Hinge front-loading

A methodology to enable the near instantaneous selection of an optimal hinge

Arthur Alglave

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday January 25, 2019 at 14:00.

Student number:	4626095	
Project duration:	February 12, 2018 – January 25, 2019	
Thesis committee:	Prof. dr. ir. L. L. M. Veldhuis,	TU Delft, Chair
	Dr. ir. G. La Rocca,	TU Delft, Supervisor
	Dr. ir. T. van den Berg,	Fokker Aerostructures, Supervisor
	Dr. ir. W. J. C. Verhagen,	TU Delft

This thesis is confidential and cannot be made public until January 25, 2024.

TU Delft

Acknowledgements

Before starting this report, I would like to express my gratitude to everyone who supported me and contributed to completing this thesis project.

First and foremost, I would like to thank my supervisors at Fokker Aerostructures, Tobie van den Berg and Ton van der Laan. Your continuous input, guidance and unconditional support were essential, especially during the rough patches of the project. You helped me become a better student and hopefully set me on the right track to becoming a better engineer. I also want to thank Gianfranco La Rocca, my supervisor at the TU Delft. Thank you very much for your time, your patience and understanding. Your feedback was always very valuable.

I would also like to thank experts at Fokker for their precious help, especially Tim Janssen, Ronald van der Aa, and Martijn van Rij. I sincerely appreciated your involvement in the project and it could not have been successful without you. Finally, I would like to thank Akshay Raju Kulkarni and Maurice Hoogreef for sharing their knowledge and experience about this topic.

Last but not least, I would like to thank my friends and family for their love and encouragements throughout this year. A special thank you to Timo, Lorenzo, and Peter, who helped me tremendously.

Arthur Alglave
Delft, January 2019

Acronyms

BFS	Best First Search
CAD	Computer Assisted Design
DOE	Design of Experiments
DSM	Design Structure Matrix
GA	Genetic Algorithm
HDOT	Hinge-System Design and Optimization Tool
IDEALISM	Integrated & Distributed Engineering Services Framework for MDO
ILS	Informed Linear Search
KBE	Knowledge Based Engineering
MDO	Multidisciplinary Design and Optimization
MS	Margin of Safety
OEM	Original Equipment Manufacturer
OML	Outer Mold Line
PDP	Product Development Process
PIDO	Process Integration and Design Optimization
OOP	Object Oriented Programming
GUI	Graphical User Interface
UML	Unified Modeling Language
RF	Reserve Factor
RIAM	Rudder In A Month
RSM	Response Surface Methodology
RTS	Random Tree Search

Nomenclature

The following symbols will be used within the body of the document:

Subscripts

be	Bearing
fb	Fixed bushing (bushing between sliding bushing and outer lug)
il	Inner Lug (also referred to *Central* or *male* lug)
in	Refers to *inner* dimension, closer to bolt axis (e.g., $r_{be,in}$ in figure 1.4)
ol	Outer Lug (also referred to *Clevis* or *female* lug)
out	Refers to *outer* dimension, further from bolt axis (e.g., $r_{be,out}$ in figure 1.4)
sb	Sliding bushing (bushing between bolt and fixed bushing)
$total$	Total attribute of the hinge, e.g. $m_{total} = m_{bolt} + m_{be} + m_{il} + m_{ol} + m_{sb} + m_{fb} + m_{nut}$

Other Symbols

$\mathbf{p}$	Optimization parameters
$\mathbf{x}$	Design vector
D	Diameter code (integer variable to lookup characteristics)
n	Index of a data structure (integer variable to lookup characteristics)

Superscripts

$*$	Optimized design variable
c	Continuous design variable
i	Integer design variable
L	Lower bound of a design variable
U	Upper bound of a design variable

Hinge variables

α	In-plane load angle (see figure 2.2)	[°]
β	Taper angle of the lug (see figure 2.2)	[°]
σ	Material ultimate tensile strength	$[MPa]$
θ	Out-of-plane load angle (see figure 2.3)	[°]
c	Cost	[euros]
e_{ax}	Axial lug eccentricity	$[mm]$
e_{tr}	Transverse lug eccentricity	$[mm]$
F	Radial load	$[kN]$

g	Gap between the bearing and the fixed bushing (see figure 2.19)	$[mm]$
m	Mass	[g]
mat	A material with given properties (yield strength, density, cost density, etc.)	
r	Radius	$[mm]$
t	Lug thickness (see figure 2.3)	$[mm]$
t_{sb}	Thickness ($d_{out} - d_{in}$) of the sliding bushing	$[mm]$
v	Volume	$[mm^3]$

Contents

List of Figures

List of Tables

List of Algorithms

I

Part A - Thesis

1 Introduction

1.1. Design of control surfaces at Fokker

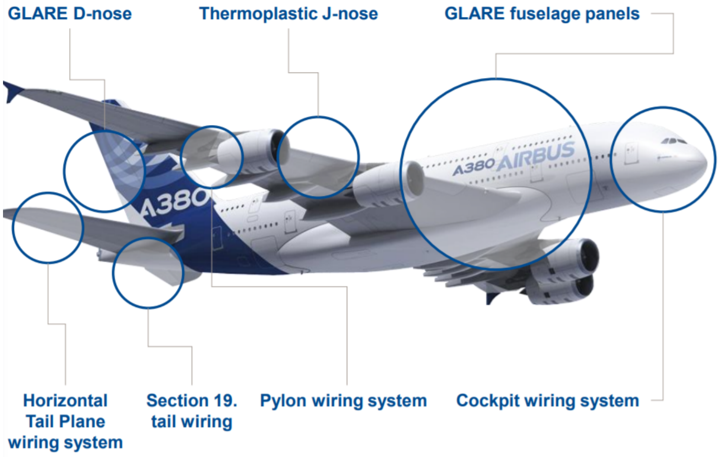


Figure 1.1: Elements of the Airbus A380 designed and/or manufactured by Fokker [9].

Original Equipment Manufacturer (OEM) companies such as Airbus and Boeing outsource part of the manufacturing and design work to subcontractors, called *Tier-1* suppliers [1]. For Tier-1 companies, such as Fokker, the development of a product, from the OEM proposal to the first delivery, is a crucial step of the Product Development Process (PDP) [37]. Those months of development work determine if the product will be profitable during the aircraft's lifetime. In this context, the objectives of Fokker are as follows:

- Increase the quality and reduce the cost of their products
- Shorten the product design time
- Convince the OEM that the product will perform as planned

Fokker Aerostructures designs movable surfaces (rudders, elevators) for business jets and transport aircrafts. The vertical tail is composed of a fixed part (the fin or vertical stabilizer, for lateral stability) and of a movable part (the rudder, for lateral control). Those two elements are linked together with the so-called "rudder-fin interface", (or "hinge system" [18]). The interface includes several hinges that, together, enable the rotation and actuation of the rudder around the vertical axis (see figure 1.2).

The hinges need to carry internal loads to sustain the deflection during various flight maneuvers. The development of a single hinge may take up to six weeks [18, 33]. The *internal* loads on the hinge depend on the *external* loads on the rudder, as well as on the numbering and positioning of the hinges. The design of the hinges is limited by the rudder Outer Mold Line (OML). The hinges must fit within the available space to avoid intersection with the OML geometry.

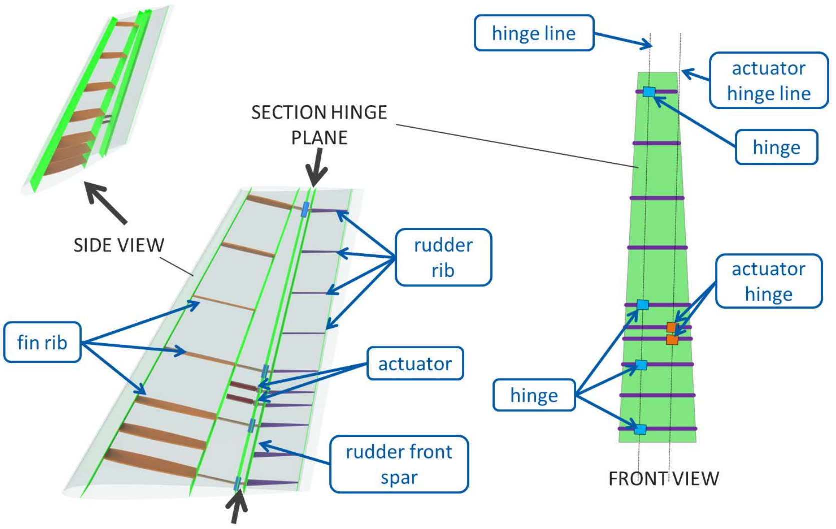


Figure 1.2: The rudder-fin interface consists of several hinges. The surface rotates around the rotation hinge line (hinges in blue) when the actuators push or pull the actuator hinge line (hinges in red) [16].

The vertical tail design is decomposed into subsystems (fin, rudder, interface) and into sub-disciplines (design; stress, cost and mass analysis) in order to enable the repartition of the workload within different team members at Fokker. This approach results in different professionals (engineers, designers) working in separate teams, each on a different sub-problem. This organizational structure creates dependencies. For instance, the *fin* team needs to know the location and number of hinges, which is determined by the *hinge system* team. This has two consequences:

1. Obtaining an optimal design is possible, but (a) only on a sub-system and (b), generally, the result is only optimal with respect to only one criterion (mass or cost).

2. The workflow needs to be structured in a way that enables the different teams to work concurrently, without having to assume too many design parameters.

The vertical tail design process starts with the high-level definition of the rudder-fin interface. From the given interface plane, the position and number of hinges are approximated, based on experience and previous rudder design cases. Then, the detailed hinge design can be carried out concurrently to tailor the hinge for the the given fin and rudder designs. The rudder-fin interface might be optimal by itself (e.g., minimum mass for the given loads), but might lead to a heavier (or worse, unfeasible) structure for both the fin and the rudder. In that case, the process has to be started all over again. As a result, the completion of a single feasible rudder design can take up to 24 months.

Since 2010, the Tools and Methods department at Fokker Aerostructures, together with researchers from the Aerospace Engineering faculty at the TU Delft, have been investigating the application of advanced design methodologies such as Knowledge Based Engineering (KBE) and Multidisciplinary Design and Optimization (MDO) to the design of movable surfaces [32, 36]. In 2017, a drastic decrease in rudder design-time has been achieved in the context of the Integrated & Distributed Engineering Services Framework for MDO (IDEALISM) cooperation [34]. The development of the *Rudder Generator* KBE application enables to automate many parts of the rudder design process.

1.2. The hinge front-loading problem

With the current rudder design process at Fokker, the interface needs to be determined first before detailed design steps can be carried out. Minimizing the time required to execute this first step would enable Fokker to reduce the design-time of the whole vertical tail.

Since each hinge of the interface must be analyzed for different requirements (loads to carry, available space, and fail-safe functionality), several hinge optimizations must be executed for each interface design iteration. Instead of repeating this search several times for each hinge, engineers at Fokker have been planning to run the hinge optimization *before* handling the rudder-fin interface problem. All the hinge optimization results would be stored, in order to be able to instantly obtain an optimal hinge "*off-the-shelf*" in the course of the movable surface design. This way, the optimal hinge is obtained in an instant, whatever the requirements on the hinge in the given rudder-fin interface design iteration. In the literature, this methodology is called *front-loading*, which is defined as identifying and solving problems early in the PDP [31].

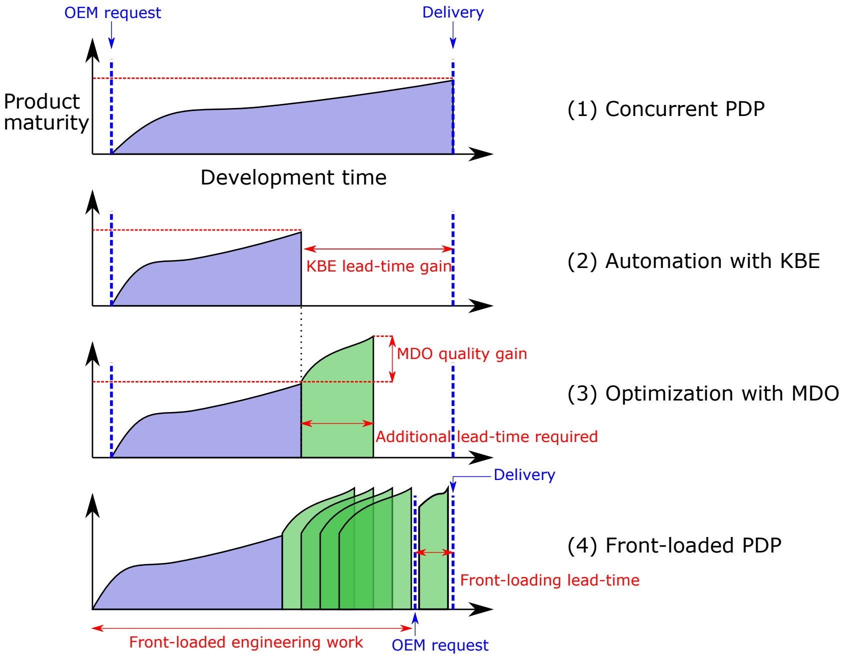


Figure 1.3: Effects of front-loading on product quality and lead-time. Front-loading is achieved by developing a KBE application - (1) to (2) - and an optimization workflow - (2) to (3). Then, the optimization is repeated for different requirements, before the OEM request is known (4).

Figure 1.3 explains the front-loading methodology adopted by Fokker. Four different PDP approaches are represented in terms of their development time (horizontal axis) and their maturity (vertical axis). The first (uppermost) process is the conventional, concurrent PDP methodology. The second is the result of automating the design process, and the third of optimizing the design. The fourth process is similar to the third one, but it optimizes various designs *before* the design process even starts (see “OEM request” marker in figure 1.3). This approach reduces the lead-time even further compared to the other approaches. Only the front-loading approach enables to achieve Fokker’s ambitious objective in terms of quality and lead-time, because the optimal product is instantly selected *off-the-shelf*.

1.3. Use-cases and research questions

The previous subsection introduced the benefits of using front-loading on the hinges of the rudder. In the light of those advantages, the main **research question** is:

How to implement the front-loading methodology for a hinge assembly?

The **first use-case** for a hinge front-loading methodology is the rudder-fin optimization:

During a rudder-fin optimization, the hinge surrogate model can be executed over a million times. Obtaining the optimal hinges very quickly (10 ms) would enable a relatively short (3 hours) rudder-fin optimization with a Genetic Algorithm.

This instantaneous availability of the results is crucial during a rudder optimization, which can require the hinge optimization to be executed a large number of times. Even a reasonably short hinge analysis run-time will generate an impractical rudder-fin optimization run-time. Kulkarni has already tackled the optimization problem in [18], and then with Hoogreef in [16]. In [26], a Genetic Algorithm (GA) was used on the rudder-fin interface optimization, with a population of 20 interfaces per generation and 30 iterations, which took about 50 hours. A near instant optimization could drastically reduce this run-time. A shorter run-time can also be leveraged to run further iterations, or into a higher fidelity rudder optimization. Being able to store and retrieve the optimal hinges, or to approximate them quickly would, therefore, be a significant improvement.

This project contributes to the research of Kulkarni and Hoogreef on three aspects:

1. Improvements on the hinge analysis and design space by using a more extensive standard parts database, completing the stress analysis offered by the Hinge-System Design and Optimization Tool (HDOT) developed by Kulkarni in 2015 [18].

2. Development of an efficient, scalable and multi-objective search algorithm to solve the hinge optimization problem.

3. Development of a method to *front-load* the hinge optimization problem by using the two previous items.

This third item enables the near instantaneous selection discussed earlier. This requirement can be rephrased as the **first sub-question**:

How to obtain Pareto-optimal hinges as quickly as possible? (< 10 ms)

Second, the results can be used to gain insight into the functioning of the system and its design space through data visualization. It would allow engineers to discuss and to justify trade-off decisions, within the multidisciplinary design team and during the discussions with the OEM. This **second use-case** can be formalized as follows:

Engineers at Fokker would like to visualize the best hinges in terms of mass and cost, in a way that enables design trade-off during the discussion with the OEM.

A front-loaded hinge dataset contains hundreds of hinges, each for a different combination of requirements on the rudder bracket, gap and load case. Each hinge design includes 30 variables that are potentially interesting to inspect. Data visualization only allows for representing a maximum of five variables at most (X, Y, Z axis, color, size on a 3D plot). Choosing the correct variables to work with the database of optimal hinges is not trivial. The use-case described above results in a **second sub-question**:

How to represent the Pareto-optimal hinges in a way that enables trade-off decisions?

1.4. Developed methodology to achieve hinge front-loading

Answering the research questions requires the following:

1. **Front-loading**: A method that finds the optimal hinges *near instantly.*

2. **Optimization**: A method that finds the optimal hinges.

3. **Model**: A method to compute the characteristics of a hinge assembly (including cost, mass, stress and design rules).

Use of KBE for the hinge modeling

Although the stress analysis of the hinge assembly is relatively simple, the process to model a complete, fully detailed hinge assembly is not straightforward. The complexity arises from several factors:

1. The model is multidisciplinary. The cost, mass, geometry and stress must be taken into account in order to evaluate and validate the hinge design.

2. Some elements of the hinge design methods are complex. For example, the fitting between the bolt and bearing and the fretting between the components are constrained by design tables and rules.

3. The hinge components cannot be analyzed separately because their analysis is coupled.

4. Standard parts and materials result in many variables. For instance, the selection of a bolt determines about 20 geometrical parameters and the choice of a material defines 13 material properties (e.g., density, yield and ultimate strength).

Together, those factors make it challenging to write a hinge modeling as a simple function. If the following function f represented the tool, it could be written as:

$$\underbrace{(\mathrm{RF}, m_{nut}, c_{bolt}, \ldots)}_{\text{30 outputs}} = f\underbrace{(\mathrm{bolt}, \mathrm{nut}, \sigma_{bolt}, \ldots)}_{\text{30 inputs}} \tag{1.1}$$

With this paradigm, the function above should return at least 30 outputs (variables that have to be inspected), and require more than 30 inputs. All the procedures to determine the cost, mass and reserve factors of all the components would need to be precisely defined, in a specific sequence of instructions.

Using an Object Oriented Programming (OOP) approach helps to deal with this complexity. OOP enables the data to be nested into *objects*, so that the sentence "*If the margin of safety of the left element of the outer lug is not respected*" can be written as “`if hinge.outer_lug[0].RF < 1`”. With pure OOP, however, the order of execution of the various analyses needs to be prescribed.

A better approach is to use KBE for the modeling of the hinge. In [19], La Rocca defines KBE as “*a technology based on the use of dedicated software tools called KBE systems, which are able to capture and systematically reuse product and process engineering knowledge, with the final goal of reducing time and costs of product development by means of: (1) Automation of repetitive and non-creative design tasks and (2) Support of MDO in all the phases of the design process*”. KBE enables to create the model with a **declarative programming** approach, i.e., declare *what* the system is, not *how* the simulation workflow looks like. It makes the development more straightforward and the code implementation more concise. Moreover, the use of KBE enables the model to be more than just a function that is iterated over:

1. Direct **visualization of the geometry** enables inspection of the data modeled. This is of high importance for Fokker. The model needs to be transparent and visual so that it can be trusted.

2. With the Graphical User Interface (GUI), the model can become an **interactive tool for design**.

3. Lastly, using KBE would enable the hinge model to be easily and seamlessly **integrated** within the existing Fokker framework, notably in the *Rudder Generator*.

As a conclusion, the hinge modeling will be done using a KBE platform, since it enables a fully detailed, declarative and lazy[1] hinge model.

Mixed discrete-continuous, multi-objective optimization methods

The optimization aims at obtaining the best hinge assemblies in terms of mass and cost. The stress analyses of the hinge components result in several non-linear constraints that ensure that the components will carry the loads without failing. As for any structural optimization problem, those constraints are typically in contradiction with the objective, which tries to minimize the amount of material used. If the material properties of the components are known, the sizing of the assembly is solvable by standard optimization methods such as quadratic programming. However, three characteristics of the hinge design make the optimization problem unconventional.

Multi-disciplinary Design and Optimization The hinge optimization problem is an MDO problem, not only in the literal sense (three disciplines: stress, cost and mass) but also in the more prominent aspect of MDO, as we have a set of coupled analyses that have conflicting interests [30]. For example, if the outer lug thickness t_{ol} increases, the allowable load on the lug increases but the bending moment on the bolts increases, reducing the allowable load of the bolt.

Combinatorial optimization With standard parts, the hinge optimization problem becomes more of a *combinatorial* rather than a *continuous* optimization problem. A combinatorial optimization problem consists of finding an optimal object from a finite set of objects [29], e.g., select a sequence of components $S = [\mathrm{bolt}, \mathrm{bearing}, \ldots]$ one after the other. Each component has multiple characteristics that are coupled and all of them affect the hinge. For example, figure 1.4 focuses on the dimensions of the bearing in the hinge assembly. Rounding the bolt radius would affect the bearing inner radius, which in turn changes the other bearing dimensions and affects the sliding bushing and lugs. Attempting

[1] Also referred as call-by-need, the program evaluates the function only when needed. Lazy evaluation for the Hinge Generator is discussed in subsection 2.6.2.

to solve the problem as a quasi-discrete problem with continuous variables (i.e., $\min(m(R,L))$, with $R \in [R_1,\ldots,R_n], L \in [L_1,\ldots,L_n]$), where the design variables could be rounded off will have severe drawbacks on the mass and cost of the hinge.

Multi-objective Both the cost and mass of the hinge must be taken into account.

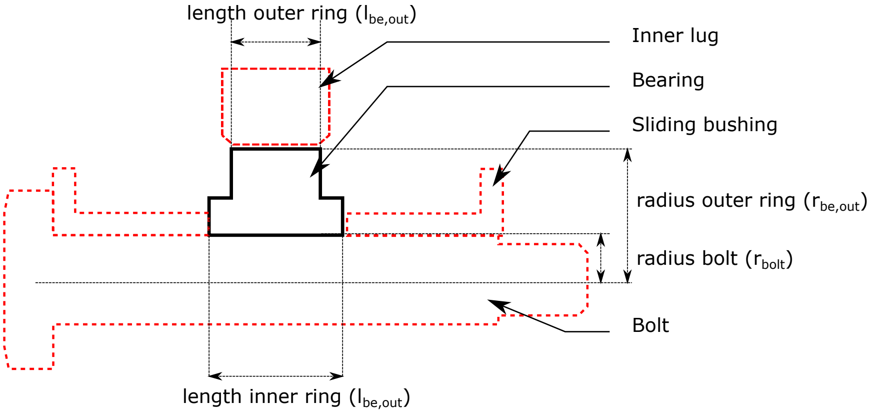

length outer ring ($l_{be,out}$)
Inner lug
Bearing
Sliding bushing
radius outer ring ($r_{be,out}$)
radius bolt (r_{bolt})
Bolt
length inner ring ($l_{be,out}$)

Figure 1.4: The bearing (dark, full lines) is in contact with three other components (dashed, red lines). Therefore, the bearing choice influences the other hinge components. Rounding the bolt radius affects the bearing, which snowballs and affects the sliding bushings and lugs.

As a conclusion, the optimization method must handle multi-objective, non-linear inequality constraints and a mix of integer and continuous design variables. A first approach to solve this problem is to use a **meta-heuristic optimization** algorithm that can handle all those characteristics, such as a GA. A second approach is to develop an **ad-hoc search method** inspired by combinatorial optimization techniques.

Front-loading through optimal hinge selection

The premise of the project is that front-loading may be achieved by repeating the optimization for an extensive database of different requirements. To fully answer the first sub-question (*How to obtain Pareto-optimal hinges as quickly as possible? (< 10 ms)*), the developed method should comply with the following criteria:

1. The optimal hinges must be obtained in less than 10 milliseconds. Therefore, it is unlikely that any analysis (and likewise, any search procedure) can be carried out because that would significantly increase the number of operations and, hence, the run-time of the hinge search.

2. In this front-loading PDP, the objective is to produce high-fidelity results that are directly usable. The optimal hinges need to correspond to an actual, existing hinge assembly and not to an approximation resulting from an interpolation or classification model.

The front-loading approach chosen is to select the nearest feasible point in the database, instead of trying to predict the optimal hinge through regression or classification modeling techniques. The benefits of this approach are numerous:

1. This approach could satisfy both the near *instantaneous* (no analysis is run, only a query) and the *exact* (only actual optimization results are given) requirement.

2. This approach is easy to implement, independently of the system and the optimization method.

3. Efforts can be focused on developing an optimization method, which can then be used by itself *or* included in this front-loading framework, instead of developing a method solely based on the front-loading PDP.

The main drawback of this approach is that “selecting the nearest feasible point” is not trivial and should be done carefully in order to make sure that the point is feasible and truly optimal. Moreover, the hinge optimization involves many parameters linked to the rudder design. The number of optimizations could be high, which represents a risk of increasing both the required computation time and the selection time.

1.5. Report outline

Subsection 1.4 introduced the three main elements of the methodology: a KBE model, an optimization method and a front-loading method. The three following chapters details those elements.

Chapter 2 discusses the KBE model. To achieve the desired modeling, it is necessary to study the design of the fin-rudder interface in order to derive high-level requirements on the design of the hinges, as described in section 2.1. Section 2.2 provides further details on the hinge design. Then, the cost, mass and stress analyses of the hinge are discussed in sections 2.3, 2.4 and 2.5. The different analyses, the design rules and methods described in those last four sections are aggregated in the KBE model in section 2.6. Finally, section 2.7 summarizes the key points of interest. Further technical details on the implementation of the KBE model are also given in the second part of this report, in chapter 7.

Chapter 3 discusses the optimization method. After the problem statement in section 3.1 and a short introduction on optimization techniques and on search performance criteria in subsections 3.2.1 and 3.2.2, existing methods for hinge optimization are assessed. Then, four different optimization methods are implemented. The first is a meta-heuristic algorithm (GA in section 3.4). It leads to the development of ad-hoc methods in section 3.5, with the Random Tree Search (RTS) in subsection 3.5.2 and the Best First Search (BFS) in subsection 3.5.3. The last method, the Informed Linear Search (ILS) in subsection 3.5.4 yields satisfactory results. The ILS is used in section 3.6 to study the effects of different inputs on the hinge design variables. A summary of all the developed methods and their results are discussed in section 3.7. The key features of the search implementations are also stated in the second part of this report, in chapter 8.

Chapter 4 discusses the methodology used to front-load the hinge optimization. The front-loading framework is defined in section 4.1. Then, the developed front-loading methods are presented. The tool to answer the data visualization use-case is described in section 4.2, before discussing the method to solve the rudder-fin optimization use-case in section 4.3. The discussions in section 4.4 argue about the benefits and drawbacks of this front-loading approach. The implementation of this framework is detailed in the second part of this report, in the chapter 9.

Conclusions are drawn in chapter 5. The research questions are answered, and the critical steps of the front-loading methodology are summarized. Finally, chapter 6 presents the recommendations for future work. It gives steps that could be carried out to validate (section 6.1) and extend (section 6.2) the results of the hinge front-loading methodology.

2

Hinge design, analysis and modeling

As outlined in the methodology, the hinge modeling is required to enable the optimization and front-loading. This section discusses hinge design, analysis and modeling. For each aspect, the question asked is the following:

1. **Design**: What are the requirements on the hinge that resulted in this configuration of components?
2. **Analysis**: What methods are available to carry out a validation of the design, in terms of cost, mass and stress?
3. **Modeling**: How can hinge design and analysis be simulated to enable hinge optimization and front-loading?

First, a high-level overview of the hinge design is given. The rudder-fin interface design is discussed in the section 2.1, as it dictates the design of the hinges which is introduced in section 2.2. The methods used to compute the stress of in the hinge components are discussed in section 2.3. The cost and mass analysis are discussed in sections 2.4 and 2.5. Those analyses are coupled with the design methods in the hinge model, as described in section 2.6. Existing hinge models are described in subsection 2.6.1, while the KBE application developed (the "`Hinge Generator`") in the context of this MSc thesis is presented in subsection 2.6.3. Subsection 2.6.2 argues about the strengths and weaknesses of using KBE for this purpose. Subsection 2.6.4 explains how the `Hinge Generator` can be used for hinge design. Further technical details about the model implementation are given in the second part of this report, in chapter 7.

2.1. Rudder-fin interface design

The *interface* is the system which enables the articulation of the movable part of the control surface. The design of the interface of the movable surface dictates the design of the hinges. Fokker receives two types of requests from the OEM: (1) either the rudder is fully defined and only needs to be manufactured or (2) Fokker is provided with requirements for the hinge plane location and the OML of the vertical tail. This study focuses on the second case, for which the whole design of the rudder-fin interface needs to be determined.

2.1.1. Overview of the rudder-fin interface

This subsection lays out the main elements of the rudder-fin interface.

Figure 2.1 shows a simplified top view of the rudder-fin interface.

The movement of the rudder is created by enabling rotation around the rotation hinge (on the left side in figure 2.1 while pushing or pulling the actuator hinge (on the right side in figure 2.1) with the actuators. From figure 2.1, the static equilibrium of the moments can be deduced:

$$\sum \vec{M}.\vec{z} = M_{aero\rightarrow rudder} + M_{actuator\rightarrow rudder} = 0 \qquad \Leftrightarrow \qquad F_{aero\rightarrow rudder} \cdot d_{aero} - F_{actuator\rightarrow rudder} \cdot d_{arm} = 0 \tag{2.1}$$

Equation 2.1 leads to an important design guideline for the rudder-fin interface. It is common practice to place the actuator and rotation hinges as far away from each other as possible, to make the lever arm

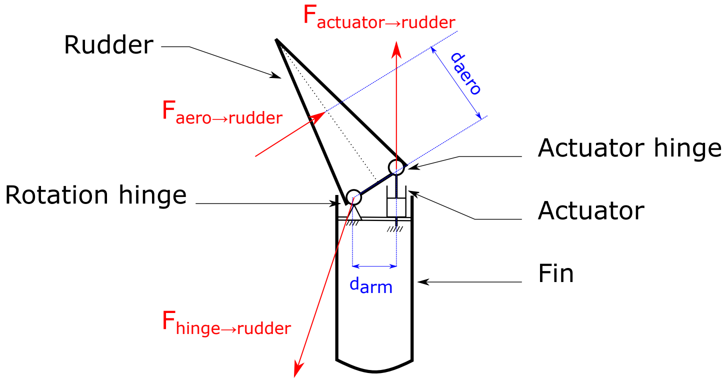


Figure 2.1: Simplified free body diagram of the fin-rudder interface (section cut, top view). The forces indicated in red are not drawn to scale. The greater d_{arm} is, the smaller $F_{actuator\rightarrow rudder}$ can be.

d_{arm} as large as possible. This positioning reduces the forces the hinges must carry. This was confirmed by Raju Kulkarni et al., as shown in [26]. Practical experience at Fokker with different types of aircraft (business jets, transport aircraft, fighters) confirms the importance of the lever arm too. The order of magnitude of the loads on the hinges is similar for both business jets and small transport aircraft, because the larger OML of the transport aircraft allowed for a larger lever arm, whereas the business jet OML is much more restrictive [38].

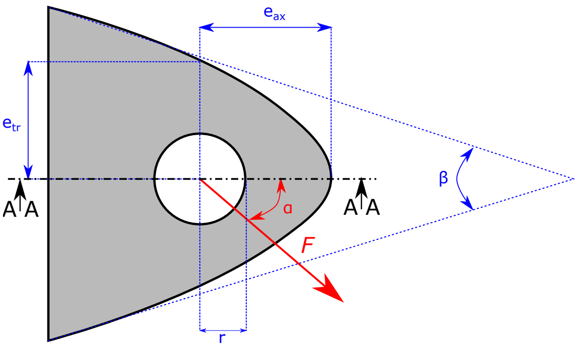


Figure 2.2: Lug lateral section cut and in-plane load definition.

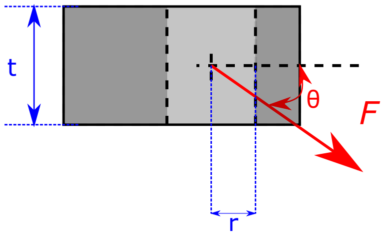


Figure 2.3: Lug top section cut and out-of-plane load definition.

The hinge positioning underlines the importance of the available space constraint during the design of the hinge. During the front-loading, it will be expressed with the following inequality constraints:

$$e_{il} \leq e_{max} \tag{2.2}$$

$$e_{ol} \leq e_{max} \tag{2.3}$$

Where e is the eccentricity of the lug, as depicted in figure 2.2. Typically, it is the inner lug inequality constraint that is limiting, because its dimensions are dictated by the dimensions of the bearing, as shown in figure 1.4. This space constraint also underlines the importance of the choice of the discrete components. If a slightly larger bolt radius is chosen, it will in turn result in a larger sliding bushing, fixed bushing and outer lug. Likewise, it will also result in a larger bearing and inner lug. Although the influence of those components by themselves could be small (e.g., the mass of the fixed bushing is small compared to the mass of the other components, $m_{fb} \ll m_{hinge}$), their effect of the dimensions of the assembly could be critical.

The rudder-fin interface is depicted more accurately in figure 2.4. On this example, the rudder is attached to two actuator hinges in red, and rotation is enabled via four rotation hinges in blue. Having several rotation and actuator hinges span-wise ensures a more uniform load repartition and limits bending, both for the hinges and for the skin panels and ribs of the rudder and fin. Although in theory, only two hinges per line would be necessary, in practice a greater number of hinges reduces the load per hinge, and therefore the size, cost, and mass of the whole interface. Designs of experiments conducted by Raju Kulkarni et al. in

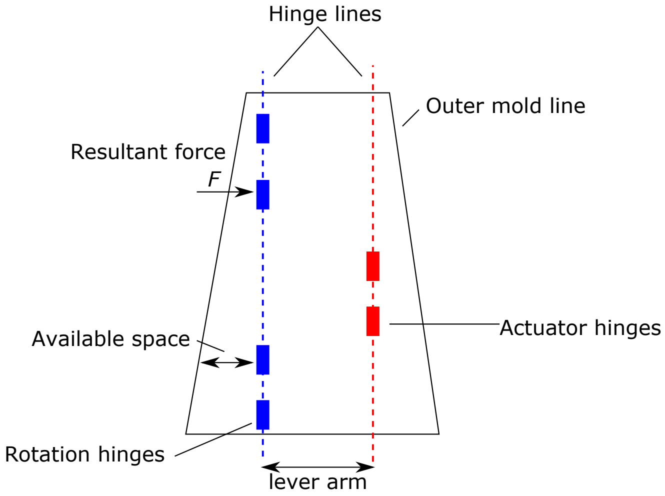


Figure 2.4: Fin-rudder interface (section cut, back view). Placing the hinge lines as far apart as possible allows for a larger lever arm, at the cost of a di-symmetry of the movable surface rotation.

[26] showed that rudder-fin interfaces with 4 hinges were almost always lighter. A four hinges configuration would, however, not meet redundancy requirements expressed in subsection 2.1.2.

One could have expected the rotation line to be located on the fin symmetry plane. However, for business jets, it is placed as close as the OML as possible, to have the largest lever arm without the hinge protruding from the OML. Likewise, the hinge lines sometimes have a small angle with respect to the fin symmetry plane, to have more space for the uppermost rotation hinge [38]. This way, the lever arm can be increased without the uppermost hinge protruding from the OML.

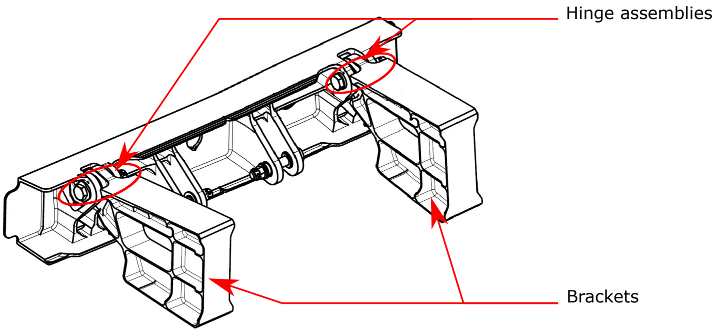


Figure 2.5: 3D drawing of the rudder-fin interface. The hinge system links the rudder and the fin through the brackets.

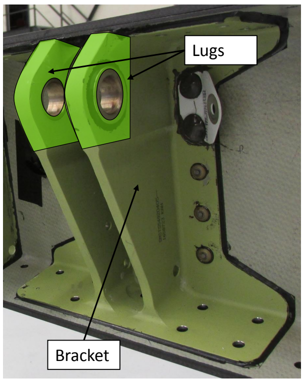


Figure 2.6: The lugs designate the underlined part of the bracket

Figure 2.5 shows a 3D drawing of the rudder-fin interface. The hinges fit within the brackets, which themselves are attached to the rudder and fin. The tip of the bracket, in which the hinge is inserted, is called the lug, as depicted in figure 2.6. Although the design of the brackets is out of the scope of this thesis, some of the variables of the bracket need to be considered as inputs for the hinge model, as they are required to analyze many components of the hinge assembly. The lug geometry will be included in the design vector (dimensions e_{tr}, e_{ax} and t, as seen in figures 2.2 and 2.3), but the lug materials (mat_{lug}) and taper angles (β) are considered to be parameters, because they depend on the bracket design.

2.1.2. Load cases

The structure of the vertical tail must sustain *external* loads during the various stages of the flight. Those loads result in *internal* loads on the hinges. Though the primary source of the internal load is the reaction to the external aerodynamic loads, other sources, namely the weight of the rudder, torsional loads and rudder inertia, have to be considered too.

In addition to the different loads, several possible failure scenarios, at different system levels (hinge, rudder, and aircraft) need to be assessed. For instance, it is required that if a rotation hinge fails, the remaining rotation hinges can still take the additional load such that the rudder-fin interface can still function as intended. The faulty hinge can then be replaced during the next inspection.

The combination of those factors implies that several scenarios, called *load cases*, need to be considered. A rudder load case consists of the following [39]:

1. What are the loads acting on the rudder (aerodynamic, inertia, torsional).
2. If a hinge has failed. The considered hinge failures are different than the failure modes described in section 2.3 (e.g., bolt bending failure), which is already accounted for in the design process. The components are designed so that failure is, in theory, impossible. However, unexpected failures could occur during the lifetime of the aircraft. For instance, a crack could appear after a shock during the assembly process or after a foreign object impact in flight.
3. If an actuator has failed. An actuator can either jam (no possible rotation or translation) or disconnect (any rotation or translation is possible).

Although it may not seem like a large number of parameters, their multiplication results in thousands of load cases for the hinges. Out of the thousands of load cases that could occur during the flight, the movable surface is designed to endure the most demanding cases. An example of failure at aircraft level is an engine failure. In that case, the rudder must be able to carry a high enough load so that the rudder moment cancels the moment from the remaining engine, to keep directional control of the aircraft.

On figure 2.7, a rudder failure case is represented.

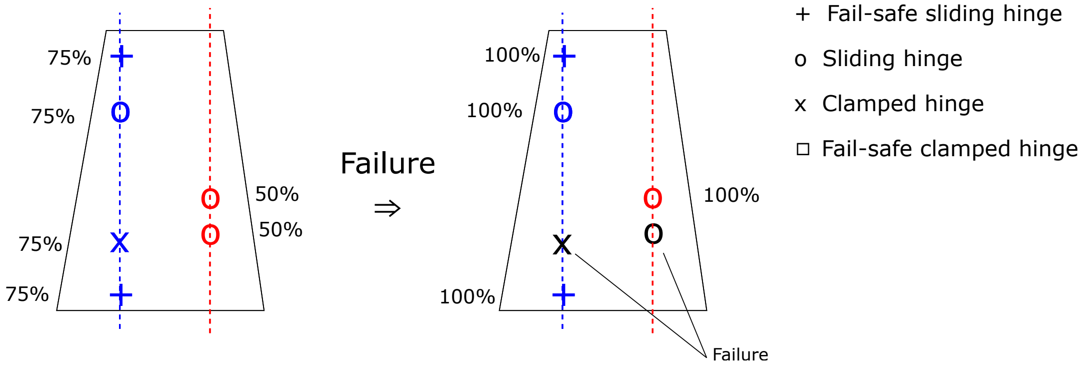


Figure 2.7: Rudder-fin interface failure case example with sample values. On the left, the nominal case where all the hinges are functional is given. The hinges are loaded at 75% and 50% of their capacity. After the failure of two hinges, the load on the remaining hinges is increased. The values are given as an order of magnitude.

The hinge load case is described by three parameters: the load magnitude F, the in-plane angle α and the out-of-plane angle θ (see figures 2.2 and 2.3). Equation 2.4 gives the range of values used for the design of the hinges of business jets at Fokker:

$$\begin{aligned} 5kN \leq F \leq 45kN \\ 0° \leq \alpha \leq 30° \\ \theta = 15° \end{aligned} \tag{2.4}$$

These values either reflect the ones currently used for the design of the hinges of business jets at Fokker (α, θ), or a slightly larger envelope of that value (i.e., $F = A \Rightarrow 0.5 \cdot A \leq F \leq 1.5 \cdot A$) to discover possibly interesting configurations. Although there should be no out-of-plane load for a sliding hinge (i.e., $\theta = 0$), a conservative value of $\theta = 15°$ is used to account for the axial component of the radial force caused by friction.

2.1.3. Requirements on the hinges

Depending on the position of the hinge in the rudder-fin interface, the requirements of the hinge are different. This leads to the two dichotomies: simple or fail-safe, and clamped or sliding.

Simple or fail-safe As with all aerospace products, the rudder-fin is required to comply with safety standards. The need for a fail-safe rudder, and therefore hinges, has been illustrated in subsection 2.1.2. The fail-safe functionality can be achieved through redundancy of the load-carrying parts. If the first element fails, the second element can still carry the loads and ensure structural integrity. Redundancy can be achieved at the hinge level, (i.e., having two simple hinges very close together), or at the hinge component level (i.e., having two components that have the same function). In general, the latter yields a lighter and cheaper hinge interface, due to minimal use of material [38].

Having fewer fail-safe hinges enables to reduce both the cost and mass of the system. Therefore, only those hinges whose failure would be catastrophic are fail-safe designs. For instance, the failure of a single rotation hinge is uncritical, provided that the remaining rotation hinges can carry the additional load without failing. From a mechanical perspective, two functional hinges are sufficient to enable the rotation of the rudder.

Most of the time at Fokker, the most inboard and outboard rotation hinges are fail-safe. If they were to fail, the rudder could be subject to high vibrations during flutter, as the span-wise rudder length that is not attached to any hinge would be substantial.

Clamped or sliding A sliding hinge allows rotation and translation around and along its axis, whereas a clamped hinge only allows rotation. The clamped hinges carry all the entire axial load (inertia and weight of rudder, in-flight maneuvers and landing) while the sliding hinges only carry the radial loads.

Only one of the rotation hinges is a clamped hinge, as having more than one clamped hinge would mean that the rudder is over constrained in the span-wise direction and could not deform freely when under load. This would result in additional loads on the hinges and rudder skin.

In general, one of the rotation hinges closest to the actuator is clamped, because the hinge in those positions are comparatively large, as the resultant forces are larger in this area. The clamped hinge is generally not fail-safe, but the rotation hinge closest to it is designed so that in case of clamped hinge failure, the sleeve would get stuck against the bearing and prevent translation, effectively becoming a clamped hinge.

Subsection 2.2.2 details the hinge designs used at Fokker to achieve those configurations.

2.2. Hinge design

High-level requirements on the hinges have been derived in the previous section:

1. One of the hinges closest to the actuator is clamped. The most common configuration at Fokker is to have the two outboard sliding hinges fail-safe, while the rest are simple sliding hinges. Automated hinge design tools could enable the study of different configurations.

2. The hinges must sustain a load case (F, α, θ) (see figures 2.2 and 2.3)

3. The hinges must fit within the available space of the rudder OML, so that $e_{il}, e_{ol} \leq e_{max}$

2.2.1. Overview of the hinge components

This subsection explains the design of a sliding hinge at Fokker. Although other designs are possible, this project is focuses on Fokker's hinge designs. A sliding hinge is an assembly of parts that enables the rotation and translation of one part (the rotor) with respect to another (the stator). A section cut of a simple sliding (as in non-fail-safe) hinge is given in figure 2.8.

In the case of the rudder-fin interface, the movable part is the rudder, which is attached to the hinge via the **central or inner lug** and the static part is the fin, which is attached via the **clevis or outer lugs**. The lugs are part of the brackets, as shown in figure 2.6. **The bolt** is the central part of the assembly, it serves as the axis of the rotation and sustains the loads from the inner and outer lugs. **The bearing** is used to allow the rotation of the rotor with respect to the stator. The lesser the friction between the two elements, the lower the force the actuator must exert to rotate the movable surface.

The sliding bushing (or sleeve) ensures that the bearing does not slide along the bolt. Moreover, it ensures that the static elements of the hinge only contact with the inner ring of the bearing. **The fixed bushing** serves as an interface between the sliding bushing and the outer lug. The fixed bushing material is chosen so that the fretting (gradual wear due to the constant friction between the bushings) is limited. A slight gap

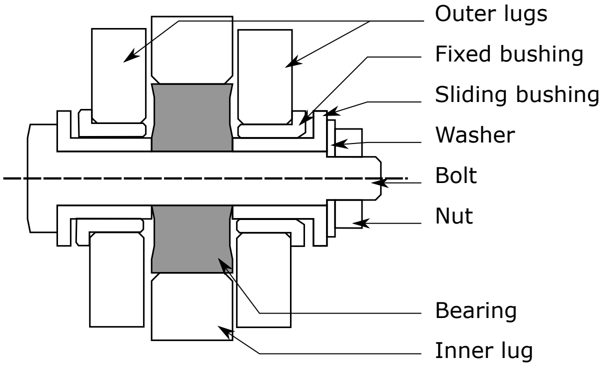


Figure 2.8: Section cut of a sliding hinge assembly. Out of the 8 components, 3 are machined (inner lug, outer lug, sliding bushing) and 5 are standard (bolt, bearing, fixed bushing, washers, nut). The main components are the bolt, bearing and lugs, the rest of the components are required to avoid fretting and to clamp the parts together.

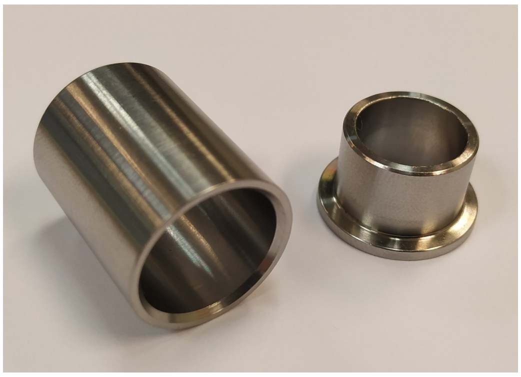


Figure 2.9: Standard bushings, without flange on the left and with flange on the right.

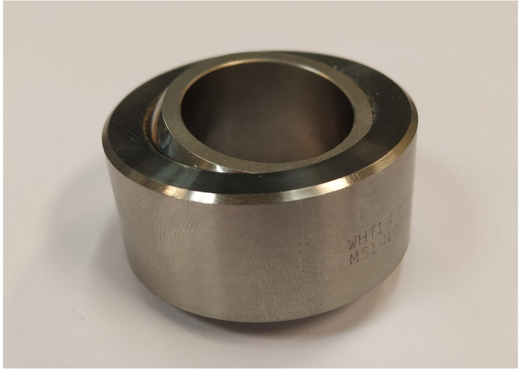


Figure 2.10: One of the standard spherical bearing used. Instead of having rolling elements, coating between the rings reduces friction.

between the sliding bushing and fixed bushing enables translation along the bolt axis[1]. Usually, a coating is applied between the bolt, the fixed bushing and the sliding bushing, to reduce corrosion and friction. **The nut** compresses the static elements around the bolt to ensure that the compound is immobile. This compression results in tension in the bolt and nut. For the bolt, it is referred as *pretension.* One or several **washers** can be inserted before the nut so that the tension caused by the torque of the nut can be exerted uniformly on the sleeve. If the nut is castellated (as in figure 3.23), washers can also be used to position the nut so that the nut slot and the bolt hole align, enabling the nut to be secured with a cotter pin.

to reduce the cost and decrease the design time, most of the hinge components are standard. Interviews conducted at Fokker determined that in most cases, the bolt, nut, washer, fixed bushing, and bearing are standard parts; while the lugs and sliding bushing are machined parts [38].

However, Fokker occasionally needs to design components because standard parts that fit the requirements are unavailable. For instance, the bolt design can be standard but is sometimes machined as standards bolts are too short to fit all the elements do not exist (see figure 2.12). Likewise, Fokker sometimes requests custom-made bearings that are not readily available in the standard parts manufacturer. The use of those specialized, custom parts can increase the total cost of the assembly by a factor of three [38].

The hinge components need to respect certain constraints so that the hinge assembly meet its functional requirements, i.e., not fail under considered load cases and enable rotation. The constraints can be divided into three types, as shown in table 2.1. Other more specific requirements and constraints exist, such as those detailed in [35], (for instance should allow disassembly for inspection) are considered out of the scope of this project.

On figure 2.14, the DSM of the sliding hinge assembly is given. The DSM matrix enables to visualize the

[1] The naming “fixed” bushing might be misleading, as the fixed bushing is sliding along the bolt axis. It is named “fixed” at Fokker, because it is fixed (or clamped) into the lugs during the manufacturing process, whereas the sliding bushing just slides onto the bolt [38].

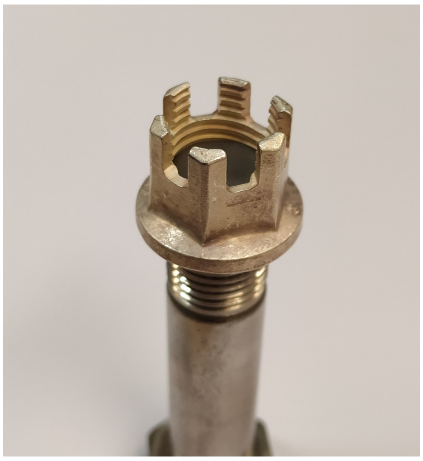


Figure 2.11: Castellated nut partially screwed onto a bolt.

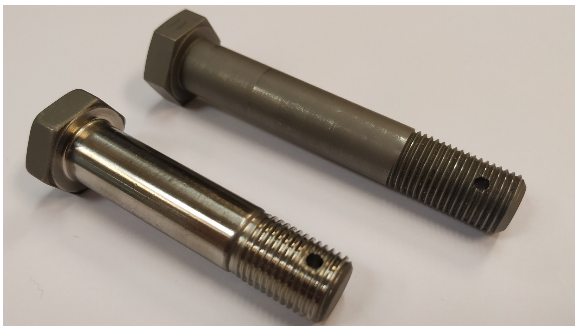


Figure 2.12: Bottom-left: standard bolt, with A286 steel. Top-right: custom machined Fokker bolt, PH13-8 Mo steel.

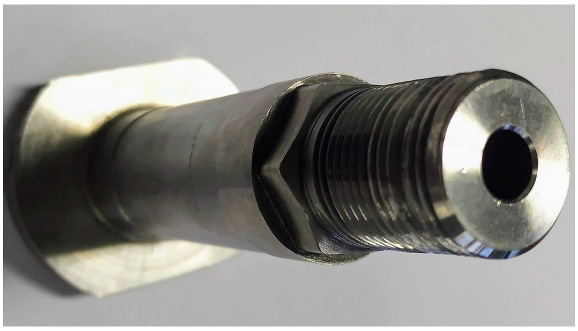


Figure 2.13: Hollowed bolt. Their use in the hinge assembly could enable mass savings but has not been implemented yet [38].

Requirement	Description	Example
Fitting	Components need to be in contact with each other	$r_{bolt} = r_{be,in}$
Fretting	Materials of components need to be compatible	if Mat_{fb} = aluminium, then $Mat_{sb} \neq$ Steel
Stress	Components must not fail under loads	$RF_{nut} > 0$

Table 2.1: Requirements on the parts of a hinge assembly.

dependencies between the various properties of each component of the hinge assembly.

The black dots represent a coupling between the geometry of two components. For example, on the first row, the *bolt outer radius* is coupled with the *bearing inner radius* and the *sliding bushing inner radius*. Looking at figure 2.8, the bolt, bearing and sliding bushing must fit together. Although most of the coupling indicated in figure 2.14 represent an equality constraint, some represent a threading that must match or an inequality constraint (i.e., the length of the fixed bushing must be lesser or equal to the length of the sliding bushing).

The blue dots represent a coupling between the materials of two components. For example, if the bolt is made of aluminium, then the sliding bushing cannot be made of bronze or titanium. With the appropriate coating, using steel or aluminium for the sliding bushing becomes feasible. This coupling is represented by the blue dot on the fourth row and fourteenth column of the matrix.

Finally, a green cell represents a coupling due to standard parts. For example, the bearing inner radius and the bearing outer radius (rows 7 and 10) have a non-linear relationship[2]. There are no bearings with a very small inner diameter and a very large outer diameter. Likewise, for a very small bolt radius (row 1), the maximum bolt shank length is smaller than the bolt shank length of a bolt with a larger radius.

[2] The correlations between the dimensions of the bearings are depicted in detail later in the report, in figure3.3

	Bolt outer radius	Bolt shank length	Bolt thread length	Bolt material	Bolt thread	Bearing material	Bearing inner radius	Bearing inner length	Bearing outer length	Bearing outer radius	Sliding bushing inner radius	Sliding bushing outer radius	Sliding bushing length	Sliding bushing material	Fixed bushing length	Fixed bushing inner radius	Fixed bushing outer radius	Fixed bushing material	Outer lug radius	Outer lug thickness	Outer lug material	Inner lug radius	Inner lug thickness	Inner lug material	Washers thickness	Nut length	Nut material	Nut thread
Bolt outer radius	1	🟩	🟩		🟩		●				●																	
Bolt shank length	🟩	2	🟩		🟩			●					●															
Bolt thread length	🟩	🟩	3		🟩																				●	●		
Bolt material				4		+								+													+	
Bolt thread	🟩	🟩	🟩		5																							●
Bearing material				+		6																		+				
Bearing inner radius	●						7	🟩	🟩	🟩	●																	
Bearing inner length		●					🟩	8	🟩	🟩																		
Bearing outer length							🟩	🟩	9	🟩													●					
Bearing outer radius							🟩	🟩	🟩	10												●						
Sliding bushing inner radius	●						●				11	🟩	🟩															
Sliding bushing outer radius											🟩	12	🟩			●												
Sliding bushing length		●									🟩	🟩	13		●										●	●		
Sliding bushing material				+										14				+										
Fixed bushing length													●		15	🟩	🟩			●								
Fixed bushing inner radius												●			🟩	16	🟩											
Fixed bushing outer radius															🟩	🟩	17		●									
Fixed bushing material														+				18			+							
Outer lug radius																	●		19									
Outer lug thickness															●					20								
Outer lug material																		+			21							
Inner lug radius										●												22						
Inner lug thickness									●														23					
Inner lug material						+																		24				
Washers thickness			●										●												25	●		
Nut length			●										●												●	26		🟩
Nut material				+																							27	
Nut thread					●																					🟩		28

Legend
● Geometry coupling (must fit)
+ Material coupling (must not fret)
🟩 Standard parts coupling

Figure 2.14: DSM of the sliding hinge assembly. On the matrix diagonal, the index of the variable is given. The order of the parameter is arbitrary and does not reflect the implementation, the matrix is only used to represent the coupling between the parts.

2.2.2. Fail-safe and clamped configurations

In the previous subsection, a simple (non-fail-safe) sliding hinge was shown. For the other three types of hinges, the design is slightly different. The configuration of the simple fail-safe hinge is shown again as a reference in figure 2.15.

On figure 2.16, a simple clamped hinge is represented. The translation along the bolt axis is blocked by clamping the outer lugs. For manufacturing purposes, only one (the left one in this case) of the two outer lugs of the bracket is actually fixed with respect to the hinge. As only one of the outer lugs is clamped, it means only one of the outer lugs will carry the axial load.

On figure 2.18, a fail-safe, clamped hinge is represented. The translation is blocked in the same manner as the simple clamped hinge represented in figure 2.16. As explained in the previous section, a higher reliability follows from introducing redundant components. The hinge of figure 2.18 has a fail-safe feature on three components, indicated in green: the bolt, the inner lug and the outer lug. For the lugs (which are then called "multi-element" lugs), their thickness is divided into two separate plates. For the bolt, a fail-safe sleeve covers the whole length of the bolt and can carry the shear load of the lugs in case of bolt failure. This multi-element approach is used in the prevention of cracks due to fatigue. Simply sizing a single element to have a very high reserve factor (RF > 5) is insufficient. A crack could remain unnoticed during an inspection and would cause failure in flight, whereas with fail-safe components, the broken component can be seen during an on-ground inspection and can be replaced straight away.

Similarly, a fail-safe sliding hinge is represented in figure 2.17. Its functioning is the same as the non-fail

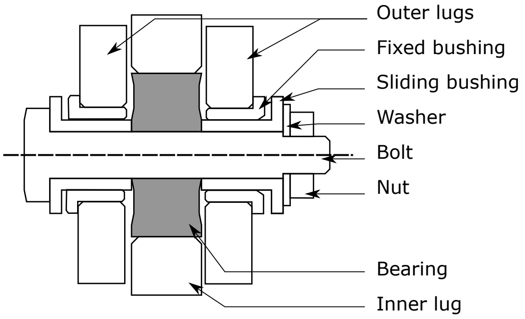


Figure 2.15: With a sliding hinge, the fixed bushing can translate over the sliding bushing.

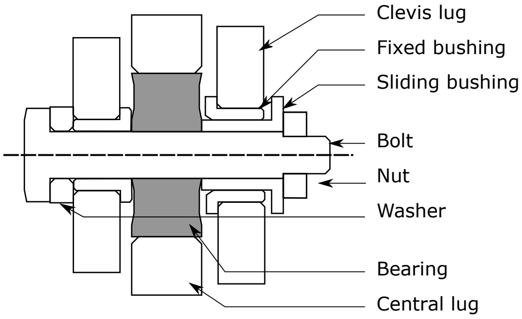


Figure 2.16: With a clamped hinge, the fixed bushing of the left outer lug, and therefore the whole bracket, cannot translate.

safe, sliding hinge depicted in figure 2.15, and the fail-safe mechanism is identical to the fail-safe clamped hinge represented in figure 2.18. In practice, fail-safe clamped hinges are seldomly used. Instead, a sliding hinge is designed so that in case of clamped hinge failure, the sleeve would get stuck against bearing and prevent rotation, effectively becoming a clamped hinge. This case is the one depicted in figure 2.7.

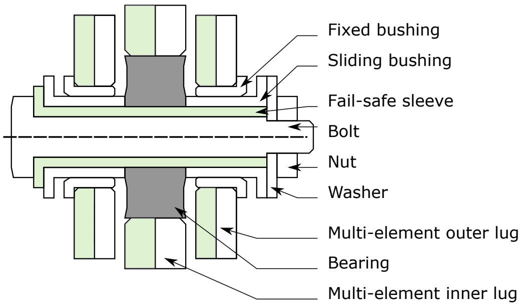


Figure 2.17: This fail-safe sliding hinge configuration contains three fail-safe elements, indicated in green: multi-elements inner and outer lugs, and a fail-safe sleeve.

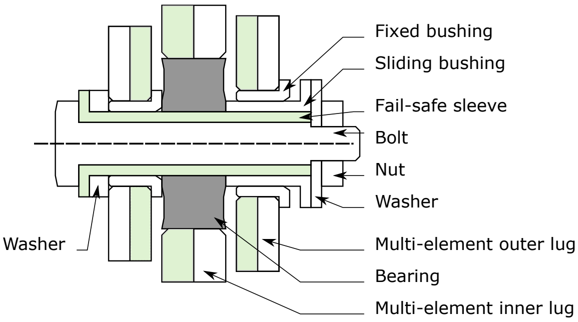


Figure 2.18: This fail-safe clamped hinge configuration contains the same fail-safe elements as the one in figure 2.17, but again with the left outer lug clamped.

2.2.3. Materials and standard part choices

The material properties of the selected material depend on its chemical composition, its heat treatment, and its manufacturing process. For the brackets (and therefore for the lugs), the component can be manufactured entirely through machining, or through forging completed by machining in a second phase [17].

Typically, the materials of the hinge components are chosen for their strength, density and cost density. For the fixed bushing, however, the material choice is a matter of its life-span, not of strength. Therefore, the sliding bushing is commonly made of bronze[3] [38]. The detailed rules for the fretting between the materials are given in tables (available within Fokker). The tables indicate the compatible materials including the consideration of the coating applied between the two components.

2.3. Stress analyses of the hinge components

The stress analysis is the most time-consuming analysis of hinge assembly. This section briefly describes the methods used at Fokker to compute the allowable forces of the hinge assembly. The symmetrical geometry of the hinge components makes it possible to have analytical methods, either based on experiments or on theory. This results in a simple, analytical stress analysis for each component. The most important stress analyses are the bolt stress analysis (subsection 2.3.2) and the lug stress analysis (subsection 2.3.3). The stress analysis of the rest of the components are detailed in subsection 2.3.4. For confidentiality reasons, this section only describes the high-level guidelines of the methodology.

[3] This choice is also studied in section 3.6.

2.3.1. Methodology at Fokker

The hinge is primarily subject to static loads resulting from the aerodynamic forces on the rudder. Therefore, only a static stress analysis is carried out for the sizing of the components. The stress at which failure occurs is computed at various critical sections of each component. Then, a fixed safety criterion is used to ensure that failure will not occur within those levels of stress.

$$RF = \frac{F_{applied}}{F_{allowable}} \qquad MS = RF - 1 \tag{2.5}$$

In this project the factor used is the Reserve Factor (RF), which is set to 1.0. Alternatively, the Margin of Safety (MS) could be used.

Although the hinge is subject to vibrations, no fatigue analysis is carried out during the sizing process. Based on experience at Fokker, fatigue analysis is not critical and can be done only after the sizing of the components as a safety check. If necessary, fail-safe designs such as those presented in subsection 2.2.2 are adopted.

2.3.2. Bolt

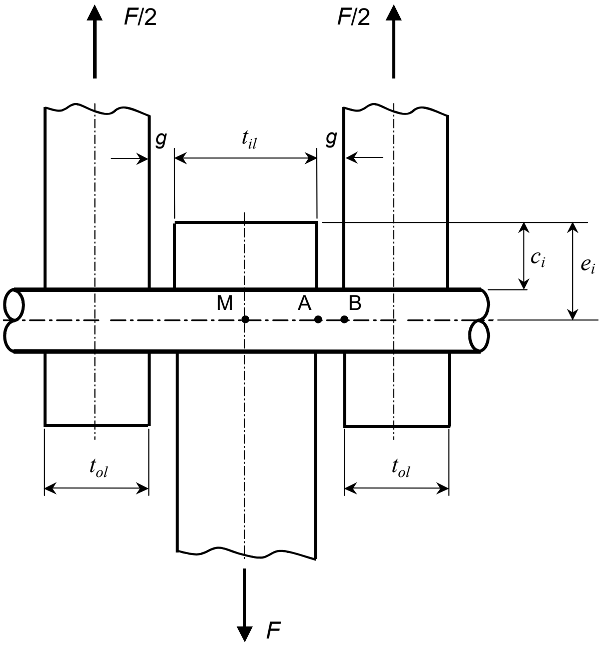


Figure 2.19: The bolt is subject to the forces F through the inner lug and $F/2$ through the outer lugs. The allowable forces are computed at the critical cross-sections in A, B and M.

The bolt stress analysis is not straightforward, because the bolt is subject to a combination of *shear*, *bending* and *tension*. The *tension* is also referred as "*bolt pretension*", as it is the tension required to clamp the elements along the bolt. The tension is effective once it is installed, before ("pre") the internal forces resulting from the aerodynamic forces on the rudder. The bolt pretension $F_{tension}$ has a maximum (if above, it would cause a failure - see failure modes below) and a minimum (if under, the components on the bolt would move along the bolt axis). The bolt pretension is computed as a function of the bolt material strength and nut material strength [8].

A simplified hinge assembly, with a bolt and two lugs, is depicted in figure 2.19. The radial load on the hinge is noted F. As the load on the male lug is F, and the outer lugs are supposed to be equidistant with respect to M, static equilibrium yields that the loads in the female lugs are $F/2$.

The *shear* is a result of the forces $F/2$ of the outer lug opposed to the force F in the inner lug. The forces F and $F/2$ also result in *bending*. As the nut must exert a clamping force on the bearing, washer(s) and sliding bushings, the bolt must also sustain a *tension* of a force equal to the clamping. The torsional stresses due to the torqueing of the nut are assumed negligible, and the shear stress distribution of the bolt is considered uniform. The technical notice [13] describes the analytical method used at Fokker to determine the allowable ultimate load (for failure) and the allowable yield load (for deformation) of the bolt. The bolt has different failure types, associated to bending, tension and shear:

- Double or single yield shear failure
- Bending failure
- Tension failures: on the thread (nut side or bolt side), or head fracture

The failures could either be static or due to fatigue.

The allowable loads are determined at the points A, B and M represented in figure 2.19. Depending on the lug thicknesses t_{il} and t_{ol}, $M_A > M_B$ or $M_A < M_B$. Therefore, the cross-sections in A and B must be designed for the combination of bending, shear and tension. At the point M, the bending is maximum and the shear is null. Hence, the cross-section in M must be designed for the combination of bending and tension.

A particularity of the bolt stress analysis is the so-called *load peaking* of the bolt. Load peaking describes that the bolt will slightly bend, changing the effective lever arm with which the bending moments is computed, as depicted in figures 2.20, 2.21 and 2.22.

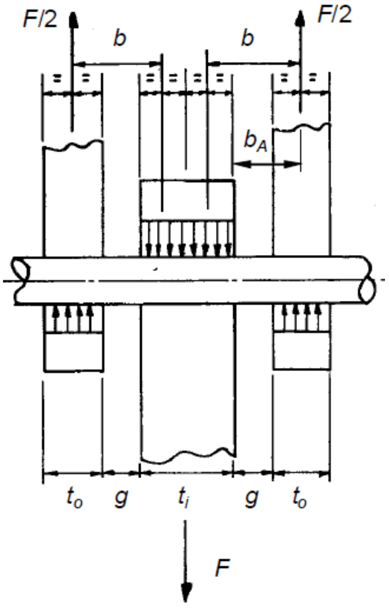


Figure 2.20: Shear stress distribution without load peaking.

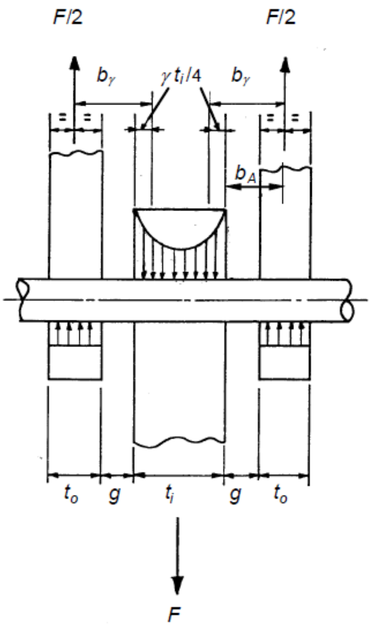


Figure 2.21: Shear stress distribution with load peaking.

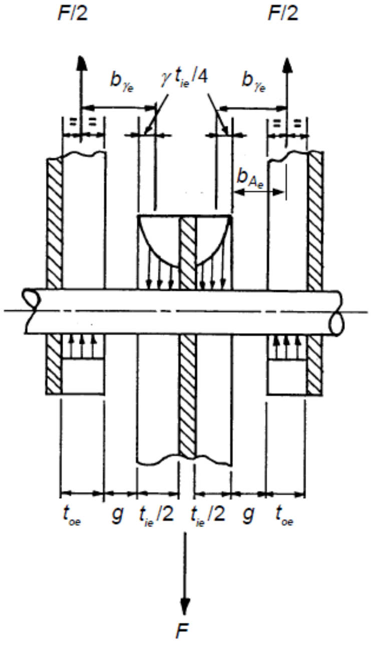


Figure 2.22: Shear stress distribution with load peaking and excess strength of lug.

On figure 2.20, there is no load peaking. The shear stress is assumed uniform and the bending moments M_A and M_B can directly be computed with the lever arms b_A and b depicted on the figure 2.20. On figure 2.21, due to the bolt deflection under transverse loading, the stresses near the shear faces of the inner lug will peak and so reduce effective the bending moment arm:

$$M = F/2 \cdot b_\gamma \qquad b_\gamma = \frac{t_{il}}{2} + g + \gamma \cdot \frac{t_{ol}}{4} \tag{2.6}$$

Where the load peaking factor γ is a function of:

$$\gamma = f(e_{in}, r_{bolt}, F_{u,in}, R_{u,in}) \qquad 0 \leq \gamma \leq 1 \tag{2.7}$$

With:

- e_{in} is the inner lug maximum eccentricity (distance from bolt center to lug end) in $[mm]$
- r_{bolt} is the bolt outer radius in $[mm]$

- $F_{u,in}$ is the allowable ultimate load of the inner lug in $[N]$
- $R_{u,in}$ is the ultimate tensile strength of the inner lug in $[MPa]$

On figure 2.22, load peaking is active again. In addition, it is assumed that the lugs are in so-called *excess strength*, and that the portion of the lugs in shaded area are inactive. The lug thicknesses t_{ol} and t_{il} are replaced by smaller values $t_{ol,eff}$ and $t_{il,eff}$, further reducing the bending moment on the bolt. This scenario happens with thick lugs, for which $t_{ol}, t_{il} \gg d_{bolt}$.

Once the allowable tension load, allowable bending moment and allowable shear load are computed, the allowable combined loading is determined from interaction equations similar to those given in [22].

2.3.3. Lugs

The lugs are the only components of the hinge assembly that are not axis-symmetrical. The technical notice [14] describes the main analytical methodology used at Fokker for the design of the lugs. The methods described in [10–12] are also required to compute parameters required by the main methodology. The load of magnitude F is decomposed in an in-plane load (in-plane angle α represented in figure 2.2) and out-of-plane load (out-of-plane angle θ represented in figure 2.3).

The methodology requires the computation of several allowable loads. First, the allowable axial load (P_{ax}) is determined. The allowable axial load is the in-plane load for which $\alpha = 0$. It is the minimum of several loads:

$$P_{ax} = \min(P_1, P_2, P_3) \tag{2.8}$$

Where P_1 is the load at tension failure in the net section, P_2 is the load at bearing failure and P_3 is the load at shear tear out failure. In general, $P_1 > P_3 > P_2$.

Various reduction coefficients η_i are computed, where $P_i = \eta_i \cdot A \cdot \sigma$. With P being the allowable load, A being a cross-section and σ the material strength. The coefficients η_i are either obtained from experimental data or from analytical formulas.

Then, the transverse load is determined. The allowable transverse load (P_{tr}) is the in-plane load for which $\alpha = 90°$. Like for the allowable axial load, it is the minimum of three allowable loads, $P_4, P_5, P6$:

$$P_{tr} = \min(P_4, P_5, P_6) \tag{2.9}$$

To compute those allowable loads, the cross-sections F_1, F_2, F_3, F_4 depicted in figure 2.23 are combined in a fictitious cross-section.

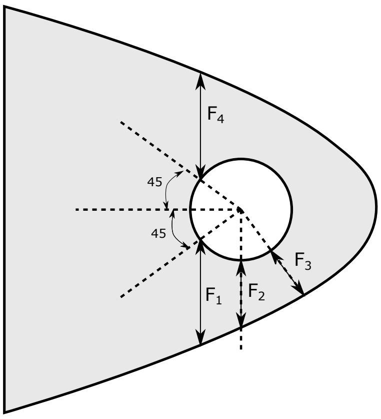


Figure 2.23: cross-sections taken into account for the transverse allowable load computation. F_3 is the smallest cross-section area. The cross-sections are combined into a fictious cross-section F_f, which is required to compute P_4, P_5, P_6.

Once the allowable axial and transverse loads are known, the allowable oblique load (load for which $0 \leq \alpha \leq 90°$) can be computed from load combination formula.

The out-of-plane allowable load (P_z) can then be computed with a similar method. Finally, the out-of-plane oblique loading can be computed with as a function of $(\theta, \alpha, P_{ax}, P_{tr}, P_z)$.

2.3.4. Other components

Sliding bushing The sliding bushing is assumed to cover precisely the length of the bolt shank. Hence, it follows that:

$$l_{sb} = \frac{l_{shank}}{2} - \frac{l_{be,in}}{2} \tag{2.10}$$

$$r_{sb,in} = r_{bolt,out} \tag{2.11}$$

The only remaining parameter to size is the sliding bushing thickness t_{sb}:

$$t_{sb} = r_{sb,out} - r_{sb,in} \tag{2.12}$$

The sliding bushing is subject to a combination of axial compression (nut clamping force) and radial compression (lug shear force). The thickness of the sliding bushing is sized so that $RF_{sb} \geq 1.0$. In practice, the values of the reserve factors are always satisfactory, as allowable compressive loads are high. Due to manufacturing constraints and life-span expectations, a minimum value for the sliding bushing is set to 0.5mm.

Fixed bushing The fixed bushing is clamped into the outer lug hole, and must fit on the sliding bushing:

$$r_{fb,out} = r_{lug,out} \tag{2.13}$$

$$r_{fb,in} = r_{sb,out} \tag{2.14}$$

$$l_{fb} = t_{ol} \tag{2.15}$$

As the hinge is assumed sliding, the fixed bushing is only subject to radial compression. As with the sliding bushing, the high allowable compressive strength resulted in high reserve factors. In this project, the fixed bushing are assumed to be standard.

Nut For the nut, the reserve factor is given by the ratio between the computed bolt pretension and the nut allowable load, given by the nut manufacturer.

2.4. Cost analysis

Designing with the mass as a single objective could result in a light but expensive design. Therefore, the cost should be included in the hinge design and optimization. As most parts of the assembly are standard, the cost analysis is often reduced to obtaining the cost of that part from the manufacturer. Unfortunately, not all the cost data are available. Some cost data is either unavailable or difficult to obtain. The data requires continuous updating and typically depend on the procurement process, e.g., the number of parts ordered and the current supply and demand of the material used.

A cost analysis method must be used for the standard parts for which reliable cost data are missing and for machined parts (lugs, sliding sleeve, bolt). For this purpose, the method described by van der Laan in [36] is used. The total cost of the part is the sum of the material cost $c_{material}$ and manufacturing cost $c_{manufacturing}$:

$$c_{total} = c_{material} + c_{manufacturing} \tag{2.16}$$

The manufacturing cost is described by equation 2.17:

$$c_{manufacturing} = \text{rate} \cdot t_{manufacturing} = \text{rate} \cdot (t_{delay} + \tau_0 \sqrt{(\frac{x}{v_o \tau_0} + 1)^2 - 1}) \tag{2.17}$$

Where:

- $t_{manufacturing}$ is the time required to manufacture the part.
- rate is the manufacturing rate (for instance in euros per hour)
- x is dimension on which the cost estimation is based, for instance an area or length

- t_{delay} is the delay time required to setup the manufacturing process
- v_0 is the steady state speed of the manufacturing operation
- τ_0 is the time required to reach 63% of the steady state speed

And the material cost is:

$$c_{material} = c_{mat} \cdot x \cdot (1 + \mathrm{sr}) \tag{2.18}$$

Where:

- c_{mat} is the raw cost of the material
- sr is the scrap rate of the part, i.e., the proportion of material that is subtracted between the raw material and the final product

This method is applicable to many parts and manufacturing processes, is simple to implement, and gives accurate results. However, it requires the parameters (t_{delay}, v_0, τ_0) of each part to be known, which can be time-consuming.

2.5. Mass analysis

The mass is often the primary objective to minimize in aircraft components. As with the cost analysis, the fact that some of the parts are standards simplifies the mass analysis, and the task is reduced to looking up the mass in the data sheet. Otherwise, to estimate the mass of a machined part, its volume is simply multiplied by its density:

$$m_{component} = \rho_{material} \cdot v_{component} \tag{2.19}$$

Where $\rho_{material}$ is the material density in $[kg/m^3]$ and $v_{component}$ is the volume of the component in $[m^3]$. The equation above does not take into account any difference between the raw material mass and the *installed* mass. The latter could differ, by including for example coating, lubricant, etc. This difference is assumed to be negligible and the mass of the components is computed with equation 2.19.

2.6. Hinge modeling

The previous sections discussed the behavior of the hinge assembly with respect to three disciplines: cost, mass and stress. The model is the implementation of those analyses, including the detailed design rules for fitting and fretting.

2.6.1. Previous tools and methods for hinge modeling

Method at Fokker The analysis of a hinge at Fokker is concurrent, manual and distributed among teams with different expertise (hinge stress engineers and hinge designers). Engineers work with specific tools for each component analysis. The operation of the tools, aggregation of the data of the hinge components, verification of fitting and fretting and reporting of the characteristics were all manual endeavors. Since 2013, Fokker took consequent steps towards the automation of this process, in the context of the Rudder In A Month (RIAM) project. This was done by using a single design methodology and having interoperable data and tools. At first, this was achieved by using Catia and Excel VBA tools, as described in [33].

Hinge model in HDOT In 2015, the hinge modeling time was again drastically reduced, with the implementation of a hinge KBE application in HDOT. This change was enabled by the adoption of the KBE platform ParaPy[4]. A hinge assembly could now be fully analyzed automatically, without any user intervention.

2.6.2. Use of KBE for the hinge modeling

Many arguments in favor of using KBE for the hinge modeling were already given in subsection 1.4. This subsection gives more details about the choice of using KBE for the hinge modeling.

[4] Discussed in greater detailed in chapter 7

Geometry The geometry is used for an accurate cost and mass analysis which requires the volume computation. The use of the geometry visualization and of the Graphical User Interface (GUI) is detailed in 2.6.4.

Class-based OOP The KBE language used in ParaPy is implemented in python and makes use of class-based OOP. OOP is advantageous from several points of view. First, data encapsulation makes the code more descriptive. The code is easier to develop, to maintain and to read than with functional programming. Second, the creation of classes is advantageous for code re-usability and modularity. One can define abstract classes, then derive behavior and data from those classes. Examples of abstract classes for the KBE model are given in the Unified Modeling Language (UML) diagrams in chapter 7. For instance, the `HingeComponent` class describes the general behavior and data of a hinge component. Therefore, all the hinge parts (bolt, lugs, bushings, etc.) inherit from `HingeComponent`.

Declarative programming approach KBE enables to create the model with a declarative programming approach, i.e., declare *what* the system is, never *how* the simulation workflow looks like. It makes the development easier and the code implementation more concise. For instance, in the Hinge Generator, the sequence in which the components are analyzed is never explicitly given. The attributes and analysis are evaluated automatically when higher-level attributes (such as the total cost or mass of the hinge) are requested.

Lazy evaluation The declarative approach is especially powerful if it is combined with lazy evaluation. Lazy evaluation is made possible by two characteristics of a KBE system: caching and dependency tracking. According to La Rocca [30], "***Caching*** *refers to the ability of the KBE system to memorize, at runtime, the results of computed values so that they can be reused when required without recomputing*". Then, the **dependency tracking** mechanism is used to keep track of the validity of the cached values.

Lazy evaluation is useful anytime the model has to be analyzed, whether it is in an optimization routine or for a manual design. If a hinge attribute (e.g., bolt radius) is changed, the same hinge instance is kept during the whole search process. This way, if an attribute of the hinge is independent of the bolt radius, the cached values are used instead of re-computing the attribute. For example, during the search methods discussed in chapter 3, if one evaluates different bearings for the same bolt, then the bolt attributes will not be re-computed. This is particularly useful for:

- The bolt volume computation. As the geometry is detailed for more precision, its evaluation is time-consuming.
- The lug stress analysis. For many component combinations, the same fixed bushing is used. In this case, the bolt and bearing can be changed without having to recompute the reserve factor of the outer lugs.

Using KBE for the hinge model also has drawbacks. The main drawback is that it requires a proprietary KBE platform. Once the platform is used for the development, it becomes convenient to use it for all the parts of the application development. Although the use of KBE enables a full operation from the GUI, it also makes the whole application dependent on the KBE platform. Simple parts of the applications that could have been written in pure OOP would become obsolete with changes in the KBE platform. Furthermore, the run-time (especially overhead time) of KBE application could be higher than that of normal OOP, as the evaluation of any attribute triggers a procedure to enable lazy evaluation.

2.6.3. Developed KBE application: the Hinge Generator

This subsection introduces the requirements for a new hinge KBE application, to substitute for HDOT's hinge sizing model Two main requirements were expressed by Fokker during this project. The first requirement was that the KBE application must use actual standard parts and materials (the database in HDOT is incomplete for confidentiality reasons). The second requirement was that the KBE application should be modular and extensible to fail-safe and clamped hinges. A new KBE application, the `Hinge Generator`, is developed to satisfy those requirements.

Contents of the hinge generator The analyses discussed in the previous subsection must all be included into a framework to enable their inter-operation. On figure 2.24, a simplified representation of the Hinge Generator framework is given. In the Hinge Generator, the hinge components contain all their own analyses and geometry. The components are children of the main object, *hinge assembly*. The *hinge assembly* object also contains properties that require inputs from several components, such as the bolt stress analysis, which requires data from the lugs, assembly and bolt.

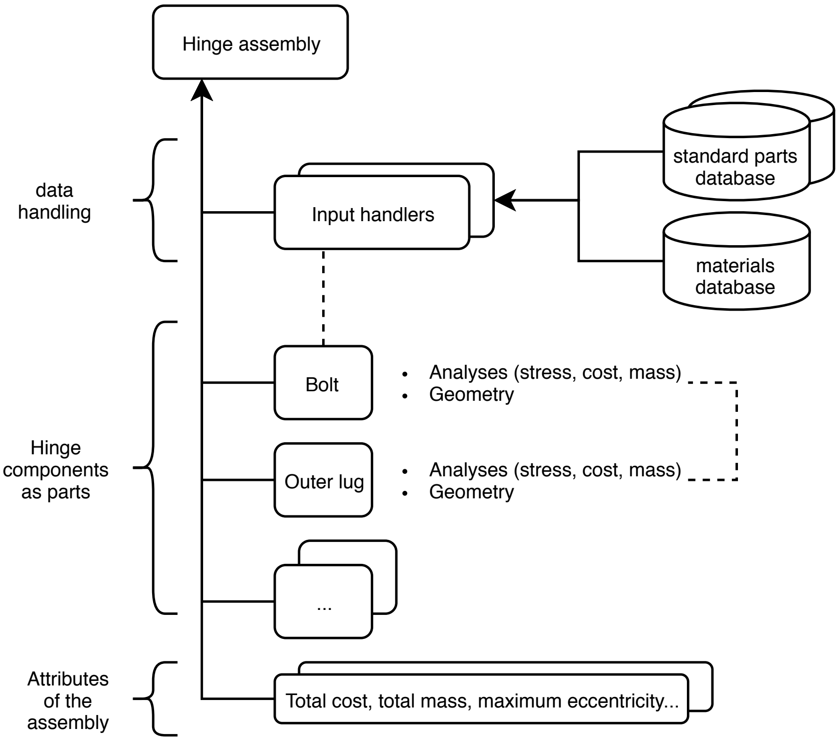

Hinge assembly
data handling
Input handlers
standard parts database
materials database
Hinge components as parts
Bolt
• Analyses (stress, cost, mass)
• Geometry
Outer lug
• Analyses (stress, cost, mass)
• Geometry
...
Attributes of the assembly
Total cost, total mass, maximum eccentricity...

Figure 2.24: The hinge assembly contains components. The components contain their analyses and geometry. The assembly reads and handles the databases before feeding them to the component objects (indicated by the dashed link between input handler and bolt). The coupling is defined in the object inputs (e.g., `bearing.radius_in = bolt.radius_out`).

The KBE application solves the problems encountered with the legacy method of subsection 2.6.1. The inter-operation of the tools is streamlined and automatic. The detailed design rules are integrated within the KBE application. Therefore, manual intervention is unecessary.

Moreover, this particular structure enables to leverage the strengths of KBE that were presented in the previous subsection. The KBE code is concise, and the workflow does not need to be defined explicitly.

Differences with the hinge sizing module of HDOT The quantitative improvements compared to the hinge KBE model in HDOT are the following:

1. Databases of standard components and materials were developed and implemented. The parts and materials used in the model are accuratly modeled and can be directly ordered.

2. The detailed lug geometry is used. Four design variables (e_{tr}, e_{ax}, t, β) are used, whereas in HDOT it is assumed that $e_{ax} = e_{tr}$ and that $\beta = 0$, therefore the only two remaining parameters are e, t.

3. The flanges, chamfers and other details of the hinge components geometry are more detailed (e.g., length of the thread of the bolt, bearing geometry flange of the bushings).

4. The lug stress analysis was improved in terms of fidelity and run-time. For a list of the differences regarding the lug stress analysis, the reader is referred to the table 7.1 in the second part of the report.

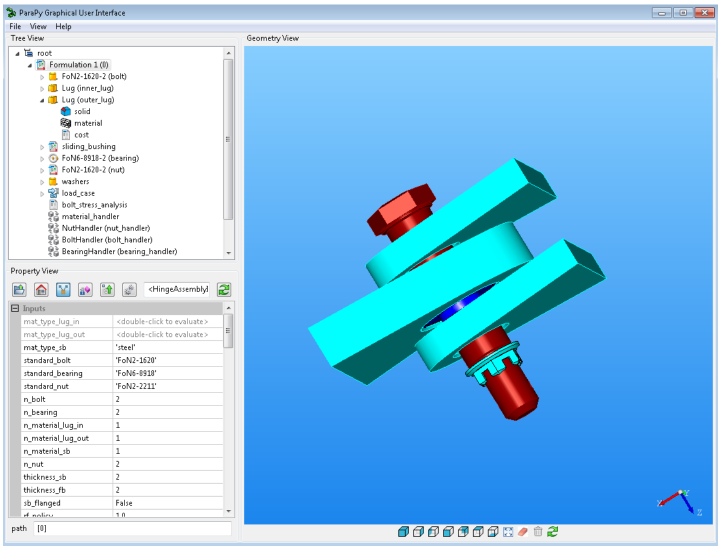
5. The total analysis run-time is of 0.5 seconds at most. It is an improvement over the run-time of HDOT which was impaired by the reading and writing of the stress analysis in Excel files.

6. The bolt preload computation has been updated.

The qualitative improvements compared to the hinge KBE model in HDOT are the following:

1. The databases and their handling were developed to be scalable (possibility to quickly include more standards parts) and easy to use (programming skills not required to add a standard).
2. The adoption of Python for the stress analysis enabled several features, such as a fail-safe mode, logging messages, bounds and assumptions, all within the stress methods themselves.
3. OOP was used to increase code modularity and extensibility compared to HDOT. The abstract class principle evoked previously has been used for the component geometry, stress analysis and search methods. Those classes can be used to easily extend the model and search to other types of hinges, such as those described in subsection 2.2.2.

2.6.4. Using the Hinge Generator for hinge design



mat_type_lug_in	<double-click to evaluate>
mat_type_lug_out	<double-click to evaluate>
mat_type_sb	'steel'
standard_bolt	'FoN2-1620'
standard_bearing	'FoN6-8918'
standard_nut	'FoN2-2211'
n_bolt	2
n_bearing	2
n_material_lug_in	1
n_material_lug_out	1
n_material_sb	1
n_nut	2
thickness_sb	2
thickness_fb	2
sb_flanged	False
rf_policy	1.0

path [0]

Figure 2.25: Visualization of a hinge assembly in the `Hinge Generator` KBE application. On the right, a visualization of the geometry of the different hinge components is displayed. On the left, the product tree (top) and attribute table (bottom) enables the user to view outputs and modify inputs of the hinge assembly.

The KBE platform includes a geometry modeling and visualization which can be used as a tool for design. The GUI of the KBE platform ParaPy can be used to inspect and tweak the model, as shown in figure 2.25.

Although the geometry is primarily used for visual inspection, it can also be used to quickly grasp some attributes of the model. On figure 2.25, the colors of the components correspond to their material type (aluminum, steel, bronze or titanium). It can also be used to display the value of the minimal reserve factor of the component, to quickly see if the stress in the components are acceptable or not. A component colored in blue indicates that it is over-sized ($RF > 2$), one colored in green indicates an acceptable size ($1 \le RF \le 2$) and one colored in red indicates that it is undersized ($RF < 1$).

The difficulty in the manual design of the hinge resides in the presence of standard parts. Although the KBE application can automate the query of the parts, the part selection is not trivial:

1. Only one integer variable is used to lookup the standard part, which will result in many different characteristics depending on the query definition. For example, looking up a bearing through an integer variable will vary all the properties of the bearing, whereas perhaps the designer intended to only change the type (cylindrical or rolling) of the bearing.
2. Changing one component will often require the other components to be changed too.

The KBE application can be used for three different use-cases, each offering a different way to query the standard parts:

Manual selection of the parts With this use-case, the user must select all the standard code to use, as well as the diameter code and length code of each part. This results in a total of 10 input variables to set. This high level of granularity offers a lot of freedom for the user to individually and precisely choose the components, but it can result in unfeasible combinations.

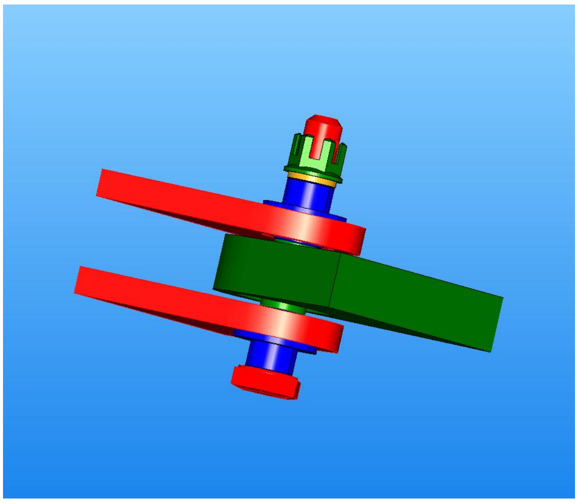


Figure 2.26: For the selected load case, the initial hinge assembly is not feasible. The bolt and outer lugs have a reserve factor smaller than one.

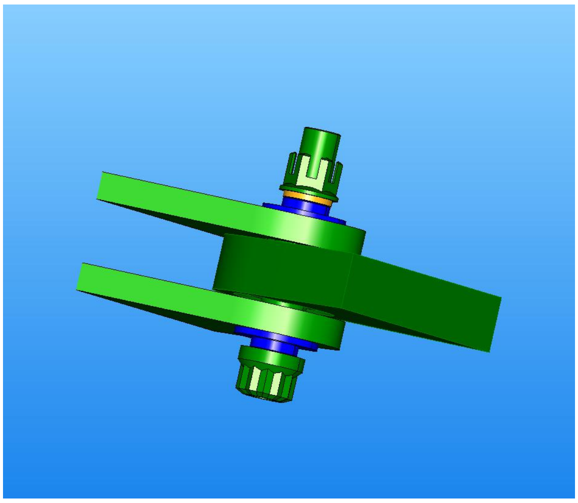


Figure 2.27: After some manipulations with the KBE application, the hinge assembly is now feasible. A slightly larger bolt diameter, together with a stronger bolt material and larger lug eccentricities increased the allowable loads.

Combination selection through integer lookup The Hinge Generator automates the selection of the standard parts. The design space is automatically pre-processed to create a set of bolt-bearing-fixed bushing-nut combinations that are feasible. This set of combinations is partially depicted on table 2.2. The columns *Bolt* and *Bearing* are the standard designations. D_{bolt}, D_{bear} and D_{fb} are indexes used to lookup the diameter codes of the bolt, bearing and fixed bushing. This table can be sorted through different criteria, and the resulting combinations will be explored in a different order.

On the figures 2.26 and 2.27, an example use is explained. On figure 2.26, the selected bolt and outer lug does not sustain the selected load case of $(F,\alpha,\theta) = (15kN,(0,30°),15°)$. The user selects a different combination using a greater integer n_{comb}. This way, a fitting combination of bolt, bearing, fixed bushing and nut is selected. This new combination has a slightly larger bolt diameter and a stronger bolt material, which can sustain the selected load case. Modifying the thickness of the outer lugs results in a higher bending moment on the bolt. Instead, the transverse and axial eccentricities of the outer lugs are increased by trial and error. This way, the outer lugs reserve factors are satisfied too, as seen in figure 2.27.

Bolt	Bearing	D_{bolt}	D_{be}	D_{fb}	r_{bolt}	m_{be}
FoN2-1623	FoN6-8918	1	1	1	3.1585	14.1
FoN2-1620	FoN6-8918	1	1	1	3.1625	14.1
FoN2-1620	MS28913	1	0	1	3.1625	27.216
FoN2-1630	FoN6-8918	0	1	1	3.1625	14.1
...	...	...	...	...	...	...

Table 2.2: Combination table generated by the `Hinge Generator`. Each row contains characteristics of the parts (bolt, bearing, fixed bushing, nut) so that they fit together.

Figure 2.28 shows the different bolt, bearing, fixed bushing and nut combinations for three different integers.

Combination selection through rounding This last application is similar to the previous one but offers yet another way to select the standard component. Instead of selecting the combination with an integer n_{comb}, the selection is done with a continuous variable from the table 2.2. The continuous variable is compared to the existing combinations, and the combination with the closest upper bound of that variable is chosen. For example, if the user queries the combination with $r_{bolt} = 3.16$, the second row will be used. However, the upper bound of one criterion might not be unique (in table 2.2, there are three rows with the upper bound $r_{bolt} = 3.1625$). In that case a second query can be used to filter the rows with the same upper bound. For

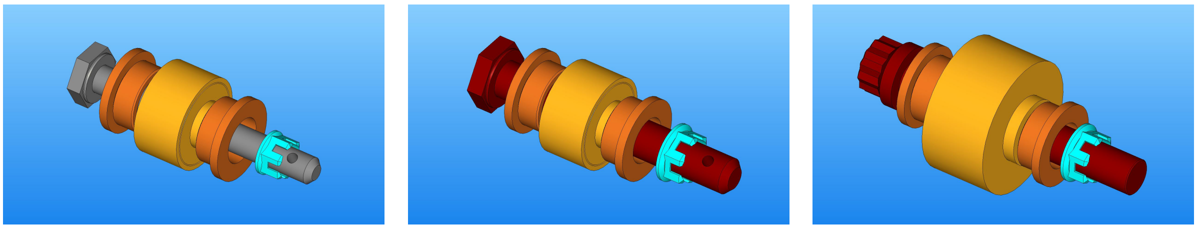


Figure 2.28: Different combinations of bolt, bearing, fixed bushing and nut. On the left: 6 mm diameter titanium bolt, rolling bearing. Right: Steel, tension bolt, spherical bearing.

example, the query $m_{be} = 0$ will yield the combination with radius $r_{bolt} = 3.1625$ and with m_{be} as close to 0 as possible, i.e., the lightest bearing.

In itself, the KBE application could enable engineers at Fokker to design the assembly in an easier way that what was possible with legacy methods. Table 2.3 summarizes the three variations developed, each providing a different compromise between precision (user having authority on the exact standard part chosen) and development speed (query tasks being fully automated). As such, it takes a step towards solving the research question already: it automates many of the design steps (material selection, standard selection, interface with different analysis tool, parametrization through GUI).

Use-case	**Inputs**	**Speed**	**Accuracy**
Manual selection	Bolt standard, bolt diameter code, etc.	1/3	3/3
Combination selection through integer lookup	n_{comb}	2/3	2/3
Combination selection through rounding	$r_{bolt} = 3.5$, $m_{be} = 0$	3/3	1/3

Table 2.3: Summary of the developed hinge KBE application. The score (1/3 to 3/3) indicates the performance of the application.

The Hinge Generator is also useful to compare two assemblies designed for the same load case. It enables the user to quickly understand the differences, rather than simply seeing that one hinge is lighter but more expensive than the other.

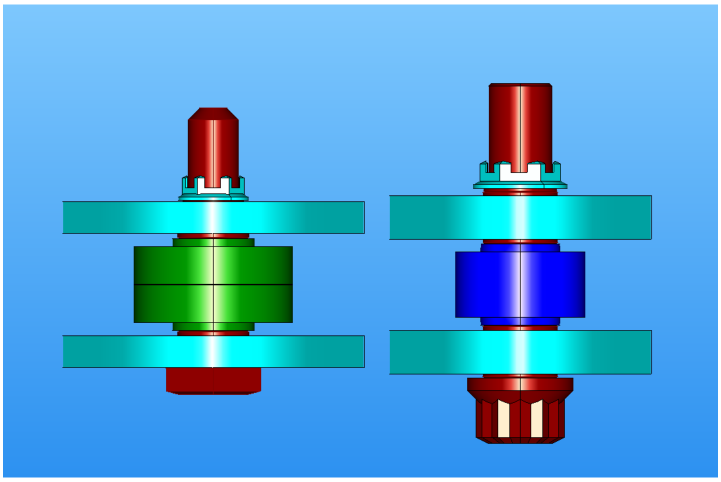


Figure 2.29: Top-view of two hinge assemblies designed the same load case. The colors of the outer lugs, bolt and nut reflect the materials, while the bearing color reflects its reserve factor. The hinge assembly on the right has a tension bolt with a stronger steel, which can carry a higher bending moment. Therefore, the thickness of the outer lug can be higher than that of the hinge on the left.

2.7. Discussions

The modeling of the interface is an important part of the rudder design process. Reducing the interface design time will facilitate the rudder design. The design requirements of the rudder-fin interface result in different hinge assembly designs. One rotation hinge needs to be clamped and two need to be fail-safe. Looking at the rudder-fin interface design also enabled to see that the internal loads (described by the parameters (F, α, θ)) and available space (described by the inequality constraints $e_{il}, e_{ol} \leq e_{max}$) are the two key requirements of a hinge in the interface.

Although the stress, cost and mass analyses rely on analytical equations, the hinge design process is complicated considering the fully detailed stress analysis, component characteristics and design rules. The use of KBE enabled to package those analysis and design rules in one single KBE model called the `Hinge Generator`. It allows the evaluation of a hinge instance in 0.5 seconds at most. Moreover, the KBE application can be used to easily select standard components and materials, so that even without a search method, the KBE model can be used to design the hinge. Three KBE application variations were developed, each providing a different compromise between precision and automation.

Even though the hinge KBE application can be used manually, this method could hardly be used to systematically obtain all the optimal hinges in terms of mass and cost. The design space is difficult to explore, and the ramifications of the geometry, material and standard parts coupling make it difficult for the engineer to predict the influences of the different parameters. Obtaining *all* the optimal hinges is important, as it could answer "*what if*" questions and limit Fokker's current use of custom-made bearings and bolts, and rely on similar cheaper, standard alternatives instead. The KBE application also enables numerical optimization, as the hinge can be evaluated efficiently and automatically. The hinge optimization is discussed in chapter 3.

3

The hinge optimization problem

The current tools and methods at Fokker require several days to design a single hinge. Moreover, the Hinge Generator presented in the previous chapter enables the generation of feasible hinge assemblies. Therefore, the available time is insufficient to answer "*what if*" questions, such as "Should we use this cheaper, but larger bolt?". Instead, Fokker often designs a single hinge, tailored to the requirement, using custom-made bearings and bolts. Using those components can result in a hinge assembly that is two to five times more expensive than one that uses standard components only [38]. Hence, obtaining all the optimal design alternatives is advantageous. It would limit Fokker's current use of custom parts and enable them to rely on cheaper, standard designs instead. A method to automatically explore the design space promises better and faster results, compared to manually varying the input parameters of the Hinge Generator.

Section 3.1 introduces and formulates the optimization problem. Necessary background information is given in section 3.2. In the light of those prerequisites, existing methods are considered and discussed in section 3.3. Four optimization methods are implemented. The first is a meta-heuristic algorithm (GA in section 3.4). The problems encountered with this method lead to the development of several ad-hoc search methods, described in section 3.5. Section 3.6 details the results associated to the Informed Linear Search. Lastly, section 3.7 summarizes the different methods developed, reflects on the research carried out in this chapter and discusses possible improvements.

3.1. Problem statement

This section gives further details on the optimization problem introduced in subsection 1.4.

3.1.1. Need for an optimization method

Manually using the hinge KBE application (as explained in subsection 2.6.4) could hardly be used to systematically obtain all the optimal hinges in terms of mass and cost. The geometry, material and standard parts coupling complicate the design choices. The *geometry coupling* refers the fitting of the parts, while the *material coupling* refers to the fretting design rules. The *Standard part coupling* refers to the fact that the dimensions of a standard part are dependent on each other. In figure 3.1, an example of the considerations to take into account for the bearing outer radius $r_{be,out}$ is given. The influence of $r_{be,out}$ are grouped in terms of disciplines (*mass*, *cost*, and *stress*).

For example, the outer radius of the bearing should *decrease* to increase the reserve factor of the inner lug. But it should also *increase* to increase the reserve factor of the bearing. Taking all these considerations into account results in conflictual decisions (both $r_{be,out} \nearrow$ and $r_{be,out} \searrow$).

Furthermore, an exhaustive search (evaluating *all* the possible designs in the design space) is excluded because it would be too time-consuming. Such a search would be feasible with a small design space (i.e., few discrete parts combinations to explore). However, the run-time would increase quickly[1] as the size of the design space increases. One of the requirements of this project is to take into account the full Fokker database of standard parts, consequently, exhaustive search should be avoided.

[1] Run-times for exhaustive search in terms of combinations are given in subsection 3.6.4.

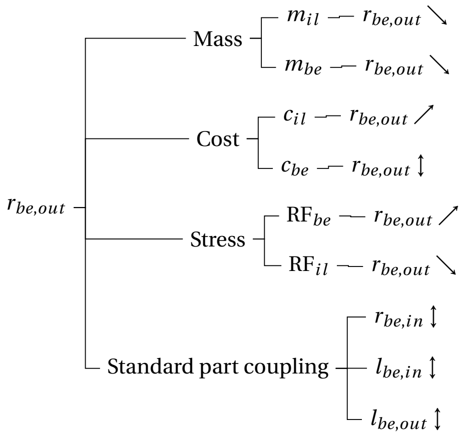

$r_{be,out}$
Mass
m_{il} — $r_{be,out}$ ↘
m_{be} — $r_{be,out}$ ↘
Cost
c_{il} — $r_{be,out}$ ↗
c_{be} — $r_{be,out}$ ↕
Stress
RF_{be} — $r_{be,out}$ ↗
RF_{il} — $r_{be,out}$ ↘
Standard part coupling
$r_{be,in}$ ↕
$l_{be,in}$ ↕
$l_{be,out}$ ↕

Figure 3.1: Considerations to take into account when trying to predict the value of the bearing outer radius $r_{be,out}$. The symbol ↘ means $r_{be,out}$ should decrease, the symbol ↗ means $r_{be,out}$ should increase and ↕ means it cannot be predicted. All things considered, $r_{be,out}$ cannot be determined from first principles.

Therefore, methods to search the design space more efficiently are necessary. The KBE application enables numerical optimization, as the hinge can be evaluated efficiently and automatically.

3.1.2. Characteristics of the optimization problem

In this subsection, the characteristics of the optimization problem introduced in chapter 1 are discussed in further detail. The problem is non-linear and contains both integer and continuous design variables. It is also multidisciplinary and multi-objective.

Non-linear The stress analyses of each hinge component result in several non-linear constraints that ensure that the ratios of applied to allowable forces are acceptable. As for any structural optimization problem, those constraints oppose the objective, which tries to minimize the amount of material used. The complete list of the inequality constraints is as follows:

$$\begin{aligned}
\mathrm{RF}_{bolt,A} &\geq 1.0\\
\mathrm{RF}_{bolt,M} &\geq 1.0\\
\mathrm{RF}_{be,axial} &\geq 1.0\\
\mathrm{RF}_{be,radial} &\geq 1.0\\
\mathrm{RF}_{il} &\geq 1.0\\
\mathrm{RF}_{ol} &\geq 1.0\\
\mathrm{RF}_{sb} &\geq 1.0\\
\mathrm{RF}_{fb} &\geq 1.0\\
\mathrm{RF}_{nut} &\geq 1.0\\
\mathrm{RF}_{washers} &\geq 1.0
\end{aligned} \tag{3.1}$$

Those constraints can also be regrouped per component, for example by using:

$$\mathrm{RF}_{be} = \min(\mathrm{RF}_{be,axial}, \mathrm{RF}_{be,radial}) \tag{3.2}$$

As outlined in section 2.3, the stress analyses result in polynomial (power 1, 2, and 4) inequality constraints depending on parts dimensions (e.g., r_{bolt}) and material characteristics (e.g., σ_{bolt}). Likewise, the mass and cost of the parts mainly depend on the volume of the parts, which will vary in polynomial terms with the part dimensions. Therefore, the sizing of the hinge assembly is solvable by well-known optimization methods such as quadratic programming.

Combinatorial optimization A combinatorial optimization problem consists of finding an optimal object from a finite set of objects [29]. With standard parts, the hinge optimization problem resembles a combinatorial optimization problem. The problem is about selecting a sequence of components [bolt, bearing, ...] from all the existing combinations $[\text{bolt}_1, \ldots, \text{bolt}_n, \text{bearing}_1, \ldots, \text{bearing}_n, \ldots]$.

The hinge optimization problem differs from a quasi-discrete problem with explicit variables (i.e., minimize $(m(R, L))$, with $R \in [R_1, \ldots, R_n], L \in [L_1, \ldots, L_n]$), where the design variables could be rounded off the closest discrete value [15]. To explain the influence of standard parts, figures 3.2 and 3.3 illustrate the bearing and its coupling with the rest of the assembly. Figure 3.3 gives the dimensions of several bearings. Significant errors would derive from rounding if the hinge was modeled with a continuous design variable r_{bolt}. Therefore, if it would be practical to describe and solve the optimization problem using the bolt radius r_{bolt}, it would not be effective with standard parts. The rounding of one variable could lead to sub-optimal component choices and sub-optimal, or even unfeasible, hinge assemblies. Further information on combinatorial optimization methods is given in subsection 3.2.2.

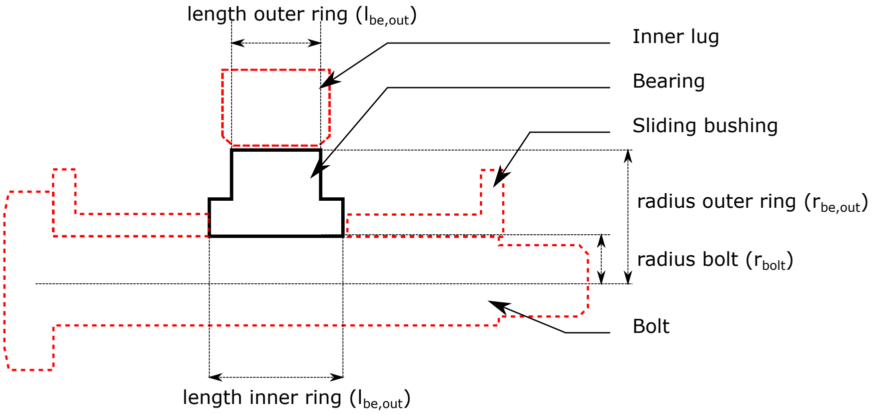

length outer ring (l_be,out)
Inner lug
Bearing
Sliding bushing
radius outer ring (r_be,out)
radius bolt (r_bolt)
Bolt
length inner ring (l_be,out)

Figure 3.2: The bearing (dark, full lines) is in contact with three other components (dashed, red lines): the bolt, the sliding bushings and the inner lug. Therefore, any of the four bearing dimensions (length outer ring $l_{be,out}$, length inner ring $l_{be,in}$, radius outer ring $r_{be,out}$ and radius inner ring $r_{be,in}$) will affect the dimensions of those components.

Multidisciplinary Design and Optimization As discussed in chapter 2, the optimization problem is an MDO problem, not only in the literal sense (three disciplines: stress, cost, and mass) but also in the more prominent aspect of MDO, as the hinge model result in a set of coupled analyses that have conflicting interests [30]. MDO is a research field that studies different frameworks to organize the decomposed system, so that the system can be optimized despite its complexity [20]. Conflicting interests within the hinge assembly have already been discussed in the previous subsection, with figure 3.1.

Further design aspects demonstrate that the hinge is an MDO problem. Having standard parts and materials implies that coupling variables y will result from one design variable x. In figure 3.3 for example, the bearing can be seen as a discipline with an input $x = r_{be,out}$ which results in three coupling variables: $\mathbf{y} = [r_{be,in}, l_{be,out}, l_{be,in}]$. This circumstance are challenging because the optimization needs to choose the set of parameters in a way that consistently increase or reduce the objectives.

The MDO decomposition can be made in terms of disciplines (stress, mass, cost) or in terms of subsystems (bolt, bearing, fixed bushings, etc.). As all of the three disciplines require the same inputs (complete geometry and material characteristics), once the components are known, the disciplines can be evaluated concurrently. Therefore, the decomposition in terms of discipline would not be particularly interesting, and the subsystems decomposition is used.

Multi-objective optimization A multi-objective optimization is an optimization where more than one objective function have to be optimized simultaneously. For the hinge, both the weight and cost must be taken into account during the optimization. The outcome of the optimization method must therefore be a set of points $(x_1^*, \ldots, x_n^*)$ instead of a single one (x^*). This set of points is called the **Pareto-optimal set**, which is

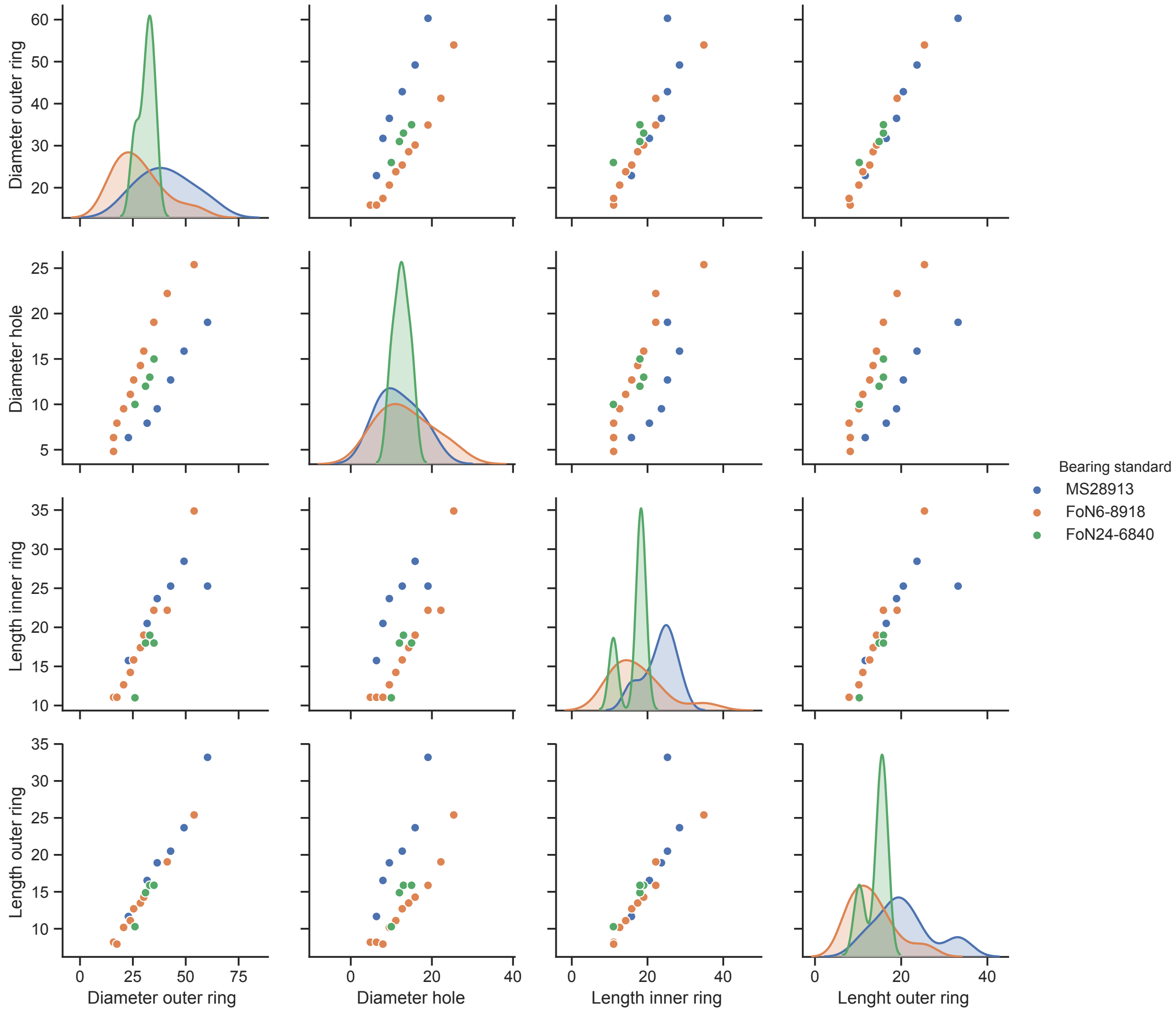


Figure 3.3: Dimensions in $[mm]$ of the bearings considered for the search. The cost, mass, and allowable forces of the bearing are not represented, but vary similarly, in a non-linear manner.

defined as the set of solutions which are superior to the rest of the solutions when all the objectives are considered, but are inferior to other solutions in one or more objectives.

Limiting the optimization to either cost or mass results in sub-optimal hinges. For example, choosing the cheapest bolt might result in a larger bolt diameter The larger bold diameter affects the attached components, resulting in a higher mass for all other components, and thus a higher total mass. In principle, having the multi-objective is similar to having several constraints and variables, but most open-source optimization algorithms only support a single objective.

As a conclusion, the search method must handle multi-objective, non-linear inequality constraints and integer and continuous design variables. The generic hinge optimization problem formulation is given in equation 3.3:

$$
\begin{aligned}
&\min \sum_{k} m_k, c_k \qquad (3.3)\\
&\text{s.t. } \forall\ k, \mathrm{RF}_k \geq 1.0\\
&\text{with } k \in [\text{bolt, bearing, fixed bushing, sliding bushing, inner lug, outer lug, nut}]\\
&\quad \mathbf{x}^i = [\text{bolt, bearing, fixed bushing, nut}]\\
&\quad \mathbf{x}^c = [e_{ax,il}, e_{tr,il}, e_{ax,ol}, e_{tr,ol}, t_{ol}, t_{sb}]\\
&\quad \mathbf{p} = [\alpha, \theta, F, g, \beta_{ol}, \beta_{il}, mat_{il}, mat_{ol}]
\end{aligned}
$$

3.2. Background information

Before tackling the hinge optimization problem, this section defines the criteria used to define the performance of the optimization or search methods (subsection 3.2.1) and introduces combinatorial search methods (subsection 3.2.2).

3.2.1. Criteria for the measure of search methods performance

The following definitions are adapted from [28]. These criteria will be used to evaluate the performance of the search methods.

Optimality is about asking "*Does the method find the optimal set of solutions?*" For example, a gradient descent optimization method applied to a convex function is optimal. An exhaustive search is by optimal too, as it explores all the design space. As the problem is multi-objective, this can also be referred to as **pareto-optimality**.

Efficiency is about asking "*How much time does the method take to find the solutions?*" In the literature, the optimization run-time is also referred as time complexity (time required by the method to find the solutions). In the rest of the project, a method is said to be efficient if its run-time is low (order of magnitude of seconds, or minutes). Efficiency concerning space complexity (memory required by the method [28]) is not taken into account during this project.

Scalability is about asking "*How does the optimization run-time scale with the size of the design space?*" An essential requirement of the KBE model is that it models the full set of standard parts that are used at Fokker. This requirement applies similarly to the search method. Ideally, the performance of the method with respect to its efficiency and optimality should be independent of the number of parts considered. The asymptotic notation $O(-)$ ("big O") characterizes the upper bound of the growth rate of the function.

A large number of combinations does not necessarily imply a higher run-time. Similar combinatorial problems with greater amounts of combinations (e.g., Traveling Salesman problem [2] for 30 cities has $30! \approx 2.65 \cdot 10^{11}$ possible paths) can be solved in a few minutes.

Framework is about asking "*How easy is it to develop and modify the search method within the front-loaded framework?*" In the literature, several frameworks that enable the optimization of complex systems. According to [4], connecting a Process Integration and Design Optimization (PIDO) tool to a KBE application is the most advantageous framework.

In this project, a custom framework containing `paraMDO`, developed by Onodi [23], is used. The framework is detailed in the second part of the report, in chapter 8.

Taking into consideration all the criteria described above, the ideal search method should yield **Pareto-optimal** hinge assemblies, within a **short run-time** (minutes), **independently of the number of standard parts** considered. The framework implementing this search method should enable the method to be easily modifiable and **extensible**.

3.2.2. Combinatorial search methods

Combinatorial optimization consists of finding an optimal object from a finite set of objects [29]. Compared to the continuous problem, seeing the hinge as a purely combinatorial problem takes away a significant portion of the information available to solve it. First, a bolt has to be chosen, then, a bearing that fits onto this bolt has to be chosen, etc. Therefore, it becomes a matter of subsequently selecting a component.

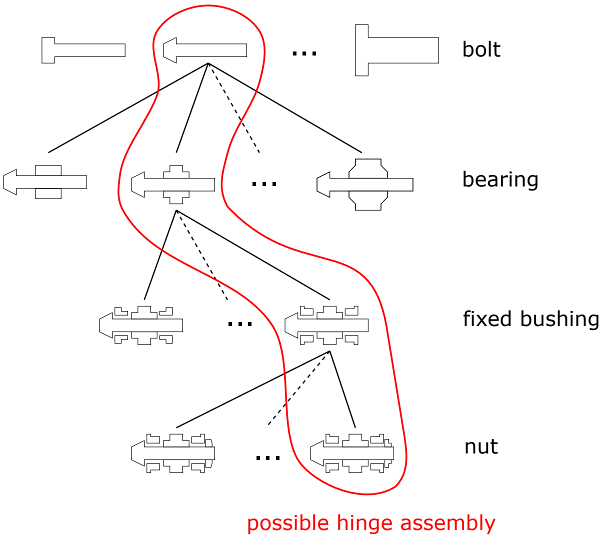
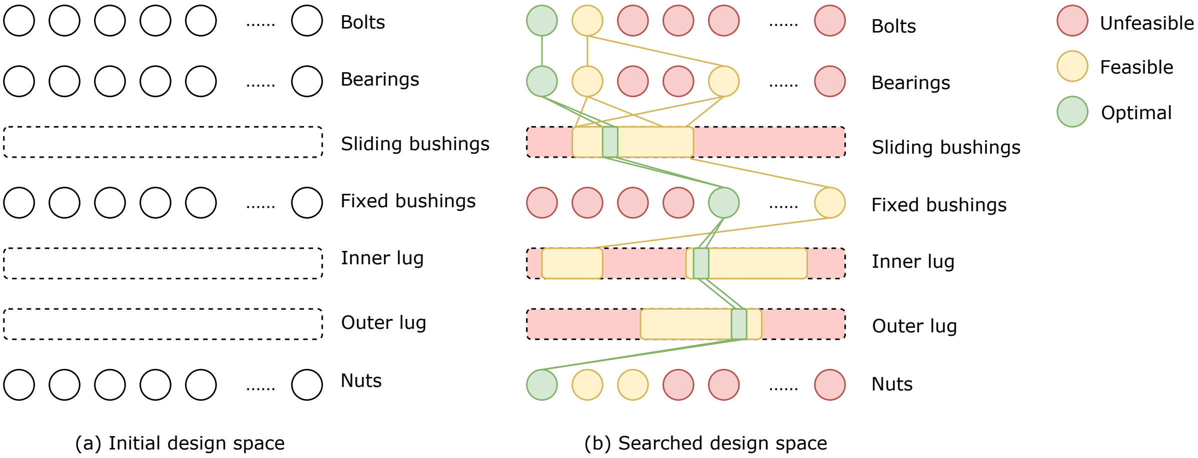


Figure 3.4: The design space of the hinge assembly can be seen as the subsequent selection of components. For each layer of component, one is expanded and the subsequent components that fit are shown underneath. A hinge assembly is a set of components (e.g., the subsequent components circled in red).

With, combinatorial problems, the design space can be seen as a **tree** and a solution can be seen as a **branch**, which is a combination of **nodes**. An example of such a tree is given in figure 3.4. It describes the design space obtained by selecting four main standard parts: the bolt, bearing, fixed-bushing, and nut. The search is the subsequent selection of each standard part of the hinge assembly. A possible hinge assembly is a branch, or a combination of each part, for instance the one indicated by the path in red in figure 3.4.

Figure 3.5 gives a more compact and complete representation of the design space of the hinge assembly. The order of the components is arbitrary, the figure is only used as an example to introduce important concepts.



(a) Initial design space

(b) Searched design space

Figure 3.5: A hinge assembly is a combination of standard components (circles) and machined components (rectangles). Out of all the possible combinations, the majority is unfeasible because it does not fit, frets, or has unsatisfactory reserve factors (red). Then, some combinations would satisfy all the requirements, but would be non-optimal in terms of mass and cost (yellow). The goal of the search is to find the optimal combinations (green).

On figure 3.5, the **depth** of three is $n = 7$ and the bearing layer (i.e. the set containing all the bearings) is at depth $d = 2$. To search for possible combinations, the algorithm must evaluate the children of the currently selected node. This process is called **expanding**. The strategy that chooses which nodes to expand to attain the optimal solution is called the **tree search strategy**. A review of different tree search strategies can be found in [28], which is given in figure 3.6.

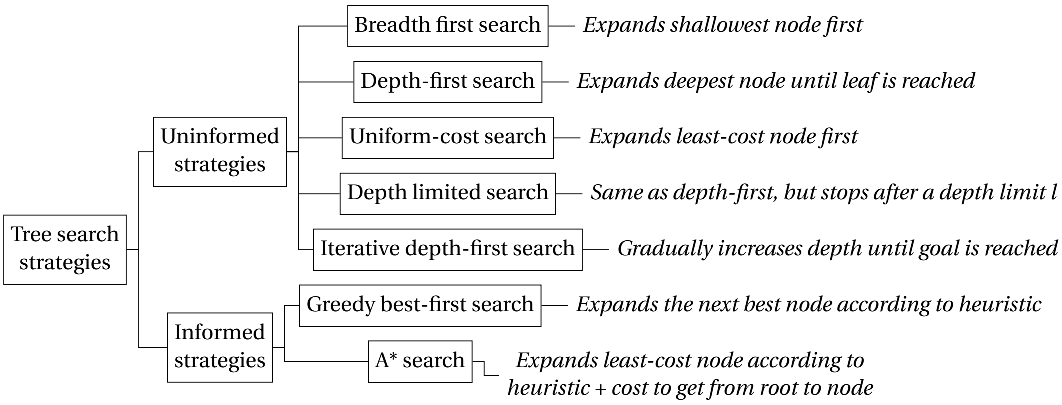

Tree search strategies
Uninformed strategies
Informed strategies
Breadth first search
Expands shallowest node first
Depth-first search
Expands deepest node until leaf is reached
Uniform-cost search
Expands least-cost node first
Depth limited search
Same as depth-first, but stops after a depth limit l
Iterative depth-first search
Gradually increases depth until goal is reached
Greedy best-first search
Expands the next best node according to heuristic
A* search
Expands least-cost node according to heuristic + cost to get from root to node

Figure 3.6: Classification and definitions of relevant tree search strategies.

Uninformed search strategies are general guidelines that describe which area should be explored next in the tree. With the **breadth-first** search, for instance, the shallowest nodes are expanded first. In figure 3.5, it would mean that all the bolts would be explored first, then all the bearings, etc. With the **depth-first** search, a bolt is chosen and all subsequent components too, until the nut is reached. The **uniform-cost** search is similar to the breadth-first search, expect that the choice of the next node considers the best cost, instead of the shallowest region A **depth-limited** search strategy could be interesting in the case of the hinge assembly, as it acts as the tree had a depth l with $l < n$. Lastly, the **iterative-depth** strategy iteratively applies the depth-limited search for different limits l. This last search can be thought of as a mix between depth-first and breadth-first. Similarly to the limited search utilization, the idea is that the search could be done iteratively until a desired level of detail is attained.

Informed search strategies use a heuristic function to estimate the final cost function of the path, which is in the case of the hinge the objective function of the complete assembly. The simplest form of informed search is the **greedy best-first** search, which only expands the node that has the best cost. Effectively, it subsequently selects the cheapest or lightest components, layer by layer. The **A-star** search slightly differs from greedy best-first search; its heuristic is given by the sum of the cost of that node and the cost to reach the node.

3.3. Existing methods

This section reviews the existing methods for hinge optimization. The methods are listed in the next three subsections, in chronological order. For each approach, the performance of the method with respect to the criteria introduced in subsection 3.2.1 is discussed. The characteristics of the previous algorithms, as well as the requirements on the one to be developed, are then summarized in table 3.1.

3.3.1. Legacy method

The legacy method used at Fokker before 2013 was manual and iterative. This methodology is documented by Kulkarni in [18]. Most of the time, the design process started with the bolt. Then, the components would be designed and analyzed in the following order: bearing; central lug; clevis lug; sliding bushing; fixed bushing; nut. As described in subsection 2.3.2, performing the bolt analysis requires the lug geometry and material. Therefore, starting the design of the hinge with the bolt requires an assumption on the lugs geometry and material. The bolt design could be satisfactory, but the lug assumption could not be satisfactory which results in the necessity to reiterate the process. This method involves a lot of iterations to get to the final, feasible design. The design-time is unsatisfactory, as this method only enables the study of a few feasible hinge assemblies, within six weeks [18].

3.3.2. Quasi-exhaustive search in HDOT

In HDOT [18], the *Hinge sizing* module contains a hinge optimization method. HDOT uses a form of combinatorial search called the “quasi-exhaustive” search. The pseudo-code of the quasi-exhaustive search is presented in algorithm 3.1.

```
1  Filter out feasible bearings;
2  for bearing in feasible bearings do
3      for material inner lug do
4          Determine $e_{il}$ by bisection search;
5          ($e^U_{il}$ = distance to OML);
6      Select the lightest inner lug;
7      Size the sliding sleeves (assumed $t_{il}/t_{ol} = 0.6$);
8      Select the bushing;
9      for material outer lug do
10         Determine $e_{ol}$ by bisection search;
11         ($e^U_{ol}$ = distance to OML, assume $t_{il}/t_{ol} = 0.6$);
12     Select the lightest outer lug;
13     Size the bolt length;
14     Select the cheapest feasible bolt;
15     Select the cheapest feasible nut;
16 Select the cheapest hinge assembly;
17 return hinge assembly;
```

Algorithm 3.1: Quasi-exhaustive search in HDOT (derived from [18]). *sizing* refers to a continuous variable, while *Select* refers to a standard component. The search is exhaustive in terms of bearings (line 2), lug materials, but only selects one bolt per bearing (line 14).

In this method, the components are chosen in a sequence: first, select the cheapest/lightest bearing; then select the cheapest/lightest lug, etc. It assumes that the lightest assembly is the combination of all lightest components. Therefore, it resembles the best-first greedy method described in subsection 3.2.2. However, the selection is only based on the weight of *one* component, whereas in the literature, best-first is based on the estimation of the total weight of the remaining components to be chosen.

Optimality The greedy heuristic of HDOT leaves out a part of the design space [18, 25], as the search automatically selects the cheapest feasible bolt for a given bearing (line 14 on algorithm 3.1) If the cheapest bolt is infeasible, any combination with this bearing is excluded, whereas the same bearing with a slightly more expensive bolt and stronger material could have yielded a good assembly

The problem with the assumption was confirmed twice. First, the comparison between the HDOT optimum and the Multidisciplinary feasible architecture optimum (see next subsection) revealed that the HDOT optimum was 42% heavier [25]. Secondly, changing the sequence in which the components were analyzed resulted in different optimal assemblies [26].

Lastly, the selection is not based on both the *mass* and *cost* of the component. Some parts of the search are done focusing on a single objective. The quasi-exhaustive search yields only one result, and that result cannot be proved to be optimal or pareto-optimal.

Scalability A best-first search strategy should scale very well with the increase of input variables, as it only expands the best node. As the rest of the nodes are ignored, the run-time should not vary with the size of the design space. However, it is not true with the quasi-exhaustive search, as the bearing search is exhaustive (line 2 on algorithm 3.1). Therefore, the search scales well with the number of bolts (as only one bolt per bearing is considered) but not with the number of bearings considered.

Framework Expanding HDOT with more hinge types and materials would be difficult, as the search is embedded into the KBE application. This approach scales poorly with complexity, a new KBE application and a new search algorithm would have to be implemented for each new hinge type. That is unsuitable with the front-loading framework, which must eventually be extended to many different hinge types.

3.3.3. GA implementation

Researchers at TU Delft worked on improving HDOT. The different disciplines that constitute HDOT were taken apart and integrated into a workflow defined in Optimus[2], a PIDO tool.

[2] https://www.noesissolutions.com/

Though the problem is formulated as an MDO problem, it can also be seen as a simple optimization in which the objective function is computed in by a sequence of sub-disciplines. The MDO architecture used was a Multi Disciplinary Feasible (MDF) architecture, for which the results are presented in [25] and [16]. To deal with the presence of integer design variables, a Genetic Algorithm (further discussed in subsection 3.4) included in Optimus was chosen.

Optimality The GA converged after 9 hours. With genetic algorithms, it is difficult to prove that the minimum was reached. As with HDOT, the cost was not taken into account. It also only yields one hinge assembly, not a Pareto-set of optimal hinge assemblies.

Scalability The evolution of the GA run-time with the size of the design space is unknown. It is uncertain how this implementation would perform with an increased number of discrete materials and standard parts.

Framework Lastly, using a PIDO tool to define the workflow and search algorithm scales well with the complexity of the model and of the optimization method. The PIDO tool enables to modify the optimization workflow if new types of hinge were to be included. Moreover, the framework defined in [16] enables a reduction of the MDO problem setup-time.

3.3.4. Discussions on the existing methods

Method	**Design space**	**Run-time**	**Optimality**	**Framework**
Quasi-exhaustive (3.3.2)	Limited database	1.5 hours	No	KBE model
GA (3.3.3)	Limited database	9 hours	Yes (1 point)	PIDO + KBE tool
To be developed	Fokker database	minutes	Yes (set of points)	Ad-hoc + KBE tool

Table 3.1: Summary of the previous optimization methods and comparison with the one to be developed. The *Limited database* indicates that few standard parts are used (less than five bolts and bearings), whereas *Fokker database* indicates that the full database of standard parts of Fokker should be used (hundreds of parts).

In a nutshell, the quasi-exhaustive search of HDOT is neither efficient nor optimal, and its implementation within the KBE model is not flexible enough to be extended to other hinge types for front-loading. Although the GA implementation cannot be proved to be optimal, it did find a 42% lighter hinge assembly than the quasi-exhaustive search.

In table 3.1, the row "*to be developed*" describes the ideal characteristics of the search method to develop during this project. The problem is often thought to be difficult because of the sheer amount of possible combinations, but the actual amount of feasible assemblies is far smaller. A method that is both optimal and efficient therefore seems like a feasible goal.

The problem tackled in the following sections is slightly different than the one described in HDOT and in the GA implementation. The materials of the lugs are then considered to be design variables, whereas, in this project, they are treated as parameters. Moreover, the model and the databases have changed significantly, as detailed in subsection 7.2 in the second part of the report. Therefore, the existing methods cannot be compared quantitatively to the developed ones regarding optimality, nor concerning efficiency.

3.4. Attempts with genetic algorithms

Genetic algorithms are a typical algorithm that would be used to tackle this problem; they have been extensively used for similar problems in the past decades [24]. Genetic algorithms can address a vast range of problems, from conventional numerical optimization problems to combinatorial optimization problems, where the design variables represent a new combination (see for instance the Traveling Salesman Problem (TSP) [2]). As it can be used out of the box and is expected to perform well, it is the first option considered for the hinge optimization problem.

The developed framework[3] uses the OpenMDAO library. OpenMDAO features a simple implementation of the Genetic Algorithm [4]. The pseudo-code is given in algorithm 3.2.

[3] Further details on the optimization framework are given in chapter 8.

[4] See documentation: `http://openmdao.org/twodocs/versions/2.5.0/_srcdocs/packages/drivers/genetic_algorithm_driver.html`

```
Parameters: Max. number of iterations $n_{iter}$, mutation rate $r_m$, crossover rate $r_c$, population size
    $n_{pop}$, penalty coefficient $p$, penalty exponent $\kappa$;
Encode design vector $x$;
while $n < n_{iter}$ do
    Perform mutation;
    Perform crossover;
    Perform tournament selection;
Decode design vector ;
return hinge assembly;
```

Algorithm 3.2: Genetic algorithm implementation in openMDAO. The steps are typical for GAs (mutation, crossover and selection). The algorithm supports inequality constraints and multi-objective functions by subtracting them to the objective.

The multi-objective nature of the problem is included with a weighted sum:

$$f = \sum \lambda_i \cdot f_i \quad \text{with} \quad \sum \lambda_i = 1 \tag{3.4}$$

The constraints are handled with a penalty function. The fitness function is constructed with the penalty parameter p and the exponent κ:

$$\Phi(x) = f(x) + p \cdot \sum (\delta_k \cdot g_k)^\kappa \qquad \text{with} \qquad \delta_k = \begin{cases} 0 & \text{if } g_k \text{ is satisfied} \\ 1 & \text{otherwise} \end{cases} \tag{3.5}$$

The general formulation given previously in equation 3.3 is not directly applicable because the categories (bolt, bearing, etc.) need to be selected from numerical values. One way to deal with these categorical design variables is to use an integer design variable as an index, which is then used to lookup the characteristics (bolt radius, material properties, etc.) of the part in a table. Two problems formulations using integer lookup (equations 3.6 and 3.9) are tested.

This first formulation In equation 3.6, integer variables are used to select the standard bearing and bolt, as well as their size.

$$\begin{aligned} &\min \sum_k m_k, c_k \\ &\text{s.t. } \forall\ k, \mathrm{RF}_k \geq 1.0 \\ &\text{with } k \in [\text{bolt}, \text{bearing}, \text{fixed bushing}, \text{sliding bushing}, \text{inner lug}, \text{outer lug}, \text{nut}] \\ &\qquad \mathbf{x}^i = [n_{bolt}, n_{bearing}, D_{bolt}, D_{bearing}] \\ &\qquad \mathbf{x}^c = [e_{ax,il}, e_{tr,il}, e_{ax,ol}, e_{tr,ol}, t_{ol}] \\ &\qquad \mathbf{p} = [\alpha, \theta, F, g, \beta_{il}, \beta_{ol}, mat_{il}, mat_{ol}] \end{aligned} \tag{3.6}$$

The integers n represent the **type of the standard** part. For example, $n_{bolt} = 1$ corresponds to a FoN1623 bolt (a titanium, tension bolt) and $n_{bearing} = 2$ corresponds to a MS28913 bearing (Military Standard roller bearing). Then, the integer D represents the **size code** (or diameter code) of the standard part. For instance, for the given bolt standard $n_{bolt} = 1$, different sizes exist, as depicted in figure 3.7.

The formulation of equation 3.6 exposes more variables in the design vector compared to the existing GA implementation described in subsection 3.3.3. The categorical design variables (bolt and bearing standard) are unordered, whereas the standard part designation is ordered by increasing radius.

This formulation is practical from a designer's perspective. The choice of the integer variables ($n_{bolt}, n_{bearing}$) influences the cost and mass of the hinge assembly significantly. If a particular class of bearing or bolt is notably less expensive than another, this formulation could converge towards this class faster. Then, the integer variables $D_{bolt}, D_{bearing}$ are used to determine if the assembly fits or not. A problem with this formulation is that some upper bounds are also depending on other design variables:

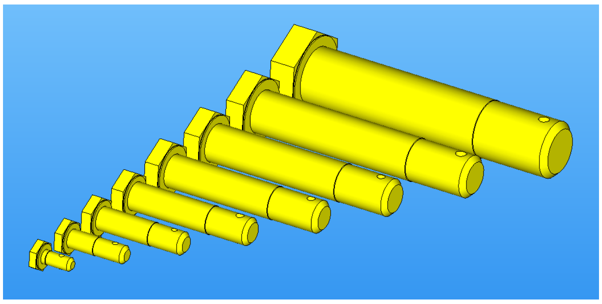


Figure 3.7: Bolts according to their diameter codes for the bolt FoN2-1620 standard. In equation 3.6, the integer variable D_{bolt} is used to lookup the characteristics of the bolts.

$$n^U_{bolt} = f(D_{bolt}) \tag{3.7}$$

$$n^U_{bearing} = f(D_{bearing}) \tag{3.8}$$

Because depending on the type of bolt chosen, a different number of bolt diameters exist. For instance, if $n_{bolt} = 0$, there are only three possible diameters $D_{bolt} = (0,1,2) \Rightarrow D = (5,6,7)$, whereas if $n_{bolt} = 1$, there are eight possible diameters $D_{bolt} = (0,1,\ldots,7) \Rightarrow D = (3.45, 3.5, \ldots, 9.65)$. Most of the points of this design space are inherently unfeasible (do not fit or fret), or because $n_{bolt} > n^U_{bolt}$ or $n_{bearing} > n^U_{bearing}$. After several attempts with this formulation (manually varying the parameters of the GA $n_{iter}, r_c, r_m, \kappa, p$), this method **failed to converge**. Optimality, efficiency and scalability can therefore not be discussed.

The second formulation Rather than checking the fitting of the assembly at run-time, at each iteration, this formulation requires a table of all the fitting combinations of components[5].

$$\begin{aligned} &\min \sum_k m_k, c_k \\ &\quad \text{s.t. } \forall\ k,\ \text{RF}_k \geq 1.0 \\ &\text{with } k \in [\text{bolt}, \text{bearing}, \text{fixed bushing}, \text{sliding bushing}, \text{inner lug}, \text{outer lug}, \text{nut}] \\ &\qquad x^i = n_{comb} \\ &\qquad \mathbf{x}^c = [e_{ax,il}, e_{tr,il}, e_{ax,ol}, e_{tr,ol}, t_{ol}] \\ &\qquad \mathbf{p} = [\alpha, \theta, F, g, \beta_{il}, \beta_{ol}, mat_{il}, mat_{ol}] \end{aligned} \tag{3.9}$$

Instead of having two integers per standard part (e.g., n_{bolt} and D_{bolt}), every integer is looked up from the table 3.2 using the integer n_{comb}. This formulation has the advantage of being inherently feasible.

n_{comb}	Bolt	Bearing	D_{bolt}	D_{bear}	D_{fb}	R
0	FoN2-1623	FoN6-8918	1	1	1	3.1585
1	FoN2-1620	FoN6-8918	1	1	1	3.1625
2	FoN2-1620	MS28913	1	0	1	3.1625
3	FoN2-1630	FoN6-8918	0	1	1	3.1625
...	...	...	...	...	...	...

Table 3.2: Combination of fitting components. The integer variable n_{comb} is used to lookup a combination of bolt, bearing, fixed bushing and nut that fits together for equation 3.9.

The performance criteria cannot be discussed, as the algorithm **did not converge**. Even after a large number of iterations, the GA displayed an erratic behavior. **Scalability** of the performances with the size of the design space is unknown too. One problem could be that the variation of n_{comb} overweight the variation of the others design variables. Here, n_{comb} is much more critical than any of the dimensions of the lugs.

[5] This table was already discussed in subsection 2.6.4 and will be also discussed in the next sections dedicated to other search methods.

Arguments in favor of an ad-hoc search method Using a meta-heuristic optimization algorithm seemed very convenient, but in practice, it is challenging to get the algorithm to converge, both with formulation 1 (equation 3.6) and formulation 2 (equation 3.9). Although the GA should have been able to solve this problem, its implementation did not yield satisfactory results.

From this unsuccessful implementation, several observations are made:

- A form of tree search could counter the problem of the first formulation (equation 3.6). As the integers are picked in a sequence, choices can be made depending on the part of the tree that has already been explored. Tree search, therefore, enables to exclude larger parts of the design space at each step. For example, the combination $D_{bolt} = 0$ and $D_{bearing} = 5$ never fits. In a tree search, this subset of the design space can be explored once and then excluded. The GA will eventually exclude this combination too, but it would take more iterations, as there are many other design variables ($e_{ax,il}, e_{tr,il}, e_{ax,ol}$...) and *all* the possibilities can be evaluated by the GA. The GA does not inherently know that $D_{bolt} = 0$ and $D_{bearing} = 5$ will always be unfeasible, no matter the values of the rest of the variables.
- To counter the problem of the second formulation (equation 3.9), it seems convenient to separate the discrete part of the problem $x^i = n_{comb}$ and the continuous part of the problem $\mathbf{x}^c$. Moreover, this separation enables the use of the gradient information in the continuous part of the problem, which could reduce the run-time of the optimization.
- With both GA implementation, the multi-objective part of the problem is addressed via the weighted sum (equation 3.4) in the objective function. However, it means that the problem would have to be solved for several values of the weight λ. There are better, multi-objective evolutionary algorithms (such as the NSGA-II algorithm [5]), but none are implemented in `openMDAO`.

As a conclusion, developing an ad-hoc search method for the hinge assembly could yield better optimality with a shorter run-time.

3.5. Developed methods

This section presents the optimization methods that have been developed during the project. Subsection 3.5.1 lays out the principles of the developed two-step method. Then, subsections 3.5.2, 3.5.3 and 3.5.4 all detail specific variations of the two-step method.

3.5.1. Proposition: ad-hoc 2-step methods

The idea of the 2-step method is to separate the search of the standard parts (combinatorial optimization problem) and the machined parts (continuous optimization problem).

Phase 1: tree search First, all the component combinations are exhaustively examined to find all the combinations of bolt, bearing, fixed bushing and nut that fit together (as seen in figure 3.8). The fitting combinations are then stored (i.e. the **tree** of combinations to explore is created). In practice, the tree is represented with a table contains attributes about the combination of components, as partially depicted in table 3.3.

Bolt	Bearing	D_{bolt}	D_{be}	D_{fb}	r_{bolt}	σ_{bolt}	m_{be}	c_{be}	$l_{be,in}$	$l_{be,out}$	$r_{be,out}$
FoN2-1623	FoN6-8918	1	1	1	3.1585	900	14.1	10.02	11.05	8.18	7.931
FoN2-1620	FoN6-8918	1	1	1	3.1625	700	14.1	10.02	11.05	8.18	7.931
FoN2-1620	MS28913	1	0	1	3.1625	700	27.216	78.2	15.748	11.6586	11.44143
FoN2-1630	FoN6-8918	0	1	1	3.1625	880	14.1	10.02	11.05	8.18	7.931
...	...	...	...	...	...	...	...	...	...	...	...

Table 3.3: Combination table generated by the `Hinge Generator`. Each row contains the part standards (columns Bolt and Bearing), diameter codes (columns D_{bolt}, D_{be}, D_{fb}) and some attributes of the combinations (7 right-most columns, e.g., r_{bolt} is the radius of the bolt), so that the components fit. In this table, the sorting criteria are $(c_1, C_2, C_3) = (r_{bolt}, \sigma_{bolt}, m_{be})$, but different ones can be used.

Then, a strategy must be used to explore that tree, as explained in 3.2.2. Each row then represents a full branch of the tree. Depending on the ordering of the rows of that table, the branches of the tree are not explored in the same order. For example, if the table is ordered by increasing bearing mass, the search can be seen as a tree search where the first layer are the bearings, ordered by mass. Then, for the same bearing, several bolts can be used. Likewise, the bolts can be ordered in terms of cost, mass, radius or tensile strength. For the two last standard components (nut and fixed bushing), two assumptions are used:

1. The fixed bushing diameter code is selected so that there is at least 0.5 mm for the sliding bushing. No smaller, or larger diameter is considered. This assumption stems from the experience with the `Hinge Generator`, with which the compression inequality constraint (described in subsection 2.3.4) was almost always satisfactory. This assumption is evaluated in subsection 3.6.2.
2. Only one nut standard is explored. This simplification was made because the influence of the nut standard is assumed to be low (less that 5% of total weight and cost). This assumption is evaluated in subsection 3.6.2.

On table 3.3, one can see the three sorting criteria (c_1, c_2, c_3) used: the bolt radius r_{bolt}, bolt tensile strength σ_{bolt}, and mass of the bearing m_{be}. For equal radius, the tensile strength increases and for equal strength, the mass of the bearing increases. Those criteria are chosen so that the higher n_{comb}, the higher the loads the hinge can sustain, but the higher its mass and cost are.

Phase 2: continuous optimization Once the combination of standard components is selected, the geometry is almost fully defined, so that the rest of the dimensions to determine are continuous, and the problem can be solved with an SQP method.

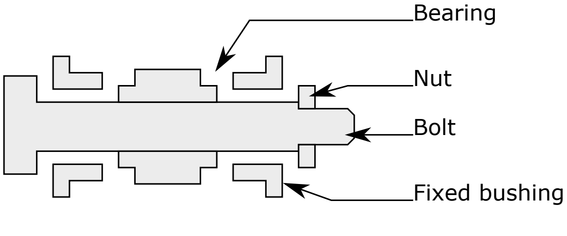


Figure 3.8: **First phase**: a bolt, bearing, fixed bushing and nut is selected. If chosen from a row of table 3.3, the components will all fit together. In practice, a combination of components is either chosen from the table 3.3 (BFS; ILS) or randomly (RTS).

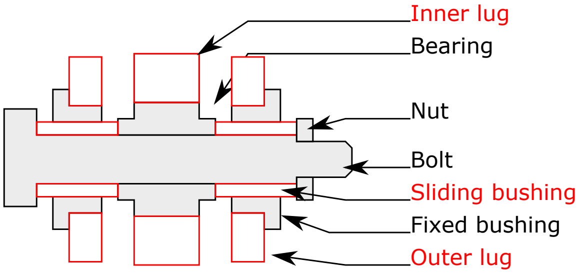


Figure 3.9: **Second phase**: the machined parts of the lug and sliding bushing (indicated in red) are sized.

If the selected branch is designated by an integer $x^i = n_{comb}$, then the continuous part of the problem can be simply reformulated as:

$$\begin{aligned} &\min \sum_k v_k(x,p) \\ &\text{with } k \in [\text{bolt, inner lug, outer lug, sliding bushing, fixed bushing}] \\ &\text{s.t. } \mathrm{RF}_{bolt} \geq 1.0 \\ &\quad \mathrm{RF}_{il} \geq 1.0 \\ &\quad \mathrm{RF}_{ol} \geq 1.0 \\ &\quad \mathbf{x}^c = (e_{ax,il}, e_{tr,il}, e_{ax,ol}, e_{tr,ol}, t_{ol}) \\ &\quad \mathbf{p} = (n_{comb}, \alpha, \theta, F, g, \beta_{il}, \beta_{ol}) \end{aligned} \tag{3.10}$$

In equation 3.10, the reserve factors of the sliding bushing and fixed bushing are not included; they are assumed to be satisfied. This assumption is verified in subsection 3.6.2. Their volume v_{sb} and v_{fb} must be included, as they contribute to the total objective and depend on the outer lug thickness. Two additional assumptions are made to achieve this simpler problem formulation:

1. The objective has been reduced to the volume v, because it is assumed that once the materials have been selected, both the cost c and mass m increase if the volume increase, for each component:

$$\frac{\partial m_k}{\partial v_k} > 0 \qquad \frac{\partial c_k}{\partial v_k} > 0 \qquad \text{with} \qquad k \in [\text{bolt, inner lug, outer lug, sliding bushing, fixed bushing}] \tag{3.11}$$

And therefore, for the total objective as well:

$$\frac{\partial m}{\partial v} > 0 \qquad \frac{\partial c}{\partial v} > 0 \tag{3.12}$$

2. The lengths of the bolt and of the outer lug are assumed continuous during the optimization, and are then rounded off to their upper bound once it is completed:

$$l_{ol}^* \Rightarrow l_{ol} = l \in [l_{fb,1}, \dots, l_{fb,n}], \quad \text{with} \quad \min(l_{ol}^* - l) \quad \text{s.t} \quad l \geq l_{ol}^* \tag{3.13}$$

$$l_{bolt}^* \Rightarrow l_{bolt} = l \in [l_{bolt,1}, \dots, l_{bolt,n}], \quad \text{with} \quad \min(l_{bolt}^* - l) \quad \text{s.t} \quad l \geq l_{bolt}^* \tag{3.14}$$

   Where $[l_{fb,1}, \dots, l_{fb,n}]$ are the possible lengths of the fixed bushing, which depends on the selected fixed bushing standard and diameter code. Similarly, $[l_{bolt,1}, \dots, l_{bolt,n}]$ are the possible bolt lengths, which also depend on the bolt standard and diameter code.

Those assumptions are discussed in the section 3.6. Figures 3.10 and 3.11 show the convergence of the second phase of the search using the SLSQP algorithm.

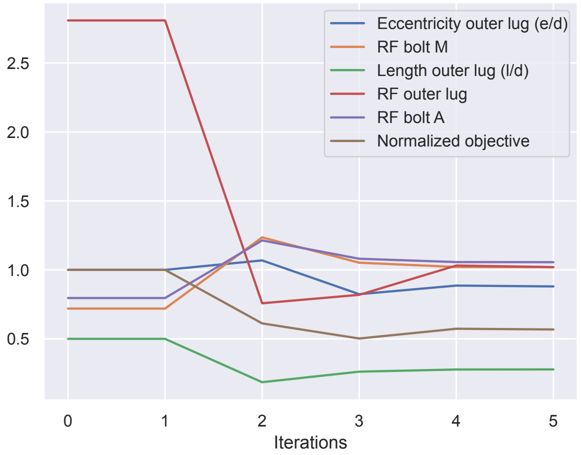


Figure 3.10: Convergence of the SLSQP algorithm. The objective $v = v_{ol} + v_{il} + v_{bolt} + v_{sb} + v_{fb}$ can be reduced by almost 50%.

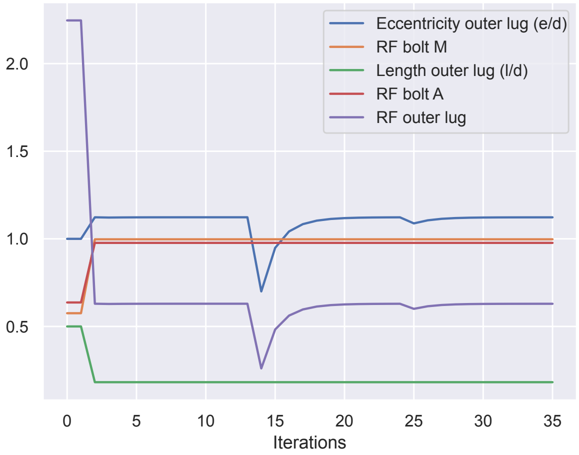


Figure 3.11: The SLSQP fails to converge because the inequality constraints $RF_{bolt,A} \geq 1$, $RF_{ol} \geq 1$, $RF_{bolt,M} \geq 1$ cannot be satisfied at the same time. The algorithm returns a inequality incompatibility constraint error.

Figure 3.10 shows a successful operation of the algorithm. Figure 3.11, however, shows that the optimization problem can sometimes not be solved. The only way to increase the allowable load on the bolt is to reduce the thickness of the outer lug t_{ol}. Therefore, if the reserve factor of the bolt is unsatisfactory for $t_{ol} = t_{ol}^L$, the algorithm will return an inequality constraint incompatibility error.

3.5.2. Random Tree Search (RTS)

The RTS method is a 2-step method that solves the optimization problem formulated in equation 3.6. This method chooses randomly standard and diameter codes so that the fretting and fitting exclusion is done as early as possible; then runs a gradient-based optimization to determine the continuous parts of the assembly. With respect to the literature, this algorithm can be classified as an iterative-depth, uninformed search.

The results of the RTS are presented in figure 3.12 for one load case. Out of all the assemblies generated, most are over-sized. The time spent checking for the feasibility of the assembly at run-time is negligible compared to the time spent optimizing the machined variables, but testing the same combinations potentially several times seems inefficient nonetheless.

The RTS is more effective than the GA, as it prunes larger parts of the design space. However, the method is not **optimal** because it does not include any convergence mechanism. Therefore, it is not **efficient** either. Scalability cannot be discussed, as neither of the previous requirements have been met. Although this implementation is not successful, it is a first step towards a more elaborated stochastic algorithm. This algorithm

```
Input: Maximum number of iterations $N_{max}$;
$feasible\_hinges = [\,]$;
while $n_{results} < N_{max}$ do
    Choose $n_{bolt}, n_{bearing}, n_{mat,ol}, n_{mat,il}$ randomly;
    if fretting then
        Prune design subset;
    Choose $D_{bolt}, D_{bearing}, D_{mat,il}, D_{mat,ol}$ randomly;
    if not fitting then
        Prune design subset;
    Precheck reserve factors;
    Setup and run SLSQP hinge optimization;
    if success then
        $feasible\_hinges$.append($assembly$);
    else
        Prune design subset;
return feasible_hinges;
```

Algorithm 3.3: The Random Tree Search (RTS) randomly selects the integer design variables checks for fretting and fitting as soon as possible. It limits the computation of unfeasible hinge designs encountered by the GA.

resembles one iteration of a population-based algorithm. The idea could be taken further but was abandoned in favor of the following search methods. The optimization framework presented in chapter 8 explains how this search could easily be used to develop more elaborated stochastic methods.

3.5.3. Best First Search (BFS)

The BFS is the simplest variation of the 2-step methods. Instead of checking for fitting at run-time, the fitting combinations are pre-processed, as described in 3.5.1. With respect to the literature, this algorithm is an uninformed greedy best-first search.

```
Parameters: $mat_{il}, mat_{ol}, F, \alpha, \theta$;
Input: Sorting criteria $(c_1, c_2, c_3)$, ;
Output: Feasible hinge assemblies;
Create feasible bolt/bearing combinations table;
Sort combinations by criteria $c_1, c_2, c_3$;
for combination in sorted set do
    Pre-check reserve factors;
    Solve optimization problem 3.10;
    if success then
        return hinge assembly;
```

Algorithm 3.4: The Best First Search (BFS) is the simplest version of the 2-step search methods. Depending on the chosen sorting criteria c_1, c_2, c_3, the search picks the smallest, cheapest or lightest components. The algorithm stops at the first successful continuous optimization.

In the first phase, the combinations are looked up linearly. Conservative tests are made to see if the components can carry the loads. If the tests are passed, an SQP optimization (phase two) is launched. If the optimization is successful, then its result is returned.

In figure 3.12, the solutions of the two search algorithms are presented in terms of cost and mass. For the RTS, the number of iterations is limited to 50. For the best-first search, the bolt radius and bearing mass are used as sorting criteria. As expected, the BFS yields much better results than the RTS, both in terms of mass and cost; the best hinges are not even reached by the RTS. It is not so surprising, knowing that there were more than 300 combinations and that the RTS was stopped after 50 iterations.

However, the problem is that the increasing the radius will also change the rest of the hinge assembly. The underlying question, already raised in [18], is:

Will the first feasible bolt (by whichever criteria we sorted the set of combination) yield the best hinge assembly?

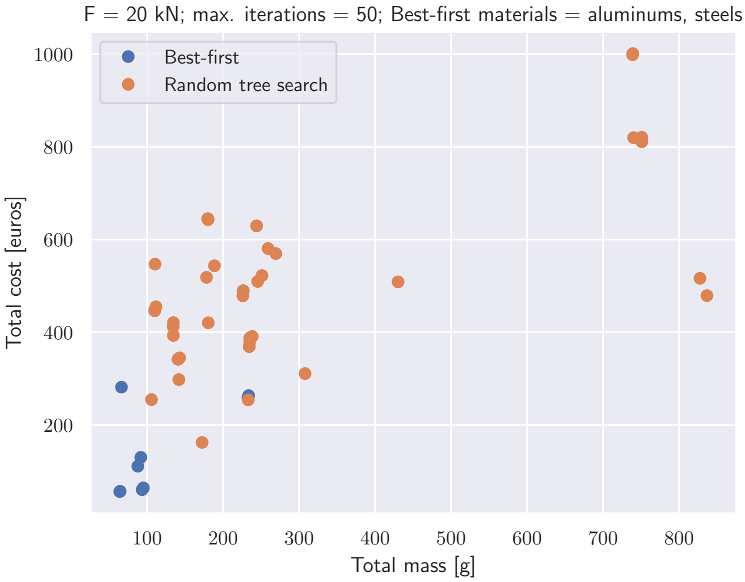


Figure 3.12: Hinges from the Best First Search outperform the hinges from the Random Tree Search, both in terms of mass and cost.

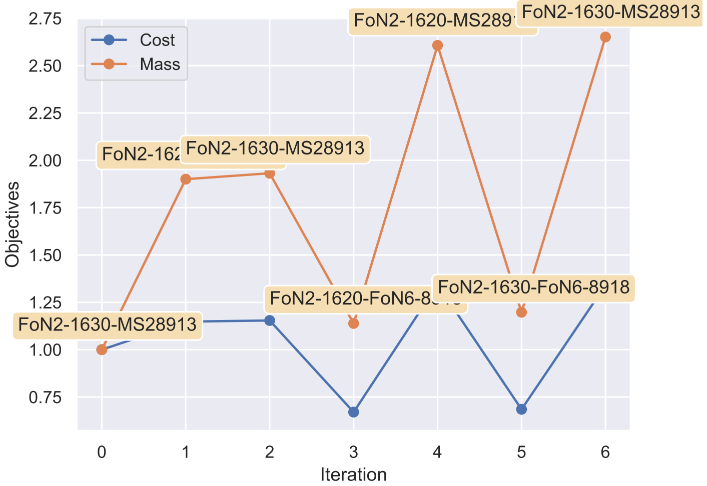


Figure 3.13: Cost and mass of the first seven iterations of an exhaustive search. The yellow labels at each iteration represent the bolt and bearing standard of the hinge. Although the cost and weight are increasing, the results at iteration $n = 3$ and $n = 5$ are cheaper than the BFS result ($n = 0$).

Optimality The general trend is indeed favorable to the assumption that the first result will be the optimal one: if $r_{bolt} \nearrow$, then $\mathrm{RF}_{bolt} \nearrow$, $m_{bolt} \nearrow$ and $c_{bolt} \nearrow$. The further away in table, the more likely it is that the bolt will carry the required load, but the heavier and more expensive the bolt will be.

However, this trend is not monotonic. For instance, if a slightly greater diameter with a different material could be worth trying, or perhaps the chosen diameter only yielded one bearing standard that is very expensive. Moreover, even if there was a single parameter from which the combinations could be ordered, it would not satisfy the multi-objective nature of the problem, as the BFS only yields one result.

A simple way to confirm that the best-first is not Pareto-optimal is to compare it to an exhaustive search. On figure 3.13, the iteration 0 corresponds to the best-first result, and the following iterations are the combinations that were skipped because they were assumed worse. We can see that the hinge assembly found in the iterations 3 and 5 could constitute interesting design choices, as they are lighter than the best-first result. Likewise, the changing the sorting criteria yields different results, and there is not a single best set of criteria to choose. Therefore, the best first search **optimality** is better than that of the RTS, but it is **not Pareto-optimal**.

Scalability and efficiency The scalability is excellent, as the run-time of the algorithm does not scale with the size of the design space. The efficiency is excellent too, as only one result is explored.

3.5.4. Informed Linear Search (ILS)

The following subsection details the implementation of the Informed Linear Search (ILS). Like the BFS and RTS, this search is a 2-step method. The ILS data workflow is given in the XDSM matrix in figure 3.14.

The first phase is similar to the BFS first phase; the combinations table is created with the criteria $(c_1, c_2, c_3) = (r_{bolt}, \sigma_{bolt}, m_{be})$ However, the ILS does not stop once a combination has been successfully optimized. Instead, the optimization result is stored, along its bolt and bearing standard. For each subsequent hinge with the same type of bolt and bearing, the following condition is checked:

$$\forall\ i,k, \quad \Delta x_i \cdot \frac{\partial f_k}{\partial x_i} > 0 \qquad \Rightarrow \qquad \Delta f_k < 0 \tag{3.15}$$

Where f_k are the objectives and x_i the design variables. Equation 3.15 is equivalent so saying that if all the design variables (x_i) that have a negative (respectively positive) impact on the objectives $(\frac{\partial f_k}{\partial x_i})$, then if all those variables are increased (respectively decreased), then the new objective will be worse, i.e., $\Delta f_k < 0$. Essentially, equation 3.15 is a usage of the gradient information to determine whether or not the next point should be expanded. In that sense, it is similar to a gradient-based search method, but more straightforward, as the output is binary instead of a search direction and step size length.

Equation 3.15 can be reduced to equation 3.16 in the case of the hinge assembly:

$$\begin{aligned} &\Delta x_i > 0 \\ &\mathbf{x} = [l_{be,in}, l_{be,out}, r_{be,out}, m_{bolt}, c_{bolt}, m_{be}, c_{be}] \end{aligned} \tag{3.16}$$

Because every element of **x** has a negative contribution to the objectives:

- If $l_{be,out}, r_{be,out}$ increase, then the size of the inner lug increase. As a previous optimal hinge assembly was already with the same bolt/bearing combination, and same lug material, this new inner lug will necessarily be non-optimal, as its volume is greater than necessary.
- If $l_{be,in}$ increases, then the bending arm increases and the allowable load of the bolt decreases. Moreover, the length, and therefore mass of the bolt increases.
- If $m_{bolt}, c_{bolt}, m_{be}, c_{be}$ increase, the mass and cost of the hinge assembly increase too.

In the algorithm 3.5, equation 3.16 is at line 7. If the condition 3.15 is true, then the previous best-first result is better than this next point.

The equation 3.16 is tested exhaustively **per bolt/bearing combination**. Usually, not all the aspects of the design problem can be accounted for by the optimization. OEM could request the Tier-1 manufacturer to only use a specific type of bearing (e.g., rolling instead of spherical), by convention or for life-cycle purposes. For instance, Fokker is currently using specialized bearings that have better characteristics but are more expensive. In table 3.3, this special case can be seen between the second and third row. Although the third row should be strictly ignored as it is worse by any of the measured metrics, it will still be explored and stored because it is a different standard. It is only after the optimization (see front-loading propositions) that the user restricts the design further.

The results of the Informed Linear Search (ILS) can be summarized with the criteria from subsection 3.2.1.

Efficiency and optimality In figure 3.15, the results from an exhaustive search and from an Informed linear Search are compared. In general, it can be difficult to assess whether the minimum of an optimization problem has been found. Fortunately, in our case the number of combinations is limited and an exhaustive search is still possible. The ILS performed as intended and found the optimal combinations, for each bearing-bolt-fixed bushing-nut combination. As a conclusion, the ILS is both **Pareto-optimal** and **efficient**.

```
Input: $\alpha, \theta, F, g, mat_{il}, mat_{ol}$ ;
Output: Best hinge assemblies;
Create feasible bolt/bearing combinations;
Sort combinations by pre-defined criteria $[r_{bolt}, \sigma_{bolt}, m_{be}]$;
for combination in sorted set do
    if combination in analyzed combinations then
        if $\forall\ i,\ \Delta x_i > 0$ then
            continue;
    Pre-check reserve factors;
    Size fixed bushing, sliding bushing, nut;
    if new cost or weight is better than previous assembly then
        Save new assembly;
    if new cost and weight are better than previous assembly then
        Save new assembly, discard old one;
    else
        Discard new assembly;
return best hinge assemblies;
```

Algorithm 3.5: The Informed Linear Search (ILS) memorizes the hinge assemblies already explored, so that only potentially interesting combinations are sized.

Scalability The ILS is also **scalable** in terms of the number of combinations considered, as no SQP optimization will be launched if the combination is not deemed interesting. The time complexity of the ILS is linear ($O(n)$) at worst and constant ($O(1)$) at best. Section 3.6 gives further details on the results obtained using this search method, notably regarding the assumptions and regarding the run-time scaling in subsection 3.6.4.

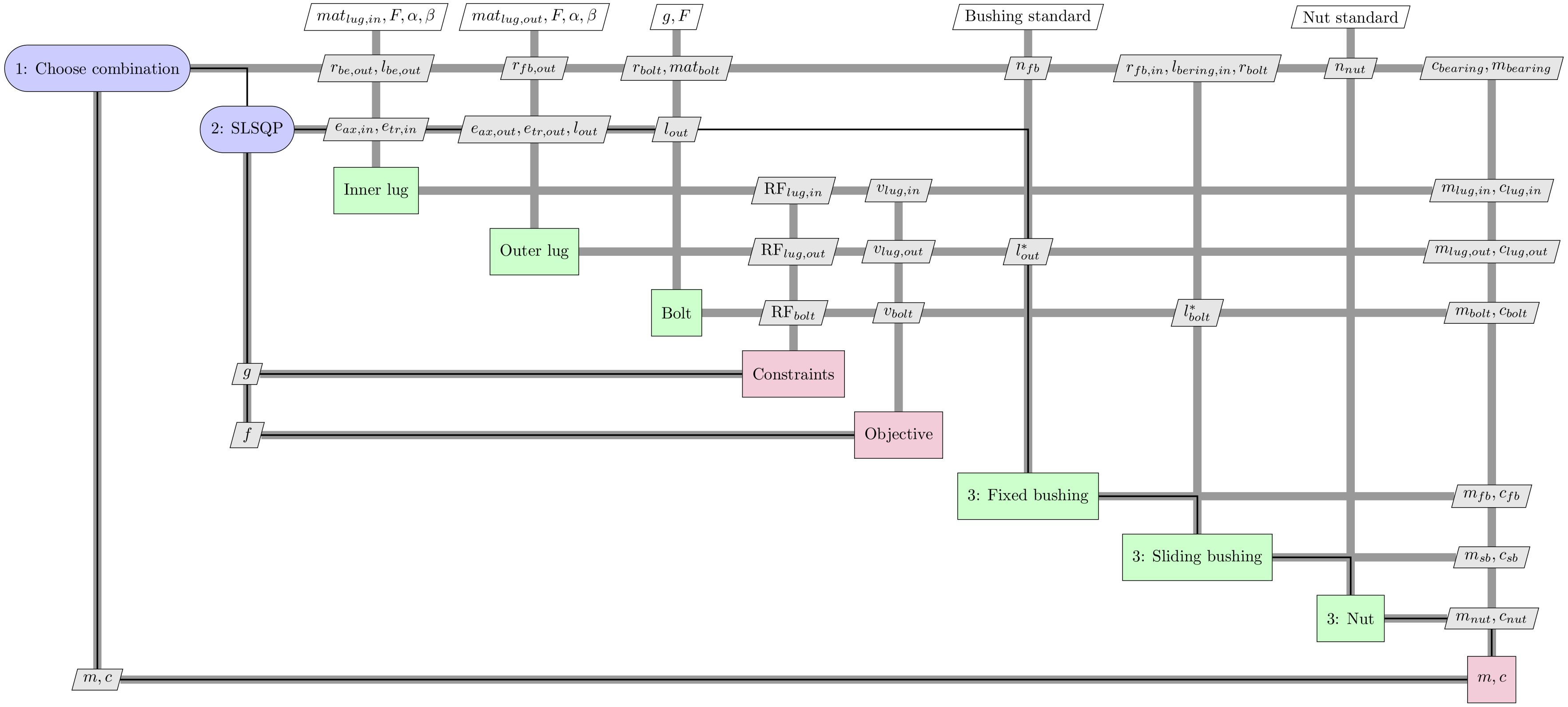


Figure 3.14: XDSM matrix of the Informed Linear Search (ILS). The search starts with the phase 1 (*1: Choose combination* block), which fixes many parameters of the assembly and enables the phase 2 (*2: SLSQP* block) to be carried out.

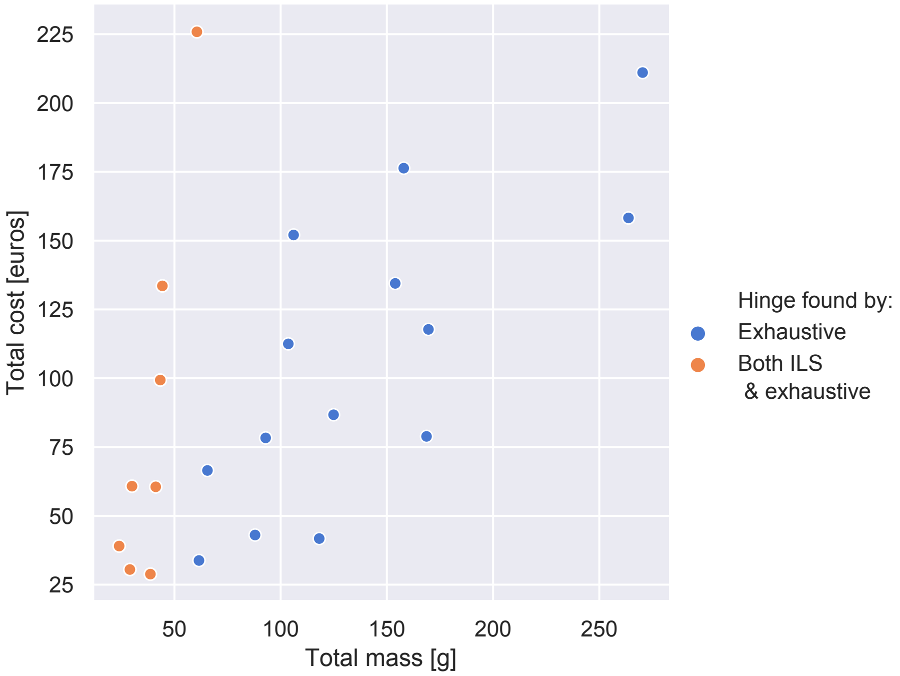


Figure 3.15: Hinge assemblies found by an exhaustive search and by the ILS. The ILS found the best hinges per unique combinations of bolt and bearing (orange), and ignored the non-optimal hinges (blue).

3.6. Results of the Informed Linear Search

The Informed Linear Search is used to describe the optimal hinges (subsection 3.6.1) and then to study the effects of different parameters, such as the load F and the outer lug material strength (subsection 3.6.2).

3.6.1. Interpretation of one load case

In figure 3.16, the results of the ILS are shown for the following parameters:

$$\begin{aligned} F &= 10\ kN \\ \theta &= 15° \\ \alpha &= (0°, 30°) \\ mat_{il} &= mat_{ol} = \text{alu1} \end{aligned} \tag{3.17}$$

The load case 3.17 corresponds to a documented load case of a business jet at Fokker.

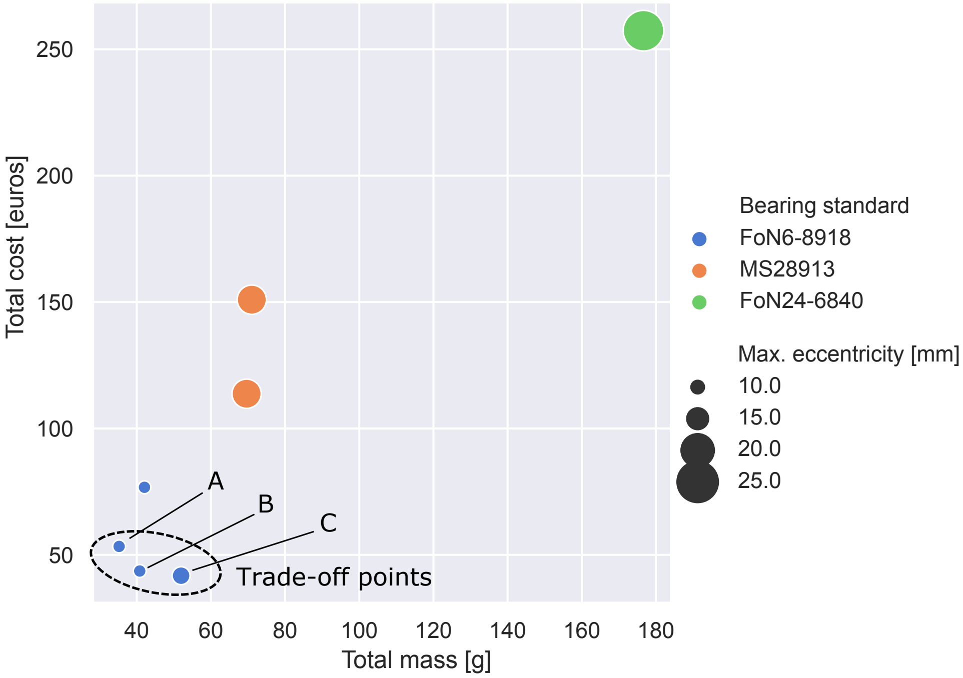


Figure 3.16: ILS results for $F = 10\ kN$, $\alpha = (0, 30°)$, $\theta = 15°$ aluminium inner and outer lugs with $\beta = 45°$. Because the relationship between the bearing cost and the bearing size is not monotonic, a trade-off between cost and mass can be made. Many assemblies are more expensive and heavier than the ones circled, but they are still part of the output of the ILS as they use a different bolt or bearing combination.

Figure 3.16 presents the results of the ILS for this load case. There are significant differences between the hinge assemblies regarding cost, mass and size. The four hinge assemblies using the bearing standard FoN6-8918 are lighter, cheaper and smaller than the other four. This observation is a confirmation that the ILS is working as intended. The best hinges were explored exhaustively in terms of bolt-bearing combination, even though some combinations resulted in heavier and more expensive hinges.

The possible trade-offs between cost and mass are underlined in figure 3.16. It shows the multi-objective character of the search: the lightest assembly is using a smaller bearing, but that happens to be more expensive than the next, more common sized one. As a result, the hinge assembly (A) is better concerning mass and size. However, the hinge assemblies (B) and (C) are also presented because despite being bigger and heavier.

These trade-off points are especially relevant in the context of the rudder-fin interface. In designing the rudder-fin interface, it is common to order standard parts in larger quantities to reduce their cost. Therefore, some of the hinges of of the interface are over-sized.

The hinge assemblies (A), (B) and (C) in figure 3.16 are also represented in figure 3.17.

Figure 3.18 shows the cost and mass of the different types of bearings considered for the optimization. Figures 3.16 and 3.18 show the same clusters per bearing type. It shows that the characteristics of the hinge

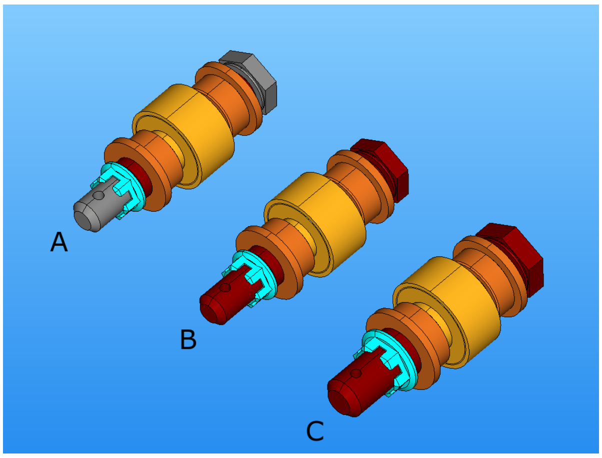


Figure 3.17: Visualization of the geometry of the pareto-optimal hinges. The cost, mass and size of the hinges A, B, C is also depicted in figure 3.16. The hinges (A) and (B) have the same bolt radius and bearing, but the first bolt is made of titanium while the second one is made of steel. Hinge (C) has the same bolt type and bearing type as (B), but the bigger bearing used in (C) is actually cheaper than the smaller, less common bearing in (B).

are heavily correlated with those of the bearings.

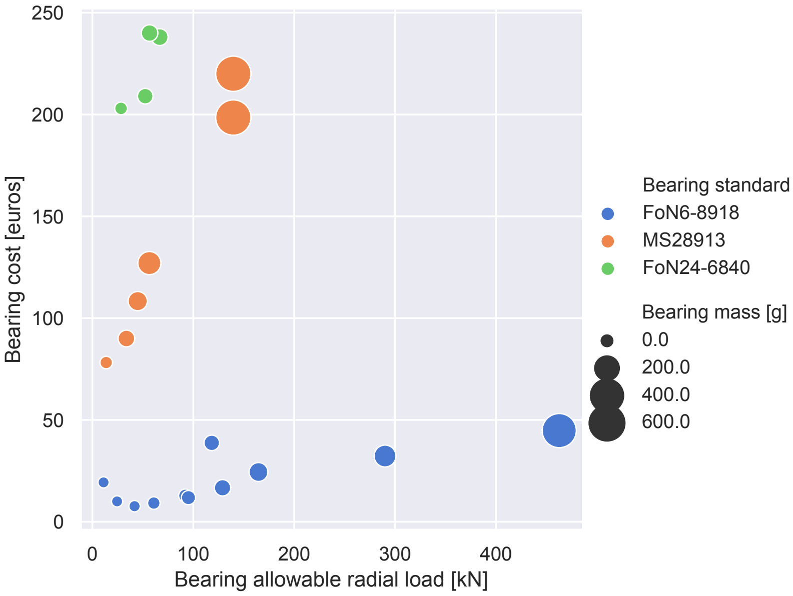


Figure 3.18: Three types of bearings are considered for the optimization. The relation between the bearing cost, mass and allowable load are not monotonic and vary significantly depending on the bearing type. Those characteristics are heavily correlated in the ILS results shown in figure 3.16.

3.6.2. Parametric studies

The previous subsection presented a single optimization run, for the load case given in equation 3.17. This subsection runs several ILS for different parameters.

The first and most important parameter is the load magnitude F. Then the parameters, such as the variables associated to the brackets and the inputs of the ILS (nut standard, bushing materials) are varied. Therefore, this study also doubles as the evaluation of the assumptions made in the first part of the ILS. If the influence of the parameters is non-negligible, then the assumptions will have to be revised.

Load magnitude influence

Figure 3.19 shows the results of several Informed Linear Search for different loads $F \in [5kN, 45kN]$.

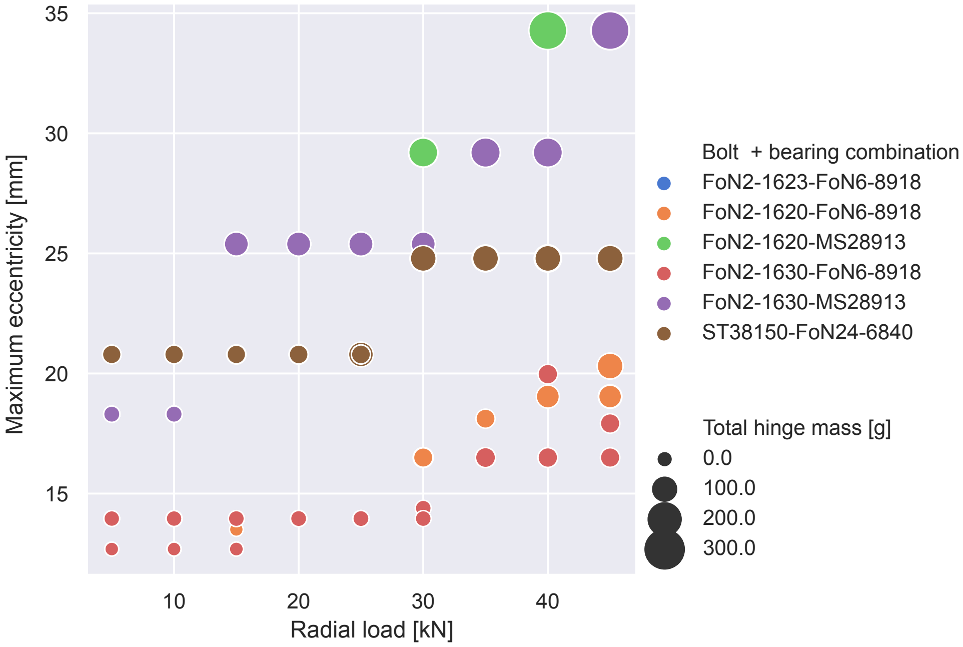


Figure 3.19: Hinge assemblies found by the ILS for $F \in [5kN, 45kN]$. As the load increases, the size of the components and of the lugs increases, therefore the maximum eccentricity of the hinge assembly increases.

Figure 3.19 captures the two main inputs of the front-loaded model: the radial load magnitude F and the maximum eccentricity e_{max}. The color legend indicates, in one variable, the selected bearing and bolt standards, while the size of the points indicates the total mass of the hinge. The general trend is that the higher the load, the heavier and larger the hinge becomes.

For a given load, several combinations are possible. Some hinge assemblies are optimal for a range of loads (e.g., brown combination from 5 to 25 kN), because it is over-sized for the lowest load. However, these hinges are still part of the output by the search algorithm because it is exhaustive in terms of combinations.

Lug material

The lug cost and mass influence that of the hinge directly, as they are included in the objective. It also influences the objective indirectly. The thicker the outer lug is, the longer the bolt, sliding bushing and fixed bushing are going to be.

The following paragraph provides an explanation on the influence of the lug material strength and lug thickness. The second phase of the ILS tries to minimize the volume:

$$\min(v) \qquad \text{with} \qquad v = v_{il} + v_{ol} + v_{sb} + v_{fb} \tag{3.18}$$

In general, the allowable load of the outer lug will vary with its cross-sectional area and its material strength:

$$F_{allowable,ol} \propto A \cdot \sigma \tag{3.19}$$

Equation 3.19 is simplified compared to the original stress analysis (described in subsection 2.3.3), which takes into account many more material properties and three different failure modes. This simple analysis does not include any of those considerations but enables to explain some of the results on a higher level. From the analytical volumes, the objectives are proportional to the thickness of the lug t and lug eccentricity e:

$$V_{ol} \propto t_{ol} \cdot e_{tr,ol} \cdot e_{ax,ol} \tag{3.20}$$

$$V_{bolt}, V_{sb}, V_{fb} \propto t \tag{3.21}$$

However, the allowable force will vary proportionally to the cross-section area A:

$$F_{allowable,ol} \propto A \cdot \sigma = t_{ol} \cdot e \tag{3.22}$$

Therefore, with respect to the lug, it is beneficial to have a thicker lug with a small eccentricity rather than a thin lug with a larger eccentricity. A thicker lug is preferable as long as the bolt can handle the added bending moment.

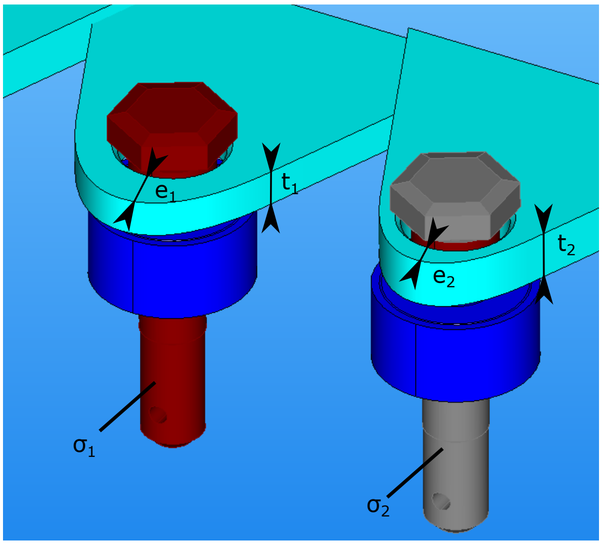


Figure 3.20: Comparison of two Pareto-optimal hinge assemblies (bearing, bolt and outer lug shown). The two assemblies use the same bearing and same outer lug material, but bolts with different material strength. The hinge on the left has the weakest bolt material ($\sigma_1 < \sigma_2$), which results in the highest outer lug eccentricity ($e_1 > e_2$) required to have both $RF_{ol} > 1$ and $RF_{bolt} > 1$

Figures 3.21 and 3.22 show that, in general, it is favorable for the eccentricity decrease instead of the thickness. This is what is happening in figure 3.21. The stronger the material, the lesser the cross-section area required to carry the loads. The thickness ratio needs to be at most $t/d = 0.4$ to avoid violating the reserve factor requirement of the bolt. Therefore, as a stronger material is used, eccentricity decreases rather than the thickness as it is more beneficial to the objective.

Frequently, for small load magnitudes, the lower bound of the eccentricity ($e/d = 0.7$ - a lower ratio and the hole and outer curvature would intersect) is reached, as in figure 3.22. Above the threshold of $e/d = 0.7$, the increasing material strength enables the thickness of the lug to be decreased without violating the lug reserve factor.

Nut standard

In the first phase of the ILS, the nut standard is a parameter. For each bolt combination, the appropriate diameter code for the FoN2-2211 standard is selected. To study the effect of the nut standard, the results of the ILS are compared for the three nut standards in the database. Table 3.4 summarizes the results of the experiments. The aluminium nuts are cheaper than the steel one. The steel nut is more expensive and heavier than the aluminum ones. As a consequence, it represents a higher percentage of the total mass and cost of the hinge assembly.

The nut accounts for less than 6% of the total hinge cost and mass. Moreover, the choice of the nut standard arises from considerations other than the mass and cost. For example, in figure 3.23, the two nut standards on the right are longer and could, therefore, be torqued further down the thread and still be secured with a cotter pin.

As a conclusion, the exclusion of the nut from the optimization is justified. The nut standard is essential, but it is not necessary to explore every nut standard for each type of bolt. With the current database, the best

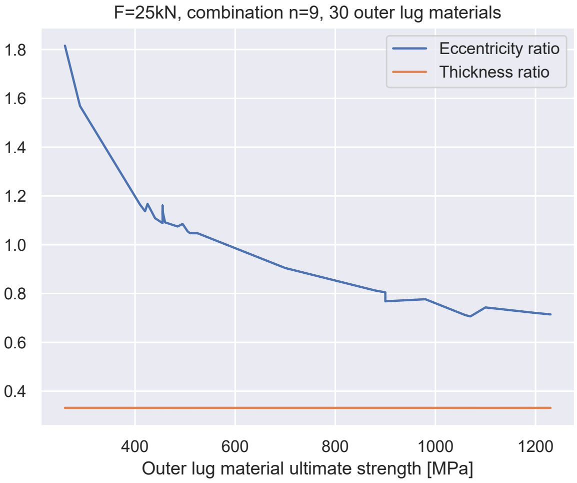


Figure 3.21: Outer lug eccentricity and thickness as a function of the outer lug material strength, $F = 25\ kN$. As the strength increases, the required lug cross-section can decrease, which is translated into an eccentricity ratio decrease.

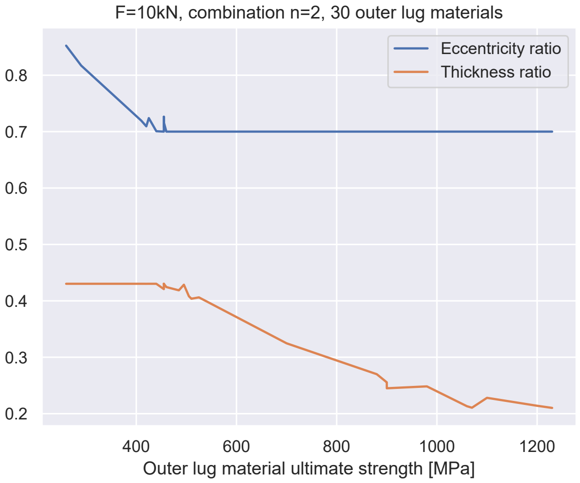


Figure 3.22: Outer lug eccentricity and thickness as a function of the outer lug material strength, $F = 10\ kN$. The same behavior from figure 3.21 is noticed until $\sigma = 450\ MPa$. Afterwards, the eccentricity lower bound is reached, therefore, the thickness of lug decreases.

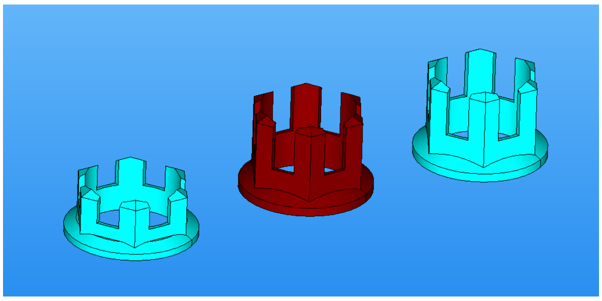


Figure 3.23: The geometry of the three nut standards considered for the ILS is very similar. The cyan color indicates a steel nut and the red color indicates a steel nut.

	FoN2-2211	FoN2-2212	FoN2-2213
c_{nut} [euros]	0.86	2.71	0.92
m_{nut} [g]	1.13	1.54	1.54
c_{nut}/c_{total}	1.98 %	5.97 %	2.12 %
m_{nut}/m_{total}	3.21 %	4.32 %	3.74 %

Table 3.4: Influence of the nut standard. The cost and mass of selected nut are given. The percentage represents the maximum value (out of the 5 ILS results).

nut standard in terms of cost and mass is the FoN2-2211. If the nut standard is to be included in the ILS, the combination tree would need to be updated to contain all the nut standards instead of the FoN-2211.

Bushing materials

During the Informed Linear Search, the materials of the sliding bushing and fixed bushings are parameters. Two arguments support this simplification:

- As evoked previously, the reserve factor of the sliding bushing was satisfactory for the chosen lower bound $r_{sb}^L = 0.5\ mm$, even for the materials with the lowest ultimate tensile strength. Table 3.5 confirms this result. For every ILS results with $5kN < F < 45kN$, $\min(RF_{fb}) = 2.31 > 1$ and $\min(RF_{sb}) = 1.91 > 1$.
- The mass and cost of the sliding bushing does not account for a large part of the hinge assembly mass and cost, as seen of table 3.5. Consequently, their importance rather lies in the constraint they exert

	Fixed bushing			Sliding bushing		
	m_{fb}/m_{total}	c_{fb}/c_{total}	RF	m_{sb}/m_{total}	c_{sb}/c_{total}	RF
Average	3.46 %	8.88 %	8.47	1.85 %	3.59 %	9.41
Minimum	1.05 %	3.48 %	2.31	0.57 %	1.28 %	1.91
Maximum	7.43 %	19.51 %	28.88	4.22 %	7.22 %	32.16

Table 3.5: Reserve factors of the sliding bushing, for several ILS results with $5kN < F < 45kN$. The reserve factor is always above one, which confirms the simplification used for the tree search.

on the materials of the neighbor components. As the material of the lug is not a design variable but a parameter, the possible materials of the bushings are known in advance.

The maximum contributions of the bushings in table 3.5 are $c_{fb}/c_{total} = 0.19$ and $c_{sb}/c_{total} = 0.07$. The value $c_{fb}/c_{total} = 0.19$ corresponds to the smallest load case ($f = 5\ kN$) for which the minimum fixed bushing diameter is already oversized with respect to the smallest bolt and bearing.

Rather, it is the life-time of the sliding bushing in the rudder installation (i.e., how often should the sliding bushing be replaced) that should be taken into account in the optimization. However, this figure of merit was not evaluated. Therefore, this design variable is left as a parameter for the designer to select.

3.6.3. Discussion on the assumptions

During the second phase of the ILS, the objective is reduced to the volume. The cost assumption used in equation 3.23 is not correct in the general case.

$$\frac{\partial c}{\partial v} > 0 \qquad \text{with} \qquad v = v_{il} + v_{sb} + v_{fb} + v_{ol} + v_{bolt} \tag{3.23}$$

Two things can happen:

Machining cost In general, the machining cost could increase as the volume decrease. As explained in section 2.4, the cost varies with the manufacturing time ($t_{manufacturing}$ in equation 2.17), which increases if the amount of material to machine increases. In the cost implementation of the Hinge Generator, however, the dimension on which the cost estimation is based (x) is equal to the length of the component. Therefore, this phenomenon could not be observed with the current implementation.

Standard parts cost Just as the cost of the bearing is not monotonic, the increase in bolt price due to the bolt length is not monotonic either. This assumption could not be verified, as the cost data for each bolt was not known during the project. The same remark applies for the fixed bushing. An analysis of the prices of the bolt per length range is required to evaluate this assumption.

The parameters fixed during the ILS (nut standard, lug materials, bushing materials) are varied to study their influence on the total cost and mass of the hinge. Those parameters are essential but should be designed separately from the search, as they either depend on the bracket geometry or have functional objectives (reduce) more important than their mass and cost contribution. In general, thicker outer lugs are more favorable.

3.6.4. Scalability with the number of standard parts

The number of combinations grows rapidly with the number of standard components considered. The runtime of the algorithm can be split into two parts:

$$t = t_{overhead} + t_{optimization} \tag{3.24}$$

The computation time required to create the combination table $t_{overhead}$ scales linearly with the number of components combinations, because the combinations have to be ordered with a sorting algorithm. The optimization time $t_{optimization}$ is almost independent to the number of combinations because the ILS will avoid the phase 2 (SQP optimization) if the combination does not seem promising. This is an improvement compared to the quasi-exhaustive search implemented in [18], whose run-time would increase linearly with the number of combinations because of its exhaustive search.

Figure 3.24 shows that the scalability of the ILS is much better than that of an exhaustive search.

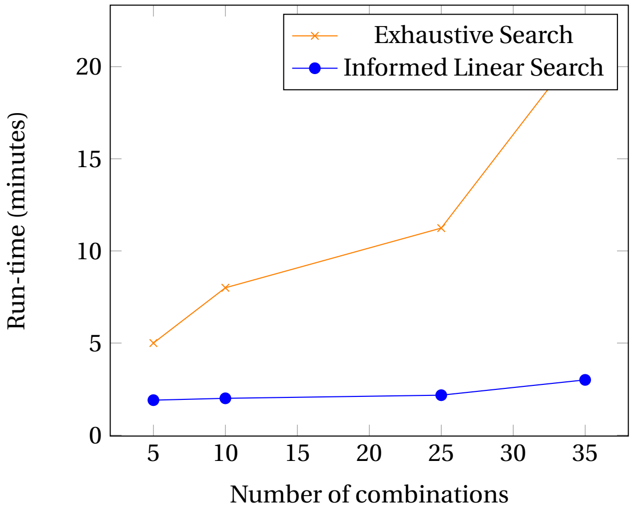


Figure 3.24: Informed Linear Search run-time and exhaustive search run-time as a function of the number of combinations (i.e., the number of rows in table 3.3) considered. As the number of combination increases, the exhaustive search run-time increases linearly, as it solves the sizing optimization problem (equation 3.10) for each combination. For the ILS, the run-time almost does not scale with the number of combinations, because the optimization problem is only solved if the combination is potentially better than previous ones.

3.7. Discussions on the developed methods

Name	**Method**		**Performance**			
	Phase 1 (Determine $\mathbf{x}^i$)	Phase 2 (Determine $\mathbf{x}^c$)	Converges	Optimal	run-time	Scalable
GA	Determine $\mathbf{x} = (\mathbf{x}^i, \mathbf{x}^c)$ through mutation, crossover, selection		No	No	-	-
RTS	Random tree search	SLSQP	No	No	-	No
Exhaustive	Pre-processing + linear exhaustive	SLSQP	Yes	Yes	40 min	No
BF	Pre-processing + greedy linear	SLSQP	Yes	No	20 seconds	Yes
ILS	Pre-processing + informed linear	SLSQP	Yes	Yes	2 min	Yes

Table 3.6: Summary of the developed methods for the hinge search problem. The run-time is given for a single load case.

Existing methods were not satisfactory from a scalability and run-time perspective. A new method had to be developed, with the updated `Hinge Generator` KBE model. The presence of both standard and machined parts makes this search difficult to tackle with out of the box meta-heuristic optimization algorithms.

The developed methods are summarized and compared in table 3.6. The use of a meta-heuristic was preferred because of its ease of implementation but did not yield fruitful results. Therefore, to solve the problem in a shorter amount of time, ad-hoc search algorithms were developed. The separation of the discrete and continuous parts of the search appeared as a good strategy. The RTS solved the problem of exploring too many unfeasible solutions by exploring the combinations in a tree search, so that more combinations can be pruned. However, the RTS cannot be used by itself as the tree search (phase 1) has no convergence mechanism.

To explore the tree structure in a more ordered manner, the standard parts combinations are pre-processed. Exploring every one of those fitting combination (exhaustive search) is feasible, and would make sure that the search is complete, but it is not efficient. On the opposite, stopping the search at a given criterion (best-first search) does not yield every interesting combination. The last developed method, the ILS, meets both ends. It is efficient, scalable and complete.

The Informed Linear Search yields several hinge assemblies, each being Pareto-optimal per bearing-bolt combination. The cost and mass of the hinge are highly correlated to the bearing cost and mass. Simple rolling bearings consistently yield the cheapest and lightest hinge assemblies. However, other results are also presented as a specific bolt-bearing combination could present a particular advantage compared to the go-to, cheaper and lighter bolt-bearing combination.

Although the achieved run-time is better than expected at the beginning of the project, using this optimization directly for the rudder-fin optimization is difficult. With a run-time of an average of two minutes, and assuming a GA that requires around $1.2 \cdot 10^6$ hinge optimizations, the rudder-fin optimization would still take at least 4.5 years. Chapter 4 proposes a solution to store and retrieve hinge optimization results so that an optimal hinge can be obtained almost instantly.

4
The hinge front-loading problem

The modeling (chapter 2) and optimization (chapter 3) are the two building blocks of the front-loading methodology. The last step is to integrate the optimization within a framework that enables the selection of an optimal hinge.

Section 4.1 defines the front-loading framework. Then, the developed front-loading methods are presented. The tool developed to answer the data visualization use-case is described in section 4.2. Subsection 4.2.1 presents the visualization framework. Then, a sample use-case of the tool is given in subsection 4.2.2. The hinge selection method developed to enable rudder-fin optimization is presented in section 4.3. The selection problem is presented in subsection 4.3.1. The selection method is presented in subsection 4.3.2, with details on the chosen input density in subsection 4.3.3. Subsection 4.3.5 evaluates the performance of the hinge selection method, concerning its efficiency and its effectiveness. Lastly, subsection 4.3.6 summarizes the strengths and weaknesses of the selection method. The discussions in section 4.4 argue about the benefits and drawbacks of this front-loading approach. The implementation of this framework is detailed in chapter 9, in the second part of this report.

4.1. Framework to enable hinge front-loading

The front-loading methodology requires a framework to enable the selection of the optimal product *off-the-shelf.*

4.1.1. IDEALISM: a product development framework for enabling front-loading

At Fokker, the *Rudder Generator* KBE application was used in 2017 to demonstrate the front-loaded PDP during the European research project IDEALISM [34]. The front-loaded rudder design process was a key industrial test-case to demonstrate the feasibility of the IDEALISM framework. The objective of IDEALISM was to define a new integration framework and design languages standards, to enable the use of KBE to support MDO in the context of a front-loaded PDP. Figure 4.1 gives a high-level overview of the IDEALISM framework and its terminology.

The interaction between the *Workbench* and the *Engineering Library* in figure 4.1 highlights the iterative nature of this methodology. For instance, one might realize after filling the design space that the KBE model was not valid in a certain design space subset, for which the structural analysis was not valid. This would lead to the creation or adjustment of the tools and/or design rules of the *Engineering Library*.

The framework illustrated in figure 4.1 describes the whole product development framework, not the front-loading framework. Nonetheless, figure 4.1 is useful to show that the developed framework presented in the following subsection uses and fits some of the higher-level ideas developed in the IDEALISM framework.

4.1.2. Developed framework for hinge front-loading

This subsection gives an overview of the entire front-loading framework. The framework implementation need to answer the two research sub-questions:

1. *How to represent the Pareto-optimal hinges in a way that enables trade-off decisions?*

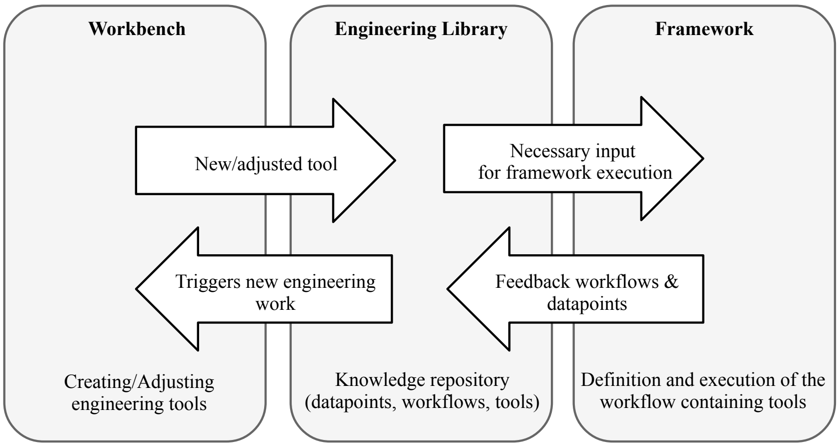

Workbench
Engineering Library
Framework
New/adjusted tool
Necessary input for framework execution
Triggers new engineering work
Feedback workflows & datapoints
Creating/Adjusting engineering tools
Knowledge repository (datapoints, workflows, tools)
Definition and execution of the workflow containing tools

Figure 4.1: IDEALISM framework (adapted from [34]). The PDP starts in the *Workbench*, where the knowledge (e.g., design rules, KBE application) is first developed. Those elements are then stored in the *Engineering Library*, and then used in an optimization workflow in the *Framework* component.

2. *How to obtain Pareto-optimal hinges as quickly as possible? (< 10 ms)*

As explained in the introduction (subsection 1.4), the chosen approach is to perform a DOE and to select an optimization result in the database. If the requested point is not in the database (e.g., lightest hinge for $F = 17.2$ kN when only 17 kN and 18 kN are in the database), then a feasible optimal point with similar requirements (here $F = 18$ kN) is selected.

Therefore, the framework should enable the following:

- The creation and "filling" of a *front-loaded* database with optimal hinges for different requirements.
- The obtention of a near-optimal hinge for a user or external requirement, even if that requirement is missing from the database.
- A quick and effective way to visualize the information in the front-loaded database.

IDEALISM's idea of iteratively filling the *Engineering Library* with data points is used. The tools and methods developed to enable front-loading are represented in figure 4.2.

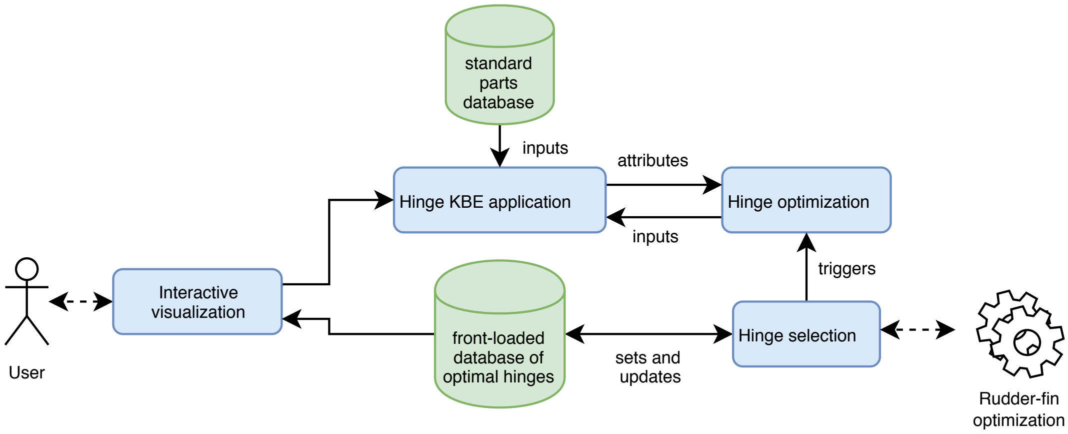


Figure 4.2: The main elements developed during this thesis (tools in blue and databases in green) are placed within the IDEALISM framework. The KBE application is used by the optimization, itself being used for the hinge selection. The hinge selection iteratively fills the front-loaded database, which can be easily visualized, to discover patters and validate results.

The key feature of the framework is to enable inter-operation of the tools and databases, as shown in figure 4.2. The first inter-operation is between the *Hinge KBE application* and the *Hinge optimization*, described in chapters 2 and 3. The optimization uses the KBE application to evaluate the attributes (cost, mass, etc.)

of a given hinge configuration. The optimization is used for different requirements. The results of each optimization are stored within the *front-loaded database*. The *Hinge selection* queries the front-loaded database. A query is a request for a specific hinge within the front-loaded database. For instance, a possible query could be the following:

Return the cheapest hinge for which $F = 26\ kN$, $e_{max} < 25\ mm$, bolt $\neq NAS6703 - 6704$

If the number of points in the front loaded database is insufficient to satisfy (e.g. it only contains optimal sets of hinges for $F = 15\ kN$ and the query is of $F = 26\ kN$), then a new optimization is *triggered*. Finally, the inspection of the front-loaded database is achieved with the *interactive visualization* tool.

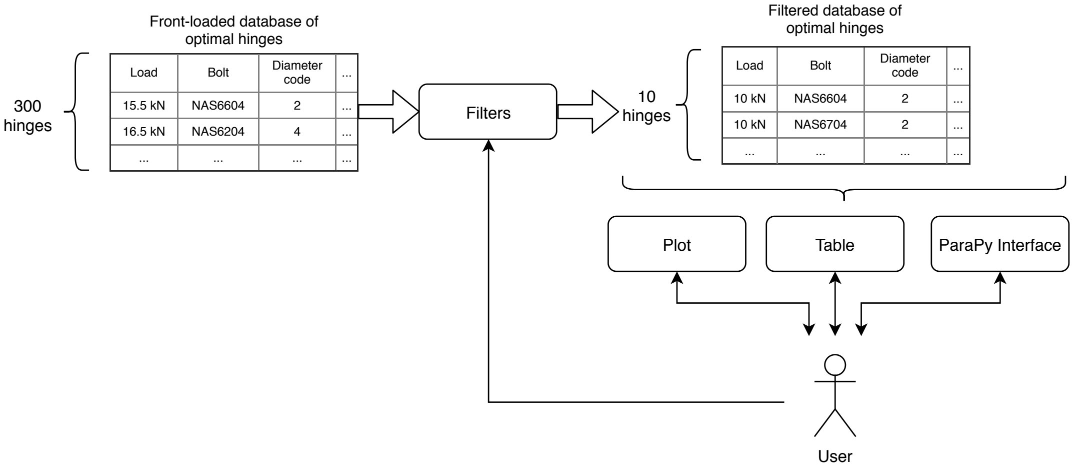
4.2. The data visualization tool

Data visualization is necessary to validate the front-loaded data set and is used as a support for collaborative work. The front-loaded hinge data set contains hundreds of optimal hinges for different requirements, each with around 30 variables that are potentially interesting to inspect. The data visualization part of the framework should enable the engineers and designers to quickly manage this large, complex set of data. Important attributes linked to different expertise and disciplines (Reserve factors, mass, cost) must all be visualized. The visualization needs to enable comparison between the attributes to facilitate trade-off decisions.

4.2.1. Presentation of the framework

Although each piece of information in the front-loaded data set could be simply visualized with a simple spreadsheet, it would be unpractical and time-consuming, particularly in collaborative meetings. In such an environment, the time frame is usually limited, which hardly allows for visualization of different sets of hinge variables considering the necessary manual steps using spreadsheets.

To reduce the time spent on those manual steps, an interactive interface is developed to enable the front-loaded data set to be easy to understand and to work with.


Front-loaded database of optimal hinges
300 hinges
Filters
10 hinges
Filtered database of optimal hinges
Plot
Table
ParaPy Interface
User

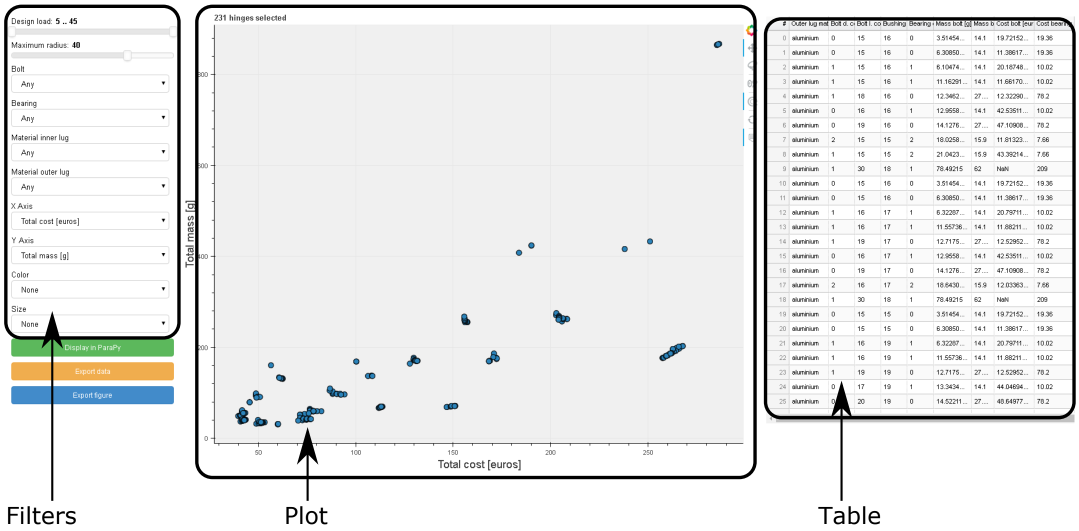
Load	Bolt	Diameter code	...
15.5 kN	NAS6604	2	...
16.5 kN	NAS6204	4	...
...	...	...	...

Load	Bolt	Diameter code	...
10 kN	NAS6604	2	...
10 kN	NAS6704	2	...
...	...	...	...

Figure 4.3: Data visualization framework. By itself, the *front-loaded database* contains too many hinges and variables to see any valuable information. The database is first filtered by the user (e.g., only display hinge for which $F = 10kN$). Then this subset of the database can be visualized through 3 tools: a plot, a table and the ParaPy interface. Each tool offers a different view and granularity on the data and can be interacted with, as indicated by the double arrows.

Figure 4.3 shows a simple representation of the data visualization framework. The framework enables to visualize data dynamically through the *Plot*, the *Table*, and to a higher-level of detail with the *ParaPy Interface*. The data can be queried, filtered and represented dynamically, allowing for a much easier and automated inspection process. It enables the user to change any dimension (X-axis, Y-axis, color, size) of a 2D *Plot*, as well as to filter out elements of the front-loaded database. Once a hinge assembly is selected, it can directly be opened in the KBE application through the ParaPy interface, to inspect and/or tweak the model. Therefore, this framework enables quick, detailed inspection of the front-loaded database.

Figure 4.4 shows the *Dynamic plot* part of the framework. On the left panel, the front-loaded database can be filtered (e.g. only display for $F < 30kN$ and aluminum lugs). The X axis, Y axis, color and size can be chosen from a given list of attributes.



Design load: **5 .. 45**

Maximum radius: **40**

Bolt: Any

Bearing: Any

Material inner lug: Any

Material outer lug: Any

X Axis: Total cost [euros]

Y Axis: Total mass [g]

Color: None

Size: None

231 hinges selected

#	Outer lug mat	Bolt d. c	Bolt l. co	Bushing	Bearing	Mass bolt [g]	Mass b	Cost bolt [eur	Cost bearin
0	aluminium	0	15	16	0	3.51454...	14.1	19.72152...	19.36
1	aluminium	0	15	16	0	6.30850...	14.1	11.38617...	19.36
2	aluminium	1	15	16	1	6.10474...	14.1	20.18748...	10.02
3	aluminium	1	15	16	1	11.16291...	14.1	11.66170...	10.02
4	aluminium	1	18	16	0	12.3462...	27....	12.32290...	78.2
5	aluminium	0	16	16	1	12.9558...	14.1	42.53511...	10.02
6	aluminium	0	19	16	0	14.1276...	27....	47.10908...	78.2
7	aluminium	2	15	15	2	18.0258...	15.9	11.81323...	7.66
8	aluminium	1	15	15	2	21.0423...	15.9	43.39214...	7.66
9	aluminium	1	30	18	1	78.49215	62	NaN	209
10	aluminium	0	15	16	0	3.51454...	14.1	19.72152...	19.36
11	aluminium	0	15	16	0	6.30850...	14.1	11.38617...	19.36
12	aluminium	1	16	17	1	6.32287...	14.1	20.79711...	10.02
13	aluminium	1	16	17	1	11.55736...	14.1	11.88211...	10.02
14	aluminium	1	19	17	0	12.7175...	27....	12.52952...	78.2
15	aluminium	0	16	17	1	12.9558...	14.1	42.53511...	10.02
16	aluminium	0	19	17	0	14.1276...	27....	47.10908...	78.2
17	aluminium	2	16	17	2	18.6430...	15.9	12.03363...	7.66
18	aluminium	1	30	18	1	78.49215	62	NaN	209
19	aluminium	0	15	15	0	3.51454...	14.1	19.72152...	19.36
20	aluminium	0	15	15	0	6.30850...	14.1	11.38617...	19.36
21	aluminium	1	16	19	1	6.32287...	14.1	20.79711...	10.02
22	aluminium	1	16	19	1	11.55736...	14.1	11.88211...	10.02
23	aluminium	1	19	19	0	12.7175...	27....	12.52952...	78.2
24	aluminium	0	17	19	1	13.3434...	14.1	44.04694...	10.02
25	aluminium	0	20	19	0	14.52211...	27....	48.64977...	78.2

Figure 4.4: The interactive data visualization is used to select a subset of the front-loaded database. The hinge total cost and mass are chosen as the X and Y axis. The hinge assemblies are colored as a function of their bolt standard.

To inspect and visualize more specific attributes of each hinge assemblies, the points on the plot can be selected and directly displayed in the `Hinge Generator` (as depicted in previous figures, for instance figure 2.25).

4.2.2. Sample use-case

This application *automates* the post-processing and data analysis tasks. Usually, these tasks would be carried out using Excel and plotting tools. The typical use-case consists of six steps, illustrated by figures 4.5 to 4.10. The user can easily carry out those steps in the left panel of the GUI, without any scripting or manual procedure.

Data preprocessing The original dataset depicted in figure 4.5 cannot be used as such, as it contains all the optimal hinges obtained for different load cases and other parameters. The first step is to select the requirements on the load and available space, as shown in figure 4.6. Then, the hinges can be further filtered by specifying the lug materials, bolt standard and bearing standard. An objective of this filtering could be to meet a particular requirement, that is either functional (e.g., select a bearing that is spherical) or circumstantial (e.g., the go-to bolt standard just had a price increase and cannot be selected anymore). This procedure is illustrated in figure 4.6.

Data analysis With conventional tools (e.g., Excel, or plotting libraries in Python or Matlab), changing properties of the plot is easy but time-consuming. On the other hand, with this tool, the X-axis, Y-axis, size, and color of the points can be adjusted by a single click. The first *analysis* method is, therefore, to modify the plot to reflect an assumption on the behavior of the hinge. For example, the user could be interested in the relationship between bolt radius and hinge price. This trend can be observed by choosing the bolt radius for the Y-axis and the design load for the X-axis. This way, the user may uncover a trend that can be generalized to all hinge designs. Such a pattern can lead to general design recommendations and further reduce the design time of a hinge.

Another example of data analysis is the trade-off between several hinges designed for the same load. The user wants to know the compromises between mass and cost between the different hinge assemblies. The choice of axis needs to be combined with the size and color of the points. The result of these choices is shown in figure 4.7. The features of the plot facilitate the apparition of clusters and correlations.

Another method to analyze the data is to use the attributes of the hinges presented in the table on the right in figure 4.9.

Inspection and validation The hinges in the front-loaded database only contain a limited number of attributes, deemed useful to carry out the data analysis.

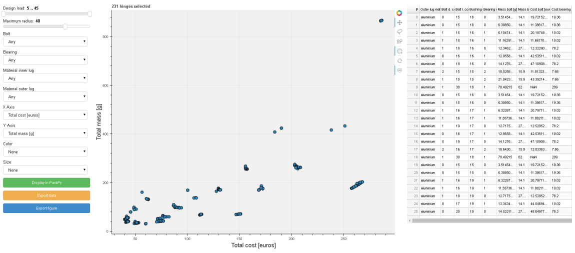


Figure 4.5: Original front-loaded database with default filters and axis.

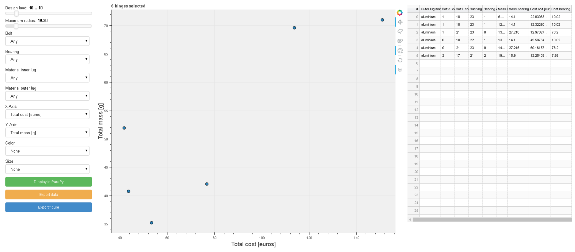


Figure 4.6: Filter the database by limiting the available space and design load. Select other parameters (e.g. available bolt types, bracket material, etc.)

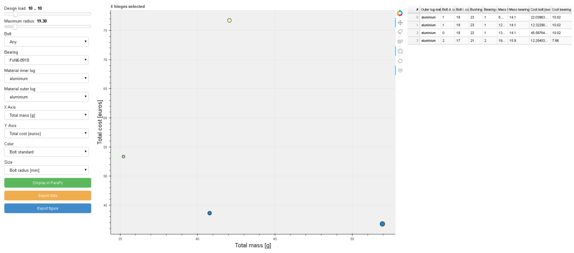


Figure 4.7: Customize the plot: choose X, Y, color and size of points.

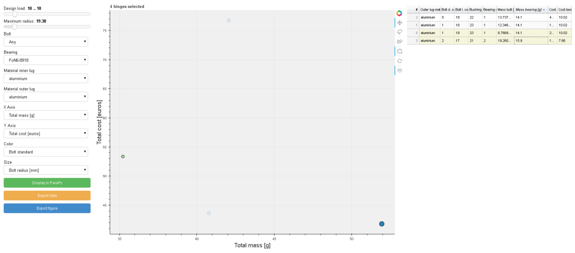


Figure 4.8: The table can also to visualize more attributes of the hinges. Columns can be sorted and selected to see to which hinge which data points correspond.

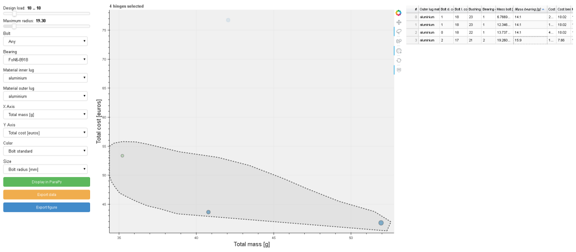


Figure 4.9: Hinge assemblies can be selected by clicking or using a lasso tool.

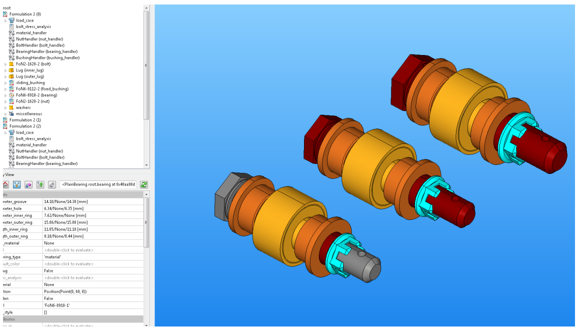


Figure 4.10: Once one or several hinges are selected, they can be opened in the KBE application.

Therefore, inspecting the hinge assembly through the KBE application (as seen in figure 4.10) can be useful if one wants a higher level of detail. This way, the strengths of the Hinge Generator as a design tool (discussed in subsection 2.6.4) are re-accessed.

This part of the data visualization process doubles as a validation tool. Only the inputs of the model are set, while the attributes used to validate and assess the model are recomputed on the fly by the Hinge Generator.

As a conclusion, the data visualization framework automated the process of inspecting and analyzing the front-loaded dataset.

4.2.3. Strengths and weaknesses of the tool

The sub-question being answered with the hinge selection method was introduced in chapter 1: *How to represent the Pareto-optimal hinges in a way that enables trade-off decisions?*

Strengths Usually, different tools (e.g., Python scripts, Excel sheets) and databases needed to be processed manually. The main advantage of this framework is that it provides a user-friendly and quick way to get a handle on the front-loaded database, by automating those data post-processing tasks. The structure is based on the Bokeh library, as detailed in chapter 9. Therefore, the developed application can be customized and extended to provide a better user experience.

Weaknesses By giving several points of view on the dataset (plot, table, ParaPy GUI), the application possibly becomes less straightforward and more difficult to access for new users. Moreover, although this type of application can be easily created for similar databases with systems other than hinges, it does require significant development time. Using an Excel spreadsheet is more time-consuming, but easily accessible and more straightforward for someone familiar with Excel but not with the developed application.

4.3. The hinge selection method

The previous section presents a method to visualize, inspect and understand the front-loaded dataset but does not address the use of the dataset for a rudder-fin optimization.

The research question "*How to obtain Pareto-optimal hinges as quickly as possible? (< 10 ms)*" is detailed in the problem statement (subsection 4.3.1). The method developed is presented in subsection 4.3.2, with details on the chosen input density in subsection 4.3.3. Subsection 4.3.6 summarizes the strengths and weaknesses of the selection method.

4.3.1. Problem statement

The goal of front-loading is to provide exact results instead of an approximation. As giving a result only based on an regression (i.e., prediction of continuous variables) or classification (i.e., prediction of discrete variables) algorithms does not provide exact results. It is preferable to give a result of a previous optimization or to run a new optimization altogether.

The chosen approach is to store the results of several optimizations for different requirements, and then to select the appropriate optimal point for new requirements. The problem statement is the following:

$$
\begin{aligned}
&\text{Solve the hinge optimization } (eq.\ 3.3) \text{ near instantly} \\
&\forall(\alpha,\theta,F,g,\beta_{il},\beta_{ol},mat_{il},mat_{ol},e_{max}) \text{ such that:} \\
&\alpha=(0,30°) \\
&\theta=15° \\
&F\in(5,\ldots,45kN) \\
&g\in(1,2,3,4mm) \\
&(\beta_{il},\beta_{ol})\in(45°,80°) \\
&(mat_{il},mat_{ol})\in(\text{alu1, alu2, alu3, steel, titanium})
\end{aligned}
\tag{4.1}
$$

Equation 4.1 shows that the combinations of the parameters are limited. Although some parameters are finite (e.g., mat_{il} can only take five values), the load magnitude F is continuous. A weakness of this method becomes evident: not all the continuous values should be computed exhaustively. An appropriate density has to decided. If the density is too low (e.g., $\Delta F = 10\ kN$), then the design space is not properly described and the minimums can be missed. On the other hand, if the density is too high (e.g., $\Delta F = 0.001\ kN$), many optimizations would return almost the same output and computational resources are wasted. This issue is discussed in subsection 4.3.3.

Instead of exhaustively exploring the design space and selecting a neighbor point, another option would be to create a surrogate model. This option is discussed in subsection 4.3.4. In the rest of the section, the exhaustive approach, coupled with the appropriate input density is chosen.

The selection of a neighbor point problem is represented graphically in figure 4.11. If the desired design point is missing from the front-loaded dataset, a candidate must then be chosen from the neighbor, feasible hinges, circled in a dotted black line in figure 4.11. The requirements on the hinge depend on the load case of the rudder and the position of the hinge in the rudder-fin interface. They can be translated into an "available space" requirement (the hinge must not intersect with the front cover of the rudder, Y-axis in figure 4.11) and an "internal loads" requirements (the hinge must sustain the resulting load without failing, X-axis in figure 4.11). Those two requirements can be represented by linear inequality constraints, as represented by the lines in figure 4.11. The grayed-out region indicates that those hinges are unfeasible either regarding available space, allowable load, or both.

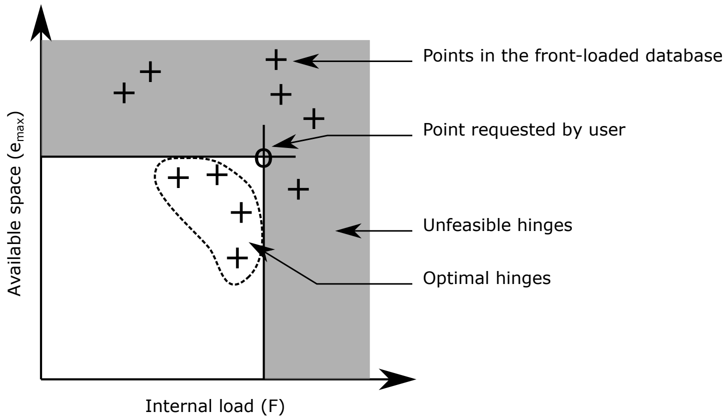
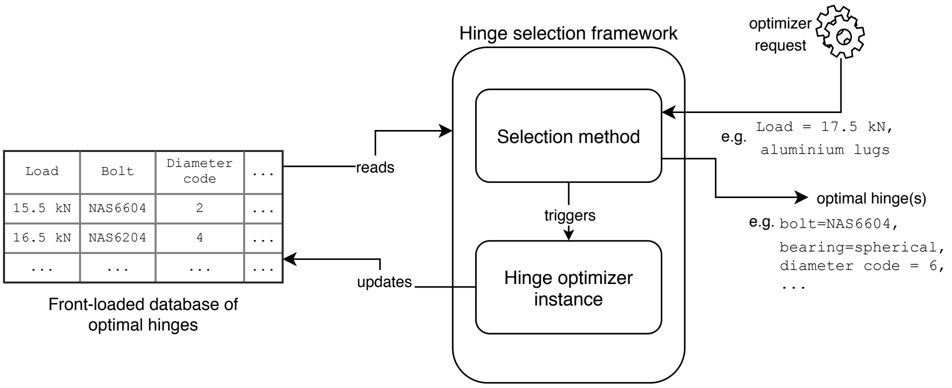


Figure 4.11: Hinge selection from the front-loaded database. The user or optimization requires a hinge that carries a certain load F and fits within a maximum available space e_{max}. Hinges in the database that do not respect those inequality constraints are in the grayed area. Hinges in the database that respect the inequalities are potential optimal design choices.

4.3.2. Framework and selection method


Hinge selection framework
optimizer request
Selection method
triggers
Hinge optimizer instance
reads
updates
e.g. Load = 17.5 kN, aluminium lugs
optimal hinge(s)
e.g. bolt=NAS6604, bearing=spherical, diameter code = 6, ...
Front-loaded database of optimal hinges

Load	Bolt	Diameter code	...
15.5 kN	NAS6604	2	...
16.5 kN	NAS6204	4	...
...	...	...	...

Figure 4.12: Hinge selection framework. The framework takes the *front-loaded database* which is a table, where the rows correspond to an optimization case and the columns to attributes of the optimal hinge. If the request point is not in the database, a new optimization is triggered, and the results are stored in the front-loaded database.

Figure 4.12 shows a simple representation of the hinge selection framework. The front-loaded database represented in figure 4.12 contains all the characteristics of a hinge required to carry out a full analysis. For example, it includes the full designation of the bolt (its standard, diameter code, length code and material code) such that the detailed specifications of the part (length, material strength, mass, etc.) can be either retrieved in the standard parts and material databases or can be recomputed at run-time with the Hinge Generator KBE application. Besides, the front-loaded database also contains important attributes (such as the reserve factors of the different parts) so that the database can be inspected with a text editor or through the data visualization tool presented earlier.

If the requested hinge is available in the database, then the selection method will return that hinge. Else, then two situations can occur. Either the requested case is deemed to be too far from the existing points in the database, and a new optimization is *triggered*, and the front-loaded database is *updated*. Otherwise, a neighbor hinge (i.e., slightly oversized or with lug materials that have slightly lower tensile strength) is selected.

The selection method is detailed in the algorithm 4.1. The instructions in lines 6 to 9 describe how the information is retrieved from the front-loaded database.

An advantage of the method is that the list of inputs can be extensive. For instance, the *additional user inputs* (line 1) could contain a constraint on the types of bearing used (e.g., only return roller bearings), which are then filtered in line 8. By default, the method will return *all* the Pareto-optimal hinges. If a single hinge is desired as an input, the results can be filtered again by a weighted sum $f = \lambda c + (1-\lambda) m$ combining the mass and cost.

```
Input: $x = (e_{max}, F, g, mat_{il}, mat_{ol})$, additional user inputs ;
Output: Best hinge assemblies;
if $\forall i, \in x_i > \Delta x_i$ then
    Run Informed Linear search with new inputs: $x = (e_{max}, F, g, mat_{il}, mat_{ol})$;
    Update front-loaded database;
Filter allowable loads $F \geq F_{database}$;
Filter allowable space $e \leq e_{max}$;
Filter additional user inputs;
Filter pareto-optimal points;
return Best hinge assemblies;
```

Algorithm 4.1: The hinge selection algorithm enables to select an appropriate hinge in the database, or to run a new optimization and to update the database if necessary.

The key point of interest lies in the determination Δx_i, which is discussed for each parameters in the next subsection.

4.3.3. Determination of the appropriate input density

The parameters Δx_i are set depending on the influence of the parameter or design variable on the problem. If the influence $\frac{\partial f}{\partial x_i}$ (where f is the objective) is small, then a larger Δx_i can be chosen.

Load case influence The load case is the most critical requirement. The load magnitude was varied from $5kN$ to $45kN$, with a step size of $\Delta F = 0.5kN$. As explained in subsection 2.1.2, the lower and upper bounds are selected to cover $+/-50\%$ of the nominal value of a hinge in a business jet rudder. The step size was obtained iteratively, by dividing the previous step size by two, until the composition of the hinge between two loads does not change.

The load case angles were not varied during this study. The in-plane angle α is a range between 0 and 30°, and the out-of-plane angle β is set to 15°, which is the standard value used at Fokker. Those values are independent of the location of the hinge within the rudder-fin interface.

Lug parameters influence The mass and cost of the bracket is far greater than the one of the lugs. The lugs influence the hinge assembly through three parameters. First and foremost, the thickness of the outer lug influences the length of the fixed bushing, sliding bushing and bolt. This influence means that a greater lug thickness results in an increased cost and mass for all those components. Furthermore, the increase in thickness increases the bending moment applied on the bolt. Lastly, the lug material influences the load peaking factor, which in turn influences the bolt bending moment too. The possible material combinations are counted in hundreds at Fokker. However, once the manufacturing constraints have been selected (plates of a certain thickness) and that the best materials have been chosen, the actual number of materials is much lower. Two criteria are made for the selection of the lug material. First, the material type of the requested hinge (i.e., aluminum, steel, titanium or bronze) must be the neighbor hinge. The material type is primordial because it determines the density, cost density, and fretting rules. Second, the ultimate strength of the material must be within $\Delta\sigma = 100\ MPa$. With this value of $\Delta\sigma$, the lug thickness variation is minimal and does not influence the cost and mass of the hinge.

Lastly, the lug sweep angle β is varied between two values: $\beta \in [45°, 70°]$. Although intermediate values could be chosen, this corresponds to business jets and small transport aircraft, which result in a different rudder design, and thus bracket design. Additionally, the influence of the sweep angle is low (it only influences one of the cross-sections of figure 2.23, which is itself only used for the allowable transverse load computation).

Available space influence The available space is an inequality constraint applied *after* the ILS. Therefore there is no rounding problem, any hinge assembly that fits within the OML (i.e. with $e < e_{max}$) can be selected.

Gap The influence of the gap g between the lugs (represented in figure 2.19) on the hinge assembly highly relevant for the hinge design. The gap size influences the bending moment, which plays a critical role in the allowable load of the bolt. Therefore, a low value of Δg is desirable. However, the requirements on the gap are standardized and have a fixed set of values, which is reflected in equation 4.1. It stems from the possible deformations that can occur during certain flight conditions.

Using those values of Δx ensures that the result of the selection method yields not only a feasible but a near-optimal point. The errors obtained using this input density are presented in subsection 4.3.5.

A problem persists with this approach, no matter how small Δx is. It is difficult to capture the exact maximum load an optimal hinge can sustain. For example, if the load $F = 17.5kN$ yields a hinge assembly A and $F = 18\ kN$ yields a hinge assembly B. The front-loading framework fails to provide a precise answer as to which exact load $F \in [17.5, 18]$ assembly A is optimal before assembly B has to be used. the question to be asked must be In order to solve this problem, the question to be asked must not be "*What is the maximal load F_{max} the hinge A can carry?*", instead of "*What is the optimal hinge for the load F?*".

4.3.4. Possible alternative approach: hinge surrogate model

An alternative method to achieve front-loading would be not run several optimizations for different requirements (i.e., find the best assembly for a given load) but to explore the possible loads for different points in the design space (i.e., find the optimal load for all the hinges in table 3.3). Algorithm 4.2 presents a possible implementation for this approach.

```
Input: hinge assembly, intial load $F_0$, intial step $\Delta F_0$, minimum step size $\Delta F_{min}$ ;
Output: Maximum allowable load for the hinge assembly;
$F = F_0$;
$\Delta F = \Delta F_0$;
while $\Delta F > \Delta F_{min}$ do
    Solve optimization problem (equation 3.10) for $F$;
    if success then
        $F \Leftarrow F + \Delta F$
    else
        $\Delta F \Leftarrow \frac{\Delta F}{2}$;
        $F \Leftarrow F - \Delta F$;
return $F$;
```

Algorithm 4.2: The inverse search does not search through the design space of combinations of components. Instead, it searches for the maximum allowable load for a given hinge assembly.

However, this does not actually give the *optimal* hinge assembly. The method gives a list of hinges which can sustain the load. Once this maximum load is obtained for each hinge, the best assembly for the given load would still have to be chosen from the set with acceptable maximum loads. Then, the phase 2 of the ILS would still have to be launched to determine the dimensions of the lugs. This would take a few seconds and therefore would violate the *instantaneous* criteria. Although this approach is feasible for small numbers of combinations, it does not scale well with as the number of combinations increases.

4.3.5. Results of the hinge selection method

The performance of the hinge selection method needs to be assessed concerning its efficiency (run-time, in subsection 4.3.5) and its effectiveness (validity of the results, in subsection 4.3.5).

Scalability and run-time

The run-time of the selection needs to be as short as possible to answer the second use-case. Assuming that the rudder-fin optimization uses a GA, with 10 000 iterations, and a population of 20 interfaces per generation, each with an average of 6 hinges, the hinge optimization will be triggered a total of $10000 \cdot 20 \cdot 6 = 1.2 \cdot 10^6$ times. With a run-time of 2 hours per hinge optimization, the total computation would have been unfeasible (273 years). The algorithm 4.1 bypasses this problem by querying existing results instead of running a new optimization.

The implementation of algorithm 4.1 requires several vectorized operations on the front-loaded database (lines 6 to 9). The computation time required to carry out those operations scales with the size of the front-loaded database. Table 4.1 shows the evolution of three run-times as a function of the size of the front-loaded database. The overhead time is the computation time required to load the database into memory. The run-time is the execution time of algorithm 4.1. This run-time is multiplied by $1.2 \cdot 10^6$ and added to the overhead time to obtain an order of magnitude of the time required to perform a rudder-fin optimization with a GA.

Front-loaded database size	Overhead time	Selection run-time	GA optimization run-time
32 000	13 seconds	3.3 ms	1:06 hours
65 000	1:15 minutes	4.7 ms	1:34 hours
127 000	3:00 minutes	5.7 ms	1:57 hours

Table 4.1: Run-times for different front-loaded database sizes. The query time varies in a linear manner with the size of the database but remains very small even for very large ones (5.7 ms for 127 000 hinges).

Table 4.1 shows that for very large databases, the run-time of algorithm 4.1 remains very small. The overhead time grows rapidly with the size of the database, but it does not affect the GA run-time, as the database only needs to be loaded once. The small run-time of algorithm 4.1 is key to obtain a short GA run-time. This result demonstrates that the front-loading approach can be applied to very large databases.

Errors and input density

Table 4.2 presents an example of the rounding error obtained with algorithm 4.1. The only difference in the two load cases lies in the dimensions of the lugs, which results in a small absolute difference. The error between the total cost and mass is, therefore, small as well.

F	17.2 kN		17.5 kN		Δ[%]	
	mass	cost	mass	cost	mass	cost
Hinge 1	46.5	71.5	47.0	71.6	0.2	1.1
Hinge 2	252.7	156.9	252.9	157.1	0.1	0.1
Hinge 3	258.5	207.9	258.6	207.9	0.0	0.0

Table 4.2: Rounding error examples for $\Delta F = 0.3 < \Delta F_{max}$. The rows present the total cost and mass of three hinge assemblies given by the ILS, for $F = 17.2kN$ and $F = 17.5kN$. As ΔF is small, the components of those assemblies are identical.

4.3.6. Strengths and weaknesses of the selection method

The sub-question being answered with the hinge selection method was introduced in chapter 1: *How to obtain Pareto-optimal hinges as quickly as possible? (< 10 ms)*

Strengths The main strength of the front-loading approach lies in its ability to provide exact results almost instantaneously. Moreover, the implementation of this approach is easy compared to the creation of a surrogate model.

Weaknesses The functioning of this approach relies on the density of the available points in the database. If the density is too high, the same combination could be analyzed and returned several times. Therefore, the time required to build the database is suboptimal, but as seen in table 4.1, it exerts a negligible influence on the GA run-time. Another problem is that this approach scales poorly with the number of driving design parameters increase. If the number of parameters is very high, the necessary time to build the front-loaded database increases, but the surrogate model presented in subsection 4.3.4 remains viable. This problem was not critical at Fokker, as the parameters (loads, gap, materials lugs) are already known and are not expected to change in the future. Therefore, the presented front-loading method should remain applicable.

4.4. Discussions

The front-loaded database needs to be included into a framework to enable its usage.

Analyzing the front-loaded dataset is challenging and time-consuming, as it contains many optimal hinges, for different requirements, each hinge having many discrete and continuous attributes. The dataset requires manual processing, such as data preprocessing, analysis and inspection. The data visualization tool automates the steps and thus enables a quicker grasp of the database.

Repeating the optimization with the appropriate input density yields Pareto-optimal hinges for the given range of requirements. The optimal hinges are then stored in a database that can be filtered depending on particular requirements or availability of the parts. A hinge selection framework was developed to use the front-loaded database in a rudder-fin optimization. With a dense enough input density, the results of the hinge selection are equal to those of the ILS. Even if the database is incomplete, it can be updated on the fly during a rudder-fin optimization. Moreover, even with very large amounts of data in the front-loaded database, the hinge selection method remains effective. The low run-time of the selection method enables its use in a rudder-fin interface optimization. Therefore, even without a front-loading database, the framework could be used in a GA and would progressively get quicker, as the hinge design space would progressively be explored and memorized.

The specific strengths and weaknesses of the developed methods were discussed in subsections 4.2.3 and 4.3.6. Another critique of front-loading is that it could result in a waste of resources. If either the optimization or model changes, then all the data analyzed for weeks or months need to be thrown away. However, once the model and the optimization are developed, experiments can be set up and executed for a small computational cost. Therefore, if resources were to be wasted as argued previously, it would only be a small part as most of the effort went into developing the model and optimization workflow. Both elements are still useful even outside of a front-loaded PDP. From that point of view, the front-loading methodology is consistent; it is the combination of advanced design methodologies (KBE and MDO) and computational power increase.

5
Conclusions

Fokker aims at offering OEMs a wide range of rudder designs trade-offs within a short timeframe. Located at the interface between the rudder and the fin, the hinges need to carry internal loads to sustain the deflection during various flight maneuvers. The hinge design constitutes a bottleneck in meeting Fokker's objective, as their design is done manually and may take up to 6 weeks for a single hinge.

Moreover, the design of a single hinge must be repeated several times, for different positions and loads in the rudder-fin interface. Therefore, pre-processing the hinge optimization and storing the related results would be advantageous, as it would avoid repeating computations. This methodology is called *front-loading*. It consists of using KBE and MDO extensively *before* the requirements on the product are fully known. Accordingly, the main research question of this thesis is as follows:

How to implement the front-loading methodology for a hinge assembly?

The front-loaded methodology was achieved by pre-processing the hinge optimization and storing the related results. This methodology provides the best hinge assemblies in terms of mass and cost for a given set of requirements on the load case and the bracket. As a result, it is possible to select an optimal hinge "*off-the-shelf*" in the course of the rudder design process. This methodology was achieved in three steps.

First, a KBE application (the *Hinge Generator*) was developed to carry out the analysis of a hinge assembly. The Hinge Generator estimates the cost, mass, and allowable loads of all the hinge components. Moreover, it automates queries for standard parts and materials and the application of design rules. The graphical interface enables the use of the KBE model in order to quickly explore possible components combinations, as well as inspecting and tweaking the hinge design.

Second, the Hinge Generator is used with a search algorithm. Several methods were tested to find a method that yields *all* Pareto-optimal hinges efficiently. The *Informed Linear Search* (ILS) is a method that separates discrete and continuous parts of the search. The ILS finds the Pareto-optimal hinges and is time-efficient, including when it is used with large databases of standard components. This short run-time provides a significant advantage over previously used methods, such as Genetic Algorithms. The results of the ILS were validated through comparison with an exhaustive search. However, using the ILS by itself is suboptimal, as similar hinge optimizations would be repeated throughout the rudder design.

Finally, the hinge optimization is *front-loaded*. The ILS is applied repeatedly, considering different requirements on the bracket and loads. The results are stored and provide a front-loaded hinge database. This database contains hundreds of hinges with various attributes to inspect, which raises the sub-question:

How to represent the Pareto-optimal hinges in a way that enables trade-off decisions?

An interactive interface was developed to facilitate visualization of the database of optimal hinges. The interface enables filtering, querying, and dynamic visualization of the data, as well as geometry inspection. It is a valuable tool to perform trade-off choices.

Moreover, the front-loaded database should be considered in the context of a rudder-fin interface optimization. During the optimization, the hinge design is triggered a large number of times, hence a quick selection method would result in a shorter rudder-fin interface optimization.

How to obtain Pareto-optimal hinges as quickly as possible? (< 10 ms)

The hinge selection method is developed to retrieve results from the front-loaded database. The selection is achieved by integrating the ILS into a framework that enables optimal results to be stored and retrieved at run-time. Depending on the differences between an existing optimal hinge and a requested point, either an existing optimal result will be used, or a new search will be launched. With its quasi-instantaneous run-time, the selection method enables rudder-fin optimization, which was previously limited by the hinge optimization run-time. Although the rounding-off approach is not applicable to every parameter of a system, the run-time of the selection method remains very small, even for large databases. This result shows that this framework is applicable to other systems, including with higher numbers of parameters.

6
Recommendations

This chapter summarizes the steps that could be carried out to validate (section 6.1) and extend (section 6.2) the results of this front-loading methodology. The order of the subsections reflects their priority. Further code-specific recommendations and remarks are also given in the second part of this report.

6.1. Validation of the results

Stress analyses

Currently, the validation of the stress analyses are made with respect to two examples given in the Fokker documents [13] and [14]. Therefore, the stress analyses are only validated for a single geometry and material, for the bolt and lug stress analyses by themselves, not for the full hinge assembly. Hence, the following are required:

- Further testing of the lug and bolt stress analyses methods, to ensure the analyses are valid on the prescribed design space.

- A complete stress analysis at the level of the hinge. Particularly, a comparison with existing hinges for business jets at Fokker could be included. For this purpose, the standard parts databases will require extension.

SLSQP termination criteria

During the experiments, inconsistent results regarding the second part of the 2-phase search were sporadically noticed. The equation 3.10 could be solved for a given hinge assembly for $F = F_1$, but not for $F_2 < F_1$ The termination criterion relies on the exit status of the gradient-based algorithm used in the second part of the search. It depends on the starting point, the tolerance and maximum number of iterations of the algorithm. This kind of inconsistency was noticed during the project, but its cause is unknown. A first step to tackle this problem would be to use a different optimization method.

User-based tools

The user-based parts of the tools that were developed (Hinge Generator for design in subsection 2.6.4 and Interactive visualization in section 4.2) have neither been tested by engineers nor by designers at Fokker. Submitting the tools to practical, hands-on tests and interviews at Fokker will ensure that the requirements of the project are well defined and satisfied.

Cost analysis

The cost analysis methodology developed by Risueno with HDOT [27] has higher fidelity than the cost analysis used in this tool, described in section 2.4. The current results in terms of cost have only been validated in terms of variations and orders of magnitude. The CERCOPS tool was not implemented in this project due to lack of time but could be incorporated in the future.

Detailed fretting and fitting design rules

The Hinge Generator currently uses a simplified set of rules for the fitting and fretting between the components.

For the fitting, the tolerances of the components given in their standard part description are taken into account. The fitting criterion is arbitrarily set to 0.001 mm. In reality, the criterion depends on the parts being assembled, as described in [6, 7].

For the fretting, the simplification used is similar. The fretting criteria depends on a specific type of material (e.g., the fretting rules are different for CRES [1] - 300 series and CRES - 400 series [38]), whereas the Hinge Generator takes into account the material type (e.g., steel, which includes both CRES 300 and 400 series).

6.2. Extension of the results

The result can be extended at various levels. Each recommendation can require changes on the KBE model, search method and hinge selection method.

Further work on the hinge optimization

Several improvements could be made to reduce the run-time of the ILS.

A first solution is to choose the first index value in a more informed manner. One could use an approach similar to binary search, which would scale better with the number of combinations (time complexity $O(n)$ vs. $O(\log(n))$). However, with the hinge, the lookup time is minimal compared to time spent optimizing continuous characteristics, therefore implementing those solutions would only result in minor computation time improvements.

A second solution would consist in restricting the optimization launch more strictly. Indeed, it would often happen that the stress inequality constraints can be validated one by one but not all at the same time. In that case, the gradient-based algorithm can be stuck for some time trying to solve an unfeasible problem.

The unsuccessful attempts with the genetic algorithms described in section 3.4 do not prevent their application to the hinge optimization problem. Possibly, “in-between” problem formulations, e.g., having one integer variable per standard parts - as opposed to two variables for the first formulation, could yield better results.

Further work on the hinge surrogate model

Although the developed hinge front-loading framework yields good results in terms of run-time, a surrogate model would consume less computational resources to create the front-loaded database. Moreover, investigating the surrogate modeling of the hinge could be useful to gain insight in the application of front-loading to other systems. A possible path to investigate was already layed out in subsection 4.3.4.

Extension to fail-safe and clamped hinges

The Hinge Generator currently only models sliding hinges.

Subsection 2.2.2 and figures 2.15 to 2.17 showed that the configurations display some differences compared to the sliding hinge. All the stress analyses of the components are already included in the Hinge Generator, as well as the geometry of the components. The main tasks will be to add the new components, modify their size (e.g., `radius_sliding_bushing = radius_bolt` will become `radius_sliding_bushing = radius_fail_safe_sleeve`) and update the load cases.

The extension of the Informed Linear Search to different types of hinges will be more complicated. The depth criteria (equation 3.16) used to reduce the run-time is still usable for other types of hinges. The assumptions used to simplify the discrete part of the 2-phases search will have to be re-evaluated for the new configurations. For example, the case of a fail-safe sleeve (figure 2.17) is different than that of the sliding bushing. Both the sleeve thickness and material should be considered as design variables, as the reserve factor of the sleeve will be critical (it will be subject to the same stress analysis as the bolt, as it needs to take the load of the lugs in case the bolt fails) and its mass will most likely represent a more substantial part of the total assembly mass too.

[1] Corrosion-Resistant Steel

Extension to hinges for elevators

The work done during this thesis is targeted at rudders. The lugs in the elevator interface will carry the weight of the control surface in the in-plane (figure 2.2) instead of out-of-plane (figure 2.3), like the rudder.

Therefore, it becomes advantageous to have an unsymmetrical lug geometry. To this extent, the geometry modeling of the lugs should be modified. The lug stress analysis of the Hinge Generator already accounts for unsymmetrical lug geometries.

Moreover, hinges with two male lugs and three female lugs are sometimes used in elevator interfaces. If that design is to be included, the stress analyses would be very similar. The positioning and number of components in the KBE model would have to be updated.

Inclusion of machined parts

Currently, the question "*Should we use a machined or standard bolt?*" cannot be answered quantitatively, as machined bolts are not taken into account in the ILS. Likewise, the question "*Should we use a machined or standard fixed bushing?*" cannot be answered either, as the fixed bushing is by definition a standard part in the ILS.

Including machined bolts in the Hinge Generator should be straightforward. As every object in the KBE application is parametric, modeling the custom geometry is only a matter of taking an existing bolt standard and setting the radius or length to the custom, machined value. Likewise, the material characteristics, stress analyses, and mass analyses will be identical. However, the cost modeling will have to be updated.

Extending the ILS to take into account both machined and standard parts is, again, more challenging. A simple solution would be to run two searches, one with standard bolts and one with machined bolts. This approach could be viable, as the discrete/machined distinction is only applied to the bolt. However, the number of cases to run would increase quickly if the distinction needs to be made for other components.

Search including material variables

This work done in this thesis does not give a satisfying answer on how to deal with discrete materials in optimization problems. In [16], the hinge optimization was performed with more than 30 materials, both for the nut and the sliding bushing. Treating the same problem with the ILS would not be very efficient, as the sliding bushing and nut standards would be explored exhaustively.

A method to deal with this problem would be to create a surrogate model of the material so that it can be replaced by one continuous, descriptive variable such as the tensile strength. This method is presented in Appendix A.

II

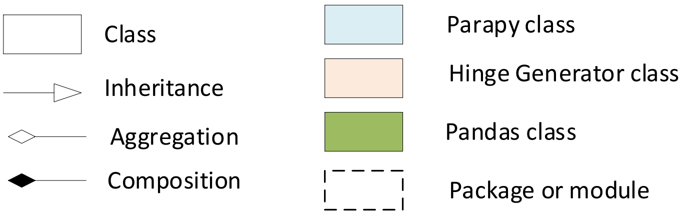
Part B - Code report

Introduction

This part of the report contains technical details about the implementation of the frameworks and methods presented in the Part A.

- The `data` folder contains the input spreadsheets for the material and standard parts databases and output data (material surrogate model, front-loaded optimization results).
- The `doc` folder contains the code documentation.
- The `hinge_generator` package contains the `Hinge Generator` KBE application (chapter 7).
- The `hinge_fl` package contains the discrete part of the optimization framework (chapter 8) and the front-loading framework (chapter 9).
- The `tests` folder contains unit tests for elements of the model and optimization methods. Abstract classes inheriting from `unittest.TestCase` enable to define tests for new components and standard parts quickly.

Several Unified Modeling Language (UML) class diagrams are used to describe the contents of the Hinge Generator. The UML is intended to provide a standard way to visualize the design of a system [3].



UML diagram legend used in the second part of the report

7

Hinge Generator

Chapter 2 presented the design of a hinge assembly, with an introduction to the Hinge Generator in subsection 2.6.3. This chapter discusses the architecture and contents of the Hinge Generator in greater details. Section 7.1 gives the key elements of the application structure. Then, section 7.2 focuses on the implementation of the stress analyses and of the material and standard parts databases. Section 7.3 presents the main libraries used in the Hinge Generator. Finally, section 7.4 raises some considerations to take into account to improve the application.

7.1. Description of the KBE application

The composition of the KBE application was already introduced in subsection 2.6.3.

Structure of the hinge assembly Figure 2.25 showed that the hinge components are children of the hinge assembly. For example, the bolt is modeled as a `bolt` child, which can be an instance of the `HexagonalBolt` class. The UML diagram on figure 7.1 gives a more precise overview of the integration of the hinge components into the assembly. The bearing, bolt, nut and bushings are modeled as `DynamicType`, which enables to have a variable geometry. The components then inherit from a more abstract class (e.g., `AbstractBolt`), which itself inherits from the generic `HingeComponent` class.

The UML diagram in figure 7.2 gives further details on the implementation of the components. This UML diagram shows the integration of the lug component and its associated elements. The rest of the hinge components follow the same layout: each component has a stress analysis (`TH3582` for the lugs), a material, a cost, and a mass. The hinge components also include a generic geometry definition ($r_{in}, r_{out}, l_{in}, l_{out}$), so that the fitting of the components can be checked systematically for any sequence of `HingeComponent`.

The application is developed in a way that enables separation of the different functionalities and disciplines required by the hinge assembly. In the UML diagram in figure 7.2, the geometry of the lug is generated in the `Lug` class. The lug stress analysis is separated from the lug; it is in the `TH3582` class. If approximations on the material characteristics are needed during the stress analysis, they will be computed in the `Material` class.

Unmodeled features in the current implementation In figure 7.3, the DSM of the sliding hinge assembly is given. The cells underlined in red correspond to an unmodeled coupling:

- The fretting between the bearing and the bolt and inner lug is not taken into account.
- The washers are not sized nor numbered to position the castellated nut with respect to the bolt hole and the bolt thread length.
- Material properties between the nut and the bolt are not taken into account.

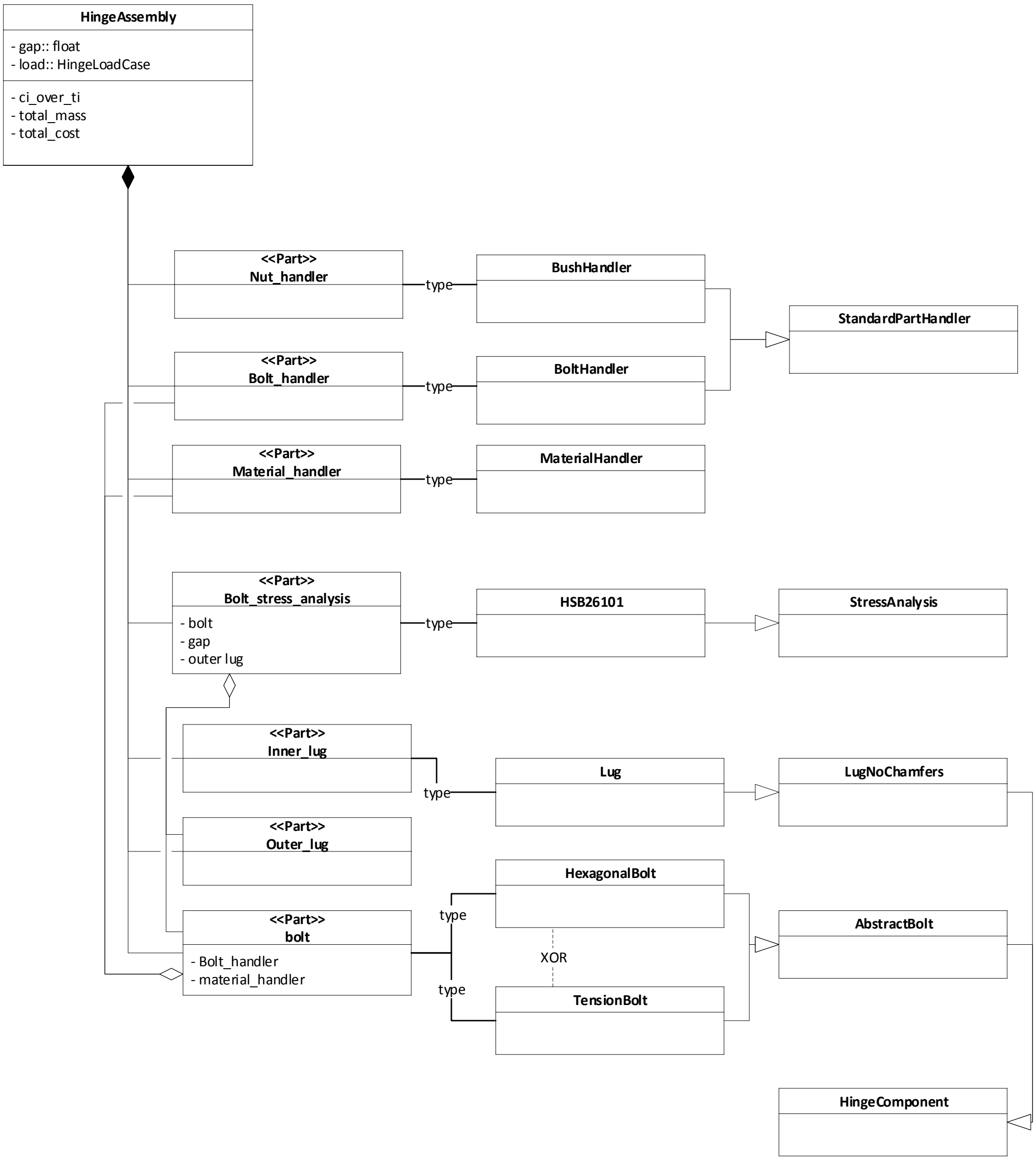


Figure 7.1: The hinge assembly contains several parts, notably data handlers, components and the bolt stress analysis. The bolt stress analysis is placed at this level (and not integrated to the component) because it requires data from the lugs and bolt. The `bolt` is a `DynamicType`, which can be an instance of `HexagonalBolt` or `TensionBolt`.

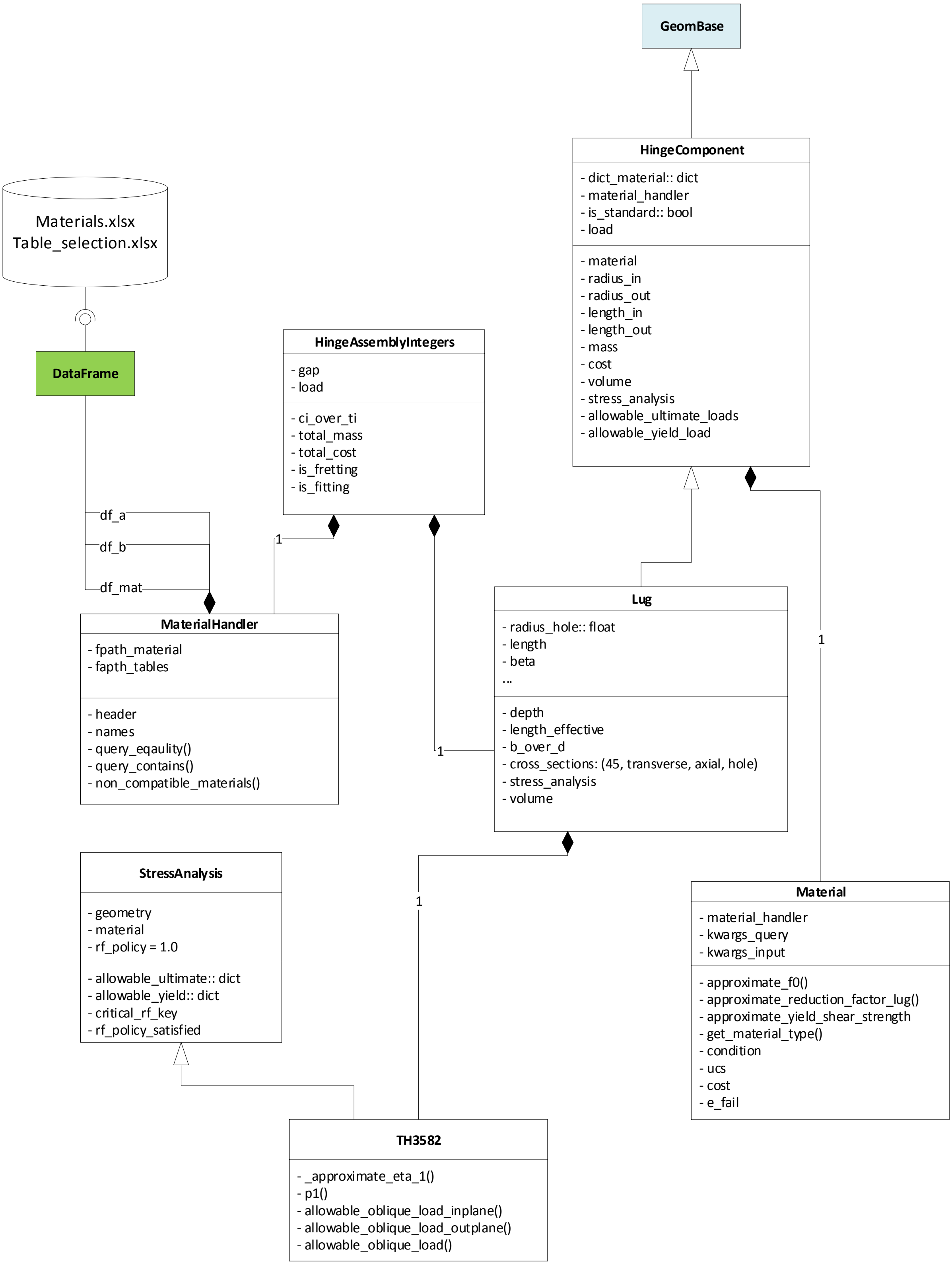


Figure 7.2: The main object representing the hinge is the `HingeAssembly`. Disciplines (stress, cost and mass analyses), geometry and data handling are all separated.

	Bolt outer radius	Bolt shank length	Bolt thread length	Bolt material	Bolt thread	Bearing material	Bearing inner radius	Bearing inner length	Bearing outer length	Bearing outer radius	Sliding bushing inner radius	Sliding bushing outer radius	Sliding bushing length	Sliding bushing material	Fixed bushing length	Fixed bushing inner radius	Fixed bushing outer radius	Fixed bushing material	Outer lug radius	Outer lug length	Outer lug material	Inner lug radius	Inner lug length	Inner lug material	Washers thickness	Nut length	Nut material	Nut thread
Bolt outer radius	1						●				●																	
Bolt shank length		2						●					●															
Bolt thread length			3																						🟧●	🟧●		
Bolt material				4		🟧+								+													🟧+	
Bolt thread					5																							●
Bearing material				🟧+		6																		🟧+				
Bearing inner radius	●						7				●																	
Bearing inner length		●						8																				
Bearing outer length									9														●					
Bearing outer radius										10												●						
Sliding bushing inner radius	●						●				11																	
Sliding bushing outer radius												12				●												
Sliding bushing length		●											13		●										🟧●	🟧●		
Sliding bushing material				+										14				+										
Fixed bushing length													●		15					●								
Fixed bushing inner radius												●				16												
Fixed bushing outer radius																	17		●									
Fixed bushing material														+				18			+							
Outer lug radius																	●		19									
Outer lug length															●					20								
Outer lug material																		+			21							
Inner lug radius										●												22						
Inner lug length									●														23					
Inner lug material						🟧+																		24				
Washers thickness			🟧●										🟧●												25	🟧●		
Nut length			🟧●										🟧●												🟧●	26		
Nut material				🟧+																							27	
Nut thread					●																							28

Legend
● Geometry coupling (must fit)
+ Material coupling (must not fret)
🟧 Not modeled

Figure 7.3: DSM of the sliding hinge assembly. On the matrix diagonal, the index of the variable is given. The order of the parameter is arbitrary and does not reflect the implementation, the matrix is only used to represent the coupling between the parts. The parts of the problem that are not modeled are indicated in orange.

Structure of the code

The `hinge_generator` package includes the following elements:

- The `geometry` package contains geometries used to create the geometrical modeling of the hinge assembly components. The UML diagram in figure 7.9 gives an example of the geometry implementation.
- The `components` package contains specific component implementations, including geometry, stress and mass analyses.
- The `stress` package includes the stress analyses files.
- The `utility` includes methods, classes and data that provides generic functions for several elements of the project.
- The `cost_analysis.py` file implements the cost analyses classes.
- The `hinge_assembly.py` file contains the different hinge assembly KBE applications. The three KBE application variations are presented in subsection 2.6.4.
- The `hinge_sizing.py` file contains the continuous part of the optimization, implemented with the `para_MDO` classes and represented in the UML diagram in figure 8.2
- The `input_handlers.py` file contains the databases handling classes represented in the UML diagram in figure 7.6.
- The `loads.py` file implements classes modeling the load cases of the hinge (`HingeLoadCase`, which has a `BoltLoadCase` and two `LugLoadCase`).
- The `material.py` file implements the `Material` and `SurrogateMaterial` classes used to handle data from the material handler and to approximate material properties.

7.2. Modules

The tool uses two databases, the standard parts database and the material database. Figure 7.4 represents their content and interaction with one another. The data should be structured in a systematic manner and in a way that enables it to be easily queried by a script.

Materials database

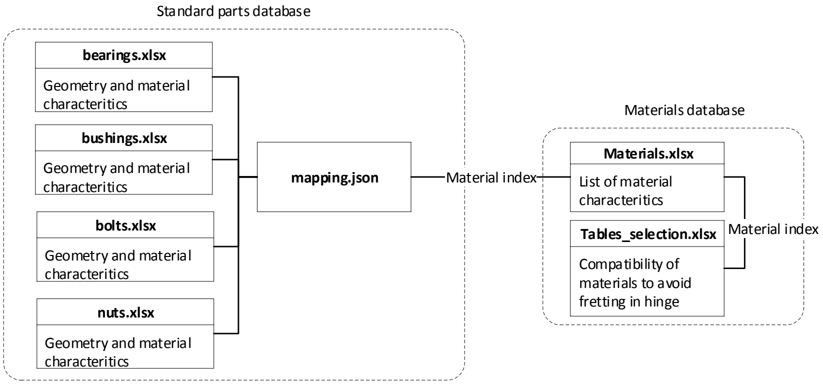

Standard parts database
bearings.xlsx
Geometry and material characteritics
bushings.xlsx
Geometry and material characteritics
bolts.xlsx
Geometry and material characteritics
nuts.xlsx
Geometry and material characteritics
mapping.json
Material index
Materials database
Materials.xlsx
List of material characteritics
Tables_selection.xlsx
Compatibility of materials to avoid fretting in hinge
Material index

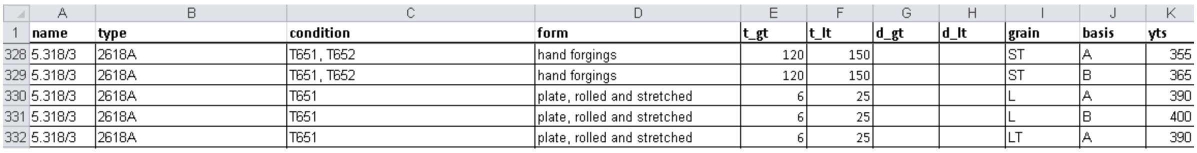
Figure 7.4: Interaction between the material and standard parts databases. The *mapping* file enables to have the standard parts database containing exclusively information about the part that is identical to the original PDF description of the part.

The material database is composed of the `materials.xlsx` file, which contains detailed characteristics of many materials and of the `tables_selection.xlsx` file, which includes the fretting rules.

Fokker works with certified and recognized sources for their data, such as the MMPDS[1]. However, they do not directly use those databases, for several reasons. First, Fokker wants to test the values, given that they can come from different suppliers with different testing methods. For example, they examine the yield strength, ultimate strength, and elongation at failure. Second, Fokker wants to merge the different suppliers, naming, and practices from the United States and European Union, as Fokker provides products for both continents.

The Fokker material database used for this project is the result of efforts to constitute a centralized source of knowledge on all the materials used at Fokker, in a way that is easily readable by a machine. Therefore, Fokker modifies, wraps and organizes those standards with their test results and data. There are no additional, manual steps to carry out to go from the material used in the model, which was often a conservative simplification and the actual material. By using this custom, fully detailed table, the material used in the model is identical to the one to be ordered from the supplier. The table contains, in addition to the material properties:

- The raw material form: sheet, plate, etc.
- The material grain direction L (long), LT (long transverse), ST (short transverse).
- The statistical basis: A-basis (when there is a unique load path) or B-basis (when there is a redundant load path in case of a failure of a primary load path member, e.g., a layer in a composite sandwich)



	A	B	C	D	E	F	G	H	I	J	K
1	name	type	condition	form	t_gt	t_lt	d_gt	d_lt	grain	basis	yts
328	5.318/3	2618A	T651, T652	hand forgings	120	150			ST	A	355
329	5.318/3	2618A	T651, T652	hand forgings	120	150			ST	B	365
330	5.318/3	2618A	T651	plate, rolled and stretched	6	25			L	A	390
331	5.318/3	2618A	T651	plate, rolled and stretched	6	25			L	B	400
332	5.318/3	2618A	T651	plate, rolled and stretched	6	25			LT	A	390

Figure 7.5: Sample contents of the `materials.xlsx` file. Different rows refer to the same material ("5.318/3"), but with different conditions, manufacturing processes (forgings, plate) and geometries (e.g., thickness of the plate must be between `t_gt` and `t_lt` - for lesser and greater than) and statistical basis (A or B).

Standard parts databases and handlers

An important part of the framework is to handle (read, transform, organize and query) the data for the standard components and materials. This is illustrated for the standard materials in figure 7.2 with the spreadsheets databases `materials.xlsx` and `table_selection.xlsx` that are converted into a `DataFrame` object in the `MaterialHandler` instance.

As shown in figure 7.4, the standard parts database is composed of a file per component and of a mapping file. The spreadsheet contains the raw data, as it is found in standard Fokker handbook.

Then, in the mapping file, the standard variable names are associated to the appropriate variable names used in the tool. This enables anyone to add a standard part to the library and the database to be more modular. Only the mapping of the standard to the ParaPy class requires knowledge of the Hinge Generator. For example, if a new bolt is added to the database, one will have to associate the dimensions of the standard, as given in the PDF, to the associated variable in the ParaPy class. Besides, the mapping contains other attributes. For instance, it contains the meaning of the standard expression - e.g., NAS6703DU02 corresponds to the NAS67 bolt standard, with the diameter code 3, the thread drilled "D", a material code "U" and length code 2. It also contains the associated ParaPy class - e.g., NAS67 bolt corresponds to a `HexagonalBolt`.

With this mapping "layer", the dependencies between the different parts of the tool are limited, meaning that the geometry generation capabilities of the `Hinge Generator` are relevant even if the databases are deprecated or heavily updated, and vice-versa. In addition, the data part of the framework is quickler to update. However, this structure makes the process of adding a standard part possibly less straightforward, as two files need to be updated. The addition of a new standard must be done in three main steps: adding the data to the spreadsheet, describing the standard with respect to the ParaPy class in the mapping and finally including tests to ensure the proper functioning of the tool. This process is documented in a separate "how-to" document, which is out of the scope of the report.

Figure 7.6 shows the classes used to handle the material and standard parts databases.

The class `StandardPartHandler` contains abstract methods to systematically read and transform the standard parts databases. This was achieved by using the same data format for each element.

[1] Metallic Materials Properties Development and Standardization, `https://www.mmpds.org/`

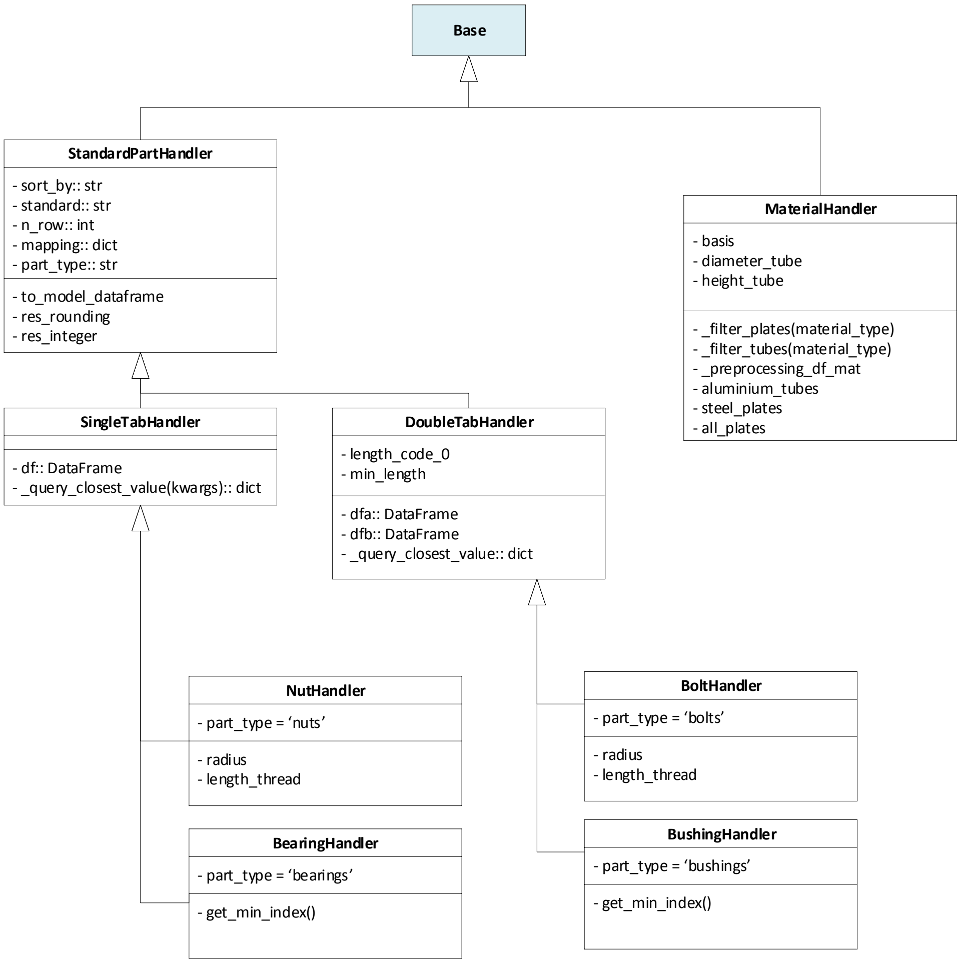


Figure 7.6: The standard parts and material databases are read through so-called handlers. Those classes include important features that facilitate the query and storing of the information.

Stress analyses package

As explained in subsection 2.6.3, a stress analyses in HDOT were implemented as Excel files [18]. Having the stress tools in Excel enables a separation between the stress analyses and the KBE model, which was appreciated at Fokker because it enbales engineers without Python experience to develop and test components of the KBE application.

For the Hinge Generator, several python tools were developed to replace them:

- The generic stress analysis for hollowed tubes (`stress_analysis.py`)
- The main bolt analysis file (`hsb26101.py`)
- The lug analysis file (`TH3582.py`)
- The plasticity coefficient computation file (`SM123.py`)
- The bearing strength computation file (`TH3232.py`)
- The bolt pretension computation file (`bolt_pretension.py`)

The use of python stress analyses files instead of Excel spreadsheets enabled several benefits:

Ease of development During the development of the lug stress analysis, errors were encountered in certain parts of the design space. A geometry that deviates too much from the nominal one yielded errors at a certain part of the analysis. Having the stress analysis in python made it easier to identify, isolate and resolve those issues. This problem would have been much harder to spot by calling a "black-box" analysis (such as an Excel file), which would simply return an error.

Transparency Using a mix of functional and object-oriented programming makes for a clear and concise instruction sequence. Having all the information about the stress analysis is easier if the program is directly operated in Python. In the Hinge Generator, the underlying stress assumptions are collected as they are used during run-time, enabling them to be retrieved by the user later on.

Interoperability The stress analysis can be imported and launched directly, there is no need to read and write files to interface with an external tool.

Run-time Python is an interpreted language and therefore is not necessarily the fastest programming language to use. The heavier parts of the stress analysis (integrations, root finding, vectorized operations) are done with the `scipy` and `numpy` packages which are implemented in C, reducing the computational time.

Two UML diagrams are used to describe the stress analyses of the Hinge Generator. The UML diagram in figure 7.7 illustrates the generic hollowed tube stress analyses defined in the `stress_analysis.py` file. The classes are built so that, again, smaller fundamental classes such as `BendingTube` can be used in several contexts, as a building block.

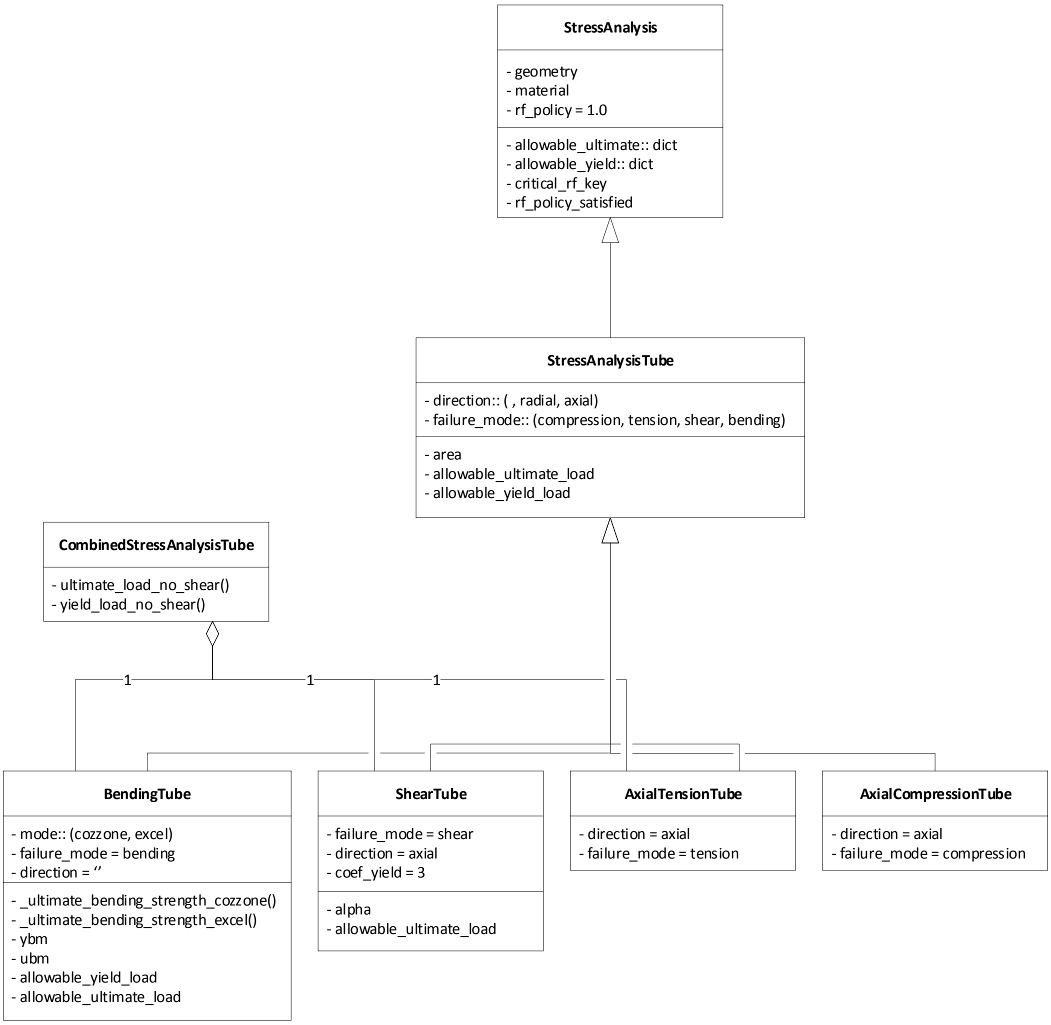


Figure 7.7: The bolt and bushings have the geometry of a hollowed tube. Before defining the stress methods specific to Fokker's methodology, generic methods related to hollowed tubes can be defined. Specific failure modes (e.g., `BendingTube`) are defined from the abstract `StressAnalysisTube` class. For the bolt which is subject to multiple loads, those analyses are combined in `CombinedStressAnalysis`.

The UML diagram in figure 7.8 shows the higher-level stress analyses used for the hinge components.

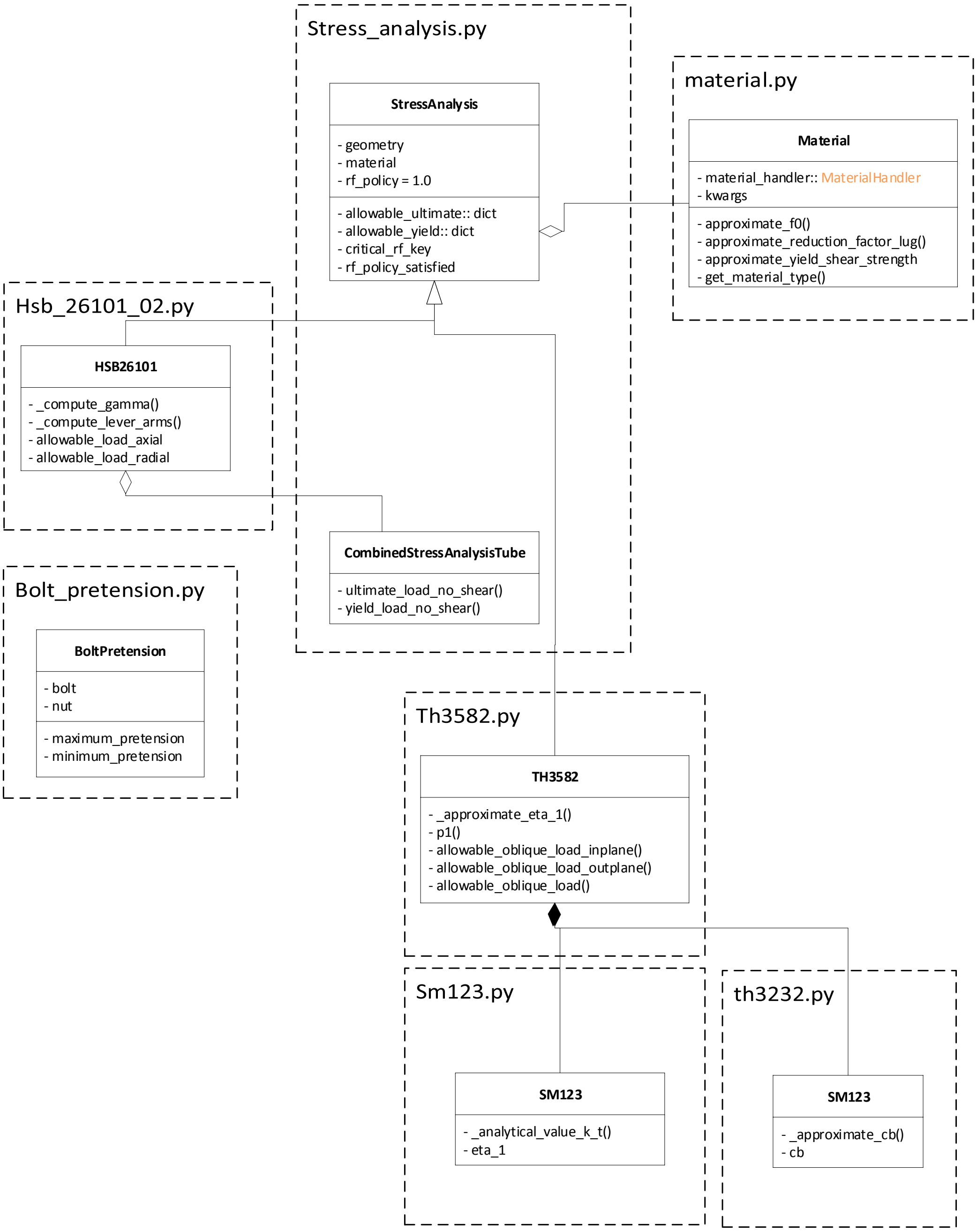


Figure 7.8: The most complicated stress analyses (for the bolt in file `hsb_26101_02.py` and lug in file `th3582.py`) all inherit from the generic `StressAnalysis` class and are defined from smaller stress analysis classes, such as `SM123`.

Finally, the stress analyses methods differ with respect to the ones used in HDOT. Below are the main differences for the bolt and lug stress analyses. For the bolt, the method implemented is identical to the one described in the HSB document [13], whereas the previous version assumed a full bolt and did not include the allowable bending strength approximation. For the lugs, the table 7.1 summarizes the steps in which the Hinge Generator lug stress analysis differs from HDOT lug stress analysis.

Geometry package

The UML diagram in figure 7.9 illustrates the inclusion of the geometry primitives into hinge components. The geometries are composed of smaller elements (e.g., `ChamferedHexagonalHead` for the bolt or `OuterRing` for the bearing) created from boolean operations from primitive solids. This way, those simpler geometries

Step	HDOT	Hinge Generator
Computation of η_1	Digitalized curve for one material	Analytical method [10]
Computation of η_4	Digitalized curve for one material	All curves digitalized + round for inexisting materials
Computation of η_5	Digitalized curve for one material	All curves digitalized + round for inexisting materials
Computation of C_b	-	Digitalized curve
Allowable load in z direction	Simplified formula $f(\sigma_b, d, t)$	Analytical method $f(C_b, \beta, d, t, \sigma_b, \tau_b)$

Table 7.1: Differences between the `HDOT` [18] lug stress analysis and the `Hinge Generator` stress analysis. A *digitalized curve* refers to an experimental curve modeled by a polynomial fit.

can be reused as building blocks to quickly create new hinge components geometries.

7.3. Libraries and platforms used

Building the model is enabled by the usage of several libraries for the Python programming language.

ParaPy

ParaPy[2] is a software framework that enables engineers to build parametric, rule-based software applications that automate engineering design processes. It defines its own, Python-based, declarative programming language, which includes dependency tracking, run-time caching and lazy evaluation. The adoption of ParaPy at Fokker Aerostructures, together with the author's previous experience with the software, are favorable to using ParaPy. ParaPy wraps the C++ geometry kernel OpenCascade[3] to enable the modeling of complicated, 3D geometries.

Pandas

The framework requires storing and retrieving a lot of data at different stages of the process, such as the material and standard parts databases for the Hinge Generator, and optimization results in the front-loaded database. Therefore, the question of data management needs to be addressed. A database is a data (piece of information) organized systematically, to make its collection easy. This choice profoundly affects the efficiency of the tool of the tool: its of run-time and memory usage, its scaling as the database grows larger, its ease of use and its ease of development.

Many different file formats and their associated query libraries exist. For instance, the XML (Extensible Markup Language) language and the python library `ElementTree`, or the SQL (Structured Query Language) language together with a SQL wrapper such as `SQLite` could be used as a data management framework.

Instead the Hinge Generator uses Pandas[4]. Pandas is an open-source library providing high-performance data structures and data analysis tools for the Python programming language. Below are the key points that encouraged the choice of `Pandas` instead of the options listed above.

- In practice, the `pandas` library offers very similar means to query and filter data as SQL[5].

- Pandas provides libraries to interface (read and write) to other file formats, such as `.xlsx` and `.json`. The former is important as the database needs to be easily readable by engineers, especially in the current context at Fokker. Developments toward automation are being made quickly, and adoption of those new tools by experts unfamiliar with coding is a significant concern.

- In terms of data analysis, operations on a `DataFrame`[6] are much more versatile than an SQL database [21]. This facilitates filtering and querying of data, which is used in many parts of the framework: for the organization of the standard parts in the Hinge Generator, for the creation of the combination tables for 2-phases search and the front-loading selection method.

[2] `www.parapy.nl`

[3] `https://www.opencascade.com/`

[4] `https://pandas.pydata.org/`

[5] See comparison with SQL in Pandas documentation: `https://pandas.pydata.org/pandas-docs/stable/comparison_with_sql.html`

[6] A 2D table - the main data structure of Pandas

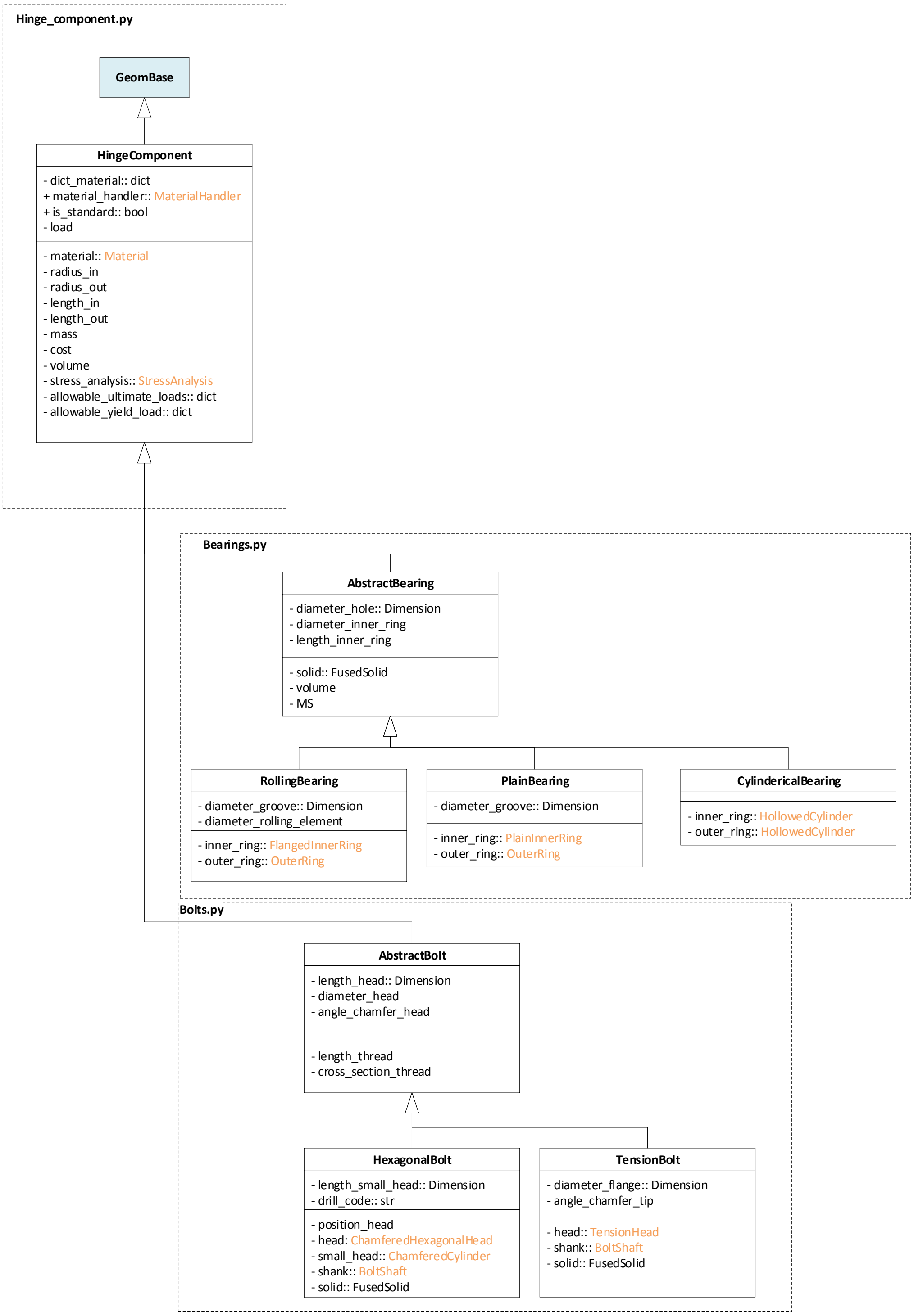


Figure 7.9: Every component of the hinge (bolt, bearing, lugs, etc.) inherits from the HingeComponent class. It is passed through abstract components classes (such as AbstractBearing), which themselves parent specific components geometries (such as RollingBearing).

Scipy and Numpy

Scipy includes many efficient numerical routines, such as routines for numerical integration and optimization. Numpy provides efficient data structures for matrix operations. Those libraries are used in the stress analyses and in the optimization.

Discussion on the use of third-party libraries

The libraries listed above enable the majority of the functions of the Hinge Generator. ParaPy and Pandas are especially necessary to the Hinge Generator. If, for any reason, one of the above libraries is no longer used at Fokker, substantial portions of the Hinge Generator would become obsolete. Likewise, if one of these library changes significantly (e.g., a functionality is deprecated or a bug remains unsolved for a long period of time), the parts of the code would need to be fixed.

Overall, this dependency problem is outweighed by the gain in development-time the libraries offer.

7.4. Remarks and recommendations for the code

The following section raises some specific points of discussion. Those remarks are not impairing the functioning of the code but could be relevant considering the code maintainability and run-time.

Single source of truth

Several design choices in the Hinge Generator were made with the goal of separating analyses, data and methods. This enables first to have engineers with different skills to work on different parts of the code, and second to have a more modular system. This design leads to a common drawback where frameworks and methods wrap and interface with each other.

For the hinge generator, it leads to the duplication of data, or to “multiple sources of truth”, which makes the code less flexible and harder to support and extend. For example, modifying only a geometry file (e.g., changing a variable name `bolt_radius` to `radius_bolt`) requires the exact same changes to be made in elements of the mapping and of the search method.

Optimization of lazy-evaluated attributes

The lazy-evaluation of ParaPy should be used to ensure that as few computations as possible are realized.

The bolt, bearing and bushings use the `DynamicType` ParaPy class, so that the appropriate geometry (e.g., `TensionBolt` or `HexagonalBolt`, depending on the bolt standard) is used depending on the selected standard. If any of the arguments on which the dynamic part depends is invalidated, the whole object is thrown away and must be re-evaluated.

The reading of the standard part databases is included into a single function that must be re-evaluated if the query changes. Although that works well for most of the standard parts, it is not optimal in the case of bolts and bushes as they are described by two tables. To read the characteristics of a bolt or bushing, both the length code and the diameter code must be provided. Therefore, during the SQP optimization, the length code must be assumed and then updated once the outer lug thickness has been determined. A more efficient, clearer option would be to separate the reading of the two tables and to only read the latter once the length code is known.

Dimension class

The `Dimension` class is a base class that could be used for engineering problems containing a dimension, such as a length in meters or a force in kilo-Newtons. This class also includes tolerances, so that the detailed fitting criteria for the hinge can be computed. The Dimension class contains five input slots: nominal value, lower bound, upper bound, tolerance and units.

Including the units and tolerances in the analyses can be valuable. During the stress analysis, it is useful to specify that the yield strength is given in mega pascals, while the internal load is given in kilo-Newtons. Likewise, dimensions come handy to compute areas and to account for worst-case dimensions during the computation of the allowable loads.

Using this class, the equation can be written plainly as `d3 = d1 + d2` in the code, without having to worry about the units, or tolerances. The listing 7.1 demonstrates the use of the Dimension class for the implicit unit conversion.

The listing 7.2 demonstrates the use of the Dimension class for checking the equality of two dimensions with tolerances.

The computing time difference between using a `Dimension` and `float` has not been evaluated. Replacing many float objects by lazy-evaluated ParaPy classes will definitely have adverse effects on the run-time of the

```
d1 = Dimension(units='mm', nom=2, tol=0.1)
d2 = Dimension(units='in', nom=0.4, tol=0.1)
d3 = d1 + d2
print('(%s) + (%s) = %s' % (d1, d2, d3))
>> (2.00 +/- 0.10 [mm]) + (0.40 +/- 0.10 [in]) = 12.16 +/- 2.54 [mm]
```

Listing 7.1: Using the `Dimension` class to perfrom an addition between two lengths with different units.

```
d1 = Dimension(units='mm', nom=5.0, ub=5.1)
d2 = Dimension(units='mm', nom=5.1)
res = d1 == d2
print('Is %s == %s? %s' % (d1, d2, res))
>> Is 5.00/5.00/5.10 [mm] == 5.10 [mm]? True
```

Listing 7.2: Using the `Dimension` class to check if two dimensions are equal. This feature is useful for the Hinge Generator, for example to check if a bearing with imperial dimensions (inches) fits with a bolt with metric dimensions (mm).

Hinge Generator. Moreover, though the examples `d3 = d1 + d2` and `d1 == d2` are simple, it could arguably be confusing because operations are happening implicitly, without the user's knowledge.

A compromise has to be found between level of detail, explicit coding and code maintainability. The problem with the `Dimension` class is that it forces its use in all the rest of the software, as it is the most basic data type that is passed between the objects. The dimension is then used in many parts of the code, even when there is no doubt about the validity of the operation in the first place. For this reason, the use of the `Dimension` class is not recommended.

8

Hinge Searcher

Chapter 3 proposed methods to solve the hinge optimization problem. This chapter gives further details on this implementation and its benefits and drawbacks.

8.1. Description of the optimization framework

This section gives a short overview of the framework developed to implement the two-step search methods. Figure 8.1 two separate objects are dedicated to the discrete and continuous part of the search.

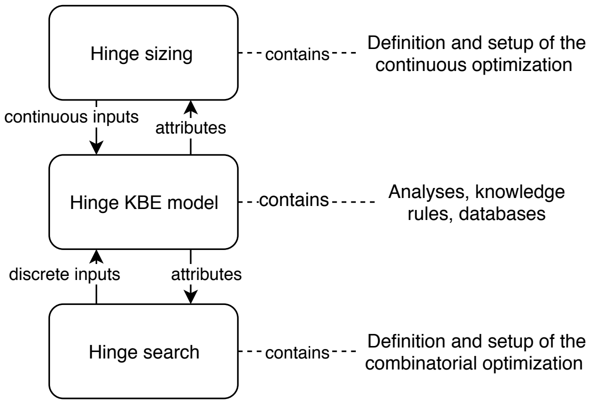

Hinge sizing
contains
Definition and setup of the continuous optimization
continuous inputs
attributes
Hinge KBE model
contains
Analyses, knowledge rules, databases
discrete inputs
attributes
Hinge search
contains
Definition and setup of the combinatorial optimization

Figure 8.1: Overview of the framework developed to implement 2-step search methods. The optimization setup, definitions and methods are separated from the KBE model. The discrete part of the problem is handled by an ad-hoc *Hinge search* class, while the continuous part of the problem is handled by a ParaMDO, which uses OpenMDAO and Scipy to carry out the optimization.

Two UML diagrams are given to further explain the framework. The UML diagram in figure 8.2 shows the definition of the continuous part of the problem created with the ParaMDO library, which wraps the optimization framework OpenMDAO, while the UML diagram in figure 8.3 illustrates the classes developed to perform the tree search detailed in chapter 3.

Continuous optimization framework

ParaMDO is an optimization framework is developed by Onodi [23]. It enables the integration of OpenMDAO[1], a framework for efficient multidisciplinary optimization, into ParaPy.

Figure 8.2 shows the UML diagram of the BaseHingeSizing class, which contains children of type paraMDO classes. It features a domain specific language to formulate optimization problems. It automatically collects the design variables, constraints and objective (see classes DesignVariable, InequalityConstraint and Objective in figure 8.2) from the KBE model. Then, it defines the OpenMDAO optimization workflow automatically.

[1] http://openmdao.org/

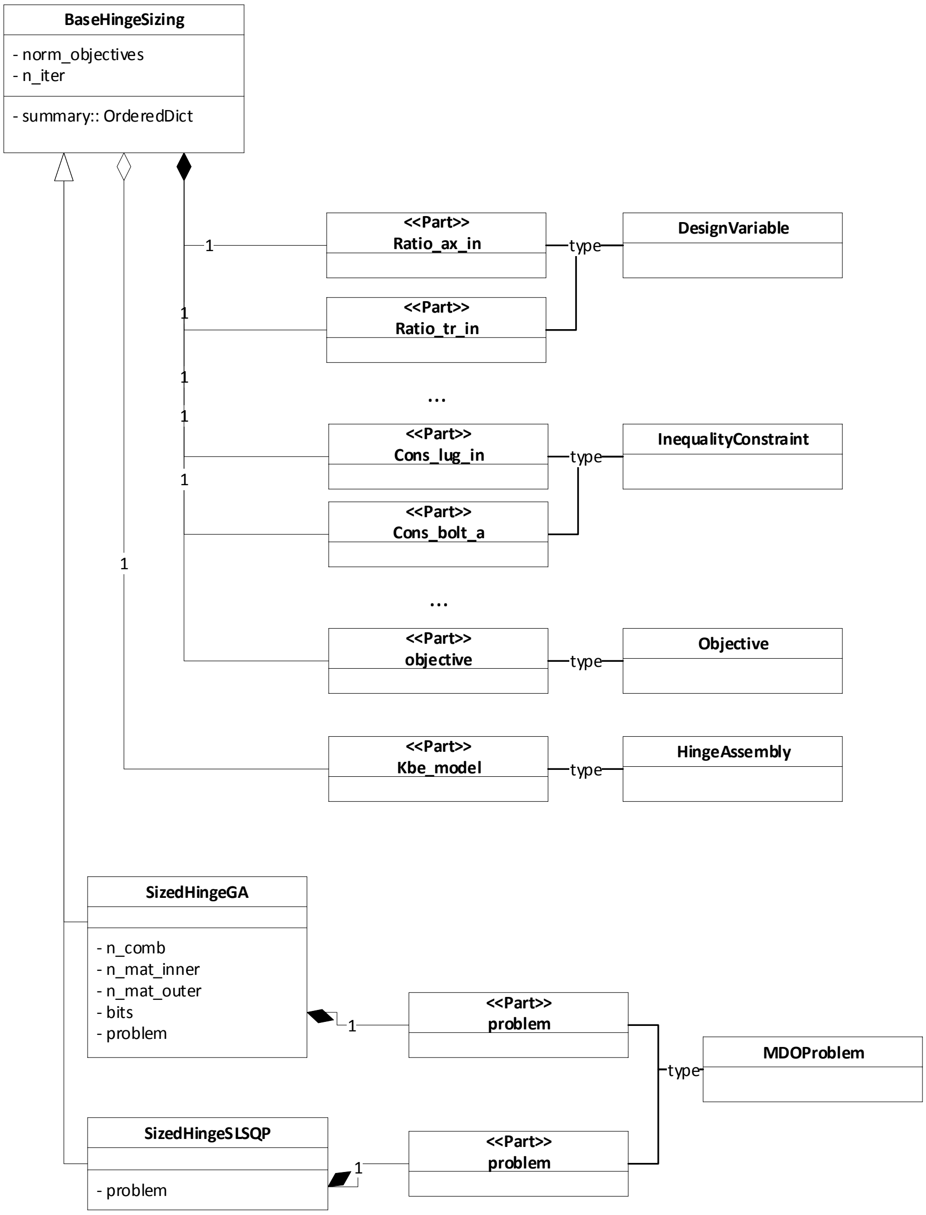


Figure 8.2: The generic continuous hinge optimization problem is defined in the `BaseHingeSizing` class, which defines the constraints, design variables and objectives with respect to the slots of the hinge assembly `model`. Those attributes are then collected in more specific problem definitions such as `SizedHingeSLSQP`.

The code in the listing 8.1 details how a design variable is defined with respect to the model. It highlights the conciseness and ease-of-use of the `ParaMDO` framework. Moreover, the code in the listing 8.1 shows the advantage of having all the parts of the framework (KBE model, optimization workflow definition, stress analyses) developed in Python.

ParaMDO also includes pre and post processing features for the user to interact with, through ParaPy's GUI. For example, the optimization method can be changed and checked directly from the GUI, without modifying the code. This makes trial-and-error optimization quicker to carry out.

This framework fits well in the context of this project, as it shortened the time required to experiment with different search methods. The only downside to `ParaMDO`'s use during this project is that it adds a dependency, as discussed in the chapter 7.1.

```
@Part
def ratio_tr_out(self):
    """
    :return: Design variable corresponding to the transverse eccentricity of the outer lug
    :rtype: DesignVariable
    """
    return DesignVariable(model_inputs=self.model.outer_lug.etr_over_d,
                          lower=TH3582.validity['e_over_d'][0],
                          upper=TH3582.validity['e_over_d'][1],
                          normalize=self.norm_dvs,
                          )
```

Listing 8.1: Definition of a design variable $e_{tr,ol}$ in the BaseHingeSizing class (see UML diagram 8.2). The design variable is linked to the model's input (outer_lug.etr_over_d). The upper and lower bounds ($e^L_{tr,ol}$ and $e^U_{tr,ol}$) are directly defined from the lug stress analysis class TH3582. The design variable can be normalized with the boolean input norm_dvs.

Discrete optimization framework

The interaction depicted earlier in figure 8.1 is also represented in the UML diagram in figure 8.3. The interaction is applied on the object HingeAssembly, which is a child of HingeSizing, itself child of BaseSearcher.

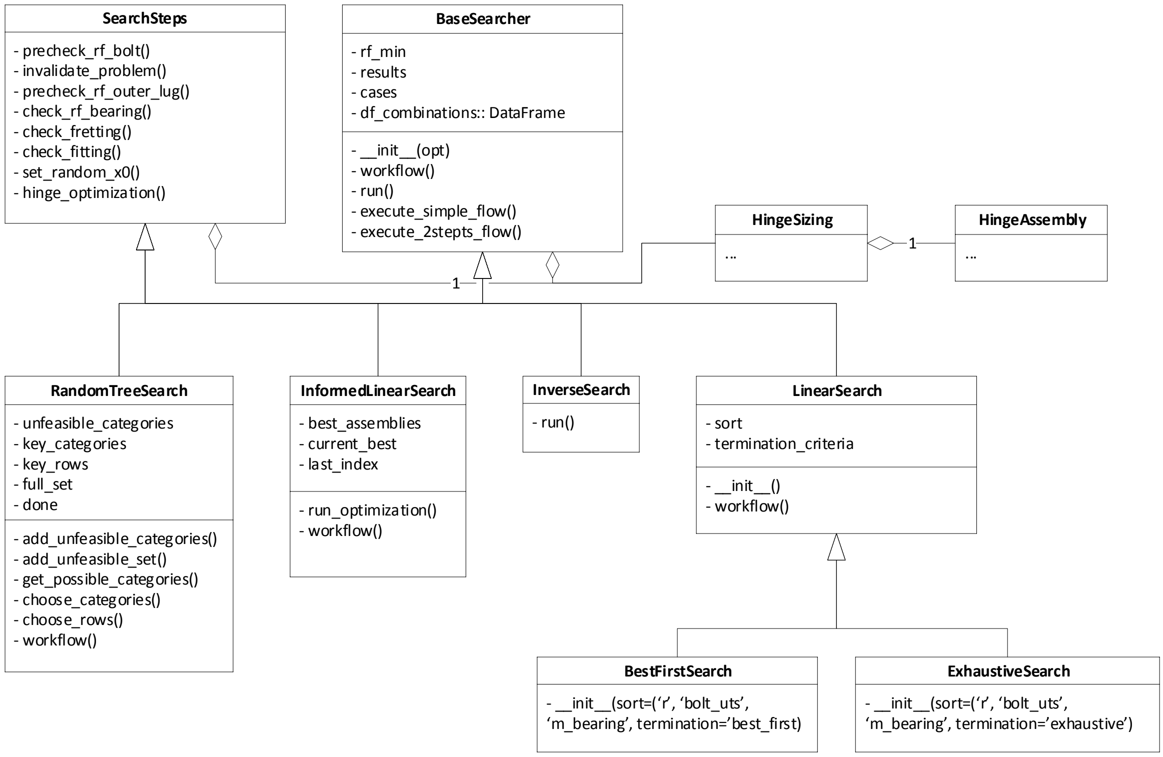


Figure 8.3: UML diagram of the optimization methods described in chapter 3, such as the ILS modeled in the class InformedLinearSearch. The search classes inherit from the class SearchSteps, which contains simple building bricks that compose more complicated search methods, and BaseSearcher which contains the methods to define search methods workflow.

In previous ad-hoc methods, nested if-then loops specific to the KBE model are defined. This framework, on the contrary, enables the quick creation of new search methods independently of the hinge KBE model. The main strength of the framework depicted in 8.3 is that new variations of search methods can be defined very simply from the existing data and methods already defined in the SearchSteps and BaseSearcher classes. It makes use of inheritance from the base classes to facilitate the creation, maintenance, and tests of new methods. The functions execute_simple_flow() and execute_2steps_flow() are used to formulate a sequence of instructions in a concise way. Using both of those features, new methods can be defined with a single class and often simply by overwriting the workflow method (see listing 8.2). Likewise, elements of random search are defined in RandomTreeSearch and could be re-used to develop a more extensive ad-hoc

```
flow = [
        self.precheck_rf_bearing,
        self.precheck_rf_bolt,
        self.precheck_rf_outer_lug,
        self.set_x0_hinge,
        self.run_slsqp_optimization,
        self.invalidate_problem,
]
```

Listing 8.2: If the search method constains a sequence of instructions, a simple list can be passed to the functions `execute_simple_flow()` and `execute_2steps_flow()`.

```
ils = InformedLinearSearch(mat_type_lug_out=['aluminium', 'steel'], radial_load=range(5, 46, 5))
ils.run()
ils.save_results('run_materials')
```

Listing 8.3: Example use of the ILS. The arguments `mat_type_lug_out` and `radial_load` are inputs of the hinge KBE model. By specifying those keywords, the search is automatically performed for full-factorial design of experiments.

search.

Any input from the KBE model can be set from optimization class. For instance, if one wishes to vary both the load magnitude and the material, the prototype of listing 8.3 can be used.

8.2. Strengths and weaknesses of the framework

Strengths The key advantages of the framework are the following:

Performance From the performance point of view, the framework is advantageous with respect to others (e.g., PIDO + KBE), because everything is operated within the same environment. No interface (reading and writing inputs and output files) is required to communicate between the elements, as every element is programmed in Python. Another key element of this framework is that the KBE model is only instantiated once. It enables the framework to leverage the lazy evaluation feature of the KBE model to only re-compute what is needed.

Ease of use The framework is simple but results in a powerful, efficient, high-level optimization and design of experiment framework that can be simply called in a few lines of code, as it is demonstrated in the listing 8.3.

Modularity The three main elements (discrete search, continuous search and KBE model) are modular. They can be used independently and exchanged to quickly develop new search methods, or apply search methods to different KBE models. Both the continuous and discrete part of the framework make it possible to modify and extend the continuous and discrete search methods.

Weaknesses The main drawback to this implementation is that the different parts of the project (KBE model, search framework, front-loading framework) all need to share the same keyword attributes. In order not to have those keywords, the functionality would have to be defined in the same framework, which would make the code less modular.

8.3. Recommendations

General recommendations are given in chapter 6. The following are in-depth points about the implementation:

- Sometimes, the SLSQP optimization launched by openMDAO will converge with an inequality constraint value that is not completely satisfied. For instance, it sporadically (about 1% of the time) happens that $\text{RF}_{ol} = 1.02$ and that the optimizer returns a successful value, when the inequality condition specified $\text{RF}_{ol} \geq 1.05$.

- The `invalidate_problem` method invalidates many of the attributes of the `HingeSizing` class. In principle, fewer attributes could be unvalidated.
- The classes `InformedLinearSearch` and `LinearSearch` were separated for development purposes. In principle, `InformedLinearSearch` should inherit from `LinearSearch` and implement the depth criteria.

9
Hinge Front-loading

Chapter 4 presented the hinge front-loading framework, which includes two main components. The data visualization framework (answers the sub-question "*How to represent the Pareto-optimal hinges in a way that enables trade-off decisions?*") is presented in section 9.1. Section 9.2 presents the implementation of the hinge selection method, which answers the sub-question "*How to obtain Pareto-optimal hinges as quickly as possible? (< 10 ms)*".

9.1. Description of the hinge data visualization framework

This section presents the hinge data visualization framework, contained in the `HingeApplication` class. A single abstract class `AbstractApplication` is defined, to contain the generic elements and functions used for the visualization. Most of the elements of the application are tailored to the front-loading hinge problem and are therefore defined in the `HingeApplication` class.

The main library enabling this interactive data visualization is `Bokeh`[1]. It generates JavaScript elements that are displayed in a web-browser but does so only with concise, high-level Python code. Those high-level Python objects are converted to JSON and used by the `BokehJS` library and can also be synchronized in the other direction. This way, Bokeh enables to use both the computation and query power of python and the UI and tools and visualizations of JavaScript and its libraries, see for instance `D3.js`[2] and `p5.js`[3].

The `HingeApplication` class wraps all the Bokeh objects, Pandas utility functions, and Tornado[4] loop. The `HingeApplication` class implements `Bokeh` objects, such as the `Figure` for the 2D plot and `Table` for the table of characteristics. This way, all the elements of the framework are self-contained, and the interactive application can be simply used by instantiating the class and then using the `start` method. The listing 9.1 shows how to start the application.

```
app = HingeApplication(fname='5-45kN.xlsx')
app.start()
```

Listing 9.1: The interface is instantiated with the file name of the front-loaded database as an argument. Once the `start` method is called, the application can be opened in a web browser, at the URL `http://localhost:5006/`.

The `hinge_fl` package contains:

- The `app_fl` package contains the interactive selection framework.
- The `hinge_selection` file implements the hinge selection method.
- The `post_processing` file implements methods to modify, analyze, import and export information contained in the front-loaded dataset.

[1] `https://bokeh.pydata.org/en/latest/`

[2] `https://d3js.org/`

[3] `https://p5js.org/`

[4] Tornado is a Python web application framework `https://www.tornadoweb.org/en/stable/`

```python
surrogate_hinge = HingesSelection(fname='5-45kN.xlsx')
hinge1 = surrogate_hinge.get_hinge(radial_load=24.5,
                                   mat_type_lug_out='aluminium',
                                   gap=1.0)
```

Listing 9.2: The selection method is wrapped in the class `HingeSelection`.

9.2. Description of the hinge selection method

As for the data visualization framework, the hinge selection method is contained in a single class, `HingeSelection`.

The functions of the `HingeSelection` class are the following:

- Provide the query functionality, e.g.:

  `Return the cheapest hinge for which` $F = 26\ kN$, $e_{max} < 25\ mm$, `bolt` $\neq NAS6703 - 6704$

  Should return that result from the front-loaded database as quickly as possible.
- Check if the Δx is admissible.
- Trigger an optimization if required.
- Update the database with the new results on the fly.

Each of those points is implemented as a class function in `HingeSelection`. As with the other classes presented thus far, the selection method is called by instantiating the class and calling the main method, as shown in the listing 9.2.

9.3. Recommendations

For the interactive visualization framework:

- The table and data elements of the visualization framework come out of synchronization after the first interaction with the widgets. The synchronization is re-established by hovering the mouse on the plot after selecting hinge assemblies in the table. The cause of this bug is unknown.
- Exporting the plots to vector formats (e.g., .svg and .PDF) could not be achieved due to the inability to install required libraries on the machine used for this project.
- A dynamic legend would enable a better understanding of the color classification.

For the selection method:

- The method enabling the query of the point according to the Δx is specific to this method. A more general method to query a specific row in a `DataFrame` according to several tolerances Δx_i present in the columns can be achieved.
- The overhead time of reading and writing large spreadsheet increases with the size of the database, as shown in table 4.1. During this project, Excel spreadsheets were used. Using a different format, such as CSV, might reduce both the overhead writing time and database file size.

A

Surrogate models for hinge components and materials

By performing simple data analysis on the material data sets and standard parts data sets reveals that the variables are linearly correlated. This can be seen for example in figure 3.3 for the standard bearings and in figure A.1 for the materials. This chapter presents a method to create a surrogate model of a material or of hinge components. In both of those cases, the variables linearly depend on each other.

Motivations

The surrogate model could be used:

1. To provide a heuristic in a hinge search method.
2. To assess the benefits of using custom machined components instead of standard ones. It would yield an optimal hinge, not the best combination of components in the existing database is.
3. To simplify an optimization problem including material variables. This problem was faced in [16] by using a GA but this solution proved to be time consuming.

Standard parts and material contain several characteristics that are required for the analyses. The goal of the surrogate model is to simplify the optimization problem by approximating all the characteristics from **one** descriptive variable. For example, the bearing in the hinge assembly is described by four dimensions. The bearing surrogate model approximates three dimensions:

$$\mathbf{x} = (l_{be,in}, l_{be,out}, r_{be,in}, r_{be,out}) \qquad \Rightarrow \qquad \begin{cases} \mathbf{x} = (l_{be,in}) \\ (l_{be,out}, r_{be,in}, r_{be,out}) = f(l_{be,in}) \end{cases} \tag{A.1}$$

Methodology

A first option is to have a *single* linear model per variable. A second option is to have a *multiple* linear model per variable. The difference between the two options can also be thought of by looking the plot in figure A.2. The first option (many linear regressions) consists of looking the off-diagonals sub-plots by themselves, whereas the second option (multiple linear regressions) consists of looking at the entire row.

The second option consistently yields an approximation with a higher coefficient of determination r^2. Therefore, for a surrogate model incorporating n variables, it is better to have n Multiple Linear Regression (MLR) models (one per row), each with $n-1$ explanatory variables, rather than having $(n^2-n)/2$ (Half the matrix above the diagonal) Single Regression models with one explanatory variable each.

Applying a multiple linear regression on each of the $n-1$ variables of the model will yield $n-1$ equations:

$$x_i = f(x_j) \qquad j \in [1,\dots,i-1,i+1,\dots,n], \quad i \in [1,\dots,k-1,k+1,\dots,n] \tag{A.2}$$

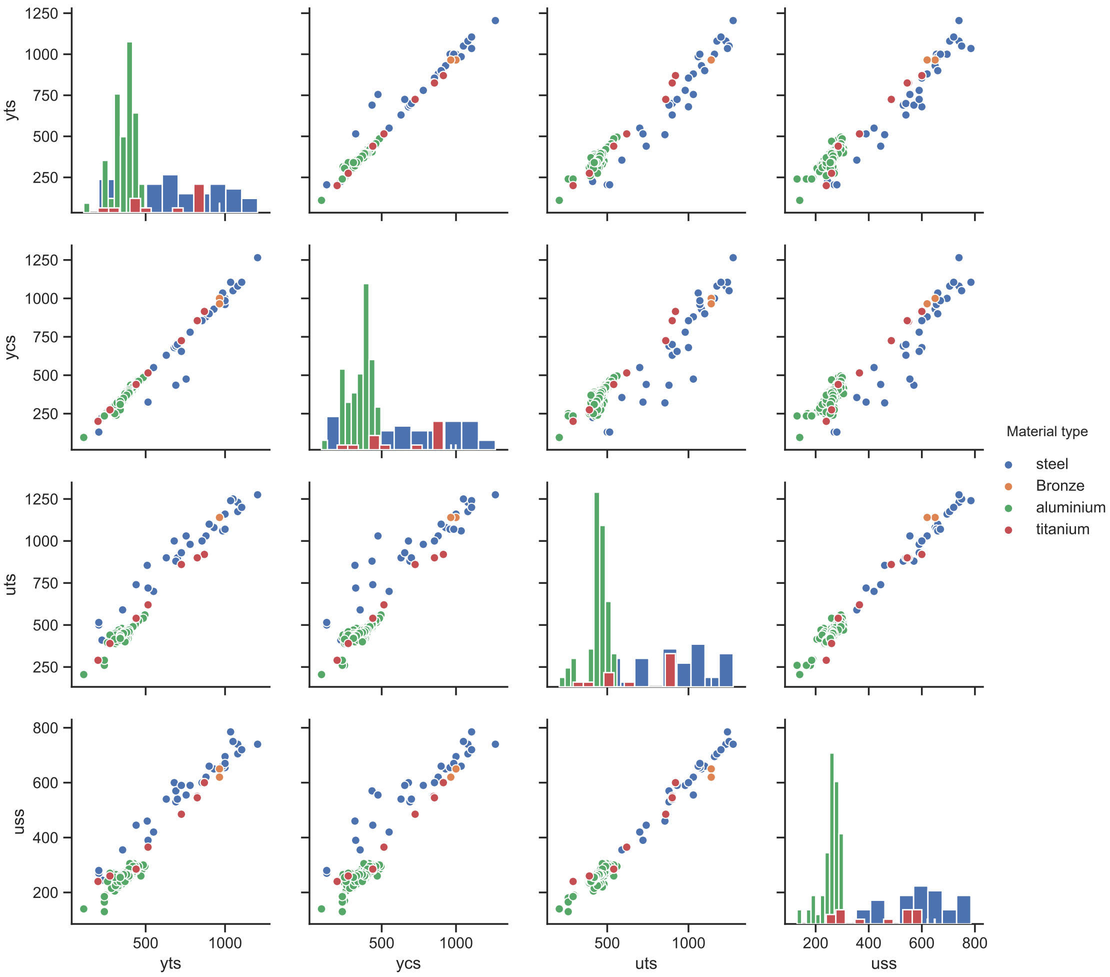


Figure A.1: Strengths in $[MPa]$ of different materials considered for the optimization. *yts*: Yield Tensile Strength, *ycs*: Yield Compressive Strength, *uts*: Ultimate Tensile Strength, *uss*: Ultimate Shear Strength.

Where k is the chosen descriptive variable. To have a surrogate model with only one descriptive variable as an input, those need to be transformed in order to have any of the variables as a function of the descriptive variable. Equation A.2 needs to be transformed into equation A.3:

$$x_i = f(x_k), i \in [1, \ldots, k-1, k+1, \ldots, n] \tag{A.3}$$

The following paragraphs show how to achieve this transformation, first with four variables and then in the general case of n variables.

Example with 4 variables For a surrogate model incorporating 4 variables:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & 0 & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & 0 & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \tag{A.4}$$

If chosen design variable is x_1, x_2, x_3 and x_4 need to be determined. The last three rows of equation A.4 are equivalent to:

$$\begin{aligned} x_2 &= a_{2,1}x_1 + a_{2,3}x_3 + a_{2,4}x_4 + b_2 \\ x_3 &= a_{3,1}x_1 + a_{3,2}x_2 + a_{3,4}x_4 + b_3 \\ x_4 &= a_{4,1}x_1 + a_{4,2}x_2 + a_{4,3}x_3 + b_4 \end{aligned} \tag{A.5}$$

As x_1, $a_{i,j}$ and b_i are known, equation A.5 is a system of 3 equations with 3 unknowns. Equation A.5 is equivalent to:

$$\begin{pmatrix} x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & a_{2,3} & a_{2,4} \\ a_{3,2} & 0 & a_{3,4} \\ a_{4,2} & a_{4,3} & 0 \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} b_2 + a_{2,1}x_1 \\ b_3 + a_{3,1}x_1 \\ b_4 + a_{4,1}x_1 \end{pmatrix} \tag{A.6}$$

This Multiple Linear Regression System (MLRS) can be solved.

General case The previous example (4 variables and choosing the first one as design variable) can be generalized. Considering a surrogate model with n variables, with the chosen explanatory variable $k \in [1, n]$:

$$X = \underbrace{\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}}_{n\times 1} \qquad A = \underbrace{\begin{pmatrix} 0 & & a_{1,n} \\ & \ddots & \\ a_{n,1} & & 0 \end{pmatrix}}_{n\times n} \qquad B = \underbrace{\begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}}_{n\times 1} \tag{A.7}$$

The surrogate model generated is a set of n MLR equations:

$$X = A \cdot X + B \quad \Rightarrow (I_n - A) \cdot X = B \tag{A.8}$$

The row k of the chosen explanatory variable is removed and $A \cdot X$ is subtracted from equation A.8. Then, the last step is to solve the linear matrix equation $A'X = B'$, with:

$$X = \underbrace{\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}}_{(n-1)\times 1} \qquad A' = \underbrace{\begin{pmatrix} 1 & & -a_{1,n} \\ & \ddots & \\ -a_{n,1} & & 1 \end{pmatrix}}_{(n-1)\times(n-1)} \qquad B' = \underbrace{\begin{pmatrix} b_1 + a_{1,k}x_k \\ \vdots \\ b_n + a_{n,k}x_k \end{pmatrix}}_{(n-1)\times 1} \tag{A.9}$$

Material surrogate model

To create a material surrogate model, 12 material properties need to be modeled from one property. Most of the material properties (such as the strengths shown in figure A.1) vary linearly with one another, in which case the MLRS methodology presented in the previous subsection can be used. Other material properties (such as the cost and mass density) depend on the material type (aluminium, steel, bronze, titanium).

Table A.1 summarizes which material characteristics can be modeled, either with a MLRS or with a simple if-then rule (e.g., `material_type = aluminium`, then $\rho = 2700\ kg/m^3$)

Discussions

The surrogate material has been implemented in the Hinge Generator with the `SurrogateMaterial` class, a subclass of `Material`. The listing A.1 gives an example of a surrogate material definition. However, it has not been implemented in an optimization algorithm.

```python
x1 = 430
x2 = 'aluminium'
x_material = SurrogateMaterial(des_vars={'uts': x1, 'material_type': x2})
print(x_material.yts)
>> 421.5
```

Listing A.1: Implementation of the surrogate material methodology in the `Hinge Generator`. The `SurrogateMaterial` can easily be used in an optimization by setting two design variables x_1 and x_2.

Characteristic	**Given by**	**Used for**	**Model type**
Material type	x_1	TH3582	Explanatory variable 1
Ultimate tensile strength	x_2	SM123, TH3582, HSB	Explanatory variable 2
Ultimate shear strength	x_2	TH3582, HSB	MLRS
Yield tensile strength	x_2	SM123, TH3582, HSB	MLRS
Yield compressive strength	x_2	HSB	MLRS
Bearing ultimate strength (b_us_15)	x_2	TH3582	MLRS
Bearing ultimate strength (b_us_20)	x_2	TH3582	MLRS
Ultimate compressive strength	uts	HSB	Fokker rule
Yield shear strength	ycs, yts	HSB	Fokker rule
Young modulus	x_1	SM123	If-then
Elongation at failure	x_1, x_2	SM123	MLRS
Density	x_1	weight evaluation	If-then
Cost density	x_1	cost evaluation	If-then

Table A.1: Summary of the material surrogate model. The 11 material properties can be modeled from two parameters: the material type and the ultimate tensile strength.

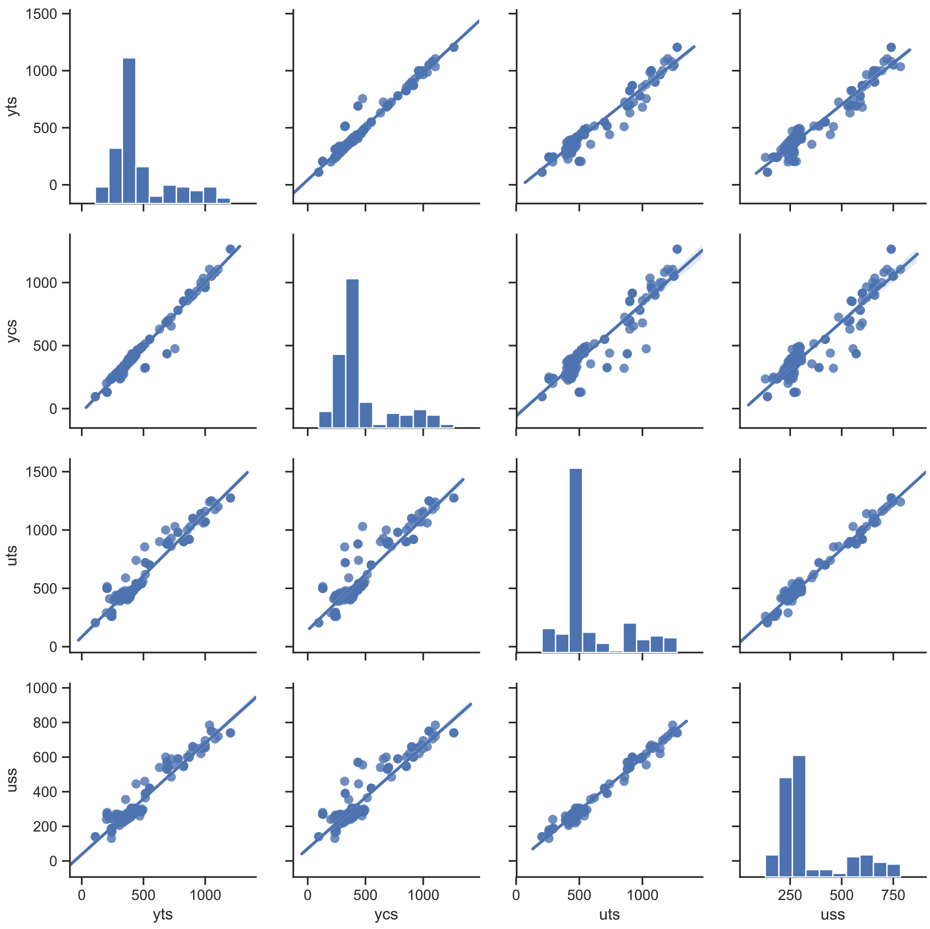


Figure A.2: Linear regressions on the strengths in [MPa] of the materials considered for the optimization. This figure corresponds to the case where the material type (indicated by the color legend in figure A.1) is not given and the linear regression is done considering all the materials.

Although the results cannot be discussed, this method would certainly be a useful heuristic to speed up the material search process during an optimization.

Bibliography

[1] ATC Aerospace. *The Aerospace Supply Chain*, 2014. `http://www.argoturbo.com/blog/the-aerospace-supply-chain` [Accessed: 20.04.2018].

[2] Marek Antosiewicz, Grzegorz Koloch, and Bogumil Kaminki. Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed .

[3] Grady Booch. *The unified modeling language user guide*. Pearson Education India, 2005.

[4] Hugh C Briggs. A survey of integrated tools for air vehicle design, part ii. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0803, 2015.

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[6] Fokker Engineering Detailed Design. *Selection of tolerances and fits for shafts and holes (FE-0017)*, 2000.

[7] Fokker Engineering Detailed Design. *Housing and shaft dimensions for bearings (FE-0129)*, 2000.

[8] Fokker Engineering Detailed Design. *Wrench Torques for bolted and screwed joints (FE-0030)*, 2000.

[9] Fokker. Fokker Technologies Corporate Presentation , 2015.

[10] Fokker-confidential. *A method to calculate the allowable tension fracture load of a lug (SM123)*, 1982.

[11] Fokker-confidential. *Allowable bending moment of metallic circular tubes (HSB52130*, 2004.

[12] Fokker-confidential. *Bending strength in the plastic range (TH3.232)*, 2008.

[13] Fokker-confidential. *Fokker Technical Handbook - Static failure of bolts in metallic double shear lugs*, 2011.

[14] Fokker-confidential. *Fokker Technical Handbook - Allowable loads of lugs*, 2012.

[15] Richard L Fox. *Optimization methods for engineering design*. Addison-Wesley Pub. Co., 1971.

[16] MFM Hoogreef. *Advise, Formalize and Integrate MDO Architectures: A Methodology and Implementation*. PhD thesis, 2017.

[17] Tim Janssen. Personal interviews conducted with tim janssen, stress methodology lead at fokker aerostructures, 2018.

[18] Akshay Raju Kulkarni. Development of a knowledge based engineering tool to support fin-rudder interface design and optimization. Master's thesis, Delft University of Technology, 2015.

[19] Gianfranco La Rocca. Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design . *Advanced engineering informatics*, 26(2):159–179, 2012.

[20] Joaquim RRA Martins and Andrew B Lambe. Multidisciplinary design optimization: a survey of architectures . *AIAA journal*, 51(9):2049–2075, 2013.

[21] Wes McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, pages 1–9, 2011.

[22] NASA. *Criteria for preloaded bolts, NSTS 08307*, 1998. `https://snebulos.mit.edu/projects/reference/NASA-Generic/NSTS_08307_RevA.pdf` [Accessed: 10.12.2018].

[23] P. Onodi. Integrating multidisciplinary design optimization and knowledge based engineering. Master's thesis, Delft University of Technology, 2019.

[24] S Rajeev and CS Krishnamoorthy. Discrete optimization of structures using genetic algorithms. *Journal of structural engineering*, 118(5):1233–1250, 1992.

[25] Akshay Raju Kulkarni, Maurice Hoogreef, and Gianfranco La Rocca. *Combining semantic web technologies and KBE to solve industrial MDO problems* . AIAA, 2017. doi: 10.2514/6.2017-3823.

[26] Akshay Raju Kulkarni, Gianfranco La Rocca, Tobie van den Berg, and Reinier Dijk. A knowledge based engineering tool to support front-loading and multidisciplinary design optimization of the fin-rudder interface. *To be published*, (232):1–20, 2017.

[27] J. Page Risueno. Development and implementation of a cost assessment methodology for multidisciplinary design optimization of aircraft components. Master's thesis, Delft University of Technology, 2017.

[28] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach* , volume 2. Prentice hall Upper Saddle River, 2003.

[29] Alexander Schrijver. *A course in combinatorial optimization* . TU Delft, 2000.

[30] Jaroslaw Sobieszczanski-Sobieski, Alan Morris, and Michel van Tooren. *Multidisciplinary design optimization supported by knowledge based engineering*. John Wiley & Sons, 2015.

[31] Stefan Thomke and Takahiro Fujimoto. The effect of "front-loading" problem-solving on product development performance . *Journal of product innovation management*, 17(2):128–142, 2000.

[32] Tobie van den Berg. *Harnessing the potential of Knowledge Based Engineering in manufacturing design* . PhD thesis, 2013.

[33] Tobie van den Berg, Ton van der Laan, and Martijn van Rij. Aerospace solution optimization through front loading techniques. 650:1–12, 2015.

[34] Stefan van der Elst. IDEALISM D2.3: Industrial service-oriented process methodology . 2017.

[35] Jan Hein van der Flier. Hinges knowledge section, rudder in a month intranet, 2016. [Accessed: 20.04.2018].

[36] Ton van der Laan. *Knowledge based engineering support for aircraft component design* . PhD thesis, 2008.

[37] Ton van der Laan, Luc Hootsmans, and Marco Panzeri. Aerospace Europe 6th CEAS Conference Robust optimization of a rudder hinge system taking into account uncertainty in Airframe parameters Aerospace Europe 6th CEAS Conference . (957):1–12, 2017.

[38] Martijn van Rij. Personal interviews conducted with martijn van rij, manager design engineering at fokker aerostructures, 2018.

[39] K.B van Zomeren. Assessment of design optimization strategies on a composite rudder using a knowledge-based engineering tool. Master's thesis, Delft University of Technology, 2017.