use the output file since it in all cases overestimated the maximum stress. However, the sample size and choice of simplified cases (simple circular or rectangular rods) might skew how the output actually behaves. This is something which became more apparent later in the work, when running simulations on the whole assembly which is complex in geometry and contains contact surfaces to compute for. Thus it might be difficult to justify using only the FEM stress verification to base any conclusions on. However it provided a good indication in earlier stages of work to how the software behaves, and in combination with later performed verifications it is argued to serve as a good initial approximation.

Secondly, when verifying the mesh settings for the assembly an arbitrary starting value was chosen which then was decreased for each simulation run. This provided a deeper understanding of the inherent discrete nature of the simulations, where variation of mesh sizes resulted in different outputs of stress and displacement. In addition, the same mesh settings resulted in different outputs when running multiple times. Which is problematic when trying to prove the consistency and robustness of the simulation setup, but the verification showed that the stresses converged and the difference between simulation runs became smaller for finer mesh sizes. To try and explain or give some reasoning behind why the differing results happen, there are two main reasons that might cause this behavior. On one hand, the approximate nature of finite element analysis might introduce some variation but when the mesh is identical, this should not be the case. On the other hand, the mesh might not be exactly identical, since for each run we let the software re-mesh the structure and thus there might be some difference in the mesh which can cause the different results.

In addition, the way of actually reading the stress from nodes in combination with the slight differences in mesh between simulation runs is likely the culprit of causing the differing results. If the maximum stress is read from a node, there is no guarantee that the actual maxima is within the element and thus node-maxima might not be the actual maxima in the output. This could be alleviated by changing how the procedure of reading maximum stress is performed, either by changing what output file to read or to make Inventor Nastran produce more detailed descriptions of each simulation case. This could change and provide more context to where the stress actually is read from.

6.3 Choice of variables for metamodel

When creating a metamodel it is a good idea to try to reduce the number of variables since each variable increases the number of simulations needed to get a certain accuracy. Therefore it was decided to perform a sensitivity analysis which helped in making well-grounded decisions in what variables to keep for the metamodel. Although the result cannot be used directly without analyzing what it means and if it is reasonable.

One important factor to consider when analyzing the result plot (Figure 5.5) is that the result is depending on the chosen intervals for the variables. A large interval will result in bigger influence for that variable, so when choosing intervals it is important to try to scale the intervals similarly for all variable. This however can be difficult when some variables are measured in millimeters and others in meters.

This scaling issue was considered when choosing intervals and therefore most of the intervals have the upper bounder set so that $ub = 10 \cdot lb$ except in cases where the variable has other size constraints for the model to fit together.

Because of this uncertainty all the outliers where controlled with manual simulations to see if the result was reasonable. The length, width and carriage thickness did indeed have a big effect on the stress, even tough the lower boundary for the carriage thickness was set lower than appropriate. The width was first set to be included in the first round of simulations, but when using the metamodel from these simulations the result was always that the width should be minimized. When manually analyzing what effect the width had on the stress there was a obvious reason why smaller width resulted in lower stress. The reason was that the two largest beams (part number 5, see table 5.1) gets closer to the bearing when the width decreases and therefore the stress also decreases. For this reason it was decided that for the second round of simulations the width should always be as small as possible with the only limitation that all components needs to fit between the beams.

The bottom thickness however could not be found to have any major effect on the stress in the manual simulations. A probable cause for why this showed that large influence in the plot is that there could be some specific case where the combined dimensions resulted in some error in the model where parts intersect and therefore result in a large change in stress. A case like this could however not be found when trying to find extreme dimension combinations that could be the cause of this. It could also stem from a mesh or contact error in the simulation or it could be a valid reason that could not be found.

6.4 Metamodel

In the initial evaluation of the metamodels performance, LS Boost looked like the clear winner for predicting stress. Later when testing this metamodel in the configuration tool it did not behave as expected. There were mainly two problems found, the range of accepted inputs was smaller than expected and there was sometimes large jumps in dimensions that did not seem reasonable. The cause of these problems is probably that LS Boost gives a discrete metamodel, this means it has a finite number of different solutions to choose from. While a continuous metamodel (like SVR) has infinite number of different points, which should remove the large jumps in dimensions and accept a larger range of inputs. There was also another benefit in using a continuous metamodel which is that Fmincon can be used to solve the optimization problem which is both faster and more accurate than Pattern search. This reduced the solving time by 96% and increased the hit rate from 30% to 80%. This shows that it is important to test the metamodel on the actual use case before choosing which one should be used. The relative error and comparison plots can give good indications on what metamodel is best but seeing the resulting dimensions in a CAD model can reveal important aspects about how the metamodel behaves and if the output is useful or not.

When examining the stress comparison plots (Figure 5.6-5.7) there are 2-3 outliers to the right side. At these points the metamodels underestimates the stress by more than 30% and this could be a problem for the configuration tool since there is a big risk for the optimization algorithm to choose points in this area since they look better than they actually are. It could look like there are solutions with low mass and acceptable stress level in this area which is what the algorithms are looking for.

The comparison (Table 5.3) confirms that different metamodels needs to be compared for each output to determine which metamodel should be used, since there most likely does not exists one metamodel that is the most accurate for all outputs. This was also the consensus from the related works section (Chapter 3.3) where different studies reached different conclusions of the best metamodel depending on use case. Some studies also concluded that different metamodels should be chosen depending on output, one example is You [29] who reached three different winners for four separate outputs. This fact that multiple metamodels should be tested for each new use case shows that it is preferable to use a method where it is easy to test and compare multiple metamodels. In this study Matlab was used to create metamodels with functions from the Statistics and Machine Learning Toolbox, this was an advantage since all functions had the same inputs and outputs which meant it was easy to test and compare the different models.

6.5 Resulting proposal

The total outcome of the work performed is a design proposal, with accompanying proposal for drivetrain components. To address if the proposal holds any real value is hard to do outright, but it is of course of great importance to try to argue for why the proposal is useful. In the method chapter there is a section about the measures of success (Section 4.1.1) where two main outcomes for success are presented. The

results in this study lands in the second category, that the configuration tool output can be used as a early design proposal and nothing more. As is presented in the results, the mean relative error of 24% lies well above the success criteria of mean relative error of 10%.

The relative error (24%) value is based on five replications of the verification test with different inputs. This number was chosen because five repeated experiments is usually what is recommended when doing experiments in school. This number is too low to give any reliable results with high accuracy but by using statistics the accuracy level can still be interpreted. According to Hubbard [47] there is a 93.75% chance that the median of a population is between the smallest and largest values in any random sample of five from that population, which he calls the "Rule of Five". So we cannot really say anything about outliers but at least there is a great chance of the true mean to be between -23% to $+45\%$.

It can be argued that this error (24%) still can be incorporated into the safety factor, or in essence that the design and dimensions are set in such a way that the uncertainty of the result will have no impact on performance (structurally), but to maintain economic sustainability in material costs the result is deemed to be unsatisfactory for use as is. Further, the time spent using the configuration tool should be kept minimal and less than what a simulation run would take. The time for a simulation run, in the best case of the positioner assembly, takes approximately one minute and the total time spent using the configuration tool also takes approximately one minute. The testing of the configuration tool was performed by the authors of this thesis, which might be a bit unrealistic, but by comparing the time of total configuration tool usage time to only one simulation it is still deemed to be more efficient to use the configuration tool. Although comparing with the time of one simulation is also unrealistic since for a traditional design process, the majority of the time would be spent on choosing and calculating the drivetrain together with designing and dimensioning of the structure. Which would be an even more favorable comparison for the configuration tool.

Design choices performed throughout the work has been made with different objectives for each respective design change, and each serves a certain purpose. To begin with, the first design was largely inspired from existing positioner solutions from Yaskawa, thus the design is from the start limited to look roughly like the existing positioners. But, since the work was not focused on redesigning a positioner we argue that the appearance is not of significance, the importance lies in the application of optimization and metamodeling, more non-standard designs is left for the reader to implement. Thus, the actual parts are not anything groundbreaking or extraordinary, but they serve the purpose of facilitating the creation and application test of the configuration tool.

It should be added that the design follows an already existing product line and thus the risk of applying the configuration tool to something which is wholly unrealistic is minimized, since the design already exists and is used. In addition, the choice of components for drivetrain as discussed above, dictate much of the design around it and thus one could argue that further exploration within the limitations of said drivetrain could be performed. To clarify, the screw system solution can be mounted in various ways, bottom mounted, top mounted (as we have currently) or dual mount in bot top and bottom of screw and thus a true optimal design might

vary between the different cases depending on load. But it was deemed to be outside of the allotted time for the project and thus was left as future work, moreover it is argued that a true optimal design was not part of the main objective to answer here.

The process of designing for simulation automation, in a broader context, can be viewed in multiple ways where emphasis on keeping the simulations as simple as possible is of value. With simple simulations there are less risk of any errors to occur, and it also is time efficient to have simple simulations. The design of what is to be simulated does impact the performance of automating the simulations, especially when multiple geometric or model changes are introduced between each simulation run. Thus, by keeping variables at a minimum reduces complexity in model updates, which saves time and reduces risk of anything going wrong along the way. During the performed work a plethora of different issues with simulations have occurred, some has been presented before in form of verifications but there were other problems as well. Most of the problems was regarding simulation, where model changes prompted from simulation automation could result in failed simulations. The problem was elusive to handle, but one solution was to simply restart the software which seemed to solve the problem. However, the core issue of the failing simulations were not found and thus it can be argued that during the 500 simulation runs, something might have gone wrong, but if that is the case it could not be noticed in the results during verification. In addition, by simply rerunning the failed simulations manually when noticed, in isolation from simulation automation, it resulted in successful simulation runs and the failed data was replaced with the new data manually. Thus, impact on metamodel will be kept minimal, since the failed simulations were drastically fewer than the bulk of data, approximately four failed simulations out of the total 500 simulations. But it is difficult to with certainty state that all simulation data is fault free, thus it is of importance to have in mind when interpreting results and drawing conclusions.

The current solution of the configuration tool uses max stress as a constraint. Minimizing mass while limiting stress would be useful for many applications, but ideally when working with planetary screws it would be valuable to add displacement of the screw as a second constraint since the planetary screw has very tight tolerances. The recommendation from Schaeffler is that tolerance for eccentricity between the nut and support bearing should be below 0.02 mm [9]. However, when automating the Nastran simulations it was only possible to get total max displacements in the three axes, which would not have given any insights of the movement of the screw. To be able to dimension the structure for the eccentric tolerance it would be necessary to switch to a FEM software where it is possible to insert probes to automatically read displacements at the bearing and nut. If probes are not an option it could also be possible to investigate if there is any relation between the max stress and eccentric displacement of the screw and in that case use the max stress as the only constraint.

The research question that was posed and that is intended to be answered is following: "How can a configuration tool be created to provide a design proposal in early stages of design work on a positioner?", and the discussion has revolved around central areas that poses questions or problems to answering the research question and areas surrounding the performed work. In addition to discussing areas closely related to the research question, environmental and economical aspects are of interest to discuss a bit about. The proposal that the configuration tool generates should

not be seen as a finished design but rather as a starting point that the engineer can further develop from. This can save the time it would have required for the engineer to manually come to this stage of the design. It can also reduce the amount of material used in the machine if the design process can start at a more optimal point which helps the engineer to make a more optimal design of the positioner. This is beneficial from both an economic and environmental standpoint.

Another use case for this proposal could be to include cost of the material and components of the model and thereby quickly be able to produce quotes for new projects. Even though the proposal does not have the final dimensions it can still be close enough to use for the quotation. This would save time for the engineers in the early stages of design.

6.6 Generalized method

The method presented in this thesis is used on a specific case, but this methodology could also be used to solve similar problems, for example on another type of positioner or other kinds of similar machines. The overarching methodology presented in Method chapter (Chapter 4), with flowchart representation in (Figure 4.2), presents a generalized overview of how configuration tools of the kind presented in this thesis can be created.

Starting with choosing design for additions to the current design. Then moving on to topology optimization of the most important parts of the structure. Thereafter a parametric CAD model can be created which should then be used in a sensitivity analysis to sort out the most important variables of the model. If off-the-shelf products are going to be used then an algorithm should be created to optimize these product choices. This together with the parametric CAD model can then be used in FEM simulations to get data points for the metamodels where the best metamodel is finally used in the configuration tool. It is then important to verify the accuracy of the results to see to what degree the results from the configurator can be used.

Each of the involved steps will have a difference in emphasis for the present use case, where for example here the emphasis was more on the simulation automation and to gather relevant and verified data from simulations. In other applications, topology optimization might have a larger role in the process, where material reduction is of utmost importance and the geometry and load cases are more intricate. Thus, by adapting, adjusting or adding additional iterations of each individual step of the method, it can be specified for many more implementations than the case-study presented in this thesis.

If someone intends to use this method on multiple similar cases it is a good idea to save knowledge and data from each results so that it can be used to save time or to be able to get better initial inputs for the different steps. Example of knowledge and data that is valuable to save is results from topology optimization and sensitivity analysis, suitable mesh sizes and suitable number of training points for the metamodel. Also it should be possible to reuse the overall structure of the codes and how to parameterize the models.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

To answer the research question this chapter will present the drawn conclusions from results, analysis and discussion. The work performed presents an engineering tool which provides a design proposal based on given use-case specific limitations, in this case load capacity and drive speed. By using a parametrized designed positioner assembly, metamodel and optimization can be directly linked to results from simulation runs on said positioner. The simulation runs are automated and performed in a fashion such that the design space is well covered (with use of LHS) in the data on which the metamodel is built. Further, the accompanying method to creating configuration tools with usage of metamodeling presents an adaptable step by step overview of what essential procedures are. The method is meant to be a guiding, step by step process that can, and should be, adjusted to the specific use-case.

Conclusions are that the results from the configuration tool provide a good initial measure of feasibility of design, but cannot be seen as anything other than an initial proposal. Usage of metamodel in design work consisting of some standardized components that can be limited in a confined design space can be concluded to be suitable. Replacing manual initial design space exploration with a configuration tool that provides a design proposal that fulfills structural limitations, streamlines the engineering work. By reducing time consuming simulations in the early stages of design the configuration tool instead gives freedom to have an initial design to base further work upon. Further it can be concluded that the configuration tool fulfills the time-efficiency objectives presented in the Measures of success section (Section 4.1.1). With a total use time of the configuration tool being on par with the best case scenario of only performing simulations.

However, the work and simulations have to be performed beforehand, in the creation of the metamodel. Thus the proposed most suitable metamodel and simulation automation set up will most probably be different and an individual assessment is needed for implementation in other areas or on other designs. In general, this thesis contributes as a case-study in application and implementation of metamodeling in engineering design. Where design and simulation automation provides the core on which a metamodel is built and integrated into a configuration tool.

In regards to the usability of the configuration tool, it can be concluded to provide a sufficient approximation of cost in early analysis of what a finalized solution might cost to manufacture. Cost analysis could, for example, be performed by using the weight of the positioner to determine total material cost, and the drivetrain compo-

nents most likely already has a price from manufacturer. This is one of the use-cases that can be recommended for a configuration tool with relative error of the level presented in this thesis, where a quick design proposal provides enough information (mass and drivetrain components) to get a estimation for final cost.

7.2 Future Work

Further development of this work can be performed in many ways, since the configuration tool presented only covers one specific product. In general the results can be refined by performing more extensive simulations, in essence gathering more data for usage in building metamodel. With more data, or inputs to the metamodel the resulting error could be smaller than what is accomplished here. Further, finding the limits of respective metamodel variant could be interesting and is also more relevant with larger sample sizes, since more extensive convergence or verification tests could be performed.

Since the performed work involves design automation and implementations in knowledge based engineering, it can be valuable to construct some methodical way of storing data from or in simulations and optimization runs. As an example, the information regarding availability of sizes of steel plates used in manufacturing can be incorporated at the 3D-model level. In the current state of the design automation, this information is not used and left for the engineer to figure out afterwards. This way of working can also be valuable when building design automation for other products, if it is assumed that the same materials are used in the new design as well. In general, in future applications it could be of interest to have knowledge surrounding manufacturing (or sourcing) baked in to the CAD work from the start since trying to add such features in post can be tedious or tricky to implement. The design automation tool could also be made to store past configuration parameters and optimization results. When a new input matches or closely resembles a previous design, the system would retrieve and present the historical data for comparison.

In addition, validation work surrounding the design proposal from configuration tool is of significance. It could be possible that the actual design proposal is correct, but the metamodel data is skewed and thus not proposing the actual optimum design. In essence, that the optimization algorithm is producing optima for the wrong stress. Some kind of validatory check on the output from configuration tool and corresponding metamodel data would provide good insight into if actual optima has been provided.

Moreover, another software or better utilization of Autodesk Inventor Nastran's capabilities could also enhance the results. When reading output data from simulations, only maxima is gathered and to our knowledge the maxima is gathered from nodes. Thus it is of interest to explore possibilities for reading element-wise data or probing specific regions to get more precise measures of maximum stress and displacement. This could possibly resolve, or reduce, some of the prediction errors present in the current metamodel.

References

[1] T. Simpson, J. Poplinski, P. N. Koch, and J. Allen, "Metamodels for Computer-based Engineering Design: Survey and recommendations," *Engineering with Computers*, vol. 17, no. 2, pp. 129–150, Jul. 2001. [Online]. Available: http://link.springer.com/10.1007/PL00007198

[2] E. Kavlakoglu, "What Is Random Forest? | IBM," Oct. 2021. [Online]. Available: https://www.ibm.com/think/topics/random-forest

[3] "Regression Using SVM." [Online]. Available: https://www.niser.ac.in/~smishra/teach/cs460/2020/lectures/lec13_1/

[4] "Workpiece Positioners - HSB | Yaskawa Europe." [Online]. Available: https://www.yaskawa.eu.com/robotics/hardware-accessories/productdetail/product/hsb_813

[5] J. P. MacDuffie, K. Sethuraman, and M. L. Fisher, "Product Variety and Manufacturing Performance: Evidence from the International Automotive Assembly Plant Study," *Management Science*, vol. 42, no. 3, pp. 350–369, 1996. [Online]. Available: https://www.jstor.org/stable/2634348

[6] M. Cederfeldt and F. Elgh, "DESIGN AUTOMATION IN SMES – CURRENT STATE, POTENTIAL, NEED AND REQUIREMENTS," *DS 35: Proceedings ICED 05, the 15th International Conference on Engineering Design, Melbourne, Australia, 15.-18.08.2005*, pp. 248–249 (exec. Summ.), full paper no. DS35_314.46, 2005. [Online]. Available: https://www.designsociety.org/publication/23007/design_automation_in_smes_%E2%80%93_current_state_potential_need_and_requirements

[7] "Load carrying capacity and life | Schaeffler medias." [Online]. Available: https://medias.schaeffler.us/en/knowledge-center/rolling-bearings/load-carrying-capacity-and-life#20514802571

[8] "Screw assemblies | Bosch Rexroth Sverige." [Online]. Available: https://www.boschrexroth.com/sv/se/media-details/8516e0f6-8d8e-47ec-9777-b3af5bac1d2a

[9] Ewellix, "Roller screws catalogue." [Online]. Available: https://medias-library.schaeffler-cdn.com/hcms/v4.1/entity/file/1270814/storage/MTI3MDgxNC8wL21hc3Rlci8vLzI0NjU4OTQ1

[10] T. Tang, L. Wang, M. Zhu, H. Zhang, J. Dong, W. Yue, and H. Xia, "Topology Optimization: A Review for Structural Designs Under Statics Problems," *Materials*, vol. 17, no. 23, Dec. 2024. [Online]. Available: https://www.mdpi.com/1996-1944/17/23/5970

[11] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Structural optimization*, vol. 1, no. 4, pp. 193–202, Dec. 1989. [Online]. Available: https://doi.org/10.1007/BF01650949

[12] D. Weinberg and J. Strain, "Topology Optimization in Autodesk Nastran In-CAD | Autodesk University," Nashville, Sep. 2025. [Online]. Available: https://www.autodesk.com/autodesk-university/class/Topology-Optimization-Autodesk-Nastran-CAD-2017#presentation

[13] F. Campolongo and J. Cariboni, "Sensitivity Analysis: How to Detect Important Factors in Large Models," 2007. [Online]. Available: https://publications.jrc.ec.europa.eu/repository/handle/JRC37120

[14] "Morris method — UQpy v4.2.0 documentation." [Online]. Available: https://uqpyproject.readthedocs.io/en/stable/sensitivity/morris.html

[15] M. D. McKay, R. J. Beckman, and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979. [Online]. Available: https://www.jstor.org/stable/1268522

[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006. [Online]. Available: https://gaussianprocess.org/gpml/chapters/RW.pdf

[17] B. Katherine, "Gaussian Processes for Dummies." [Online]. Available: https://katbailey.github.io/post/gaussian-processes-for-dummies/

[18] F. Lee, "What Is a Neural Network? | IBM," Oct. 2021. [Online]. Available: https://www.ibm.com/think/topics/neural-networks

[19] J. M. Lasaosa, "Tree Ensembles: Bagging, Boosting and Gradient Boosting," Jan. 2023. [Online]. Available: https://towardsdatascience.com/tree-ensembles-theory-and-practice-1cf9eb27781/

[20] E. Arora F. Wendell Miller Distinguished Professor, *Introduction to Optimum Design*. Chantilly, UNITED STATES: Elsevier Science & Technology, 2004. [Online]. Available: http://ebookcentral.proquest.com/lib/bthbib-ebooks/detail.action?docID=286706

[21] K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms. Wiley, New York*, 01 2001. [Online]. Available: https://www.researchgate.net/publication/220045365_Multiobjective_Optimization_Using_Evolutionary_Algorithms_Wiley_New_York

[22] C. Audet and J. E. Dennis Jr., "ANALYSIS OF GENERALIZED PATTERN SEARCHES." *SIAM Journal on Optimization*, vol. 13, no. 3, p. 889, Oct. 2002. [Online]. Available: https://research.ebsco.com/linkprocessor/plink?id=875292a5-e8e5-3b48-ad50-57825846f848

[23] P. Cegielski, D. Golański, P. Kołodziejczak, A. Kolasa, T. Sarnowski, B. Silbert, and Y. Krykavskyy, "Study of design and research of new welding manipulators on the example of an L-type positioner," *Welding Technology Review*, vol. R.

91, nr 7, 2019. [Online]. Available: http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-3ec2b8fb-1d64-4808-ab91-ac2dc76aeacf

[24] C. a. I. Lei, G. U. O. Jianfei, L. I. Bin, and Z. Jin, "Design development and experimental research of positioner for industrial robot," *Manufacturing Technology & Machine Tool*, no. 11, pp. 40–44, Nov. 2022. [Online]. Available: https://1951.mtmt.com.cn/en/article/doi/10.19287/j.mtmt.1005-2402.2022.11.006

[25] J. Wang, T. Gao, M. Li, J. Zhu, L. Song, and W. Zhang, "Topology optimization of modular structures with multiple assemblies and applications to airborne shelves," *Chinese Journal of Aeronautics*, vol. 37, no. 4, pp. 321–332, Apr. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1000936123004296

[26] M. Tarkian, *Design automation for multidisciplinary optimization: a high level CAD template approach*. Linköping: Department of Management and Engineering, Linköping University, 2012. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A556208&dswid=-318

[27] M. Zawidzki and Jankowski, "Multiobjective optimization of modular structures: Weight versus geometric versatility in a Truss-Z system," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 11, pp. 1026–1040, Nov. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/mice.12478

[28] J. Persson, *Efficient Optimization of Complex Products : A Simulation and Surrogate Model Based Approach*, ser. Linköping Studies in Science and Technology. Dissertations. Linköping: Linköping University Electronic Press, Apr. 2015, vol. 1655. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-115939

[29] Y. Yongmin, "Optimal design of pmsm based on automated finite element analysis and metamodeling," *Energies*, vol. 12, p. 4673, 12 2019. [Online]. Available: https://www.researchgate.net/publication/337842338_Optimal_Design_of_PMSM_Based_on_Automated_Finite_Element_Analysis_and_Metamodeling

[30] Q. Shen, F. Vahdatikhaki, H. Voordijk, J. van der Gucht, and L. van der Meer, "Metamodel-based generative design of wind turbine foundations," *Automation in construction*, vol. 138, no. Journal Article, p. 104233, 2022. [Online]. Available: https://go.exlibris.link/RpdLcdCj

[31] H.-S. Park and X.-P. Dang, "Structural optimization based on CAD–CAE integration and metamodeling techniques," *Computer-Aided Design*, vol. 42, no. 10, pp. 889–902, Oct. 2010. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0010448510001107

[32] W. J. C. Verhagen, P. Bermell-Garcia, R. E. C. van Dijk, and R. Curran, "A critical review of Knowledge-Based Engineering: An identification of research challenges," *Advanced Engineering Informatics*,

vol. 26, no. 1, pp. 5–15, Jan. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S147403461100036X

[33] R. Pastor, M. Mihola, Z. Zeman, and A. Boleslavský, "Knowledge-Based Automated Mechanical Design of a Robot Manipulator," *Applied Sciences*, vol. 12, no. 12, p. 5897, Jun. 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/12/5897

[34] C. B. Chapman and M. Pinfold, "The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure," *Advances in Engineering Software*, vol. 32, no. 12, pp. 903–912, Dec. 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997801000412

[35] R. Cinar, H. K. Celik, M. Ucar, and A. E. W. Rennie, "CAD–FEA Integrated Automation Platform for Structural Design, Deformation Simulation, and Size Optimization of Housings in External Gear Pumps," *Applied Sciences*, vol. 15, no. 23, p. 12564, Jan. 2025. [Online]. Available: https://www.mdpi.com/2076-3417/15/23/12564

[36] V. Grković, M. Kolarevic, A. Petrović, and M. Bjelić, *CAD configurator for automatic configuration of modular strongrooms.* Faculty of Technical Sciences Novi Sad, Department of Industrial Engineering and Management, University of Novi Sad, 2020, accepted: 2024-04-08T06:37:59Z Publication Title: The 9th International Conference on Mass Customization and Personalization - Community of Europe - MCP-CE 2020. [Online]. Available: https://scidar.kg.ac.rs/handle/123456789/20517

[37] J. v. Brocke, A. Hevner, and A. Maedche, *Introduction to Design Science Research*, 09 2020, pp. 1–13.

[38] "Topology Optimization in Autodesk Nastran In-CAD | Autodesk University." [Online]. Available: https://www.autodesk.com/autodesk-university/class/Topology-Optimization-Autodesk-Nastran-CAD-2017#presentation

[39] A. T. Nguyen and S. Reiter, "A performance comparison of sensitivity analysis methods for building energy models," *Building Simulation*, vol. 8, Dec. 2015.

[40] "gamultiobj - Find Pareto front of multiple fitness functions using genetic algorithm - MATLAB." [Online]. Available: https://se.mathworks.com/help/gads/gamultiobj.html

[41] F. Bre and V. Fachinotti, "A computational multi-objective optimization method to improve energy efficiency and thermal comfort in dwellings," *Energy and Buildings*, vol. 154, Aug. 2017. [Online]. Available: https://www.researchgate.net/publication/319080996_A_computational_multi-objective_optimization_method_to_improve_energy_efficiency_and_thermal_comfort_in_dwellings

[42] "Design calculation program of Bosch Rexroth | Bosch Rexroth USA." [Online]. Available: https://www.boschrexroth.com/en/us/products/industrial-solutions/linear-motion-technology/linear-motion-designer/

[43] "Optimal Latin Hypercube Technique." [Online]. Available: https://abaqus.uclouvain.be/English/IhrComponentMap/ihr-c-Reference-OptimalLatin.htm

[44] M. Meckesheimer, A. Booker, R. Barton, and T. Simpson, "Computationally Inexpensive Metamodel Assessment Strategies," *AIAA Journal*, vol. 40, pp. 2053–2060, Oct. 2002. [Online]. Available: https://www.researchgate.net/publication/216301102_Computationally_Inexpensive_Metamodel_Assessment_Strategies

[45] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," *International Journal of Forecasting*, vol. 32, no. 3, pp. 669–679, Jul. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207016000121

[46] J. Pustavrh, M. Hočevar, P. Podržaj, A. Trajkovski, and F. Majdič, "Comparison of hydraulic, pneumatic and electric linear actuation systems," *Scientific Reports*, vol. 13, no. 1, p. 20938, Nov. 2023. [Online]. Available: https://www.nature.com/articles/s41598-023-47602-x

[47] D. W. Hubbard, *How to Measure Anything: Finding the Value of Intangibles in Business*, 2nd ed. Hoboken, New Jersey: John Wiley & Sons, 2014.

Appendix A

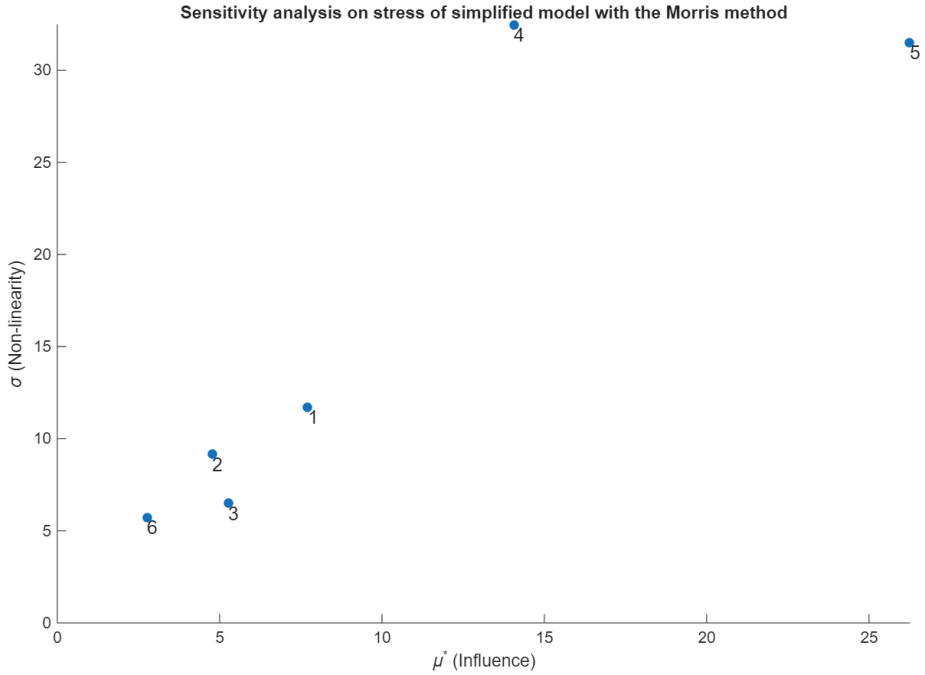
Supplemental Information

A.1 Tables

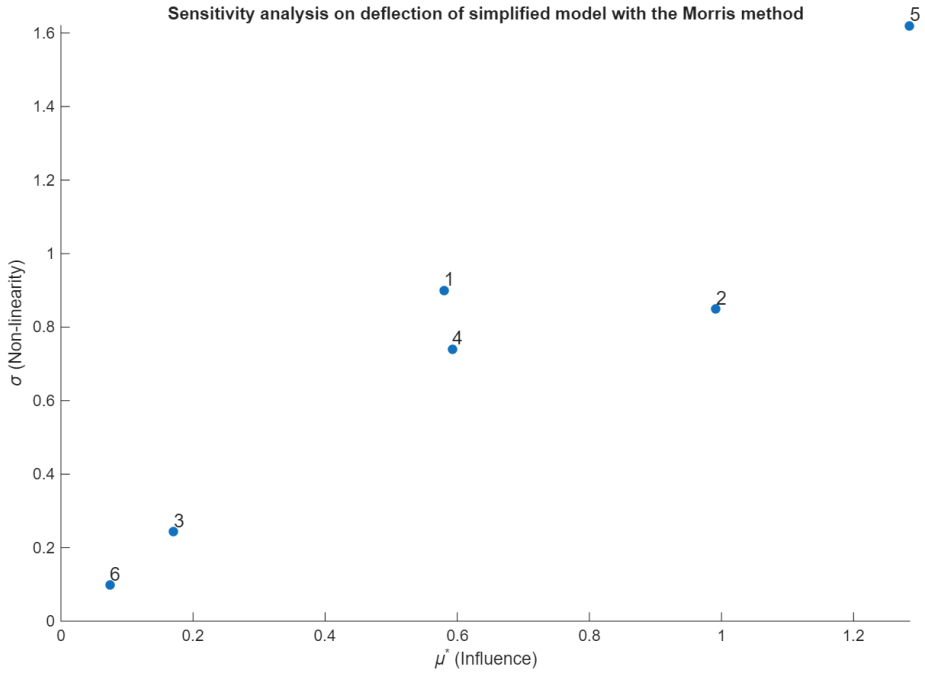
Table A.1: Boundaries of the parametric model

Variable	**Lower boundary**	**Upper boundary**
Height	1250 mm	5000 mm
Width	500 mm	2500 mm
Length	400 mm	2500 mm
TopThickness	5 mm	50 mm
SideThickness	5 mm	50 mm
BottomThickness	5 mm	50 mm
FrontThickness	5 mm	50 mm
BackThickness	5 mm	50 mm
NumHolesSide	0	10
HeightHolesSide	5 mm	300 mm
WidthHolesSide	5 mm	400 mm
SkottFlange1	10 mm	200 mm
SkottFlange2	10 mm	100 mm
SkottThickness	5 mm	50 mm
NumSkott	1	10
DistSkott	50 mm	750 mm
PLSWidth	25 mm	250 mm
PLSThickness	5 mm	50 mm
CarriageThickness	5 mm	50 mm

A.2 Figures



(a) Morris test of stress on simplified model.



(b) Morris test of deflection on simplified model.

Figure A.1: Morris sensitivity test on simplified model.

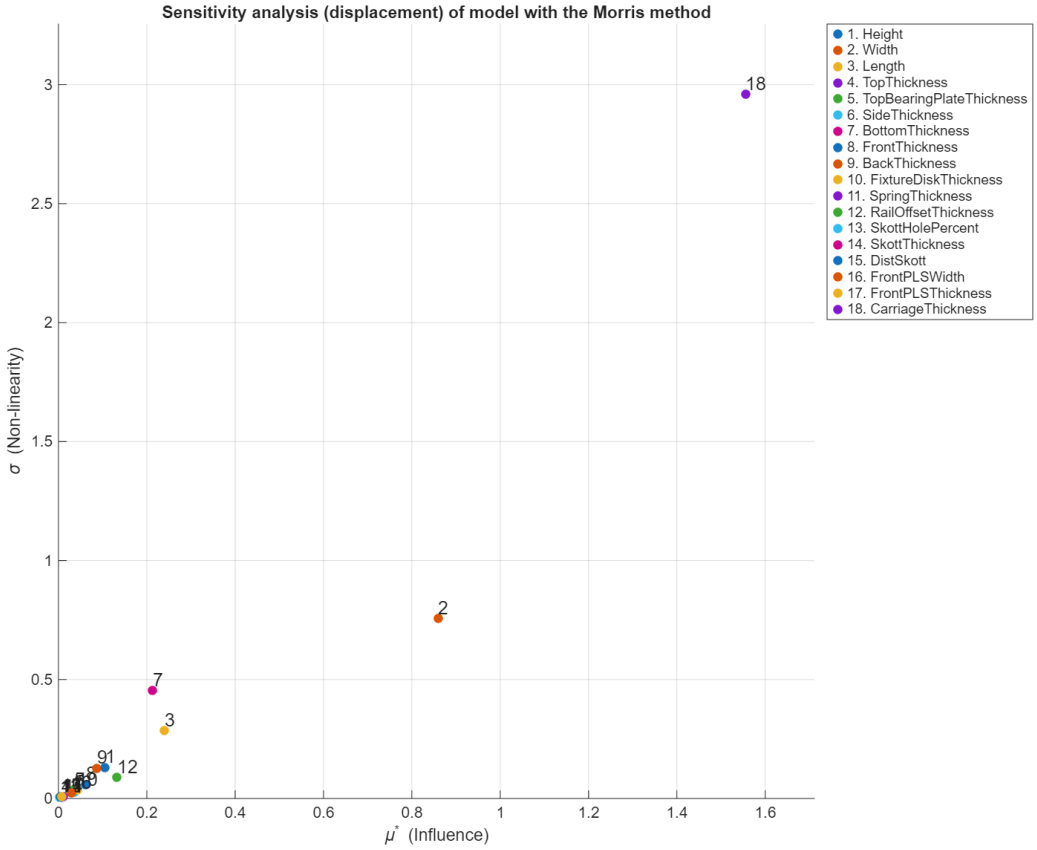


Figure A.2: Sensitivity analysis of the complete model showing parameter influence on displacement.

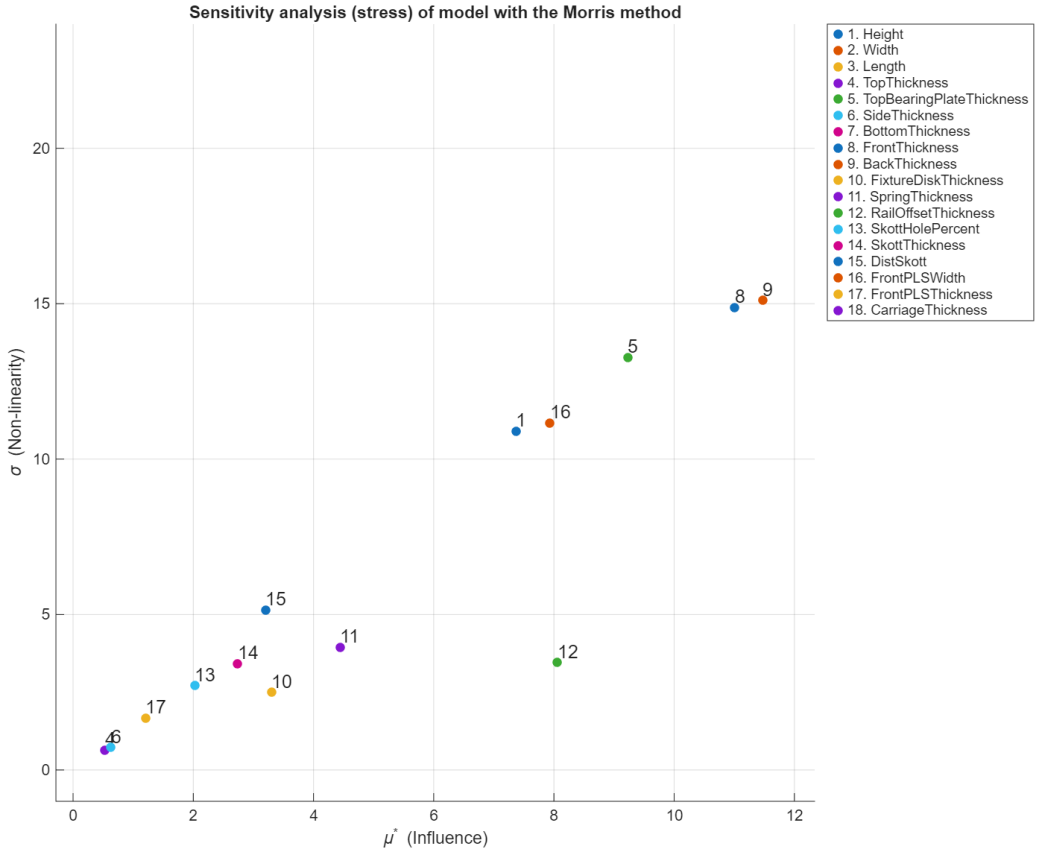


Figure A.3: Sensitivity analysis of the complete model showing parameter influence on stress, zoomed

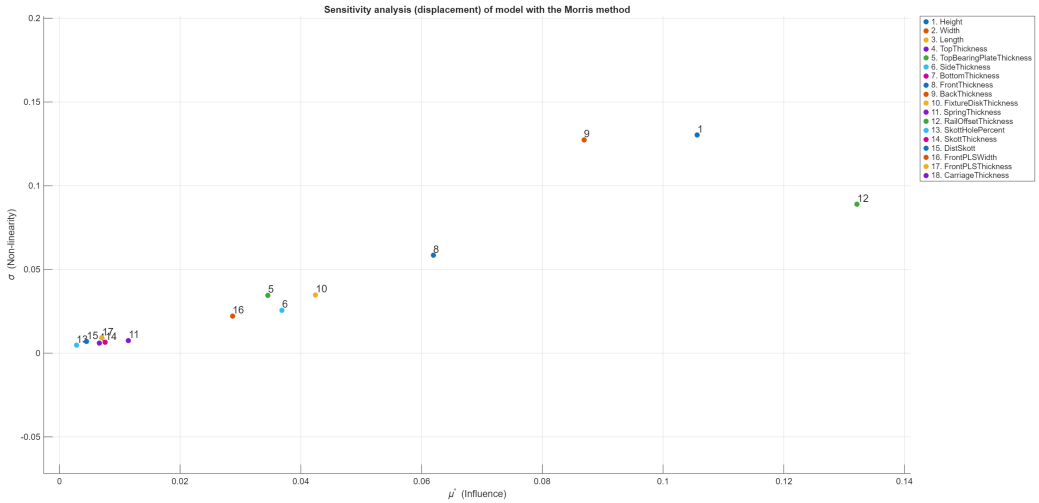


Figure A.4: Sensitivity analysis of the complete model showing parameter influence on displacement, zoomed.

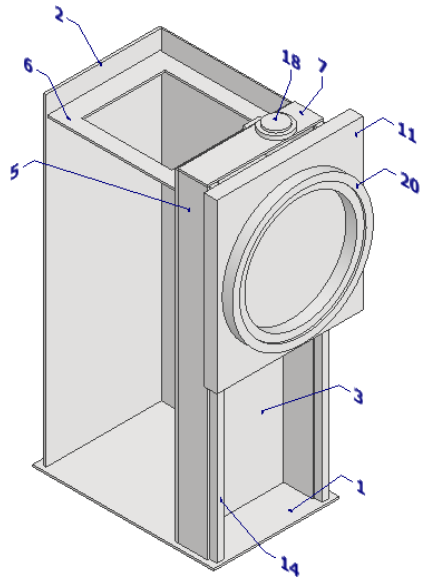


Figure A.5: Positioner assembly in Autodesk Inventor with part numbers attached to respective part. See table 5.1 for part descriptions.

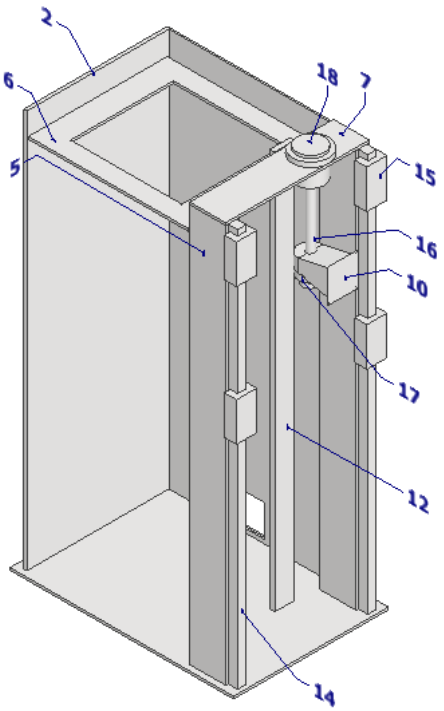


Figure A.6: Positioner assembly in Autodesk Inventor with part numbers attached to respective part. See table 5.1 for part descriptions.

A.3 Matlab code

Algorithm 1 Matlab code: Morris screening.

```
clc;clear all;close all; % Trajectory design matrix
k=19; %Number of variables
delta=2/5; %Step
function [B_star,P_star,D_star]=Morrispath(k,delta)
    p=4; %Number of levels

    x_star=randi([0,p-1],1,k)*(delta/2);
    D_star=diag(randi([0,1],1,k)*2-1);

    A = eye(k);

    P_star=A(randperm(k),:);
    J=ones(k+1,k);
    B=tril(ones(k+1,k),-1);

    B_star=(J.*x_star+(delta/2)*[(2*B-J)*D_star+J])*P_star
       ; %Path
end

%% Choose and save path
function d = calculate_dml(B_m, B_l)
    [num_points, k] = size(B_m);
    d = 0;
    for i = 1:num_points
        for j = 1:num_points
            point_dist = sqrt(sum((B_m(i,:) - B_l(j,:))
               .^2));
            d = d + point_dist;
        end
    end
end
M = 500; %Number of paths to choose from
all_candidates = cell(M, 1);
all_P = cell(M, 1);
all_D = cell(M, 1);

for i = 1:M %Calculate all paths
    [all_candidates{i},all_P{i}, all_D{i}] = Morrispath(k,
        delta);
end

```

```
dist_matrix = zeros(M, M);
for m = 1:M %Calculate distances between all paths
    for l = m+1:M
        dist_matrix(m,l) = calculate_dml(all_candidates{m}, all_candidates{l});
        dist_matrix(l,m) = dist_matrix(m,l); %Symmetry
    end
end

r = 10; %Number of paths
selected_indices = randi(M); %Start with a radom path
remaining_indices = setdiff(1:M, selected_indices);

for i = 2:r
    %For each remaining trajectory find the smallest distance to the
    %selected trajectories.
    min_distances = min(dist_matrix(remaining_indices, selected_indices), [], 2);
    %Find the index of the largest minimum distance
    [~, best_idx_in_rem] = max(min_distances);

    new_best_idx = remaining_indices(best_idx_in_rem);
    selected_indices = [selected_indices, new_best_idx];
    remaining_indices(best_idx_in_rem) = [];
end

final_paths = all_candidates(selected_indices);
all_P = all_P(selected_indices);
all_D = all_D(selected_indices);

% Define the bounds
lb = [1250, 625, 400, 5, 5, 5, 5, 5, 0, 5, 5, 10, 10, 5, 1, 50, 25, 5, 15]; % Lower bounds, Height, Width, Length, TopThickness, SideThickness, BottomThickness,
% FrontThickness, BackThickness, NumHolesSide (pos 9), HeightHolesSide,
% WidthHolesSide, SkottFlange1, SkottFlange2,
% SkottThickness,NumSkott (pos 15),DistSkott,PLSWidth, PLSThickness,CarriageThickness
ub = [5000, 2500, 2500, 50, 50, 50, 50, 50, 10, 300, 400, 200, 100, 50, 10, 750, 250, 50, 150]; % Upper bounds
%intager defines if the variable should be an integer or not
integer = [false,false,false,false,false,false,false,false,true,false,false,false,false,false,true,false,false,false,false];
```

```
range = ub-lb;
paths_scaled=cell(r,1);
for i=1:r %Scale the paths to correct dimensions and round
    the values
    path_i=final_paths{i};
    for ii=1:k+1
        path_i(ii,:) = lb+path_i(ii,:).*range;
        for iii=1:k
            if integer(iii)
                path_i(ii,iii)=round(path_i(ii,iii));
            else
                path_i(ii,iii) = round(path_i(ii,iii),3,"
                   decimals");
            end
        end
    end
    paths_scaled{i}=path_i;
end

filename = "C:\Users\ander\OneDrive - BTH Student\KEVIN
   SAMUELSSON's files - Exjobb_Inventor\Parts\
   ParametricSettings.xlsx";
for i=1:r %Write the paths to Excell
    writematrix(paths_scaled{i},filename,'Sheet','Sheet1',
       'Range','H'+string((i-1)*(k+1)+1))
end

%% Get result from simulations
res_filename = "C:\Users\ander\OneDrive - BTH Student\
   KEVIN SAMUELSSON's files - Exjobb_Inventor\Simulation
   automation\Morris_result.xlsx";
Stress = cell2mat(readcell(res_filename, 'Range', 'A1:A'+
   string(r*(k+1)), "Sheet","Blad1"));
Deflection = cell2mat(readcell(res_filename, 'Range', 'B1:
   B'+string(r*(k+1)), "Sheet","Blad1"));
%% Calculate influence for stress
%Similar code for deflection
EEkr=zeros(k,r);
for ii=1:r
    EEk=zeros(k,1);
    D_star=all_D{ii};
    P_star=all_P{ii};
    B_star=final_paths{ii};
    for i=1:k
        delta_sign=delta*D_star(i,i);
        param=find(P_star(i,:));

```

```
113         EEk(param,1)=(Stress((ii-1)*k+i+1)-Stress((ii-1)*k
                +i))/delta_sign;
114     end
115     EEkr(:,ii)=EEk;
116 end
117
118 mu_star_k=mean(abs(EEkr),2)
119 mu_k=mean(EEkr,2);
120 sigma_k=sqrt(mean((EEkr-mu_k).^2,2))
121
122 figure;
123 scatter(mu_star_k,sigma_k,'filled')
124 title('Sensitivity analysis of simplified model with the
    Morris method')
125 xlabel('\mu^* (Influence)')
126 ylabel('\sigma (Non-linearity)')
127 text(mu_star_k, sigma_k, string(1:k), 'Vert','bottom', '
    Horiz','left', 'FontSize',12)
128 xlim([0 max(mu_star_k)])
129 ylim([0 max(sigma_k)])
```

Algorithm 2 Matlab code: GA_optimization.

```
function best_solution=GA_Optimization(Load, Height, Stat, Life, MaxSpeed)
    global All_Bearings All_Nuts Motors Screws Gearboxes
    %[Bearing, Nut, Motor, Gearbox]
    nvars=4;
    A = []; % Linear inequality constraints
    b = []; % Linear inequality constraints
    Aeq = []; % Linear equality constraints
    beq = []; % Linear equality constraints
    lb = [1,1,1,1]; % Lower bounds
    ub = [length(All_Bearings), length(All_Nuts), length(Motors), length(Gearboxes)]; % Upper bounds
    intcon = [1,2,3,4]; % Integer constraints

    options = optimoptions('gamultiobj','PopulationSize',500); %Set populations size
    [x,fval] = gamultiobj(@(x) Objective_GA_multi(x,Load, Height, MaxSpeed),nvars,A,b,Aeq,beq,lb,ub,@(x) nonlcon(x,Load, Height, Stat, Life),intcon,options);

    % Choose the "best" point
    f_norm = normalize(fval,1,'range');
    dist = sqrt(f_norm(:,1).^2+f_norm(:,2).^2);
    [~, best_idx] = min(dist);
    best_solution = x(best_idx, :);

    [c, ceq] = nonlcon(best_solution, Load, Height, Stat, Life);
    if any(c > 0)
        warning('The solution does not follow the constraints');
        GA_result(best_solution,Load, Height)
        nonlcon(best_solution, Load, Height, Stat, Life)
    end
    %GA_result is used to print some results, same calculations as in
    %Objective_GA_multi and nonlcon
    GA_result(best_solution,Load, Height)
end
```

Algorithm 3 Matlab code: Objective of component optimization.

```
function fitness=Objective_GA_multi(vars,Load, Height, MaxSpeed)
    global All_Bearings All_Nuts Motors Gearboxes
    F_r=Load*9.82;

    %Motor
    d_2=cell2mat(All_Nuts(vars(2),13)); %Root diameter
    d_0=cell2mat(All_Nuts(vars(2),10)); %Nominal diameter
    n_p=160000/d_0; %Permissable speed
    l_cr=Height-250; %Critical length
    f_ncr=4.3; %Bearing setup coefficient for no bearing below
    n_cr=f_ncr*(d_2/l_cr^2)*10^7; %Critical speed
    n_crp=n_cr*0.8; %Critical speed with safety factor

    P_h=cell2mat(All_Nuts(vars(2),11)); %Lead
    eta=0.8; %Efficiency for planetary screw
    M_ta=(F_r*P_h)/(2000*pi*eta); %Drive torque needed

    P_motor=cell2mat(Motors(vars(3),2)); %Motor power (kw)
    eta_gearbox=cell2mat(Gearboxes(vars(4),7))/100; %Efficiency gearbox
    n_allowed_power=P_motor*60*10^6*eta_gearbox/(F_r*P_h); % max rpm allowed wrt motor power
    Motor_speed=cell2mat(Motors(vars(3),5)); %Max motor speed allowed
    %Gearbox
    Gearing=str2double(cell2mat(Gearboxes(vars(4),6)));

    Rotation_speed=min([Motor_speed/Gearing,n_crp, n_p, n_allowed_power]);
    speed=Rotation_speed*P_h; %mm/min
    speed=min(speed,MaxSpeed);
    steel_density=7.8*10^-6; %kg/mm^3
    m_screw=l_cr*pi*d_0^2/4*steel_density;
    mass=cell2mat(All_Bearings(vars(1),7))+cell2mat(All_Nuts(vars(2),9))+cell2mat(Motors(vars(3),8))+cell2mat(Gearboxes(vars(4),4))+m_screw;
    fitness = [mass, -speed];
end
```

Algorithm 4 Matlab code: Non linear constraints of component optimization.

```
function [c,ceq]=nonlcon(vars,Load, Height, Stat, Life)
    global All_Bearings All_Nuts Motors Gearboxes
    F_r=Load*9.82; %Radial load (N)
    p=3;
    n1=150; %Typical running speed
    n2=0;
    q1=10; %10% duty cycle
    q2=100-q1;
    S_0min=Stat; %Static safety factor
    L_10hmin=Life; %Minimum life (h)

    n=(q1*n1+q2*n2)/100; % Equvalent rpm
    %Bearing
    C_0=cell2mat(All_Bearings(vars(1),6))*1000;
    C=cell2mat(All_Bearings(vars(1),5))*1000;
    S_0_B=C_0/F_r;
    L_10h_B=16666/n*(C/F_r)^p;
    %Nut
    C_0=cell2mat(All_Nuts(vars(2),8))*1000;
    C=cell2mat(All_Nuts(vars(2),7))*1000;
    S_0_N=C_0/F_r;
    L_10h_N=16666/n*(C/F_r)^p;
    %Screw
    Max_length=cell2mat(All_Nuts(vars(2),12));
    screw_length=Height-250;

    %Motor
    P_h=cell2mat(All_Nuts(vars(2),11)); %Lead
    eta=0.8; %Efficiency for planetary screw
    M_ta=(F_r*P_h)/(2000*pi*eta); %Drive torque needed

    M_motor=cell2mat(Motors(vars(3),4)); %Torque from
       motor

    %Gearbox
    M_allowed=cell2mat(Gearboxes(vars(4),3)); %Allowed
       torque in gearbox
    Gearing=str2double(cell2mat(Gearboxes(vars(4),6)));
    eta_gearbox=cell2mat(Gearboxes(vars(4),7))/100; %
       Efficiency gearbox
    M_tot=Gearing*M_motor*eta_gearbox; %Total torque from
       motor and gearbox
    % Constraints (must be <= 0)
    c(1) = S_0min-S_0_B;
```

```
41      c(2) = L_10hmin-L_10h_B;
42      c(3) = S_0min-S_0_N;
43      c(4) = L_10hmin-L_10h_N;
44      c(5) = M_tot-M_allowed;
45      c(6) = M_ta-M_tot;
46      c(7) = screw_length-Max_length;
47      ceq = [];
48  end
```

Algorithm 5 Matlab code: Latin Hypercube sapmpling.

```
%% Define Design Space and Sample Count
N_samples = 500; % Number of samples
N_dim = 10;      % Number of variables

%[Height,Length,TopBearingPlateThickness,FrontThickness,
   BackThickness,SpringThickness,RailOffsetThickness,
   FrontPLSWidth,CarriageThickness, Load]
% Define the bounds
lb = [1350, 400, 2, 5, 5, 5, 15, 20, 20, 1000]; % Lower
   bounds
ub = [5000, 1500, 40, 20, 20, 50, 60, 360, 100, 20000]; %
   Upper bounds

% Generate the LHS Matrix (Latin hypercube)
LHS = lhsdesign(N_samples, N_dim, 'criterion', 'maximin');

% Scale the LHS Matrix to the Actual Bounds
range = ub-lb;
LHS_Scaled = round(lb+LHS.*range,3);
%%
filename = "C:\Users\ander\OneDrive - BTH Student\KEVIN
   SAMUELSSON's files - Exjobb_Inventor\Simulation parts\
   InputFile_sim.xlsx";
writematrix(LHS_Scaled,filename,'Sheet','LatinHyper','
   Range','A1')
Stat=6; %Static safety
Life=10^5; %Life (h)
MaxSpeed = 2000;
PartMatrix=zeros(N_samples,4);
for i=1:N_samples
    PartMatrix(i,:) = GA_Optimization(LHS_Scaled(i,10),
       LHS_Scaled(i,1), Stat, Life,MaxSpeed);
end
%%
writematrix(PartMatrix(:,1:2),filename,'Sheet','LatinHyper
   ','Range','K1')
```

Algorithm 6 Matlab code: Metamodels.

```
Resfile_new="C:\Users\kevsa\OneDrive - BTH Student\Documents\Inventor\Exjobb_Inventor\Simulation parts\Data Simulering 2026-04-16\Input_Resultat0416.xlsx";
Input = cell2mat(readcell(Resfile_new, 'Range', 'A50:J501', "Sheet","InputOchResultat100"));
Result = cell2mat(readcell(Resfile_new, 'Range', 'N50:S501', "Sheet","InputOchResultat100"));
Input_ref = cell2mat(readcell(Resfile_new, 'Range', 'A2:J49', "Sheet","InputOchResultat100"));
Result_ref = cell2mat(readcell(Resfile_new, 'Range', 'N2:S49', "Sheet","InputOchResultat100"));
R=1;
%% Gaussian process regression
% Stress metamodel
Mdl_sigma = fitrgp(Input(:,1:end), Result(:,R), ...
    'KernelFunction', 'matern52', ...
    'BasisFunction', 'pureQuadratic', ...
    'Standardize', true, ...
    'Optimizer', 'quasinewton');
% Mass metamodel
Mdl_mass = fitrgp(Input(:,1:end), Result(:,6), ...
    'KernelFunction', 'matern52', ...
    'BasisFunction', 'linear', ...
    'Standardize', true, ...
    'Optimizer', 'quasinewton');

sigma_pred_ref = predict(Mdl_sigma, Input_ref);
mass_pred_ref = predict(Mdl_mass, Input_ref);
rel_error_sigma = abs((sigma_pred_ref - Result_ref(:,R)) ./ Result_ref(:,R)) * 100;
rel_error_mass = abs((mass_pred_ref - Result_ref(:,6)) ./ Result_ref(:,6)) * 100;
mean_err_sigma = mean(rel_error_sigma)
mean_err_mass = mean(rel_error_mass)

figure
scatter(Result_ref(:,1),sigma_pred_ref)
hold on
plot([0 200], [0 200])
title("GPR stress prediction")
xlabel("Stress from FEM (MPa)")
ylabel("Predicted stress (MPa)")
axis square;

```

```
figure
scatter(Result_ref(:,6),mass_pred_ref)
hold on
plot([1000 8000], [1000 8000])
title("GPR mass prediction")
xlabel("True mass (kg)")
ylabel("Predicted mass (kg)")
axis square;

%% Neural network
rng(3);
Mdl_sigma = fitrnet(Input(:,1:end), Result(:,R), ...
    'Standardize', true, ...
    'LayerSizes', [5 5], ...
    'Lambda', 3, ...
    'IterationLimit', 100);

Mdl_mass = fitrnet(Input(:,1:end), Result(:,6), ...
    'Standardize', true, ...
    'LayerSizes', [2 2], ...
    'Lambda', 4, ...
    'IterationLimit', 100);
% ...Plot and calculate mean relative error...

%% Random forrest or Least squares boosting
Mdl_sigma = fitrensemble(Input(:,1:end), Result(:,R), ...
    'Method', 'LSBoost', ...          % Can choose LS
      Boost or Bag(Random forrest)
    'NumLearningCycles', 70, ...
    'Learners', templateTree('MaxNumSplits', 6), ...
    'LearnRate', 0.25);

Mdl_mass = fitrensemble(Input(:,1:end), Result(:,6), ...
    'Method', 'LSBoost', ...
    'NumLearningCycles', 400, ...
    'Learners', templateTree('MaxNumSplits', 2), ...
    'LearnRate', 0.2);
% ...Plot and calculate mean relative error...

%% Support vector machines
Mdl_sigma = fitrsvm(Input(:,1:end), Result(:,R), ...
    'Standardize', true, ...
    'KernelFunction', 'gaussian', ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 200, ...
    'Epsilon', 1);

```

```
83 Mdl_mass = fitrsvm(Input(:,1:end), Result(:,6), ...
84     'Standardize', true, ...
85     'KernelFunction', 'polynomial', 'PolynomialOrder',4,
         ...
86     'KernelScale', 'auto', ...
87     'BoxConstraint', 300, ...
88     'Epsilon', 0.2);
89 % ...Plot and calculate mean relative error...
90
91 %% Fmincon
92 % Define Design Space and Sample Count
93 N_samples = 10; % Number of samples
94 N_dim = 10;       % Number of variables
95
96 %[Height,Width,Length,TopBearingPlateThickness,
     FrontThickness,BackThickness,SpringThickness,
     RailOffsetThickness,FrontPLSWidth,CarriageThickness,
     Load]
97 % Define the bounds
98 lb = [1350, 400, 2, 5, 5, 5, 15, 20, 20, 1000]; % Lower
     bounds
99 ub = [5000, 1500, 40, 20, 20, 50, 60, 360, 100, 20000]; %
     Upper bounds
100
101 % Generate the LHS Matrix (Latin hypercube)
102 LHS = lhsdesign(N_samples, N_dim, 'criterion', 'maximin');
103
104 % Scale the LHS Matrix to the Actual Bounds
105 range = ub-lb;
106 LHS_Scaled = round(lb+LHS.*range,3);
107
108 Height=3000;
109 Load=5000;
110 R_el=70;
111 obj = @(x) predict(Mdl_mass,x);
112 A=[];
113 b=[];
114 Aeq=[];
115 beq=[];
116 lb = [Height, 400, 2, 5, 5, 5, 15, 20, 20, Load]; % Lower
     bounds
117 ub = [Height, 1500, 40, 20, 20, 50, 60, 360, 100, Load]; %
      Upper bounds
118 fvalmin=10^6;
119 xoptmin=zeros(1,11);
120 tic
121 for i =1:N_samples
```

```
122     x0=LHS_Scaled(i,:);
123     [x_opt,fval, exitflag, output] = fmincon(obj, x0, A, b
           , Aeq, beq, lb, ub,@(x) nonlconloc(x,Mdl_sigma,R_el
           ))
124     if fval<fvalmin && nonlconloc(x_opt,Mdl_sigma,R_el) <
           R_el
125         xoptmin=x_opt;
126         fvalmin=fval;
127     end
128 end
129 toc
130 function [c,ceq]=nonlconloc(x,Mdl_sigma,R_el) %Constraints
131     sigma_pred = predict(Mdl_sigma, x);
132
133     % Constraints (must be <= 0)
134     c(1) = sigma_pred-R_el;    % stress <= allowable
135     ceq = [];
136 end
137
138 %% Genetic Algorithm
139 tic
140 for i=1:10
141     nvars=10;
142     A = []; % Linear inequality constraints
143     b = []; % Linear inequality constraints
144     Aeq = []; % Linear equality constraints
145     beq = []; % Linear equality constraints
146     lb = [Height, 400, 2, 5, 5, 5, 15, 20, 20, Load]; %
           Lower bounds
147     ub = [Height, 1500, 40, 20, 20, 50, 60, 360, 100, Load
           ]; % Upper bounds
148     intcon = []; % Integer constraints
149
150     options = optimoptions('ga','PopulationSize',1000, '
           CrossoverFraction',0.8, 'MaxGenerations',10);
151     [x_opt,fval] = ga(obj, nvars, A, b, Aeq, beq, lb, ub
           ,@(x) nonlconloc(x,Mdl_sigma,R_el), intcon, options
           )
152
153     [c, ceq] = nonlconloc(x_opt,Mdl_sigma,R_el);
154     if any(c > 0)
155         warning('The solution does not follow the
               constraints');
156         nonlconloc(x_opt,Mdl_sigma,R_el)
157     end
158 end
159 toc
```

```
160
161 %% Patternsearch
162
163 % Define Design Space and Sample Count
164 N_samples = 10; % Number of samples
165 N_dim = 10;       % Number of variables
166
167 % Define the bounds
168 lb = [1350, 400, 2, 5, 5, 5, 15, 20, 20, 1000]; % Lower
       bounds
169 ub = [5000, 1500, 40, 20, 20, 50, 60, 360, 100, 20000]; %
       Upper bounds
170
171 % Generate the LHS Matrix (Latin hypercube)
172 LHS = lhsdesign(N_samples, N_dim, 'criterion', 'maximin');
173
174 % Scale the LHS Matrix to the Actual Bounds
175 range = ub-lb;
176 LHS_Scaled = round(lb+LHS.*range,3);
177
178 obj = @(x) predict(Mdl_mass,x);
179 A=[];
180 b=[];
181 Aeq=[];
182 beq=[];
183 lb = [Height, 400, 2, 5, 5, 5, 15, 20, 20, Load]; % Lower
       bounds
184 ub = [Height, 1500, 40, 20, 20, 50, 60, 360, 100, Load]; %
       Upper bounds
185 optsc = optimoptions("patternsearch",Algorithm="classic");
186
187 fvalmin=10^6;
188 xoptmin=zeros(1,10);
189 tic
190 for i =1:N_samples
191     x0=LHS_Scaled(i,:);
192     [x_opt,fval] = patternsearch(obj, x0, A, b, Aeq, beq,
          lb, ub,@(x) nonlconloc(x,Mdl_sigma,R_el),optsc)
193     if fval<fvalmin && nonlconloc(x_opt,Mdl_sigma,R_el) <
          R_el
194         xoptmin=x_opt;
195         fvalmin=fval;
196     end
197 end
198 toc
```

Algorithm 7 Matlab code: Optimization for configurator.

```
% Define Design Space and Sample Count
N_samples = 10; % Number of samples
N_dim = 10;       % Number of variables

%[Height,Length,TopBearingPlateThickness,FrontThickness,
   BackThickness,SpringThickness,RailOffsetThickness,
   FrontPLSWidth,CarriageThickness, Load]
% Define the bounds
lb = [1350, 400, 2, 5, 5, 5, 15, 20, 20, 1000]; % Lower
   bounds
ub = [5000, 1500, 40, 20, 20, 50, 60, 360, 100, 20000]; %
   Upper bounds

% Generate the LHS Matrix (Latin hypercube)
LHS = lhsdesign(N_samples, N_dim, 'criterion', 'maximin');

% Scale the LHS Matrix to the Actual Bounds
range = ub-lb;
LHS_Scaled = round(lb+LHS.*range,3);

obj = @(x) predict(Mdl_mass,x);
A=[];
b=[];
Aeq=[];
beq=[];
lb = [Height, 400, 2, 5, 5, 5, 15, 20, 20, Load]; % Lower
   bounds
ub = [Height, 1500, 40, 20, 20, 50, 60, 360, 100, Load]; %
    Upper bounds
fvalmin=10^6;
xoptmin=zeros(1,10);

for i =1:N_samples
    x0=LHS_Scaled(i,:);
    [x_opt,fval, exitflag, output] = fmincon(obj, x0, A, b
       , Aeq, beq, lb, ub,@(x) nonlconloc(x,Mdl_sigma,R_el
       ))
    if fval<fvalmin && nonlconloc(x_opt,Mdl_sigma,R_el)
       <=0
        xoptmin=x_opt;
        fvalmin=fval;
    end
end

```

```
36 function [c,ceq]=nonlconloc(x,Mdl_sigma,R_el) %Constraints
37     sigma_pred = predict(Mdl_sigma, x);
38
39     % Constraints (must be <= 0)
40     c(1) = sigma_pred-R_el;    % stress <= allowable
41     ceq = [];
42 end
43
44 PartMatrix = GA_Optimization(Load,Height,Stat,Life,
       MaxSpeed);
45 Stats=GA_result(PartMatrix,Load, Height);
46
47 filename = "C:\Users\kevsa\OneDrive - BTH Student\
       Documents\Inventor\Exjobb_Inventor\Simulation parts\
       InputFile_sim.xlsx";
48 if xoptmin(1)>0
49     writematrix(xoptmin,filename,'Sheet','Config','Range',
           'A1')
50     writematrix(PartMatrix,filename,'Sheet','Config','
           Range','K1')
51     writematrix(Stats,filename,'Sheet','Config','Range','
           P1')
52     writematrix(1,filename,'Sheet','Config','Range','O1')
           %Found a valid minimum point
53 else
54     writematrix(0,filename,'Sheet','Config','Range','O1')
           % Did not find a valid minimum point
55 end
```

A.4 iLogic code

Algorithm 8 iLogic code: Run simulations main code.

```
GoExcel.Save
GoExcel.Close
Dim outputPath As String = "C:\Users\ander\OneDrive - BTH Student\KEVIN SAMUELSSON's files - Exjobb_Inventor\Simulation parts\ResultFile_sim.xlsx"
Dim ParamPath As String = "C:\Users\ander\OneDrive - BTH Student\KEVIN SAMUELSSON's files - Exjobb_Inventor\Simulation parts\Parameters_sim.xlsx"
Dim InputPath As String = "C:\Users\ander\OneDrive - BTH Student\KEVIN SAMUELSSON's files - Exjobb_Inventor\Simulation parts\InputFile_sim.xlsx"
Dim feaPath As String = "C:\Users\ander\OneDrive - BTH Student\KEVIN SAMUELSSON's files - Exjobb_Inventor\Simulation parts\Assembly_sim\InCAD\FEA\oi1sn9kvp.OUT"
Dim ScrewNutBearingPath As String = "C:\Users\ander\OneDrive - BTH Student\KEVIN SAMUELSSON's files - Exjobb_Inventor\Simulation parts\Parameters_ScrewNutBearing_sim.xlsx"
SharedVariable("outputPath_s") = outputPath
SharedVariable("ParamPath_s") = ParamPath
SharedVariable("InputPath_s") = InputPath
SharedVariable("feaPath_s") = feaPath
SharedVariable("ScrewNutBearingPath_s") = ScrewNutBearingPath

For i As Integer = 150 To 150
        SharedVariable("row") = i
        iLogicVb.RunRule("Change size")
        GoExcel.Save
        GoExcel.Close
        ThisDoc.Document.Update2(True)
    Dim asm As AssemblyDocument = TryCast(ThisDoc.Document, AssemblyDocument)
    If Not asm Is Nothing Then
        Dim refs As DocumentsEnumerator = asm.AllReferencedDocuments
        For Each d As Document In refs
            Try
                d.Update2(True)
            Catch
                'Ignore docs that cannot update (read-only, unresolved, etc.)
```

```
28              End Try
29          Next
30          asm.Update2(True)
31      End If
32          System.Threading.Thread.Sleep(1000)
33      InventorVb.DocumentUpdate()
34          System.Threading.Thread.Sleep(1000)
35          iLogicVb.RunRule("Main")
36          GoExcel.Save
37          GoExcel.Close
38          ThisDoc.Document.Update2(True)
39
40      If Not asm Is Nothing Then
41          Dim refs As DocumentsEnumerator = asm.
                AllReferencedDocuments
42          For Each d As Document In refs
43              Try
44                  d.Update2(True)
45              Catch
46                  'Ignore docs that cannot update (read-only
                        , unresolved, etc.)
47              End Try
48          Next
49          asm.Update2(True)
50      End If
51
52      InventorVb.DocumentUpdate()
53          iLogicVb.RunRule("Run Simulation Load")
54          GoExcel.Save
55          GoExcel.Close
56  '       MessageBox.Show("Run Simulation" & i)
57          System.Threading.Thread.Sleep(1000)
58          iLogicVb.RunRule("Get stress and displacement")
59          GoExcel.Save
60          GoExcel.Close
61  '       MessageBox.Show("Get stress/Displacement" & i)
62          Dim MaxStress As Double = SharedVariable("
                MaxStress")
63          Dim MaxDisp As Double = SharedVariable("MaxDisp")
64          Dim T1 As Double = SharedVariable("T1")
65          Dim T2 As Double = SharedVariable("T2")
66          Dim T3 As Double = SharedVariable("T3")
67
68          GoExcel.Open(outputPath, "Sheet1")
69          GoExcel.CellValue("A" & i) = MaxStress
70          GoExcel.CellValue("B" & i) = MaxDisp
71          GoExcel.CellValue("C" & i) = T1
```

```
72          GoExcel.CellValue("D" & i) = T2
73          GoExcel.CellValue("E" & i) = T3
74          GoExcel.CellValue("F" & i) = iProperties.Volume
               *7.85*10^(-6)
75          GoExcel.Save()
76          GoExcel.Close()
77          ThisApplication.CommandManager.ControlDefinitions.
               Item("AppReturnBaseEnv").Execute
78          System.GC.Collect()
79          System.GC.WaitForPendingFinalizers()
80
81      If System.IO.Directory.Exists(feaPath) Then
82          For Each f In System.IO.Directory.GetFiles(feaPath
               , "*.OUT")
83          Try
84              System.IO.File.Delete(f)
85          Catch
86              MessageBox.Show("Kunde inte rensa")
87          End Try
88          Next
89      End If
90  Next
```

Algorithm 9 iLogic code: Change size.

```
iLogicVb.UpdateWhenDone = True
iLogicVb.RunRule("SharedVariables")
GoExcel.Save
GoExcel.Close
Dim asm As AssemblyDocument = TryCast(ThisDoc.Document,
   AssemblyDocument)

Dim outputPath As String = SharedVariable("outputPath_s")
Dim ParamPath As String = SharedVariable("ParamPath_s")
Dim InputPath As String = SharedVariable("InputPath_s")
h = 1
Dim startCol As Integer = 1
Dim endCol As Integer = 9
Dim readrow As String = h
GoExcel.Open(InputPath, "Config")
Dim Inputs(13) As Double
'Store parameter changes in array "Input"
Inputs(1) = GoExcel.CellValue("A" & h)' Height
Inputs(2) = GoExcel.CellValue("B" & h)' Length
Inputs(3) = GoExcel.CellValue("C" & h)'
   TopBearingPlateThickness
Inputs(4) = GoExcel.CellValue("D" & h)' FrontThickness
Inputs(5) = GoExcel.CellValue("E" & h)' BackThickness
Inputs(6) = GoExcel.CellValue("F" & h)' SpirngThickness
Inputs(7) = GoExcel.CellValue("G" & h)'
   RailOffsetThickness
Inputs(8) = GoExcel.CellValue("H" & h)' FrontPlsWidth
Inputs(9) = GoExcel.CellValue("I" & h)' CarriageThickness
Inputs(10) = 0
Inputs(11) = GoExcel.CellValue("K" & h)'Bearing
Inputs(13) = GoExcel.CellValue("J" & h)'Load (N)
Inputs(12) = GoExcel.CellValue("L" & h)'Screw/Nut
'MessageBox.Show(Inputs(1))
GoExcel.Save
GoExcel.Close
GoExcel.Open(ParamPath, "Sheet1")
For i As Integer = startCol To endCol
    GoExcel.CellValue("B" & i+1) = Inputs(i)
Next
GoExcel.Save
GoExcel.Close
Dim Loadkg As Double = Inputs(13)
Dim ModelStateRail As String
If Loadkg < 2000
```

```
42          Component.ActiveModelState("Slew bearing_sim:1") =
                "403"
43          Component.ActiveModelState("Fixture disk_sim:1") =
                "550"
44          Component.ActiveModelState("L_Bracket_sim:1") = "
               550"
45          ElseIf Loadkg< 4000
46                  Component.ActiveModelState("Slew
                       bearing_sim:1") = "589"
47                  Component.ActiveModelState("Fixture
                       disk_sim:1") = "600"
48                  Component.ActiveModelState("L_Bracket_sim
                       :1") = "600"
49          ElseIf Loadkg< 7500
50                  Component.ActiveModelState("Slew
                       bearing_sim:1") = "695"
51                  Component.ActiveModelState("Fixture
                       disk_sim:1") = "800"
52                  Component.ActiveModelState("L_Bracket_sim
                       :1") = "800"
53          ElseIf Loadkg< 10000
54                  Component.ActiveModelState("Slew
                       bearing_sim:1") = "1079"
55                  Component.ActiveModelState("Fixture
                       disk_sim:1") = "1100"
56                  Component.ActiveModelState("L_Bracket_sim
                       :1") = "1100"
57          ElseIf Loadkg<= 20000
58                  Component.ActiveModelState("Slew
                       bearing_sim:1") = "1144"
59                  Component.ActiveModelState("Fixture
                       disk_sim:1") = "1200"
60                  Component.ActiveModelState("L_Bracket_sim
                       :1") = "1200"
61  End If
62
63  If Loadkg<2500
64          ModelStateRail  = "1000"
65          Component.ActiveModelState("Rail_sim:1") =
                ModelStateRail
66          Component.ActiveModelState("Rail_sim:2") =
                ModelStateRail
67          Component.ActiveModelState("RunnerBlock_sim:1") =
                ModelStateRail
68          Component.ActiveModelState("RunnerBlock_sim:2") =
                ModelStateRail
69          Component.ActiveModelState("RunnerBlock_sim:3") =
```

```
            ModelStateRail
70          Component.ActiveModelState("RunnerBlock_sim:4") =
               ModelStateRail
71          ElseIf Loadkg<5000
72          ModelStateRail = "2500"
73          Component.ActiveModelState("Rail_sim:1") =
               ModelStateRail
74          Component.ActiveModelState("Rail_sim:2") =
               ModelStateRail
75          Component.ActiveModelState("RunnerBlock_sim:1") =
               ModelStateRail
76          Component.ActiveModelState("RunnerBlock_sim:2") =
               ModelStateRail
77          Component.ActiveModelState("RunnerBlock_sim:3") =
               ModelStateRail
78          Component.ActiveModelState("RunnerBlock_sim:4") =
               ModelStateRail
79          ElseIf Loadkg<7500
80          ModelStateRail = "5000"
81          Component.ActiveModelState("Rail_sim:1") =
               ModelStateRail
82          Component.ActiveModelState("Rail_sim:2") =
               ModelStateRail
83          Component.ActiveModelState("RunnerBlock_sim:1") =
               ModelStateRail
84          Component.ActiveModelState("RunnerBlock_sim:2") =
               ModelStateRail
85          Component.ActiveModelState("RunnerBlock_sim:3") =
               ModelStateRail
86          Component.ActiveModelState("RunnerBlock_sim:4") =
               ModelStateRail
87          ElseIf Loadkg<10000
88          ModelStateRail = "7500"
89          Component.ActiveModelState("Rail_sim:1") =
               ModelStateRail
90          Component.ActiveModelState("Rail_sim:2") =
               ModelStateRail
91          Component.ActiveModelState("RunnerBlock_sim:1") =
               ModelStateRail
92          Component.ActiveModelState("RunnerBlock_sim:2") =
               ModelStateRail
93          Component.ActiveModelState("RunnerBlock_sim:3") =
               ModelStateRail
94          Component.ActiveModelState("RunnerBlock_sim:4") =
               ModelStateRail
95          ElseIf Loadkg<12500
96          ModelStateRail = "10000"
```

```
97          Component.ActiveModelState("Rail_sim:1") =
               ModelStateRail
98          Component.ActiveModelState("Rail_sim:2") =
               ModelStateRail
99          Component.ActiveModelState("RunnerBlock_sim:1") =
               ModelStateRail
100         Component.ActiveModelState("RunnerBlock_sim:2") =
               ModelStateRail
101         Component.ActiveModelState("RunnerBlock_sim:3") =
               ModelStateRail
102         Component.ActiveModelState("RunnerBlock_sim:4") =
               ModelStateRail
103         ElseIf Loadkg<15000
104         ModelStateRail = "12500"
105         Component.ActiveModelState("Rail_sim:1") =
               ModelStateRail
106         Component.ActiveModelState("Rail_sim:2") =
               ModelStateRail
107         Component.ActiveModelState("RunnerBlock_sim:1") =
               ModelStateRail
108         Component.ActiveModelState("RunnerBlock_sim:2") =
               ModelStateRail
109         Component.ActiveModelState("RunnerBlock_sim:3") =
               ModelStateRail
110         Component.ActiveModelState("RunnerBlock_sim:4") =
               ModelStateRail
111         ElseIf Loadkg<17500
112         ModelStateRail = "15000"
113         Component.ActiveModelState("Rail_sim:1") =
               ModelStateRail
114         Component.ActiveModelState("Rail_sim:2") =
               ModelStateRail
115         Component.ActiveModelState("RunnerBlock_sim:1") =
               ModelStateRail
116         Component.ActiveModelState("RunnerBlock_sim:2") =
               ModelStateRail
117         Component.ActiveModelState("RunnerBlock_sim:3") =
               ModelStateRail
118         Component.ActiveModelState("RunnerBlock_sim:4") =
               ModelStateRail
119         ElseIf Loadkg<20000
120         ModelStateRail = "17500"
121         Component.ActiveModelState("Rail_sim:1") =
               ModelStateRail
122         Component.ActiveModelState("Rail_sim:2") =
               ModelStateRail
123         Component.ActiveModelState("RunnerBlock_sim:1") =
```

```
            ModelStateRail
124         Component.ActiveModelState("RunnerBlock_sim:2") =
            ModelStateRail
125         Component.ActiveModelState("RunnerBlock_sim:3") =
            ModelStateRail
126         Component.ActiveModelState("RunnerBlock_sim:4") =
            ModelStateRail
127         ElseIf Loadkg=20000
128         ModelStateRail = "20000"
129         Component.ActiveModelState("Rail_sim:1") =
            ModelStateRail
130         Component.ActiveModelState("Rail_sim:2") =
            ModelStateRail
131         Component.ActiveModelState("RunnerBlock_sim:1") =
            ModelStateRail
132         Component.ActiveModelState("RunnerBlock_sim:2") =
            ModelStateRail
133         Component.ActiveModelState("RunnerBlock_sim:3") =
            ModelStateRail
134         Component.ActiveModelState("RunnerBlock_sim:4") =
            ModelStateRail
135 End If
136
137 Dim excelPath As String = SharedVariable("
       ScrewNutBearingPath_s")
138 GoExcel.Open(excelPath, "Sheet1")
139 Dim targetColumn As String = "B"
140
141 Dim startColB As Integer = 5
142 Dim endColB As Integer =13
143 Dim readrowB As String = Inputs(11).ToString
144
145 Dim startColS As Integer = 23
146 Dim endColS As Integer =24
147 Dim readrowS As String = Inputs(12).ToString
148
149 Dim startColN As Integer = 16
150 Dim endColN As Integer =20
151 Dim readrowN As String = Inputs(12).ToString
152
153 For i1 As Integer = startColS To endColS
154         Dim colLetter As String = Chr(64 + i1)
155     cellValue = GoExcel.CellValue(colLetter & readrowS)
156     GoExcel.CellValue(targetColumn & i1-8) = cellValue
157 Next
158
159 For i2 As Integer = startColB To endColB
```

```
160         Dim colLetter As String = Chr(64 + i2)
161     cellValue = GoExcel.CellValue(colLetter & readrowB)
162     GoExcel.CellValue(targetColumn & i2-4) = cellValue
163 Next
164
165 For i3 As Integer = startColN To endColN
166         Dim colLetter As String = Chr(64 + i3)
167     cellValue = GoExcel.CellValue(colLetter & readrowN)
168     GoExcel.CellValue(targetColumn & i3-6) = cellValue
169 Next
170     If Not asm Is Nothing Then
171         Dim refs As DocumentsEnumerator = asm.
               AllReferencedDocuments
172         For Each d As Document In refs
173             Try
174                 d.Update2(True)
175             Catch
176             End Try
177         Next
178         asm.Update2(True)
179     End If
180 For Each doc As Document In ThisApplication.ActiveDocument
    .AllReferencedDocuments
181     doc.Update2(True)
182 Next
183 Dim BearingOuterDiameter As Double = GoExcel.CellValue("B4
    ")
184 Dim BearingFlangeDiameter As Double = GoExcel.CellValue("
    B2")
185 Dim NutOuterDiam As Double = GoExcel.CellValue("B12")
186 Dim NutVariant As String = GoExcel.CellValue("O" &
    readrowN)
187 Dim RailHeight As Double = Parameter("Rail_sim:1", "H2")
188 Dim RunnerBlockThickness As Double = Parameter("
    RunnerBlock_sim:1", "H_") -Parameter("RunnerBlock_sim:1
    ", "H2")
189 'Component.ActiveModelState("Carriage_Mount_sim:1") =
    NutVariant
190
191 GoExcel.Save
192 GoExcel.Close
193 GoExcel.Open(ParamPath, "Sheet1")
194 'MessageBox.Show(Inputs(12))
195 'B12 OuterDiamBearing
196 GoExcel.CellValue("B12") = BearingOuterDiameter
197 'B14 BearingDistFront
198 'B15 RailOffsetHeightEXCEL
```

```
Dim BearingDist As Double
If BearingFlangeDiameter < NutOuterDiam Then
        GoExcel.CellValue("B14") = NutOuterDiam/2+5
        BearingDist = GoExcel.CellValue("B14")
        GoExcel.CellValue("B15") = BearingDist +
           NutOuterDiam / 2
        If Inputs(11) < 6'Bearing
                GoExcel.CellValue("B" & 11) = NutOuterDiam
                    * 2 + 350' Width
        Else
                GoExcel.CellValue("B" & 11) = NutOuterDiam
                    * 2 + 450' Width
        End If
Else
        GoExcel.CellValue("B14") = BearingFlangeDiameter/2
        BearingDist= GoExcel.CellValue("B14")
        GoExcel.CellValue("B15") = BearingDist +
           BearingFlangeDiameter / 2
        If Inputs(11) < 6'Bearing
                GoExcel.CellValue("B" & 11) =
                   BearingFlangeDiameter * 2 + 350' Width
        Else
                GoExcel.CellValue("B" & 11) =
                   BearingFlangeDiameter * 2 + 450' Width
        End If
End If
Dim RailOffset As Double = GoExcel.CellValue("B15")
'B13 DistCarriage
GoExcel.Save
GoExcel.Close
GoExcel.Open(excelPath, "Sheet1")
GoExcel.CellValue("B17") = (RunnerBlockThickness +
   RailOffset + RailHeight - BearingDist)
GoExcel.Save
GoExcel.Close
d46 = BearingOuterDiameter/2 + 25
d47 = d46
RuleParametersOutput()
InventorVb.DocumentUpdate()
    If Not asm Is Nothing Then
        Dim refs As DocumentsEnumerator = asm.
           AllReferencedDocuments
        For Each d As Document In refs
            Try
                d.Update2(True)
            Catch
            End Try
```

```
238         Next
239         asm.Update2(True)
240     End If
241 For Each doc As Document In ThisApplication.ActiveDocument
       .AllReferencedDocuments
242     doc.Update2(True)
243 Next
244 GoExcel.DisplayAlerts = False
245 iLogicVb.UpdateWhenDone = True
246 docUpdate = ThisApplication.ActiveDocument
247 docUpdate.Update2(True)
```

Algorithm 10 iLogic code: Code necessary for simulations.

```
1  Sub Main()
2      Dim oIlogic As IiLogicAutomation
3      oIlogic = iLogicVb.Automation
4
5      ' Setting iLogic External Rules directory to Inventor
          Nastran's iLogic external rules folder
6      Dim ExternalRuleDirectory(0) As String
7          ExternalRuleDirectory(0) = "E:\Inventor Nastran
              2025\In-CAD\System\iLogic"
8      oIlogic.FileOptions.ExternalRuleDirectories =
          ExternalRuleDirectory
9      iLogicVb.RunExternalRule("IlogicExtRule_EnterInCADEnv"
          )
10
11     ' Setting iLogic Addin DLLs directory to Inventor
          Nastran's System folder
12         oIlogic.FileOptions.AddinDirectory = "E:\Inventor
              Nastran 2025\In-CAD\System\iLogic"
13
14 End Sub
```

Algorithm 11 iLogic code: Run simulation.

```
AddReference "NINIlogic"
Sub Main()
                SharedVariable("SimulationRunning") = True
        ThisDoc.Document.Update2(True)
        InventorVb.DocumentUpdate()

           iLogicVb.RunExternalRule("
              IlogicExtRule_EnterInCADEnv")
           Dim InputPath As String = SharedVariable("
              InputPath_s")
             h = SharedVariable("row")
             GoExcel.Open(InputPath, "LatinHyper")
             Dim Loadkg As Double
             Loadkg = GoExcel.CellValue("J" & h)

           Dim AddinObj As New NINIlogic.InCadAddin
           Dim Force As String = CStr(-1*Loadkg*9.82)
           Dim Moment As String = CStr(1658*Loadkg/1000)

           strCmdDeleteLoad1 = "<DeleteLoad ID=""1""/>"
           strCmdDeleteLoad2 = "<DeleteLoad ID=""2""/>"

           strCmdForLoad1 = "<Load Name=""Load 1"" ID=""1"
              " Type=""0"" SubType=""0"" EntitiesCount=""1
              "" Subcases=""2"" ArrowLength=""70""
              ArrowColor=""65280"" DisplayDensity=""3""
              Direction=""0"" CoordinateSystemID=""0""
              Display=""1"" Fx=""0[N]"" Fy="""" & Force & "
              [N]"" Fz=""0[N]"" HasVariableLoad=""0""
              HasHeatGenTable=""0"" HadConvCoeffTable=""0"
              " HasRadTable=""0"" HasTransTable=""0""
              HasVelTransTable=""0"" HasAccTransTable=""0"
              " HasVelFreqTable=""0"" HasAccFreqTable=""0"
              " HasFreqTable=""0"" HasFreqPhaseTable=""0""
               HasAccFreqPhaseTable=""0""><Entity1
              GeometryType=""2"" GeometryID=""3""
              ComponentName=""L_Bracket_sim:1""/></Load>"
           strCmdForLoad2 = "<Load Name=""Moment 1"" ID=""
              2"" Type=""1"" SubType=""0"" EntitiesCount="
              "1"" Subcases=""2"" ArrowLength=""50""
              ArrowColor=""65280"" DisplayDensity=""3""
              Direction=""0"" CoordinateSystemID=""0""
              Display=""1"" Mx="""" & Moment & "[N m]"" My=
              ""0[N m]"" Mz=""0[N m]"" HasVariableLoad=""0
```

```
                "" HasHeatGenTable=""0"" HadConvCoeffTable="
                "0"" HasRadTable=""0"" HasTransTable=""0""
                HasVelTransTable=""0"" HasAccTransTable=""0"
                " HasVelFreqTable=""0"" HasAccFreqTable=""0"
                " HasFreqTable=""0"" HasFreqPhaseTable=""0""
                 HasAccFreqPhaseTable=""0""><Entity1
                GeometryType=""2"" GeometryID=""1""
                ComponentName=""Fixture disk_sim:1""/></Load
                >"
23
24          i = AddinObj.Run(strCmdDeleteLoad1)
25          i = AddinObj.Run(strCmdDeleteLoad2)
26
27          i = AddinObj.Run(strCmdForLoad1)
28          i = AddinObj.Run(strCmdForLoad2)
29
30           strCmdForMesh = "<GenerateMesh/>"
31           strCmdForSolve = "<RunAnalysis/>"
32
33           i = AddinObj.Run(strCmdForMesh)
34           i = AddinObj.Run(strCmdForSolve)
35
36          iLogicVb.UpdateWhenDone = True
37  End Sub
```

Algorithm 12 iLogic code: Read stress and displacement.

```
Dim lines As String() = System.IO.File.ReadAllLines(
   outFilePath)

Dim maxStress As Double = 0
Dim maxDisp As Double = 0
Dim foundStress As Boolean = False
Dim foundDisp As Boolean = False

For Each line As String In lines

    If Line.Contains("MAXIMUM TET ELEMENT VON MISES STRESS
       ") Then
        Dim parts() As String = Line.Split("=")
        If parts.Length > 1 Then
            maxStress = Val(parts(1).Trim())
            foundStress = True
        End If
    End If

    If Line.Contains("MAXIMUM DISPLACEMENT MAGNITUDE")
       Then
        Dim parts() As String = Line.Split("=")
        If parts.Length > 1 Then
            maxDisp = Val(parts(1).Trim())
            foundDisp = True
        End If
    End If
Next

'Get displacements--------------

Dim T1, T2, T3 As Double
Dim foundHeader As Boolean = False

For i As Integer = 0 To lines.Length - 1

    If lines(i).Contains("M A X I M U M   D I S P L A C E
       M E N T S") Then
        foundHeader = True
    End If

    If foundHeader Then
        Dim currentLine As String = lines(i).Trim()

```

```
41          Dim parts() As String = currentLine.Split(New
               String() {" "}, StringSplitOptions.
               RemoveEmptyEntries)
42
43          If parts.Length >= 4 AndAlso parts(0) = "2" Then
44              ' parts(0) = ID (1)
45              ' parts(1) = T1
46              ' parts(2) = T2
47              ' parts(3) = T3
48
49              T1 = Val(parts(1))
50              T2 = Val(parts(2))
51              T3 = Val(parts(3))
52
53              Exit For
54          End If
55      End If
56  Next
57
58  SharedVariable("T1") = T1
59  SharedVariable("T2") = T2
60  SharedVariable("T3") = T3
61  SharedVariable("MaxStress") = maxStress
62  SharedVariable("MaxDisp") = maxDisp
63  SharedVariable("SimulationRunning") = False
```

Algorithm 13 iLogic code: Configurator main code.

```
iLogicForm.Show("Positioner configuration tool", FormMode.
   Modal)
iLogicVb.RunRule("RunMatlab")
iLogicVb.RunRule("Change_Size_Config")
iLogicVb.RunRule("Display Data")
```

Algorithm 14 iLogic code: Run Matlab.

```
' Initiate MATLAB
Dim matlab As Object
matlab = CreateObject("Matlab.Application")

matlab.Execute("load('C:\Users\kevsa\OneDrive - BTH
   Student\Documents\Inventor\Exjobb_Inventor\Matlab\
   SVRMetamodel.mat')")
' Send parameters from Inventor to Matlab
Dim h As Double = Parameter("HeightConfig")
Dim m As Double = Parameter("Load")
Dim stat As Double = Parameter("StaticSafety")
Dim life As Double = Parameter("Life")/3600
Dim Rel As Double = Parameter("R_el") / 1000
Dim Max_Speed As Double = Parameter("MaxSpeed")*2 'desired
    speed to maxspeed
matlab.PutWorkspaceData("Height", "base", h)
matlab.PutWorkspaceData("Load", "base", m)
matlab.PutWorkspaceData("Stat", "base", stat)
matlab.PutWorkspaceData("Life", "base", life)
matlab.PutWorkspaceData("R_el", "base", Rel)
matlab.PutWorkspaceData("MaxSpeed", "base", Max_Speed)

' Run Matlab codes
Dim scriptPath As String = "C:\Users\kevsa\OneDrive - BTH
   Student\Documents\Inventor\Exjobb_Inventor\Matlab"
matlab.Execute("cd '" & scriptPath & "'")
matlab.Execute("ReadExcell")
matlab.Execute("RunFromInventor")
```

Algorithm 15 iLogic code: Display solution summary.

```
iLogicVb.RunRule("SharedVariables")

Dim InputPath As String = SharedVariable("InputPath_s")
Dim ScrewNutBearingPath As String = SharedVariable("
   ScrewNutBearingPath_s")

GoExcel.Open(InputPath, "Config")
Dim Inputs(7) As Double
Dim Components(4) As String
'Store parameter changes in array "Input"
Inputs(1) = GoExcel.CellValue("P1") 'Safety Factor Bearing
Inputs(2) = GoExcel.CellValue("Q1") 'Safety Factor Nut
Inputs(3) = GoExcel.CellValue("R1") 'Life Bearing
Inputs(4) = GoExcel.CellValue("S1") 'Life Nut
Inputs(5) = GoExcel.CellValue("T1") 'Mass
Inputs(6) = GoExcel.CellValue("U1") 'Max Speed
Inputs(7) = GoExcel.CellValue("O1") 'Found solution or not
Dim FoundSolution As String
If Inputs(7) = "0"
        FoundSolution = "No"
Else
        FoundSolution = "Yes"
End if
Dim Bearing, Nut, Motor, Gearbox As Double
Bearing = GoExcel.CellValue("K1") 'Bearing
Nut = GoExcel.CellValue("L1") 'Nut
Motor = GoExcel.CellValue("M1") 'Motor
Gearbox = GoExcel.CellValue("N1") 'Gearbox
GoExcel.Save
GoExcel.Close
GoExcel.Open(ScrewNutBearingPath, "Sheet1")
Components(1) = GoExcel.CellValue("D" & Bearing) 'Bearing
Components(2) = GoExcel.CellValue("O" & Nut) 'Safety
   Factor Nut
Components(3) = GoExcel.CellValue("AC" & Motor) 'Life
   Bearing
Components(4) = GoExcel.CellValue("Z" & Gearbox) & "_" &
   GoExcel.CellValue("AA" & Gearbox)'Life Nut
GoExcel.Save
GoExcel.Close
' --- Display the inputs in a MessageBox ---
Dim msg As String = "Current Solution Data:" & vbLf & _
"Valid solution: " & FoundSolution & vbLf & _
"----------------------------" & vbLf & _
```

```
"Bearing: " & Components(1) & vbLf & _
"Nut: SRF_" & Components(2) & vbLf & _
"Motor: " & Components(3) & vbLf & _
"Gearbox: VH+ " & Components(4) & vbLf & _
"----------------------------" & vbLf & _
"Static Safety Factor Bearing: " & Round(Inputs(1),1) &
   vbLf & _
"Static Safety Factor Nut: " & Round(Inputs(2),1) & vbLf &
    _
"Life Bearing: " & Round(Inputs(3)/1000) & "*10^3 h" &
   vbLf & _
"Life Nut: " & Round(Inputs(4) / 1000) & "*10^3 h" & vbLf
   & _
"Total mass: " & Round(iProperties.Volume*7.85*10^(-6)) &
   " kg" & vbLf & _
"Mass of Drivetrain Components: " & Round(Inputs(5),1) & "
    kg" & vbLf & _
"Max Speed: " & Round(Inputs(6)) & " mm/min"

MessageBox.Show(msg, "Output Summary")
```

Faculty of Engineering, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden