Knowledge-Based Engineering Approach to the Finite Element Analysis of Fuselage Structures

S. D. De Smedt

Master of Science Thesis

TU Delft Delft University of Technology

Faculty of Aerospace Engineering

Knowledge-Based Engineering Approach to the Finite Element Analysis of Fuselage Structures

Master of Science Thesis

For the degree of Master of Science in System Engineering & Aircraft Design at Delft University of Technology

S. D. De Smedt

December 3, 2014

Faculty of Aerospace Engineering (LR) · Delft University of Technology

Word count: 29894 words
Thesis registration number: 249#14#MT#SEAD-FPP

Delft University of Technology
Department of
Aerospace Structures & Design Methodologies

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering (LR) for acceptance a thesis entitled

Knowledge-Based Engineering Approach to the Finite Element Analysis of Fuselage Structures

by

S. D. De Smedt

in partial fulfillment of the requirements for the degree of

Master of Science System Engineering & Aircraft Design

Dated: December 3, 2014

Supervisor(s):

Dr.Ir. R. Vos

Reader(s):

Dr.Ir. L. Veldhuis

Dr.Ir. D. Gransden

M.Sc.Ir. Durk Steenhuizen

Abstract

With the predicted growth in demand for air transport and the industry's goals of reducing the environmental impact of aircraft, researchers have been looking at new ways of coping with these challenges. Studies have shown that the blended wing body concept could successfully fulfill the industry's demands through a combination of high aerodynamic efficiency, low structural weight and an increase in useable cabin space for passenger allocation. However, one of the challenges faced in the design of blended wing body aircraft for passenger transportation is the pressurization of the non-circular center body.

A relatively new solution to this problem is the oval fuselage. For this concept an inside-out approach is used to define a more structurally focused shape which distinguishes itself from other solutions through an unobstructed cabin space allowing for higher cabin flexibility and passenger comfort.

In order to assess the potential of a new concept it is, amongst other things, essential to obtain a good initial estimation of its weight. To provide a general method of determining the weight of the fuselage structure during the conceptual design phase a semi-analytical weight estimation method was developed. From its application, promising results were found for the correspondence with the traditional class weight estimations methods and the possibilities the oval fuselage concept offers for future aircraft design.

During the development of the semi-analytical weight estimation method the validation of its results was limited to a finite element analysis of a quasi-two-dimensional fuselage section. However, such an analysis does not capture the influences and interactions of the oval fuselage components on a three-dimensional level. Therefore, throughout this report an evaluation is made, based on a three-dimensional finite element analysis, of whether the semi-analytical weight estimation method can be used as a conceptual sizing method for conventional and non-conventional aircraft with an oval fuselage. This validation method is based on a comparison between the load results from the semi-analytical weight estimation method and those obtained from the finite element analysis of a complete oval fuselage structure.

In order to eliminate the non-creative and time-consuming tasks related to the setup of finite element models and provide an aid in future higher-fidelity optimization a Knowledge-Based engineering project approach was used for this thesis for which the Automated Finite element

model Generator was developed. This tool automatically translates the conceptual design data into an analysis ready finite element model.

From the validation a strong correspondence of the line loads in the lateral/hoop direction is found. This may be related to the fact that the loads in this direction almost exclusively consist of the differential pressurization loads which simplifies the complex three-dimensional situation to a quasi-two-dimensional analysis which closely resembles the analysis carried out in the semi-analytical weight estimation method. The comparison of the loads in the longitudinal and shear direction showed more significant differences. For the outer skin the differences in longitudinal line loads are mainly related to the difficulties with obtaining a longitudinal balance for the oval fuselage. This is a result of the inaccuracies in the aerodynamic loads of the fuselage structure as calculated by the semi-analytical weight estimation method. For the trapezoidal box, the most significant differences were found in the loads of the floor surface. These can be related to the excessive deformations the floor surface undergoes which were not accounted for in the semi-analytical weight estimation method.

Due to the large differences found in the results of the longitudinal and shear loads, it is at this stage of the research difficult to draw a final conclusion with respect to the accuracy of the semi-analytical weight estimation method.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms

AFG	Automated Finite element model Generator
BC	boundary condition
BWB	Blended Wing Body
CAD	Computer Aided Design
FE	Finite Element
FEA	Finite Element Analysis
GenDL	General Descriptive Language
KBE	Knowledge-based Engineering
LE	Leading Edge
OML	Outer Mold Line
PCL	Patran Command Language
SAWE	Semi-Analytical Weight Estimation
TE	Trailing Edge
UML	Unified Modeling Language
XML	Extensible Markup Language

List of Latin Symbols

A	Aspect Ratio	[-]
A	Cross-sectional area	[m^2]
AftRatio	Aft ratio of the fuselage	[-]
B_s	Bending stiffness	[N/m^2]
C	Buckling coefficient	[-]
C_{D0}	Zero-lift drag coefficient	[-]
c_{torus}	Distance between y-axis and torus center-line	[m]
d	Depth	[m]

E	Youngs modulus	[N/m^2]
e	Oswald factor	[-]
F	Force	[N]
G	Shear modulus	[N/m^2]
h_1	Distance between the top of the cross-section's outer shell and cabin ceiling	[m]
$h_{1,3\text{-min}}$	Minimum value of h_1 and h_3	[m]
h_2	Cabin height	[m]
h_3	Distance between the bottom of the cross-section's outer shell and cabin floor	[m]
h_{aft}	Vertical displacement of the aft tip	[m]
h_{cabin}	Height of the cabin	[m]
h_{fuselage}	Height of the center section	[m]
h_{nose}	Vertical displacement of the nose tip	[m]
l_{aft}	Length of the aft section	[m]
$l_{\text{aft,min}}$	Minimum length of the aft section	[m]
l_{cabin}	Length of the cabin	[m]
L/D	Lift-to-drag ratio	[-]
l_{fuselage}	Total length of the fuselage	[m]
l_{nose}	Length of the nose section	[m]
$l_{\text{nose,min}}$	Minimum length of the nose section	[m]
n	Index	[-]
N	Normal line load	[N/m]
p	Pressure	[N/m^2]
q	Shear flow	[N/m]
r	Radius	[m]
S	Surface area	[m^2]
t	Thickness	[m]
t	Time	[sec]
w_c	Ceiling width	[m]
w_{cf}	Width cargo floor	[m]
W_{f}	Fuel weight	[N]
w_f	Floor width	[m]
W_{OE}	Operational empty weght	[N]
W_{payload}	Payload weight	[N]
x	Lateral axis	[-]
y	Longitudinal axis	[-]
z	Vertical axis	[-]
z_{floor}	Vertical position of the cabin floor	[m]

List of Greek Symbols

Δ	Difference operator	[-]
δ	Displacement	[m]
ϵ	Finesse ratio of the fuselage's longitudinal cross-section	[-]
ζ	Droop ratio of the fuselage's longitudinal cross-section	[-]

θ	Orientation/Angle	[rad]
σ	Normal stress	[N/m^2]
τ	Shear stress	[N/m^2]
φ	Angle spanned by circular arc	[rad]
ψ	Poisson's ratio	[-]

Sub- & Superscripts

aft	With relation to the aft part of the fuselage
belly	With relation to the lower curve of the airfoil
c	With relation to the core of a sandwhich structure
cabin	With relation to the cabin
cf	With relation to the cargo floor
crit	critical
crown	With relation to the upper curve of the airfoil
end	End
f	With relation to the facing of a sandwhich structure
f	Fuel
floor	With relation to the floor of the cabin area
fuselage	With relation to the fuselage
lat	Lateral direction
long	Longitudinal direction
max	Maximum
min	Minimum
norm	Normalized
nose	With relation to the nose part of the fuselage
OE	Operational empty
oriented	Not perpendicular to the longitudinal axis
oval	With relation to an oval cross-section
overall	Out of a range of values
payload	Payload
pb	With relation to the pressure bulkhead
start	Start
straight	Perpendicular to the longitdudinal axis

Part I

Thesis Report

Chapter 1

Introduction

With international civil air travel expected to increase by 30% from 2011 to 2016 [1] and long-term forecasts saying that scheduled passenger air traffic is to double from 2.7 billion in 2011 to 6 billion in 2030 [2], the question arises whether the current generation of aircraft will be able to cope with this predicted growth. As a result of this the industry is, now more than ever, expressing its concerns with regard to the environmental impact of aircraft. For the most part, these concerns are related to their noise and fuel pollution. With respect to the latter, the International Air Transport Association stated [3] that the industry aims to reduce the net CO_2 emissions from aircraft by half by 2050 when compared to 2005. Furthermore, they want to achieve an average fuel efficiency improvement of 1.5% per year up until 2020 to, from then onwards, achieve a carbon-neutral growth. With the ongoing increase in demand for air transport and the industry's goals to decrease the environmental impact of aircraft it seems the need for change is ever present.

Motivated by this, much work has gone into the research and design of a replacement of the conventional aircraft over the past years. One of the concepts that has been studied extensively is the Blended Wing Body (BWB) aircraft. When compared to a conventional aircraft sized for the same mission BWB aircraft have shown very promising results with respect to weight reduction, improved aerodynamic efficiency and lower fuel burn [4, 5]. This relatively new concept may be seen as a hybrid between a conventional aircraft and a pure flying wing in the sense that the wings blend into a fuselage section that also contributes to the lifting capabilities of the complete aircraft. In order to provide this additional lift, the shape of the centerbody of BWB aircraft is predominantly dictated by the aerodynamic design. As a result of this the fuselage geometries of these aircraft steer away from the traditional cylindrical shape and take on non-circular forms.

Although having been used in the very early days of air travel, the use of non-circular fuselages almost disappeared with the advent of pressurized passengers cabins during which the circular fuselage became the standard. The reason for this is that when pressurizing a circular fuselage, the outer skin will be subjected to a purely tensile stress state while for non-circular fuselages, due to the absence of rotational symmetry on its cross-section, additional bending stresses will be present in the skin.

With the promising results offered by the BWB aircraft, the research into finding methods of pressurizing non-circular fuselages were restarted. Two main concepts that are already in the later stages of both research and development are the integrated panel concept [6] and the segregated multi-bubble concept [7]. Although both concepts successfully resolve the problem of pressurizing a non-circular fuselage one of the main disadvantages of both these concepts is that the cabin space itself is obstructed by vertical poles and/or walls that are required to carry part of the pressurization loading and/or its resulting stresses. This significantly reduces the cabin flexibility and interior options of both these fuselage concepts.

A relatively new fuselage concept that aims to counter the problem faced by the two aforementioned solutions is the oval fuselage concept [8]. Unlike the other concepts there is no need for the vertical pillars and/or walls inside the cabin space to carry part of the pressurization loads. This is achieved by pressurizing the aerodynamic surface, which is formed by tangentially connecting four circular arcs, and having a trapezoidal box, which also functions as the cabin space, carry the reaction loads that are created due to the pressurization of this shell. The magnitude of the reaction load at a certain corner of the trapezoidal box depends on the radii of the circular arcs that come together at this location. By not having any pillars and or walls in the trapezoidal box area, the oval fuselage provides higher cabin flexibility with respect to interior configurations and an increased feeling of comfort for the passengers.

One of the challenges faced with these new fuselage designs is that the standard methods used to carry out the initial sizing and weight estimation do not apply. The reason being that these methods rely mostly on the use of empirical analysis to determine the initial estimates. However, seeing as the availability of comparable data for novel concepts is very limited to non-existent these methods cannot be applied to these unconventional designs. As a way to overcome this problem, Schmidt and Vos [8] proposed a new semi-analytical weight estimation method to standardize this process for conventional and oval fuselages. From their research it became clear that this method's results had a high correspondence to those of the empirical methods from open literature and showed the potential of the oval fuselage concept as a solution to the pressurization of non-circular fuselages.

As a means to validate the results obtained from the SAWE method a Finite Element Analysis (FEA) of an oval section was carried out during its development. This Finite Element (FE) model represented a quasi-two-dimensional fuselage section contained within the extent of the cabin area. From this FEA it was concluded that the assumptions made in the SAWE method are valid and its results accurate. However, due to the quasi-two-dimensional nature of this FEA the three-dimensional influences and interactions of the different oval fuselage components were not taken into account.

With the promising results obtained during the development of this semi-analytical method the need for a higher fidelity validation is crucial to ensure the accuracy of the SAWE results and provide an indication of the feasibility of the oval fuselage concept itself. Therefore a validation is described within this thesis report based on the FEA of the complete oval fuselage and all of its structural components.

At the start of this thesis it was quickly realized that a specific project approach would be required to provide an efficient analysis process. When considering the lack of comparable analysis and empirical data for oval fuselages, the relative complexity of its geometry and the large number of tests required for both the FE model verification and the SAWE method validation the manual setup of the FE models is deemed highly inefficient. Therefore, a

Knowledge-based Engineering (KBE) approach is used for this project to allow for the conceptual design results, as produced by the SAWE method, to be automatically translated into an analysis ready FE model and overcome the non-creative and time-consuming tasks related to its setup. In theory, this would mean that the user should simply run the KBE tool and hereafter be able to carry out the FEA.

With the information provided above, the goal of this thesis project can be formulated as:

Verify, based on the use of finite element analysis, the semi-analytical weight estimation method's results for aircraft with oval fuselages through the development and use of a Knowledge-based Engineering tool.

The report that follows provides a detailed description of the work done to realize this goal and it is divided into two main parts. The first part is an elaborate discussion of the higher fidelity validation carried out during this thesis project and the second provides a manual explaining the operation and functioning of the KBE tool created.

At the start of Part I, Chapter 2 provides a brief explanation of the state-of-the-art of BWB aircraft, non-circular fuselages, the SAWE method and the parametrization of the oval fuselage. Hereafter, the complete KBE tool and the modules it contains are discussed in Chapter 3. The actual validation of the SAWE method is carried out in Chapter 4 where first the test-case is described followed by a verification of the FE model and a comparison of the results from both methods. Finally, in Chapter 5, a conclusion is drawn and a number of recommendations are given with respect to this project and its outcome.

For Part II, firstly a short introduction to the manual is given in Chapter 6 followed by a discussion of the separate modules in Chapter 7. Within the latter chapter the module operation, input/output, module architecture and known errors, limitations and assumptions are explained for each module individually to provide the reader with all the information needed to operate and potentially modify/expand the KBE application.

Chapter 2

State-of-the-art

Some additional background information is provided about the BWB aircraft, the state-of-the-art of non-circular fuselages and the idea behind the SAWE method in order to further clarify the contribution and placement of this project into the overall body-of-knowledge. Hereafter, the parametrization of the oval fuselage is explained. The reason for this is twofold. Firstly, this provides the reader with a comprehensive explanation of the oval fuselage concept and secondly it allows for the differences between the parametrization used in the SAWE method and that used in this thesis project to be highlighted.

2-1 History of the Blended Wing Body

When considering the design evolutions with respect to the modern subsonic jet transport aircraft, it may be said that over the past 50 years no significant deviations from the conventional design have occurred. The conventional aircraft consists of a cylindrical fuselage with swept wings, a tail and wing mounted engines and pylons. It was this realization that inspired the McDonnel Douglas design study in 1988 in which novel concepts were to be investigated. What resulted was a three-year study in which the BWB concept came to be.

During this study, a comparison between the BWB concept, as seen in figure 2-1, and a conventional aircraft was carried out which showed that when designed for the same mission the BWB concept is lighter, has an improved lift-to-drag ratio and a significantly lower fuel consumption [4].

In another study [9] a similar comparison between the performance and characteristics of a conventional and BWB aircraft was carried out. This study obtained similar findings to the one carried out by Mc Donnel Douglas and in addition emphasized the decreased technical risk, higher commonality in production and operation and higher level of comfort and safety of BWB aircraft.

With the promising results obtained from the preliminary studies research was continued to investigate and improve several different aspects of the BWB aircraft. These aspects, that

make the BWB aircraft a potential replacement of the conventional aircraft, will be briefly discussed in the following paragraphs.

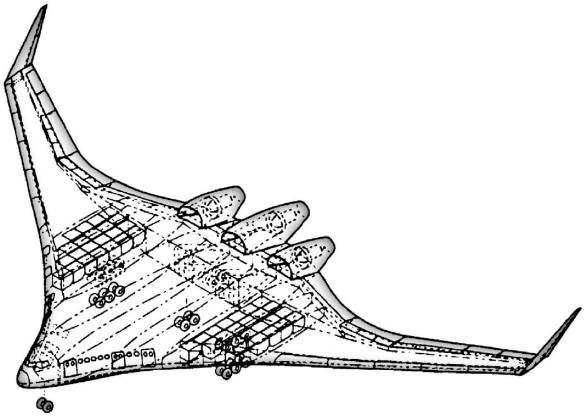


Figure 2-1: Second generation BWB concept as envisioned by R. H. Liebeck[4]

As mentioned in the introduction, one of the most pressing design objectives for future aircraft is the reduction in fuel consumption. Before addressing the ways in which the BWB aircraft's characteristics help improve upon the current consumption levels of conventional aircraft the design aspects related to the fuel required by an aircraft are briefly mentioned.

The fuel weight W_f required by an aircraft during cruise (assuming quasi-steady flight) may be determined using the Breguet equation shown below [10]. From this equation it becomes apparent that the aircraft designer may influence the required fuel weight through two parameters, i.e. the operative empty weight (W_OE) and the lift-to-drag ratio (L/D). The first of these parameters that is considered is the lift-to-drag ratio.

$$W_f = \int_{t_\mathrm{start}}^{t_\mathrm{end}} cW\frac{D}{L}dt = \int_{t_\mathrm{start}}^{t_\mathrm{end}} c\left(W_\mathrm{OE} + W_\mathrm{payload} + W_\mathrm{f}\right)\frac{D}{L}dt \tag{2-1}$$

The lift-to-drag ratio of aircraft may be investigated by using the equation that follows below [11]. This shows that by having a higher aspect ratio (A) (i.e. a larger span and smaller wetted area) and a lower induced drag (i.e. a high Oswald efficiency factor (e) and a reduced zero-lift drag coefficient C_D0) improvements in the lift-to-drag ratio may be achieved. With respect to the wetted area some studies [4, 12] have shown that BWB aircraft could provide a lower value due to the lifting capabilities of the center body and the resulting smaller outer wing. The increased span has been investigated as well in Ref [4] in which BWB aircraft with spans of up to 100m were considered which also showed improvements in the lift-to-drag ratio.

$$\left(\frac{L}{D}\right)_\mathrm{max} = \sqrt{\frac{\pi Ae}{4C_\mathrm{D0}}} \tag{2-2}$$

When it comes to the drag of the BWB aircraft a number of improvements were discovered. One of the more obvious advantages is the absence of the drag caused by empennage structures. Furthermore, in Ref [12] it was found that the interference drag was reduced for BWB

aircraft due to the blending of the wing and centerbody and the absence of sharp corners, attached engines and/or pylons. Also, the wave drag was shown to be reduced due to the natural area ruling of the aircraft which in turn allows for a higher cruise Mach number[4]. Additionally, extensive research [13] into the spanwise lift distribution across BWB aircraft has shown that both the wave drag due to shock waves during its transonic cruise and the induced drag may be reduced through the application of a combination of an elliptical and triangular lift distribution. An improvement of 16% was achieved through this optimization when compared to the unoptimized baseline geometry of the BWB aircraft.

When going back to equation 2-1, the second parameter that the designer has influence on to lower the fuel consumption is the operative empty weight of the BWB aircraft. The early studies [4] showed that a reduction of 15% in take-off weight would be possible for the BWB aircraft when compared to conventional aircraft. However, one of the biggest problems with the BWB is the design aircraft of the fuselage. As mentioned in the introduction, this is related to the lack of rotational symmetry across the cross-section of the fuselage, which results in the creation of bending loads in addition to the tensile loads that are already present in circular fuselages.

Therefore, research has been going into the development of non-circular fuselages and their pressurization. From this research a number of advantages were identified with respect to non-circular fuselages. Firstly, as mentioned earlier, the centerbody produces lift forces, which decreases the aerodynamic loading on the wings and therefore reduces the resulting lateral bending loads on the wingbox/fuselage structure. This could lead to lighter wing structures. Secondly, due to the wider, flatter and shorter nature of the fuselages, the longitudinal bending loads are also decreased, which again may lead to a reduction in structural weight. In addition, these geometric aspects of the fuselage provide easier structural integration of other aircraft components such as the wings and engines with the fuselage. These simplified connections may also add to a decrease in operation empty weight. However, as already mentioned, the pressurization of these fuselages remains one of the larger difficulties in the design of the BWB aircraft. Some of the solutions that have been proposed up to this point are discussed in the next section.

2-2 Non-circular Fuselages

In order to push the development of the BWB aircraft forward it was quickly realized that a solution needed to be found to one of the largest challenges this concept faced, i.e. the pressurization of its fuselage. To allow for the BWB to maintain its advantages over the current generation of tube-wing configuration aircraft, their fuselage should be structurally efficient while maintaining its aerodynamic qualities. In the subsections below, an explanation of the pressurization problem is provided together with three non-circular fuselage concepts that attempt to fulfill the aforementioned design constraints.

2-2-1 Pressurization

When considering the pressurization of an object that is to function as a fuselage the cylindrical shape is the most favorable one to be used. The pressure inside is carried by pure tensile stresses in the hoop and axial direction with magnitudes as described in equation 2-3 and 2-4 respectively [14]. In these equations, p represents the internal pressure, r is the radius of the cylinder and t its thickness.

$$\sigma_{hoop} = \frac{pr}{t} \tag{2-3}$$

$$\sigma_{long} = \frac{pr}{2t} \tag{2-4}$$

When pressurizing a non-cylindrical object on the other hand a non-uniform tensile stress distribution will be present in the hoop direction due to the lack of rotational symmetry in the cross-section. This unbalance in tensile stresses will result in bending stresses in the walls of the object [15]. Due to this the analysis and design of non-circular fuselages is significantly more complex than its circular counterpart. In the subsections below, three different solutions to this problem are discussed.

2-2-2 Integrated Panel Concept

One of the first solutions proposed was the use of the integrated panel concept [16]. This concept, illustrated in figure 2-2, connects a pressurized vessel in the form of a flat or vaulted shell to the outer skin of the BWB aircraft. The connection between these two components is achieved using a variety of connecting materials/structures which include honeycomb sandwich materials and spanwise and chordwise ribs. The concept of using ribs instead of honeycomb material in combination with the vaulted shell concept provided the best solution out of all the possible combinations with respect to weight reduction. Further research [17] continued the investigation into the use of the vaulted shell concept and lead to the multi-bubble concept.

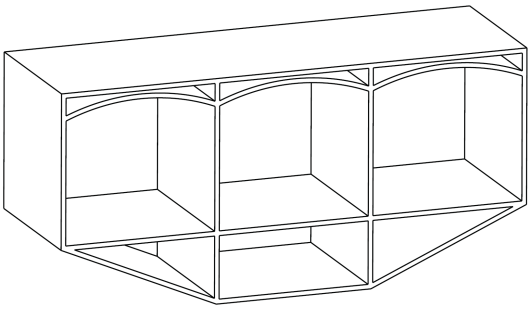


Figure 2-2: Integrated panel fuselage concept [18]

2-2-3 Multi-bubble Concept

One of the major downsides of the integrated panel concept is the fact that the skin of the pressure vessel is almost fully connected to the outer skin of the BWB aircraft. Therefore, the deformation of the pressure vessel has an impact on the aerodynamic characteristics of the aircraft's outer skin. A possible solution to this is the segregated multi-bubble concept as seen in figure 2-3 and described in reference [19]. The multi-bubble concept combines an inner collection of circular shells placed in tandem with an outer shell to provide the aerodynamic shape. Here the circular shells carry the majority of the pressurization loads in tension while the vertical walls carry the reaction loads at the intersections of the separate circular shells. In this way the loads carried by the outer skin and pressure vessel could be almost completely separated.

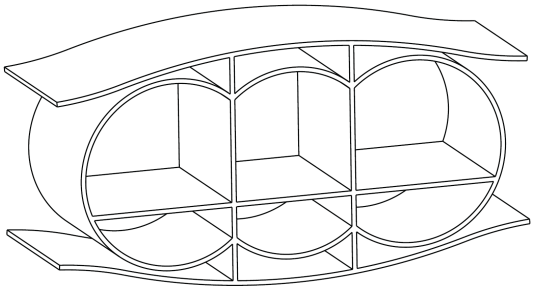


Figure 2-3: Multi-bubble fuselage concept [19]

2-2-4 Oval Fuselage Concept

Although both the integrated panel concept and the multe-bubble successfully resolve the problem of pressurizing a non-circular fuselage one of the main disadvantages of both these concepts is that the cabin space itself is obstructed by vertical poles and/or walls that are required to carry part of the pressurization loading and/or its resulting stresses. This significantly reduces the cabin flexibility, passenger comfort and interior options of both these fuselage concepts.

A relatively new fuselage concept that aims to counter these problems is the oval fuselage concept [20]. As seen from Figure 2-4, the oval fuselage concept consists of a trapezoidal box structure which houses the passengers and which is enclosed by four circular arcs that

tangentially connect at the corners of the trapezoidal box. The main way in which this concept distinguishes itself from the other concepts is through an unobstructed cabin space. Unlike the other two there is no need for the vertical pillars and/or walls within the cabin space to carry part of the pressurization loads. The way in which this is achieved is by pressurizing the aerodynamic surface and having the trapezoidal box carry the reaction loads that are created due to pressurization. In this way, the oval fuselage concept aims to provide a higher flexibility with respect to interior configurations and increased passenger comfort.

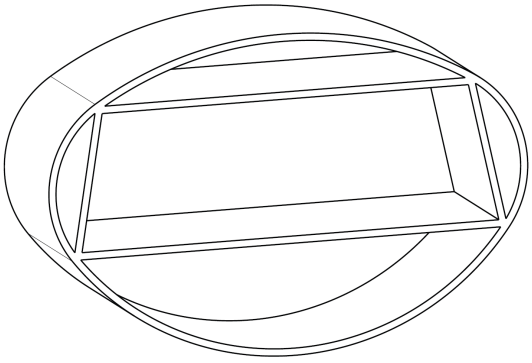


Figure 2-4: Oval fuselage concept [8]

For further details about the oval fuselage's conceptual design (i.e. the workings of the oval fuselage, the initial sizing, the load cases investigated, etc.) the reader is referred to the thesis work of K. Schmidt [21].

2-3 Semi-analytical Weight Estimation Method

When trying to determine the efficiency of aircraft types that steer away from the conventional design, one of the more difficult challenges to overcome at the early stage is the determination of the weight. Due to the lack of data on both BWBs and aircraft with non-circular fuselages, the traditional Class 2 methods, which rely heavily on empirical data, do not provide accurate initial weight estimations. Included in this is the analysis and initial weight estimation of the aforementioned fuselage concepts.

As an alternative to these methods, a SAWE method has been developed for the determination of non-circular fuselages using the oval fuselage concept[8]. This method determines the weight of the complete fuselage by combining analytical methods to determine the weight of all structural members and empirical methods for all non-structural components. The analytical methods determine the size of the structural members by carrying out a two-dimensional structural analysis on a cross-section subjected to pressurization loads in combination with the loads present during either steady-state maneuvering or landing.

For the trapezoidal inner structure, the members are sized as to satisfy the requirements on yield strength, global buckling, crippling, dimpling and wrinkling. The outer structure, i.e. the aerodynamic surface, is sized on yield strength and panel buckling using a simplified buckling analysis in which the structure is represented as a curved stiffened panel.

The SAWE method itself was created as part of a larger project called the Initiator [22]. This design tool allows for the translation of a set of top-level requirements into a conceptual aircraft design. This is done by carrying out the initial estimation of some of the aircraft's characteristics, defining the geometry based on these results and eventually running a number of analysis modules inside optimization loops which continue to run until a number of requirements are met and the aircraft properties are converged. One of these analysis modules used is the SAWE method.

The main output of the Initiator that can be used to transfer its results to another tool comes in the form of a document written in the Extensible Markup Language (XML). Within this outputted document, almost all of the results are collected as calculated by the modules contained within the Initiator. Amongst others, these include the general geometric and structural description of the aircraft, the flight conditions and accompanying aerodynamic loads, weight results, etc.

It was shown by Schmidt and Vos that this weight estimation method could successfully be applied to conventional circular fuselages with a root-mean-square of the prediction error of 13%[8]. In addition, a FEA of a small three-dimensional fuselage section was carried out in which the loads were compared to the analytical predictions from the SAWE method, which showed a good correspondence of the results. However, when analyzing a section of the fuselage representing a quasi-two-dimensional state, the interactions/influences of the separate components on a three-dimensional level are not fully represented. Furthermore, a number of assumptions were applied in order to facilitate the analysis of the oval fuselage geometries. Therefore, the need for a higher fidelity validation of the SAWE method is high to ensure the accuracy of the SAWE method and the potential of the oval fuselage concept.

However, before discussing the validation of the SAWE method the parametrization of the oval fuselage is explained to provide the reader with the knowledge required for the chapters that follow.

2-4 Oval Fuselage Parametrization

To provide a comprehensive explanation of the oval fuselage concept and illustrate the differences between the method used in the SAWE and Automated Finite element model Generator (AFG) tool the parametrization of the oval fuselage is discussed in the subsections that follow.

It should be noted that the drawings provided in this section are for illustrative purposes only and should not be used as a reference for the actual calculations needed to obtain them.

2-4-1 Two-dimensional Oval Cross-section

The three-dimensional shape of the oval fuselage concept, as described in subsection 2-2-4, is constructed in two major steps:

1. Determination and orientation of the geometry of the two-dimensional cross-sections based on the definition of the cabin-floor, cabin height and center curve.

2. Creation of the three-dimensional surfaces through the linking of the cross-sectional oval shapes by using flat, cylindrical, toroidal and spherical surfaces in combination with specific surface shapes for the formation of for example the nose, tail and bulkhead surfaces.

An oval fuselage cross-section is built-up using four circular arcs which enclose a trapezoidal box. The parameters required to fully define such a two-dimensional cross-section in the xz-plane are collected in figure 2-5. It should be noted that in this figure the black dashed areas represent sandwhich panels.

As may be seen from this figure, the four circular arcs form the outer shell of the fuselage and the trapezoidal box's members form the borders of the cabin space with the horizontal ones functioning as the floor and ceiling and the "vertical" members forming the walls.

From the set of parameters required to create the complete two-dimensional cross-section, four need to be defined while the remainder are derived from these four values. These independent parameters are the floor width w_f and the top, center and bottom height, i.e. h_1, h_2 and h_3. The determination of the remaining parameters is based on the fact that the four circular arcs need to meet at the corner points of the trapezoidal box in a tangential manner.

The first step in obtaining the values of these parameters is the determination of the width of the ceiling, w_c. This value may be found by solving equation 2-5 for w_c.

$$\tan^{-1}\frac{w_f}{h_3} - \tan^{-1}\frac{w_c}{h_1} - \tan^{-1}\frac{w_c - w_f}{h_2} = 0 \tag{2-5}$$

Despite the presence of the $\tan^{-1}$ operator, solving equation 2-5 for w_c results in an analytical but rather elaborate expression. The use of an analytical solution in this project is different than the method used within the SAWE method seeing as the latter used an iterative loop to solve equation 2-5 which leads to a less accurate value of w_c.

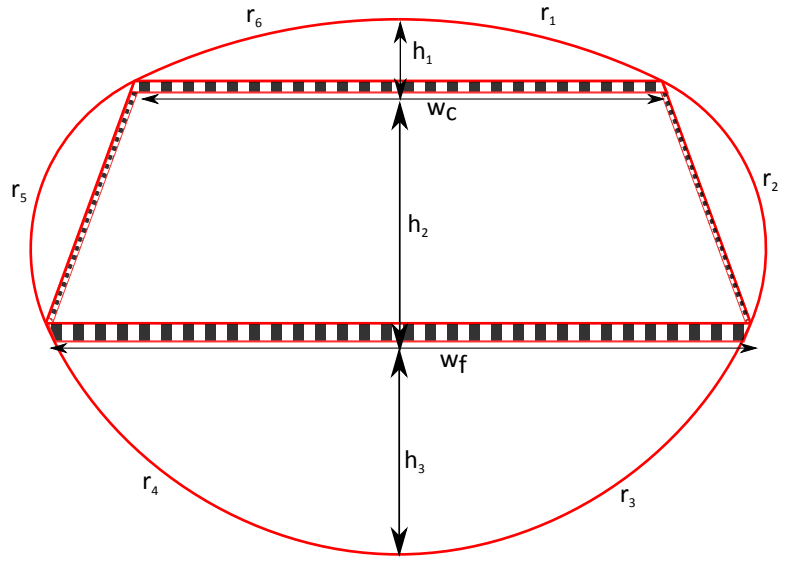
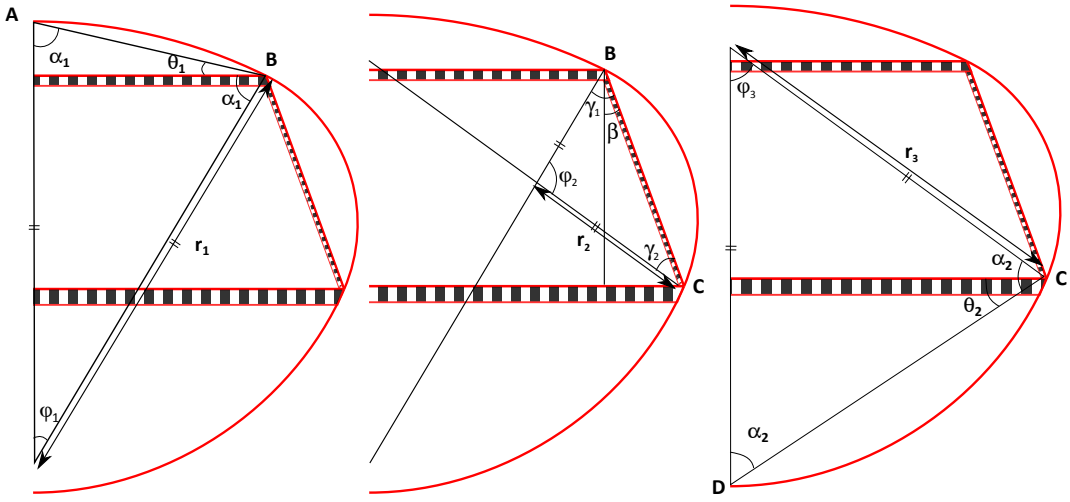


Figure 2-5: Example of a two-dimensional oval fuselage cross-section

$$
\begin{aligned}
w_c &= \frac{-h_1 h_3 - h_2 h_3 + {w_f}^2}{2 w_f} + \frac{\sqrt{\alpha}}{2 w_f} \\
\alpha &= {h_1}^2 {h_3}^2 + 2\, h_1 h_2 {h_3}^2 + 4\, h_1 h_2 {w_f}^2 + 2\, h_1 h_3 {w_f}^2 + {h_2}^2 {h_3}^2 - 2\, h_2 h_3 {w_f}^2 + {w_f}^4
\end{aligned}
\tag{2-6}
$$

The width of the ceiling is now known and the remaining parameters to be determined are the raddii of the circular arcs (r). To determine these values, the three cross-section drawings in figure 2-6 may be used. In these drawings, the upper, middle and lower arc's parameters are displayed from left to right respectively and are found using equations 2-11, 2-16 and 2-21.



(a) Radius arc 1 **(b)** Radius arc 2 **(c)** Radius arc 3

Figure 2-6: Cross-sectional dependent parameters

Upper arc's parameters:

$$\theta_1 = \tan^{-1}\left(\frac{w_c}{h_1}\right) \tag{2-7}$$

$$\alpha_1 = \tan^{-1}\left(\frac{h_1}{w_c}\right) \tag{2-8}$$

$$\varphi_1 = \pi - 2\,\theta_1 \tag{2-9}$$

$$|AB| = \frac{w_c}{\cos\left(\frac{\pi}{2} - \theta_1\right)} \tag{2-10}$$

$$r_1 = \frac{|AB|}{2\,\cos\left(\theta_1\right)} \tag{2-11}$$

Side arc's parameters:

$$\beta = \tan^{-1}\left(\frac{w_f - w_c}{h_2}\right) \tag{2-12}$$

$$|BC| = \frac{w_f - w_c}{\sin\left(\beta\right)} \tag{2-13}$$

$$\gamma_2 = \pi - 2\,\alpha_2 - \beta \tag{2-14}$$

$$\varphi_2 = \pi - 2\,\gamma_2 \tag{2-15}$$

$$r_2 = \frac{|BC|\,\sin\left(\gamma_2\right)}{\sin\left(\varphi_2\right)} \tag{2-16}$$

Lower arc's parameters:

$$\theta_2 = \tan^{-1}\left(\frac{w_f}{h_3}\right) \tag{2-17}$$

$$\alpha_2 = \tan^{-1}\left(\frac{h_3}{w_f}\right) \tag{2-18}$$

$$\varphi_3 = \pi - 2\,\theta_2 \tag{2-19}$$

$$|CD| = \frac{w_f}{\cos\left(\frac{\pi}{2} - \theta_2\right)} \tag{2-20}$$

$$r_3 = \frac{|CD|}{2\,\cos\left(\theta_2\right)} \tag{2-21}$$

With the knowledge of how an oval cross-section is defined a link can now be established between the top-level requirements a designer can set and the four independent parameters required to completely define an oval fuselage cross-section.

2-4-2 Fuselage Floor Planform and Longitudinal Cross-section

In order to define the first independent parameter w_f, the fuselage's floor planform is established by defining the width of the floor at specific positions along the longitudinal axis.

The positions at which the widths are defined are expressed by the designer as a percentage of the fuselage's cabin length. The reason for expressing these locations in terms of percentages is due to the fact that the cabin length is not defined as a settable requirement but instead is made dependent on a number of other properties of the fuselage.

The first of these properties is related to the height variations of the fuselage's longitudinal cross-section that lies within the vertical plane of symmetry. An example of such a cross-section is shown in figure 2-7. As mentioned earlier, the oval fuselage also carries out an aerodynamic function by contributing to the total lift generated by the aircraft. The extent of this contribution is mainly dictated by the shape given to this longitudinal cross-section.

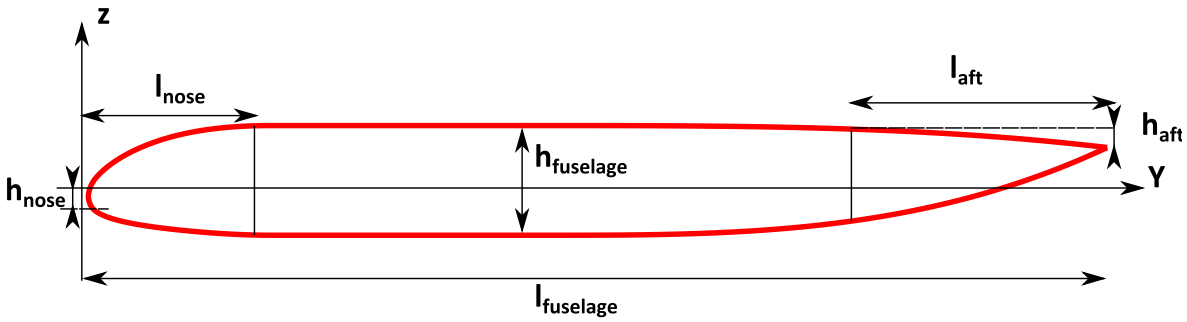


Figure 2-7: Longitudinal cross-section of the fuselage displaying parametrization method

All of the parameters contained within Figure 2-7 need to be defined. These are the total length of the nose, fuselage and aft section (l_{nose}, l_{fuselage} and l_{aft}), the vertical displacement of the front and rear most points of the fuselage (h_{nose} and h_{aft}) and finally the height of the center section of the fuselage h_{fuselage}. These parameters are fully described by the designer using the finesse and droop ratios of both the nose and aft section and the total fuselage length and height. The equations linking the finesse ratio of the nose and tail (ϵ_{nose}, ϵ_{aft}) and their droop ratio (ζ_{nose}, ζ_{aft}) to the physical parameters may be found below.

It should be noted that the equations provided below for the finesse ratios do not correspond to the ones provided in the thesis work of Schmidt [21]. Instead of the fuselage height in the denominator of these fractions (as was described in the equations of Schmidt), the fuselage length should be used to determine the finesse ratios.

$$\epsilon_{\text{nose}} = \frac{l_{\text{nose}}}{l_{\text{fuselage}}} \tag{2-22}$$

$$\epsilon_{aft} = \frac{l_{\text{aft}}}{l_{\text{fuselage}}} \tag{2-23}$$

$$\zeta_{\text{nose}} = \frac{h_{\text{nose}}}{h_{\text{fuselage}}} \tag{2-24}$$

$$\zeta_{aft} = \frac{h_{\text{aft}}}{h_{\text{fuselage}}} \tag{2-25}$$

To fully describe the outline of the fuselage's longitudinal cross-section the crown and belly curve need to be defined, which are the upper and lower curve of the longitudinal cross-section respectively. The equations used for the definition of these curves are discussed separately for the nose, center and aft sections.

Nose section: $y \leq l_{nose}$

The shape of the nose's crown and belly curve is created using an elliptical shape which is drooped down by h_{nose}. If the nose droop ratio is set to zero, then a perfect ellipse will be created which is symmetric about the horizontal axis.

$$z_{\text{crown}} = \left(\frac{h_{\text{fuselage}}}{2}\left(1-\left(\frac{(y-l_{\text{nose}})^2}{{l_{\text{nose}}}^2}\right)\right)^{\frac{1}{2}}\right) - \left(\frac{h_{\text{nose}}}{{l_{\text{nose}}}^3}\left(l_{\text{nose}}-y\right)^3\right) \tag{2-26}$$

$$z_{\text{belly}} = \left(\frac{-h_{\text{fuselage}}}{2}\left(1-\left(\frac{(y-l_{\text{nose}})^2}{{l_{\text{nose}}}^2}\right)\right)^{\frac{1}{2}}\right) - \left(\frac{h_{\text{nose}}}{{l_{\text{nose}}}^3}\left(l_{\text{nose}}-y\right)^3\right) \tag{2-27}$$

Center section: $l_{nose} < y < l_{fuselage}$-$l_{aft}$

The center section is defined by splitting the height of the fuselage (h_{fuselage}) equally below and above the y-axis.

$$z_{\text{crown}} = \frac{h_{\text{fuselage}}}{2} \tag{2-28}$$

$$z_{\text{belly}} = \frac{-h_{\text{fuselage}}}{2} \tag{2-29}$$

Aft section: $l_{fuselage}$-$l_{aft} \leq y$

For the shape of the aft section two different polynomials are used to create the crown and belly curve. When considering Figure 2-7 it may be seen that these curves are displaced vertically depending on the value of h_{aft} relative to a horizontal line at a height of $h_{\text{fuselage}}/2$. Therefore, when ζ_{aft} is set to zero, the crown curve will be a horizontal line continuing along the same height as where the crown curve ended for the center section. While the belly curve will move from the aft end position of the center section's belly curve towards $h_{\text{fuselage}}/2$

according to the equation below.

$$z_{\text{crown}} = \frac{h_{\text{fuselage}}}{2} - \left(\frac{h_{\text{aft}}}{{l_{\text{aft}}}^3} \left(y - (l_{\text{fuselage}} - l_{\text{aft}})\right)^3 \right) \tag{2-30}$$

$$z_{\text{belly}} = \frac{-h_{\text{fuselage}}}{2} + \left(\frac{h_{\text{fuselage}} - h_{\text{aft}}}{{l_{\text{aft}}}^3} \left(y - (l_{\text{fuselage}} - l_{\text{aft}})\right)^3 \right) \tag{2-31}$$

As mentioned before, the length of the cabin space is left dependent on a number of other properties. With the aerodynamic outline of the center section defined the only remaining properties that need to be set by the designer are the cabin floor height z_{floor} and the height of the cabin h_{cabin}. Using figure 2-8, the relation between these properties and the cabin length (l_{cabin}) can be explained.

By defining the vertical position of the cabin floor and the height of the cabin itself, the two dashed horizontal black lines may be drawn. Along these lines the floor and ceiling will be defined. In order to determine the length of the cabin, the intersections between the cross-section's outline and these horizontal black lines need to be found. However, due to the fact that at these intersections the values of both h_1 and h_3 are zero, no oval cross-sections can be created at these locations. Therefore, a new parameter is introduced that sets the minimum value of h_1 and h_3, namely $h_{1,3\text{-min}}$. By adding this distance to the two horizontal black lines the two dashed horizontal red lines are created. Through the intersection of these red lines with the outline of the longitudinal cross-section, the front and aft longitudinal boundaries of both the ceiling and floor can now be found.

For the front boundary of the cabin the most aft intersection is used as seen in Figure 2-8. When considering the aft boundary in Figure 2-8 it can be seen that the boundary used to define the cabin space is located further forward than the most forward intersection with the longitudinal cross-section. The reason for this is that the value set for the minimum length of the aft part of the fuselage ($l_{\text{aft,min}}$) is larger than the length of the aft part defined by the intersection. For the nose section, the length defined by the nose intersection is longer than $l_{\text{nose,min}}$ and therefore the intersection point is used rather than $l_{\text{nose,min}}$. This results in the two-dimensional cross-section of the cabin space as shown by the red rectangle in Figure 2-8.

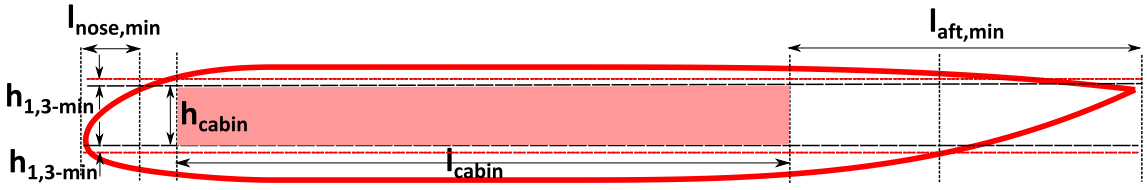


Figure 2-8: Determination of the cabin length (l_{cabin})

2-4-3 Outer-skin Connecting Parts

As currently described, the oval cross-sections are oriented at right angles to the longitudinal axis of the fuselage. However, this will create sharp corners at the edges of the fuselage's floor planform and therefore also on the outer fuselage skin, which will result in areas of stress concentrations. The oval cross-sections should therefore be rotated to ensure a smooth transition from one defined width to the next. This is achieved by making use of three types of

three-dimensional surfaces, i.e. cylindrical, spherical and toroidal surfaces. These surfaces are displayed in figure 2-9 where the cylindrical, spherical and toroidal surfaces are represented in blue, red and green respectively. The use of each type of surface depends on the way the width varies along the longitudinal axis of the cabin. When considering figure 2-10 it may be seen how the designer's width input, which is represented by the black dotted line, is transformed into a continuous outline of the floor planform where no sharp edges are present at the transitions from one width to another. Similar to figure 2-9, blue, red and green once again represent the cylindrical, spherical and toroidal surfaces respectively.

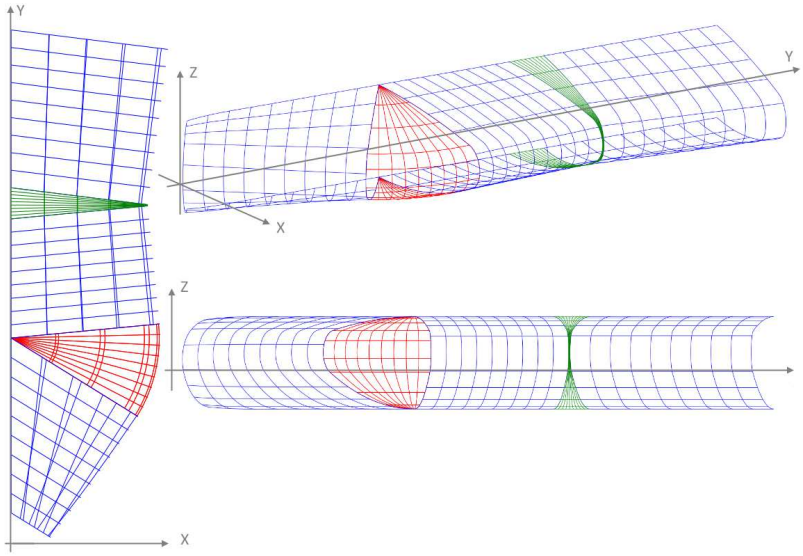


Figure 2-9: Three-dimensional connecting outer surfaces

An addition to the width transitions the SAWE method could handle are the gradual width increase and decrease, which are shown in transition 4-5-6 and 0-1-2 respectively in Figure 2-10. The reason for incorporating this into the higher fidelity analysis is to allow for small adjustments to be made to the floor planform without having to adjust the general layout as calculated by the SAWE method. It should be noted that at the writing of this thesis report this is only incorporated into the Geometry module of the AFG tool and therefore only functions as a geometric adjustment tool to for example investigate different cabin layout possibilities or improve the wing fuselage connection.

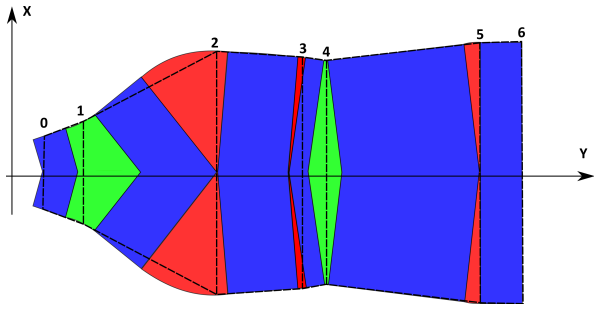


Figure 2-10: Smooth transitions along the cabin floor widths

When the center locations and angles of the oval fuselage cross-sections have been determined, the final step is to reevaluate the values required to construct these cross-sections, i.e. w_f, h_1, h_2 and h_3. The recalculation of these values is needed due to the fact that the cross-sections are not necessarily created at the locations where the widths have been defined but rather at the center locations determined for each cross-section. Examples of this are width sections 1 and 3 in figure 2-10.

2-4-4 Three-dimensional Fuselage

With the location and orientation of the oval cross-sections determined, the final step is to create the outer shell and cabin space. Three of the surfaces used to create the outer shell are already known, namely the cylindrical, spherical and toroidal surfaces. However, for the nose and aft section different surfaces will be used. In both cases, the outline of the longitudinal cross-section and the geometry of the oval cross-section closest to these parts is used as a starting point for the determination of the section's shape. This ensures a smooth transition from the cabin area's outer skin to the nose and aft section's skin.

For the nose section, the geometry may be determined using equations 2-32 to 2-37. In these equations, the *oval* subscript refers to the points located on the most forward oval cross-section, i.e. closest to the Leading Edge (LE) of the fuselage's longitudinal cross-section. The $y_{\text{oval,min}}$ and $y_{\text{oval,max}}$ parameters represent the minimum and maximum y-coordinate value found along the outline of the oval cross-section. Another minimum parameter is z_{min}, which refers to the lowest point on the longitudinal cross-section at the same y-coordinate as the y_{nose} position being considered. In relation to this, $z_{\text{min,overall}}$ is the overall lowest point on the longitudinal cross-section and is a constant that does not change in these equations when moving from one point to the next. Finally, α represents an interval of angles and is used to define n values of c_0 and c_1 along the length of the nose section.

$$x_{\text{nose}} = x_{\text{oval}}\, t_{\text{norm}} \tag{2-32}$$

$$y_{\text{nose}} = \left(\left(y_{\text{oval}} - \frac{y_{\text{oval,max}} + y_{\text{oval,min}}}{2}\right) c_0\right) + c_1 \tag{2-33}$$

$$z_{\text{nose}} = (z_{\text{oval}}\, t_{\text{norm}}) + z_{min} - (z_{\text{min,overall}}\, t_{\text{norm}}) \tag{2-34}$$

$$\alpha = \left[0..\frac{\pi}{2}\right] \tag{2-35}$$

$$c_0 = 1 - \sin(\alpha)^{1.3} \tag{2-36}$$

$$c_1 = \frac{y_{\text{min,overall}} + y_{\text{max,overall}}}{2}\, c_0 \tag{2-37}$$

For the tail section a similar set of equations can be applied as shown below. The oval cross-section used in this case is the one located most aft of the LE of the fuselage, i.e. closest to

the Trailing Edge (TE).

$$x_{tail} = x_{\text{oval}}\, c_3 \tag{2-38}$$
$$y_{tail} = (y_{\text{oval}} - y_{\text{min,overall}})\, c_2 + y \tag{2-39}$$
$$z_{tail} = (z_{\text{oval}}\, t_{\text{norm}}) + z_{min} - (z_{\text{min,overall}}\, t_{\text{norm}}) \tag{2-40}$$

$$i = = [0..n] \tag{2-41}$$
$$c_2 = [1..0] \tag{2-42}$$
$$c_3 = 1 - \left(\frac{1 - \text{AftRatio}}{(n-1)^2}\right) i^2 \tag{2-43}$$

Although the above equations are identical to the ones used in the SAWE method, the difference with the SAWE method is that it does not take the influence of the orientation of the oval cross-sections into account. This may be explained using figure 2-11. In this figure a top view of the nose section is displayed with the same oval cross-section at two different orientations, i.e. $oval_{\text{straight}}$ and $oval_{\text{rotated}}$. When considering point p_1 for both cross-sections it may be seen that their y-coordinates are different. Due to this, the values required for certain parameters will be different, i.e. t_{norm} and z_{min} and consequently also the results of the parametrization. By altering this in the SAWE method's parametrization the kink along the center-line in the nose and tail surface is changed into a smooth surface definition. With the removal of this geometrical error, also the possibility of creating areas of high stress concentrations due to geometrical errors in the nose and tail cone is removed.

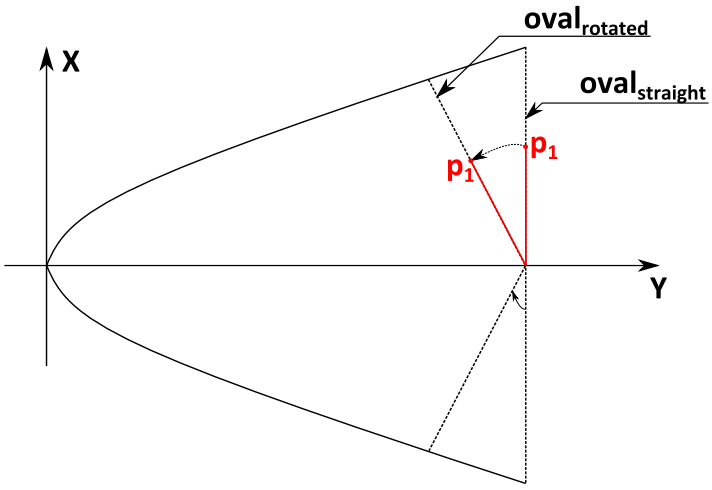


Figure 2-11: Rotated oval cross-section error explained

The last surfaces that need to be recreated are those required for the trapezoidal box structure. However, no specific shapes are required for the construction of this part of the fuselage. Instead, the smooth outlines for both the floor and ceiling surface may be used as the four longitudinal edges between which lofted surfaces can be created. These then form the trapezoidal box structure.

By using all of the aforementioned surfaces, the oval fuselage structure as defined by the SAWE method can now be recreated. An example of a conventional and BWB aircraft are shown in figure 2-12 and 2-13 respectively. One final note that should be made with respect to the parametrization of the oval fuselage is that not all geometric parts have been covered that are required to create the final FE model. The reason for this is that the creation of the remaining geometric parts, i.e. the pressure bulkheads, cargo-floor, etc., depend partly on the redefinition of the geometry as addressed in section 3-4 and will therefore be described there.

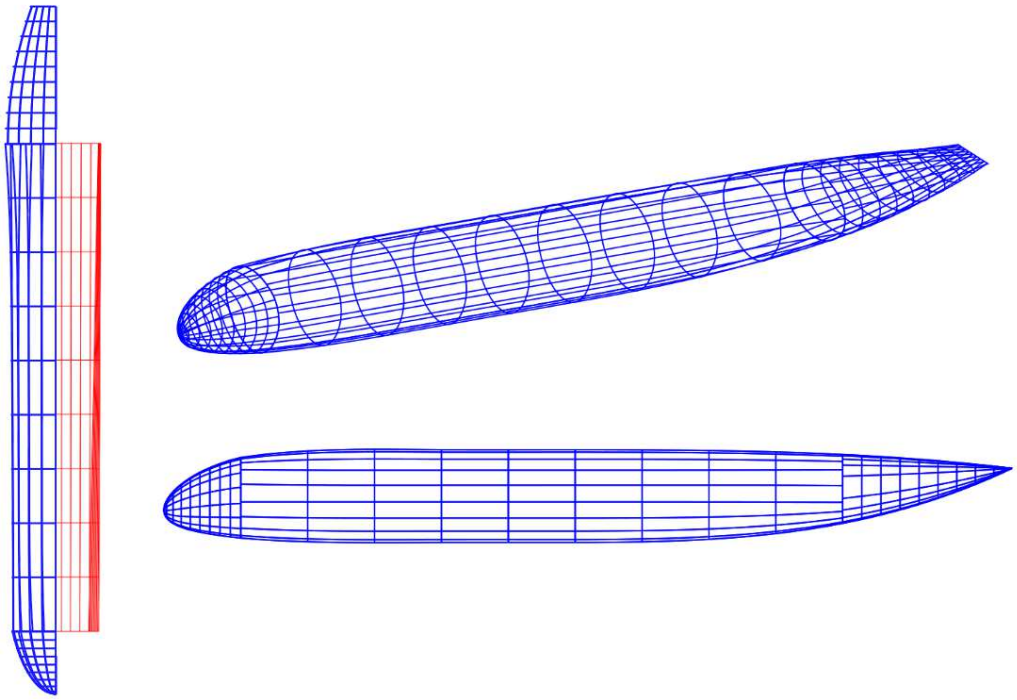


Figure 2-12: Example of a conventional oval fuselage

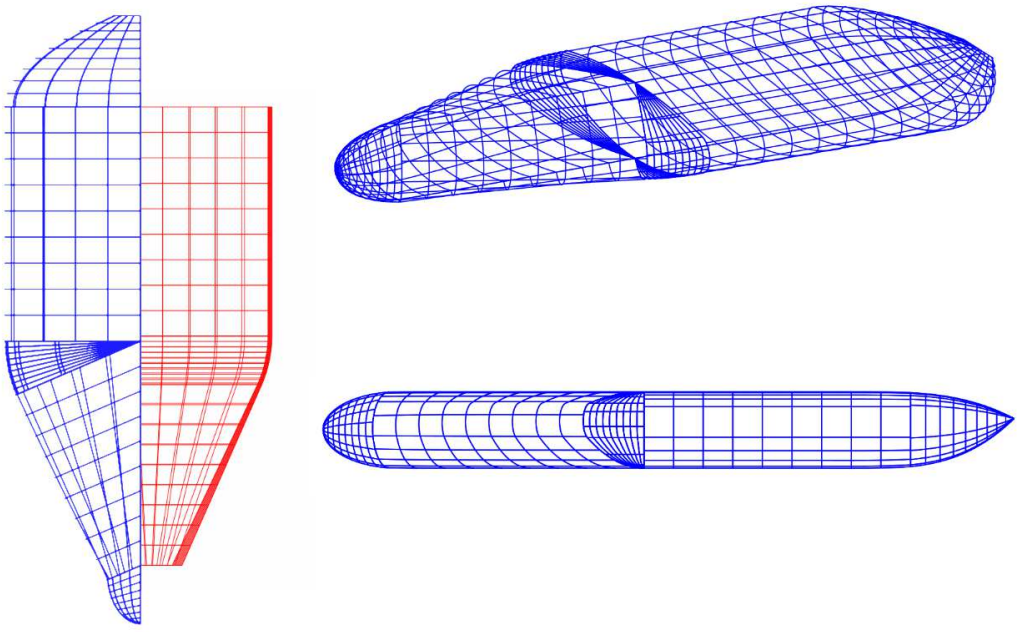


Figure 2-13: Example of a BWB oval fuselage

Chapter 3

Automated Finite element model Generator

In the sections below, the AFG tool is explained by firstly considering an overview of the complete tool and hereafter discussing the functions and capabilities of each of the modules separately.

3-1 High-level Overview

Figure 3-1 provides a high-level overview of the process required to translate the preliminary design data as obtained by the SAWE method into input suitable for the FEA software. This is accomplished through the use of five modules, i.e. the Controller module, the Initiator Controller module, the Geometry module, the FEA Code module and the FEA module, which are all collected in the AFG tool. It should be noted that Figure 3-1 only provides an illustration of the order in which the different modules should be activated rather than the actual functioning of the AFG tool. The latter is discussed in greater detail in Section 3-2.

As mentioned earlier, the SAWE method is part of a larger conceptual design tool called the Initiator. Seeing as the data required by the AFG tool is calculated by this conceptual design tool the first step in the automation process is to provide the Initiator Controller module with the required input data. This input comes in the form of the top-level requirements and properties of the aircraft that is to be designed. Using this data, the Initiator Controller creates a script that is run by the Initiator. Contained within this script are the commands that are to be executed by the Initiator, which will cause it to calculate and store the data required by the subsequent modules of the AFG tool. This data contains the:

- Geometrical and structural properties of the structural components making up the aircraft, i.e. mass, thickness, dimensions, etc.
- Mass of the non-structural components and passengers, luggage, cabin utilities, cargo etc.

- Loads acting on and generated by the different components of the aircraft and information about the different load cases investigated.
- Longitudinal, lateral and vertical positioning of the center of gravity of the different components making up the aircraft.

Within the Geometry module the geometric data of the designed fuselage is used to recreate and redefine the fuselage geometry using a KBE environment. This recreation is based on the parametrization described in subsection 2-4. The redefined geometry is then exported into a number of data files each representing a part of the fuselage structure, e.g. outer skin, frames, floors, bulkheads, etc.

After the geometry redefinition and exporting of the data, the next module to be activated is the FEA Code module. Based on the data provided by preceding modules, this module is capable of creating the code required for the automation of the FEA setup. Here the word code refers to the collection of commands written in the programming language of the FEA pre-processor, i.e. Patran Command Language (PCL), which tell the FEA pre-processor what tasks to carry out in order to create the complete FE model.

With both the data and code files for the FEA software generated, the FEA module is started. This will boot the FEA software and allow for the input of a single command which reads in the input file containing the FEA code. This will start a sequence of actions which create the complete FE model automatically. Hereafter, the model can be analyzed and the results extracted to allow for the SAWE method to be validated.

Throughout the following sections, a more detailed description of the different modules is given to provide better insight into both the workings and relations of these modules.

For more information about the actual use of the AFG tool, the user is referred to its Operational Manual which explains in great detail the input/output of each module, the commands required to run them and the calculations/processes it carries out during operation. The operational manual may be found in Part II

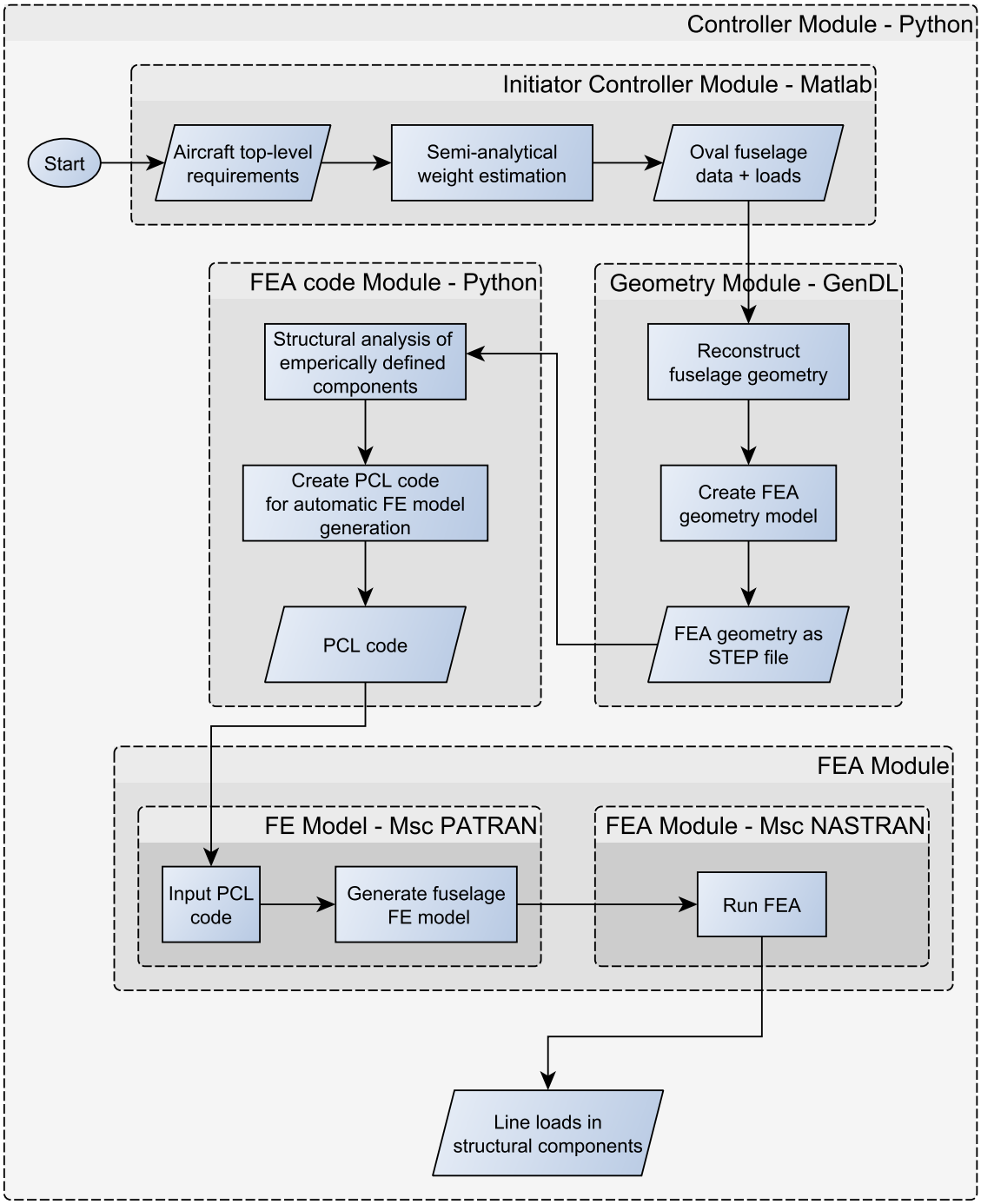


Figure 3-1: Higher fidelity analysis and optimization process

3-2 Controller Module

The complete process described in Section 3-1 is managed using the Controller module. Through this module the user is able to run specific processing modules (e.g. Geometry module, FEA Code module, etc.) and edit the settings used within the AFG tool.

As seen from Figure 3-3, the Controller module forms the center of all the procedures and is responsible for telling the different modules when they should be activated. Once the modules have been run they will store the calculated results and/or data into output files to allow for subsequent modules to access and use them.

The Controller module is housed in a command shell that accepts text-based input allowing the user to instruct the program to carry out specific tasks. The command-shell at initialization of the AFG tool is displayed in Figure 3-2.

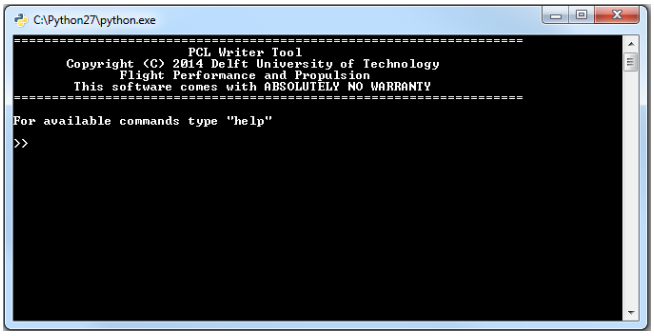


Figure 3-2: Command shell of the AFG Tool at start-up

Besides ensuring the correct program and data flow, the Controller module was also put in place to avoid the need for the user to understand all of the different programming languages and conventions used within each module. In theory, the user should only need the Controller module up until the point where the actual FEA needs to be carried out and the results need to be interpreted.

3-3 Initiator Controller Module

As mentioned earlier, the Initiator is capable of translating a number of top-level requirements into a conceptual design of an aircraft of which the results are stored in an XML file. Based on these results, the AFG tool can then automatically create the FE model of the fuselage linked to this design. Therefore, the first processing module to be run in the AFG tool is the Initiator Controller module.

This module is responsible for the following tasks:

1. Write a script in the programming language of the Initiator, i.e. Matlab programming language, containing the commands that instruct the Initiator with what tasks to carry out
2. Boot the Initiator conceptual design tool

3. Instruct the Initiator which files to load and which script to read which ensures the correct analysis module is run

4. Store the data not contained within the output XML file into a number of separate files

With relation to the last task in the list, it should be mentioned that part of the data required by the subsequent modules is not contained within the Initiator's XML file. Therefore, the script written by the Initiator Controller module contains commands that instructs the Initiator to retrieve the required data that is only available temporarily while the Initiator runs and store it in a number of files to allow for future access by the modules that follow in the chain of operation of the AFG.

As a final note, it should be mentioned that the advised analysis module to run in the Initiator is the "Class25weightEstimation" module. The reason for this is to ensure that all of the required results are calculated by the Initiator which ensures the correct functioning of the subsequent modules.

3-4 Geometry Module

Within the Initiator's output file only the basic inputs required by the parametrization of the oval fuselage are stored. Therefore, the geometry of the oval fuselage needs to be reconstructed using these parameters before it can be imported into a FE model. The main issue however with importing the geometry as it was defined in Section 2-4 directly into a FE model is that a lack of congruence between the nodes on the different component's meshes would be present. For example, when meshing the outer skin and the floor separately, the nodes on these two components may not align due to the differences in these components' geometry and dimensions. As a result of this, the FEA will not understand that a (rigid) connection exists between the floor surface and the fuselage skin. Therefore, a geometrical redefinition is required to ensure congruence between the nodes present on the different components making up the fuselage's model.

This redefinition is carried out through the use of a Knowledge-Based Engineering (KBE) platform. The choice for this project went to the General-purpose Declarative Language (GenDL) from Genworks International. GenDL is an object oriented KBE platform which uses the lisp programming language. The main reason for choosing this platform is related to the strong object-oriented Computer Aided Design (CAD) capabilities it offers. This allows for the geometrical reconstruction and redefinition of the oval fuselage concept in a way dependent solely on the parameters already defined in the SAWE method.

Although the reconstruction of the oval fuselage geometry might seem unnecessary at first, there are actually two reason for it. The first is related to the fact that the redefinition of the oval fuselage makes use of the surfaces representing the original geometry. The second reason is that by reconstructing the geometry the re-useability of the Geometry module increases. This module could for example be incorporated into a higher fidelity design/optimization routine allowing the user to create, alter and inspect the geometries of both conventional and unconventional fuselage structures. This is even further enhanced due to the object-oriented nature of GenDL KBE platform which means that the reconstructed fuselage may be seen as a building block that can be combined with other components such as the wings, empennage, etc. to form a complete aircraft model.

As a final note it should be mentioned that for a number of structural components present within the oval fuselage only a mass property is determined by the SAWE method. This is related to the fact that for these components only empirical relations were used rather than analytical methods. Therefore a number of surfaces still need to be geometrically defined. These are the cargo floor and the pressure bulkheads at the nose and at the end of the cabin. With the parametric description of these surfaces being dependent on the geometrical redefinition their parametrization is discussed after the redefinition has been explained. Additionally, due to the use of empirical relations in the SAWE method not every surface has a thickness related to it. Therefore a number of structural calculations are carried out to obtain all of the required structural data, which is described in Section 3-5.

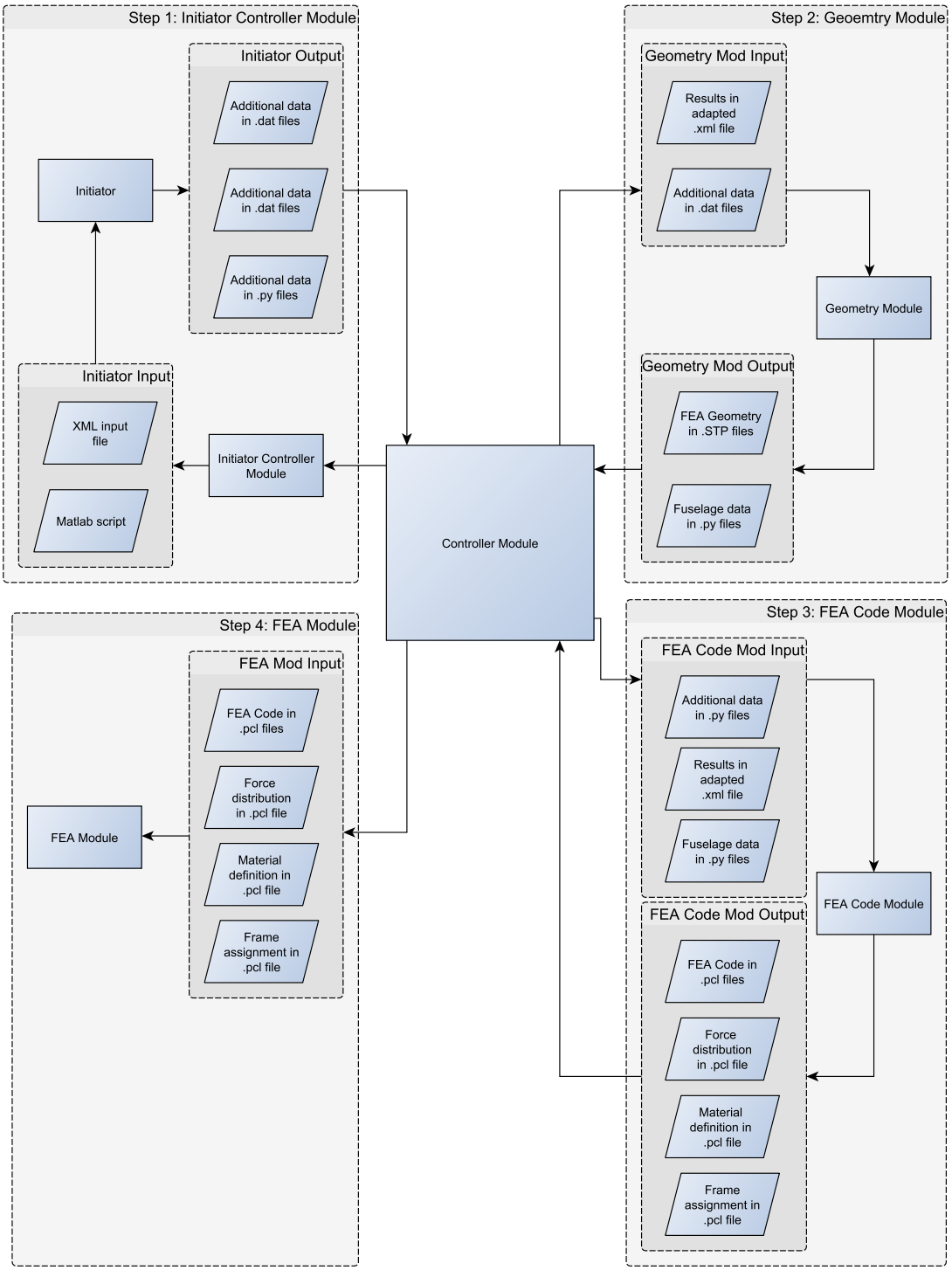


Figure 3-3: Flow of the AFG tool

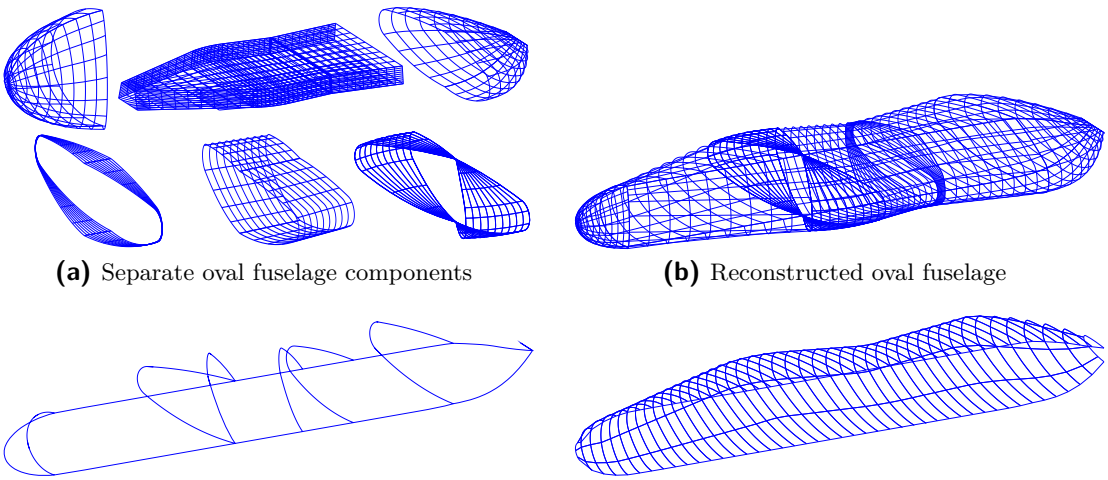
3-4-1 Geometrical Redefinition of the Oval Fuselage

As mentioned, the geometrical redefinition of the oval fuselage uses the original surfaces as defined in the SAWE method. Therefore, the first step in the redefinition process, which is illustrated in Figure 3-4, is to reconstruct the oval fuselage components. These are the nose and tail cone, the floor, ceiling and wall surfaces forming the trapezoidal box and the cylindrical, spherical and toroidal surfaces forming the outer skin around the cabin area. Examples of these are displayed in Figure 3-4a.

Once all of the separate surfaces have been redefined these components may be assembled into a complete fuselage structure. For example, Figure 3-4b shows the assembly of the different building blocks making up the outer skin of an oval fuselage. It should be noted however that in assembly form, the separate components do not yet form an integral surface. This is done for the outer skin in Figure 3-4c.

As may be seen in this figure, only half of the fuselage skin geometry is joined. The reason for this is that due to the symmetrical nature of the oval fuselage concept about the vertical plane of symmetry in the longitudinal direction, only half of the geometry needs to be analyzed in the FEA. However, this is only the case provided that the correct boundary conditions and loads are applied. The latter is discussed in greater detail in Section 3-6.

For the redefinition of the geometry model, the main idea is to use the frame locations and the floors (i.e. cargo floor, cabin floor and cabin ceiling) as the indicators of where the vertical and horizontal division planes should be positioned respectively. These planes are used to horizontally and/or vertically divide the separate structural components of the oval fuselage into smaller surfaces as shown in Figure 3-4d. Through this redefinition, the properties, loads and boundary conditions can be applied with higher accuracy and ease due to the increased traceability of the surfaces, nodes and elements. The reason for this increased traceability will be explained in subsection 3-6-1. Furthermore, due to this redefinition, the complete oval fuselage structure is made up of only rectangular surfaces which ensures the use of quadrilateral elements in the FEA.



(a) Separate oval fuselage components

(b) Reconstructed oval fuselage

(c) Half fuselage skin joined

(d) Redefined half fuselage geometry

Figure 3-4: Illustration of the steps making up the geometry redefinition process

With the redefinition being based on the original surface definitions the Outer Mold Line (OML) of the fuselage structure does not change significantly from the original. The small difference that is present with respect to the original OML is related to the representation of the curvature in longitudinal direction as is explained in the next paragraph. In the recommendations for future developments in Chapter 5 a possible solution is provided to help circumvent this limitation.

For this redefinition, the joined skin surface is divided into a number of frames (as may be seen in Figure 3-5). These frames are created by intersecting the joined fuselage skin as shown in Figure 3-4c with planes parallel to the xz-plane at the same longitudinal positions as the fuselage frames.

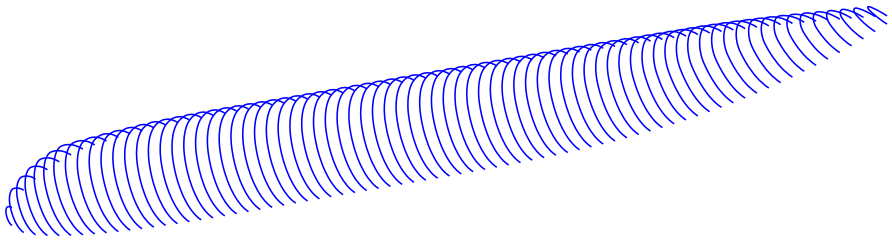


Figure 3-5: Curves used for the redefinition of the oval fuselage skin

Each of these frame curves is then divided into four separate curves by dividing them vertically by the cargo, floor and ceiling planes. By creating a lofted surface in between two adjacent curves the complete outer skin is formed. This process has been illustrated in Figure 3-6.

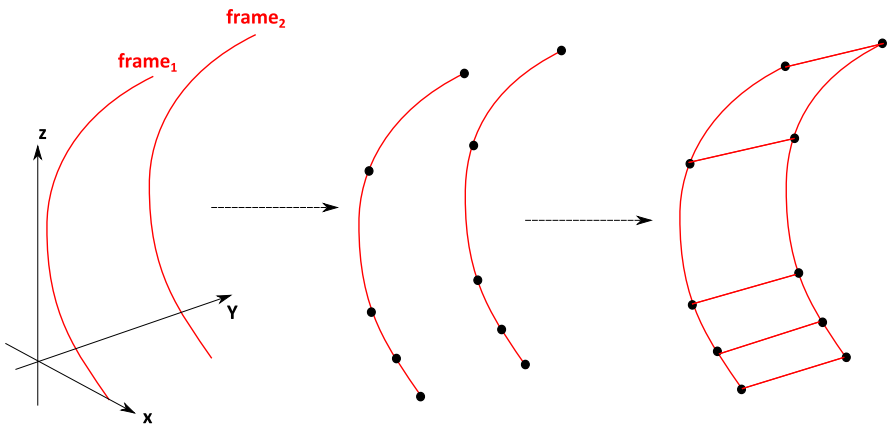


Figure 3-6: Example for the loft of the outer skin surface between two frame curves

By using lofted surfaces, straight lines are created between the two curves in order to form a surface. This means that the curvature of the skin in the longitudinal direction will be approximated by a collection of straight lines. This may be seen as a limitation of the Geometry module. However, when a small frame-spacing is used relative to the total length of the fuselage, the curvature is approximated with high accuracy and the difference becomes insignificant for the FEA.

For the redefinition of the floor, wall and ceiling surface a similar lofting technique is applied as described for the skin surface. For these surfaces, the curves used in the lofting process

need to be defined however before this technique can be applied. For the floor and ceiling, these curves are created by connecting two points by means of a straight line. The start point of the curve is formed by using the points of the vertically divided frame curve (i.e. the middle illustration in Figure 3-6) at the location of the floor or ceiling. This has been illustrated for the floor in Figure 3-7 by the "Start point skin curve 1" and "Start point skin curve 2". For the end of the curve the points with the same longitudinal position as these start points and which lie within the vertical plane of symmetry are used. These are represented by the "Point in symmetry plane 1" and "Point in symmetry plane 2" points in Figure 3-7. When these curves have been defined, a lofted surface can be created in between two longitudinally adjacent curves as represented by the grey surface. It should be noted that for the wall the start and end point of the middle arc, i.e. Arc 2, of the oval skin are used to create the curves used to create the lofts.

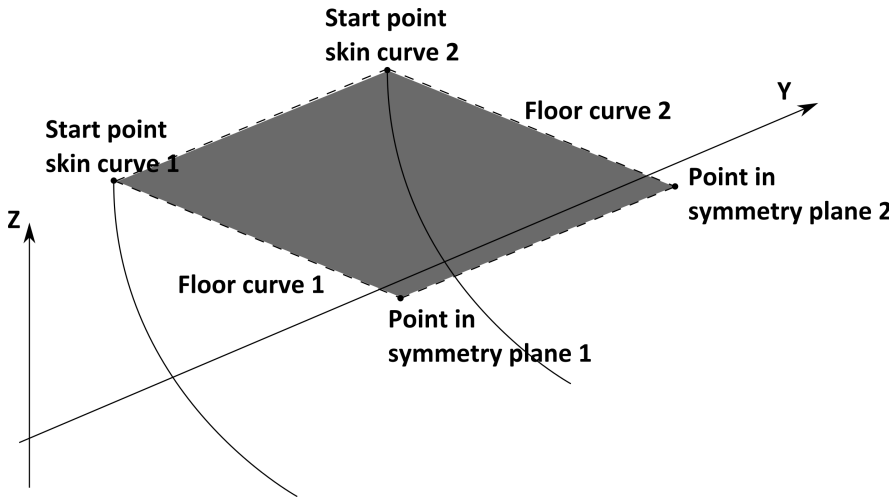


Figure 3-7: Example of floor redefinition

Seeing as the pressure bulkheads and cargo floor are not defined geometrically within the SAWE method, the Geometry module defines and calculates the dimensions of these surfaces. These geometrical definitions are discussed in the next subsection.

3-4-2 Geometrical Definition of Remaining Surfaces

Three surfaces have not been geometrically defined within the SAWE method, i.e. the nose and aft pressure bulkheads and the cargo floor. The front and aft pressure bulkheads are located directly aft of the radar dome and at the aft end of the cabin space respectively. The cargo floor is located in between the bottom of the fuselage outer skin and the cabin floor.

Front Bulkhead

The front bulkhead is modeled as a flat plate. This plate is created by lofting the frame curves at the most forward longitudinal position, i.e. closest to the nose tip, to vertical lines in the longitudinal symmetry plane having the same longitudinal position as the most forward frames and the same height as this frame it is to be lofted with.

Aft Bulkhead

For the aft bulkhead, a concave dome shape is used with respect to the cabin rather than a flat plate. The shape of this dome has been parametrized to be dependent on a single user input parameter, which is the angle θ_{pb} shown in Figure 3-8.

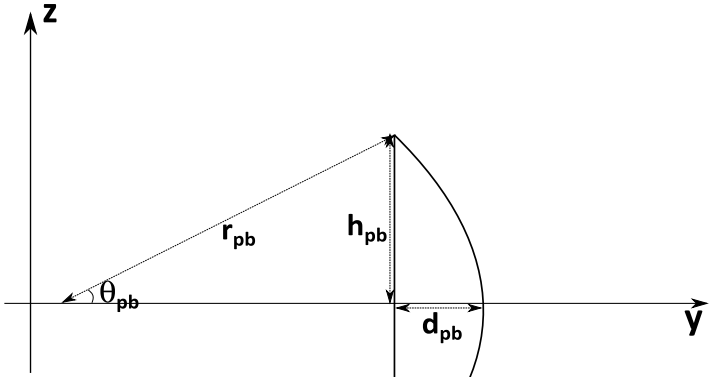


Figure 3-8: Parameterization of the aft bulkhead

Using the equations below in combination with Figure 3-8, the geometrical parameters are determined to construct the aft pressure bulkhead in the Geometry module.

$$h_{pb} = \sqrt{\frac{S_{oval}}{\pi}} \tag{3-1}$$

$$r_{pb} = \frac{h_{pb}}{\sin(\theta_{pb})} \tag{3-2}$$

$$d_{pb} = \frac{R\,(1-\cos(\theta_{pb}))}{\sin(\theta_{pb})} \tag{3-3}$$

Cargo Floor

The last surface that is not geometrically described within the Initiator is the cargo floor. In order to construct this surface, two additional parameters are extracted by the Initiator Controller module that are not stored within the XML file. These are the front and aft position of the cargo floor. Together with the vertical positioning of the cargo floor, these parameters are used to create the cargo floor in a manner similar to that used for the floor, wall and ceiling surfaces as was illustrated in Figure 3-6.

With these last surfaces geometrically described, the complete oval fuselage geometry as used in the FEA is defined. An example of a complete oval fuselage structure is provided in Figure 3-9. As may be seen from this figure, the structural components that are included into the FE model are the following: the complete outer skin (blue), the trapezoidal shaped box (red), the pressure bulkheads (green) and the cargo floor (red). The frames and floor struts are not illustrated in Figure 3-9 due to the fact that these are modeled within the FE model based on the surface edges of the previously mentioned surfaces.

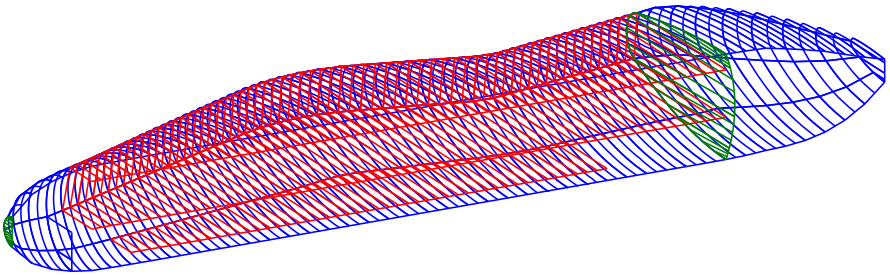


Figure 3-9: Complete oval fuselage FE model geometry

Once all the surfaces have been geometrically redefined, the Geometry module runs a script that translates the geometry created within the KBE environment into a number of STEP-files. These files allow for three-dimensional geometries to be easily transferred between different applications by representing the geometry in a standard data exchange format.

3-4-3 Geometry Dependent FEA Considerations

For the FE model, the stringers that are present on the fuselage skin are not geometrically defined despite being structural components. Instead, their load bearing contribution is included by adding their smeared thickness to the outer skin thickness. The reason being that due to the size of the geometric model, the modeling of the stringers would become computationally too expensive. Especially when taking into account the future possibilities of using this tool within an optimization routine. Furthermore, seeing as the FE model is meant to represent the global behaviour of the fuselage, the analysis of for example panel buckling should be considered separately.

Furthermore, as mentioned earlier, the symmetric nature of the oval fuselage allows for only half of the fuselage's geometry to be modeled, which results in approximately half the number of elements in the FE model. This implies that the user has the possibility to carry out an analysis using a higher mesh density on the surfaces within an approximately equal time frame as when the complete fuselage would be investigated. However, by modeling only half the geometry symmetric BCs and loads should be applied to the model at the correct locations. This is discussed in greater detail in section 3-6.

3-5 FEA Code Module

Based on the data provided by both the Initiator Controller module and the Geometry module, the FEA Code module is capable of creating the code required for the automation of the FEA setup. However, before the code can be created some structural data still needs to be determined. This is related to the lack of structural data (i.e. thickness data) of the components of the oval fuselage that were determined using empirical analysis in the SAWE. In the paragraphs that follow the structural calculations and the generation of the FEA code are discussed.

3-5-1 Structural Analysis of Empirically Defined Components

Seeing as the SAWE method is based on a semi-analytical nature, a number of structural components have either only a mass or a mass and geometric shape related to them. This means that for these surfaces no thickness has been defined yet. These surfaces are the pressure bulkheads, the cargo floor, the nose surface and the tail surface. The main idea behind these structural-calculations is to ensure that these components do not undergo a deflection larger than allowed. This allowed deflection is determined as a percentage of a geometrical dimension of the respective component.

Pressure Bulkheads

Due to the difference in geometric shape for the nose and aft pressure bulkhead two different calculation procedures need to be employed. For the aft pressure bulkhead, the simplification is made that the surface being analyzed is part of a perfect spherical shape. This idealized shape is clamped around the edge to simulate the connection between the skin and the bulkhead. Using the equation [23] below the skin thickness of the spherical dome may be calculated.

$$t = \frac{p\, {r_{\mathrm{pb}}}^2\, (1-\nu)\, (1-\cos(\theta_{\mathrm{pb}}))}{2\, E\, \Delta y} \tag{3-4}$$

The only parameter that was not defined in Figure 3-8 is the allowed deformation in the longitudinal direction, i.e. Δy. This value is determined as a percentage of r_{pb}, the radius of the aft bulkhead.

A similar analytical setup is used for the nose pressure bulkhead with the difference that the geometrical shape of the nose bulkhead is represented using a flat circular plate. Based on this, the equation below [23] may be used to determine its thickness. The allowed deformation is again defined as a percentage of the radius, however this time of the nose pressure bulkhead.

$$t = \frac{p\, {r_{pb}}^4\, \left(1-\nu^2\right) (5+\nu)}{\Delta y\, 64\, E\, (1+\nu)^{\frac{1}{3}}} \tag{3-5}$$

Nose Surface

One final surface who's structural properties are predominantly based on the pressure that acts on it is the nose surface which has been shown in Figure 3-10.

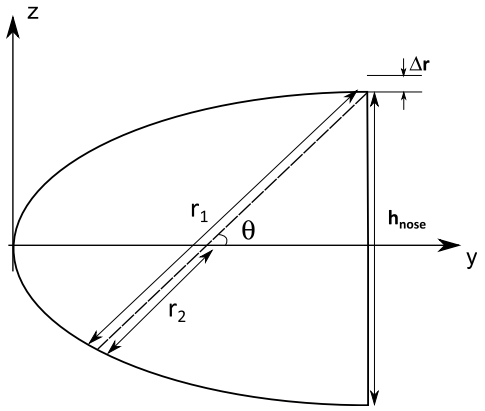


Figure 3-10: Parametrization of the nose surface for structural calculations

For the determination of the properties of this surface, the nose was represented as a smooth body of revolution. Therefore, the nose droop was ignored during the calculations and the shape in Figure 3-10 revolved to create an elliptically-shaped cone. In the following equation [23], the deformation Δr (displacement in the radial direction) is determined by taking a percentage of the height of the nose (h_{nose}).

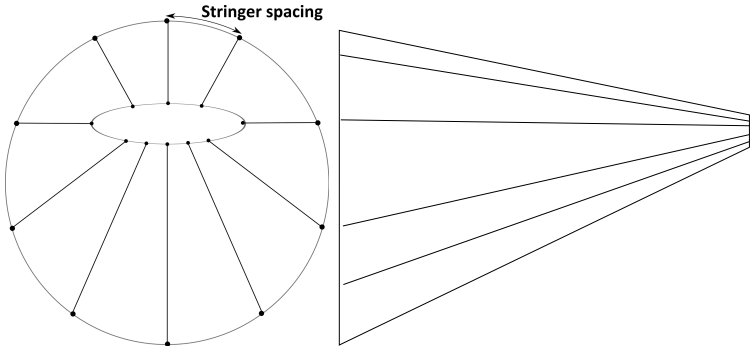
$$t = \frac{p\, {r_2}^2 \sin(\theta)}{2\, E \Delta r} \left(2 - \frac{r_2}{r_1} - \nu\right) \tag{3-6}$$

Tail Surface

Also for the definition of the tail cone an idealized model is used. Longitudinal booms and skin panels are joined together to represent the tail surface as shown in Figure 3-11. In this model, the normal and shear stresses are separated in the sense that the booms are responsible for carrying all of the normal loads while the skin carries the shear stresses. A clamped state is assumed for the edges at the connection of the tail surface with the skin.

This idealized situation is used to determine the normal and shear stresses present at each location. The main contributors to the stress are the weight of the tail cone itself and the weight and aerodynamic loads of the horizontal and vertical stabilizer that are connected to the tail.

Following the fuselage shear and bending calculation procedures described in Chapter 22 of Megson [24] the required thickness of the tail cone was determined.



(a) Front view idealized tail

(b) Side view idealized tail

Figure 3-11: Idealized tail geometry used as defined in Megson [24]

Cargo Floor

In contrast to the previous surfaces the cargo surface is a sandwich structure. In order to determine the two parameters required, i.e. the core and facings thickness, the equation below [25] may be used. In these equations l_{cf} and w_{cf} represent the length and width of the cargo floor, while B_{s} is the bending stiffness, A_{cf} the cross-sectional area of the core and G and E the shear and Young's modulus respectively. This equation is solved within a loop until the mass is approximately equal to that calculated by the SAWE method and the structure itself is strong enough to withstand the loads that act on it. The pressure load in this case is the total weight that the cargo floor needs to carry divided by the area of the floor itself.

$$\Delta z = \frac{5\,p\,{l_{cf}}^4}{384\,B_s}\,(1+3.2\,k) \tag{3-7}$$

$$k = \frac{3\,B_s}{{l_{cf}}^2\,G_c\,A_c} \tag{3-8}$$

$$B_s = \frac{E_f\,t_f\,(t_c+t_f)^2\;w_{cf}}{2} \tag{3-9}$$

Floor Strut

A final structural component that was not yet defined in the SAWE method but was necessary to resolve the excessive deformations of the cabin floor is the floor strut. These structural components are sized based on the fact that the critical buckling load should never be achieved. Therefore, by solving the following equation [23] for the radius of the floor strut, the cross-sectional area can be determined and the definition of the floor strut within the FE model completed.

$$r = \left(\frac{C\,\pi^3\,E}{4\,l^2\,F_{crit}}\right)^{\frac{1}{4}} \tag{3-10}$$

In this equation, F_{crit} represents the critical buckling load. The value of this parameter is determined by taking a percentage of the load the floor has to carry and dividing this result

by the number of floor struts present. Further, C represent the buckling factor and l the length of the floor strut.

Final Remarks Concerning Structural Recalculations

It should be noted that during the calculations discussed above, care was taken to stay as close to the structural weight of the component as determined by the SAWE methods. This to ensure that the calculated weight distribution along the fuselage is relatively unaltered and the longitudinal balance maintained.

Finally, the above equations simply present a first (conservative) estimate of the thickness of these components and should not be seen as final structural properties. These calculations simply allow for the complete FE model to be constructed and analyzed without excessive deformations/influence of the recalculated structural components.

3-5-2 FEA Code Generation

The pre- and post-processor used to create the FE model and process the analysis results is Msc. Patran which uses PCL as its programming language. This command language allows the user to communicate with the software using text-based inputs. Based on the data generated by the preceding modules the FEA Code module is capable of writing all of the PCL code required to automatically generate a complete FE model.

The tasks that are normally carried out by hand but that are now automated through code generation are the importing of the geometric data files (i.e. STEP files), creation and assignment of structural and material properties, definition and placing of loads and BCs and the combining of these loads and boundary condition (BC)s into a set of load cases.

The method used to automate the tasks related to the setup of a FE model is to translate each task into a function. This was achieved by firstly carrying out all of the aforementioned tasks by hand and taking note of the associated PCL code that was provided by the FEA pre-processor in its command window output. These PCL codes were then rewritten into a function format to allow for them to be used (and reused for future projects) to automatically write the code required for the automated setup of the FE model based on a number of inputs.

3-6 FEA Module

With the data and code files for the FEA software generated, the FEA module is started. This boots the FEA pre-processor software, i.e. Msc. Patran, and allows for the input of a single line of code. This will instruct Msc. Patran to read in the input file containing the PCL code which tells it what commands to carry out. This results in a sequence of events which create the complete FE model. Once these tasks have been completed the model is analyzed using Msc. Nastran and the results extracted.

In the paragraphs below, the setup of the FE model is discussed to provide insight into the factors that may influence the results of the analysis. The paragraphs follow the same order as that in which the code is read into the FEA's pre-processor.

3-6-1 Geometry Definition

As mentioned earlier, the geometric model is defined within the FEA pre-processor by importing STEP files. The different surfaces making up the complete fuselage are stored in separate files each representing a part of the fuselage structure. The reason for splitting the surfaces into separate files and importing these in a specific order, is related to the traceability of the different surfaces within the FEA software. This facilitates the process of assigning structural and material properties, loads, BCs, etc. The reason for this is that by splitting up the geometry into a number of separate files, the order in which the surfaces are imported into the FEA pre-processor is controlled and consequently the numbering of the surfaces can be controlled. By knowing which surface is linked to which number in the FEA one can easily retrieve the correct surfaces to assign the properties, loads and BCs to.

3-6-2 Material and Property Definition

In order to fully define the surfaces that were imported, the surfaces need material and structural properties (i.e. density, thickness, material orientation) assigned to them.

For the material properties, two types of materials are used within the oval fuselage structure, i.e. isotropic and honeycomb materials. The fuselage skin, pressure bulkheads and floor struts have isotropic materials assigned to them while the trapezoidal box and cargo floor are given sandwhich panel properties. The latter property is assigned by providing the layup of the sandwich panel through the specification of the facing and core materials and their respective thicknesses.

The last properties that needs to be defined are the frame shapes. C-frame properties are assigned only to the frames contained within the cabin area. The reason for this is that the Initiator only determines the values of the structural properties of the components present within the extent of the cabin. Furthermore, during the recalculation of the components outside of the cabin area, i.e. the nose and tail sections, these components are sized to withstand the loads that are present without the presence of these frames. This is done to simplify their structural recalculations.

3-6-3 Model Meshing

Due to the redefinition of the geometry, the FEA code module is capable of ensuring that the nodes created on a surface or beam align with those present on the ones connected to it. By equivalencing the model, the FEA then understands that these components are connected to one another.

However, one of the implications that follows from this redefinition is that due to the large variety in the components' dimensions the placement of the mesh seed cannot be done using mesh size seeing as this could result in both quadrilateral and triangular elements being created. An example of this is shown in Figure 3-12. When this happens, issues when meshing the surface and assigning the element types are caused.

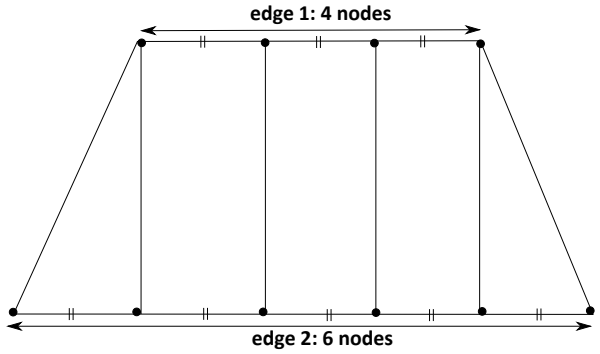


Figure 3-12: Potential error when using mesh size seed

To overcome these issues, the FEA Code module is capable of defining separate mesh seeds (based on number of seeds per edge) for the separate edges of a surface. Furthermore, this allows the user to have more control over the meshing of the model and prevent the aforementioned complication.

3-6-4 Element Types

Three main element types are used within the FEA and these are the shell, rod and beam elements. The outer skin, trapezoidal box and bulkheads are all created using shell elements while the floor struts are rod elements and the frames are represented as beam elements.

3-6-5 Loads

For the semi-analytical weight estimation, the loads considered during the analysis are the differential pressure inside the cabin and the inertial and aerodynamic loads of oval fuselage structure and the components attached to it. Within the FEA these are represented using inertial loads, pressure loads and total loads. All the loads are discussed below and illustrated in Figure 3-13 (except the cabin differential pressure). It should be noted that these representations serve a purely illustrative purpose and do not represent the actual magnitudes of the loads.

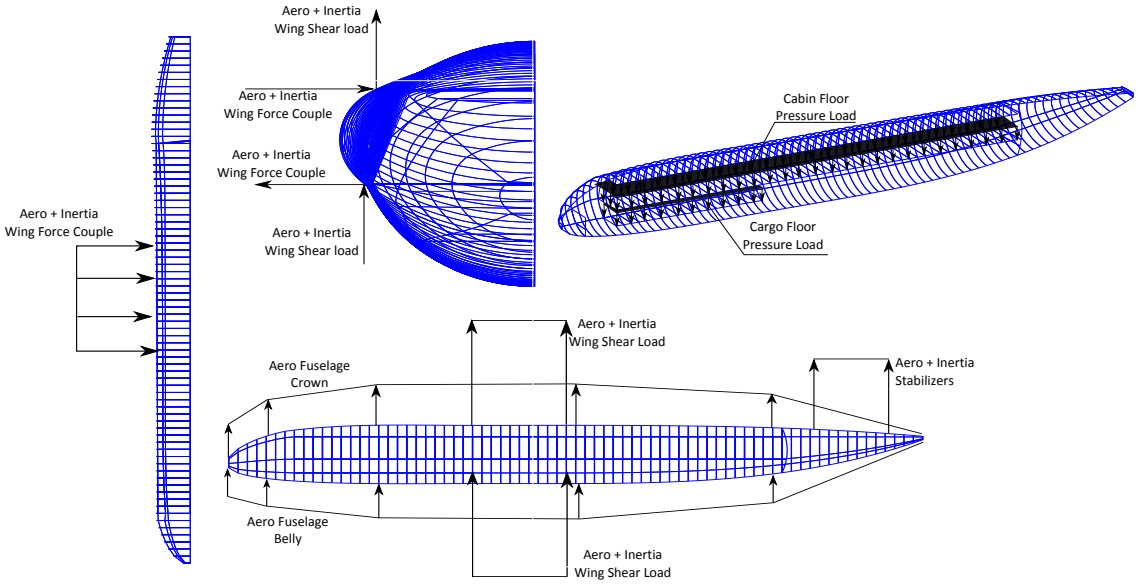


Figure 3-13: Illustration of the load application points

Pressure Loads

Pressurization loads are used within the FEA to model loads that are oriented normal to the surface they act on. For example, the cabin differential pressure is represented as a pressure load which acts on the inside of the fuselage skin contained within the front and aft bulkhead. Furthermore, seeing as the loads on the cargo and cabin floor are divided uniformly over the surface of these floors, pressure loads are also used to represent these.

Inertial Loads

The inertial loads of the structural components making up the oval fuselage are automatically incorporated due to the assignment of their material and structural properties and the creation of an inertial load representing the gravitational effect. However, those that result from the components attached to the fuselage need to be translated into loads acting on the fuselage which is done using total loads.

Total Loads

A total load is a load of which the magnitude gets uniformly divided over a length or surface area and which acts in a direction defined by the user.

For example, the inertial and aerodynamic loads from the vertical and horizontal stabilizer are modeled as a total load acting in the vertical direction on the crown curve contained within the longitudinal extent of the vertical stabilizer.

For the inertial and aerodynamic loads resulting from the wing and its attached components, i.e. engines and pylons, a force transformation is applied. The shear force and bending

moment of these components are translated into total loads representing a shear and force couple respectively. These act on the horizontal members of the trapezoidal box that are attached to the wing, i.e. the floor and ceiling surface contained in between the front and aft spar locations.

Finally, the aerodynamic load generated by the fuselage is also represented as a total load which acts on a slim strip on both the upper and lower surface of the fuselage.

Boundary Conditions

As mentioned earlier, by analyzing only half of the fuselage structure, symmetric boundary conditions need to be applied to correctly represent the analysis of a complete fuselage. These symmetric boundary conditions are applied to all the edges that are present on the symmetry plane along the longitudinal axis. These prevent any deflections that are perpendicular to the symmetry plane and any rotations about axis that are parallel to the symmetry plane, i.e. $\delta_x{=}\theta_y{=}\theta_z{=}0$.

Furthermore, the vertical and longitudinal displacements, i.e. $\delta_y{=}\delta_z{=}0$, are constrained by fixing the node on the floor and ceiling at the longitudinal position of the front spar. The reason for choosing this location to constrain these degrees of freedom is related to the fact that at the connection of the wing and fuselage the force balance is thought to be zero which in an ideal situation would mean that there are no displacements at this location. Finally, it should be mentioned that also the loads that act in or about the plane of symmetry need to be considered. For example, the aerodynamic loads of the fuselage and the horizontal stabilizer need to be divided by two in order to create a model that correctly represents the real-life situation.

Chapter 4

Application, Verification and Validation

Using the AFG tool the conceptual design data can be translated into an analysis ready FE model. However, before using the results from the FEA as a base of comparison for those of the SAWE method it is checked whether the FEA produces the results as expected.

Due to the novelty of the oval fuselage concept no comparable data is available to be used for the verification of these models. This limits the verification to the use of analytical/logical reasoning to form a conclusion on whether the results are as expected or not.

Once the FE model is verified the validation of the results of the SAWE method is carried out by plotting the longitudinal, lateral and shear load results of both methods into the same graph. This allows for the results to be compared and conclusions to be drawn with respect to the accuracy of the SAWE method.

For both the verification of the FE model and the validation of the SAWE method the same test-case is used, i.e. the same conceptual design data is used to build both FE models. Therefore, firstly the details of this test-case are discussed to provide insight into the FE models used.

As a final note it should be mentioned that the reason for separating the discussion of the FE model verification and the SAWE method validation is that the comparison of the load behaviour as calculated by both methods can be explained more clearly. This is related to the fact that for some of the results the magnitudes are significantly different. When plotting both these result sets on one graph, the load behaviour of the load with the smaller magnitude can become indistinguishable. Therefore, the two have been split to provide a clear indication of the load behaviour as predicted by both methods.

4-1 Test-Case

The test-case used within the next two sections models a straight oval fuselage which has been designed based on the top-level requirements of a large passenger aircraft with limited cargo capacity. In the paragraphs below the details of the top-level requirements, geometry and load case are discussed.

4-1-1 Fuselage Geometry

Provided in Table 4-1 are the top-level requirements that are passed on to the Initiator. These are used to create the conceptual design of an aircraft having a straight oval fuselage capable of meeting the requirements stated below.

Requirement	Value	Unit
Passengers	555	[-]
Payload	52000	[kg]
Cruise mach number	0.82	[-]
Altitude	10670	[m]
Range	12223	[km]
Take-off distance	2988	[m]
Landing distance	2104	[m]
Max lift-to-drag ratio	20	[-]
Maximum lift coefficient	1.4	[-]
Wing aspect ratio	7.5	[-]

Table 4-1: Top-level requirements for the large passenger aircraft

The result of the translation of these top-level requirements into a conceptual design results in the oval fuselage geometry displayed in Figure 4-1. The outer skin has been displayed in blue, the trapezoidal box and cargo floor in red and the pressure bulkheads in green.

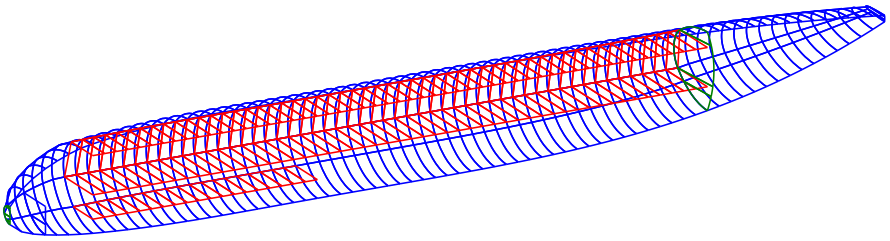
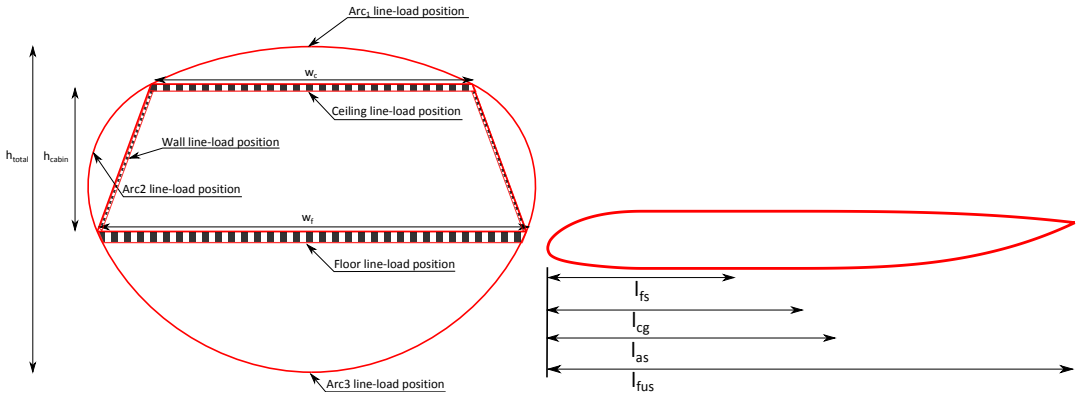


Figure 4-1: Straight oval fuselage geometry

To provide some insight into the dimensions of the oval fuselage geometry Figure 4-2 illustrates the front view of the oval cross-section and the side view of the longitudinal cross-section together with the parameters linked to them.

Figure 4-2a also illustrates the points at which the loads are extracted from the FEA. The reason for choosing these locations is to minimize the influence of the front spar boundary condition on the line load results. It should be noted that for the floor and ceiling, the loads

in the bottom facing of the sandwich structure are used while for the wall, the inner facing is considered.



(a) Oval cross-section **(b)** Longitudinal cross-section

Figure 4-2: Straight oval fuselage parameters

The values of the parameters displayed in the figure above have been collected in Table 4-2.

Parameter	Description	Value	Unit
h_{cabin}	Height of the cabin	3.1	[m]
h_{total}	Total height of the fuselage	8.1	[m]
l_{as}	Longitudinal position of the aft spar	36.2	[m]
l_{cg}	Longitudinal position of the center-of-gravity	34.2	[m]
l_{fs}	Longitudinal position of the front spar	28.2	[m]
l_{fus}	Length of the fuselage	75.0	[m]
w_c	Width of the ceiling	6.8	[m]
w_f	Width of the floor	9.6	[m]

Table 4-2: Dimensions straight oval fuselage

Some additional information concerning the SAWE's weight prediction for this test-case is provided in Table 4-3.

Parameter	Weight [N]
Maximum Take-off Mass	2204909
Operational Empty Mass	100653
Maximum Fuel Mass	114439

Table 4-3: Weight results of the SAWE method

4-1-2 Load Case

The load case used in this analysis resembles cruise flight conditions at Mean Take-off Mass (MTOM) and maximum differential pressure, fuel weight and load factor (i.e. excluding the cruise cabin pressure, all loads have been multiplied by a load factor of 2.5). The complete load case description is provided in Table 4-4. For a visual representation of the application points of the loads one can refer to subsection 3-6-5.

Load/BC	Location	Load type
Cargo inertia	Cargo floor	Pressure
Cabin floor inertia	Cabin floor	Pressure
Cruise cabin pressure	Skin between front/aft bulkhead	Pressure
Engine inertia load	Floor and ceiling between front/aft spar	Total load
Fuel aero load	Floor and ceiling between front/aft spar	Total load
Fuel inertia load	Floor and ceiling between front/aft spar	Total load
Fuselage aero load	Fuselage upper and lower surface	Total load
Oval fuselage aero load	Outside of skin on top and bottom	Pressure
Stabilizer aero load	Outside of skin at stabilizer	Total load
Stabilizer inertia load	Outside of skin at stabilizer	Total load
Wing aero load	Floor and ceiling between front/aft spar	Total load
Wing inertia load	Floor and ceiling in between front/aft spar	Total load

Table 4-4: Load case used

The magnitudes of these loads have been collected in Table 4-5. It should be noted that these loads are already multiplied (except for the cruise cabin pressure) by the maximum load factor of 2.5.

Load	Magnitude	Dimension	Line length [m]
Cabin floor inertia	-1098.0	N/m2	-
Cargo inertia	-1218.0	N/m2	-
Cruise cabin pressure	55660.0	N/m2	-
Engine force couple	-6128.0	N/m	7.5555
Engine inertia shear	-3051.0	N/m	7.5555
Fuel force couple	-90285.0	N/m	7.5555
Fuel inertia shear	-31020.0	N/m	7.5555
Fuselage aero load	4669.0	N/m2	-
Wing aero force couple	87747.0	N/m	7.5555
Wing aero shear	82597.0	N/m	7.5555
Wing inertia force couple	-23728.0	N/m	7.5555
Wing inertia shear	-8152.0	N/m	7.5555

Table 4-5: Magnitudes loads acting on test-case

4-1-3 Load Extraction and Processing

In all of the load comparison plots that follow in the next two sections, use is made of the longitudinal, lateral and shear line loads present within the front and aft limits of the cabin area. The reason for using the line loads is to investigate the loads independent of the thickness of the component and obtain a better understanding on the load behaviour within them.

Seeing as the SAWE method already carries out its analysis using the line loads these can be directly be used for comparison. However, for the FEA, the load results are expressed as stresses rather than line loads. Therefore some post-processing is required in order to allow for a valid comparison to be carried out between the two analysis methods. For the normal line loads (N_{long} and N_{lat}) and shear flow (q) the following equations were used to convert the results of the FEA to line loads.

$$\sigma = \frac{N}{t} \tag{4-1}$$

$$\tau = \frac{q}{t} \tag{4-2}$$

One final note to be made with respect to the results shown in the next two sections is that towards the aft bulkhead the results may shown extreme changes when compared to the results in front of it. The reason for this is that the presence of the aft bulkhead forms a rigid constraint at these locations and therefore significantly impacts the loads. At the front of the cabin on the other hand, no bulkhead is present and thus no additional constraint is placed on the structure at this location.

4-2 Finite Element Model Verification

As mentioned in the introduction to this chapter, the extent of verification is limited by the lack of available comparable data for oval fuselages. Therefore, the verification of the FE model is based on whether or not the results of the FEA can be justified using analytical/-logical reasoning. In the paragraphs below, the results for the longitudinal, lateral and shear line loads are discussed for the outer shell and trapezoidal box structure. For the outer shell, the three different arcs are considered while for the trapezoidal box the results of the floor, wall and ceiling surfaces.

The solid red line represents the analysis where all loads are acting on the fuselage (as described in Table 4-4) while for the dashed red line only the differential pressure is taken into account. Furthermore, in the graphs in this and the following section, the vertical lines represent the front spar, center of gravity and aft spar location.

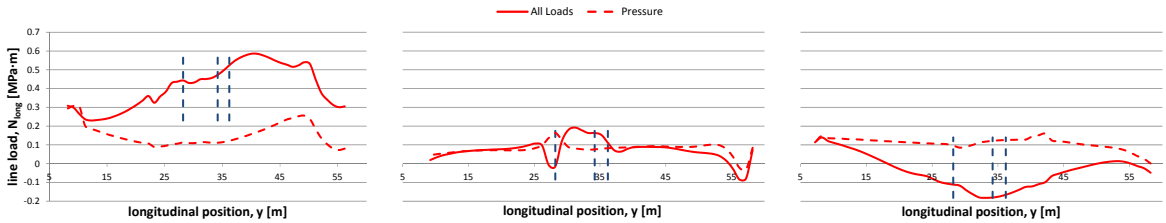
4-2-1 Outer Shell

The first components's line loads to be investigated are those of the three circular arcs forming the aerodynamic shell of the oval fuselage. Figure 4-3 shows from left to right the longitudinal loads for the top, middle and bottom arc, i.e. arc 1, 2 and 3 respectively.

When only the pressure acts on the fuselage a pure tensile state is present in all of the different arcs. This tensile load varies slightly along the longitudinal axis based on the changes in geometry of the arcs, i.e. the radius of each arc.

For the top arc, i.e. arc 1, it may be seen that the line load due to pressure decreases as one moves towards the center of gravity location. The reason for this is related to the fact that for the top arc the radius decreases towards the middle of the oval fuselage. When considering equation 2-11 this decrease can be related to the increase in the height parameter h_1 which causes a decrease in radius r_1. For the other two arcs, the variations in the height parameters, i.e. h_2 and h_3 for arc 2 and 3 respectively, are much smaller and therefore also the changes in line load.

By adding the longitudinal bending to the analysis, significant tensile and compressive loads are added to the upper and lower arc respectively, causing vertical shifts of the graphs. From this it also becomes clear that the bending loads are the most influential loads present in the longitudinal direction for these components. The effect of these bending loads is almost insignificant for the middle arc due to the high proximity of this arc to the center of gravity.



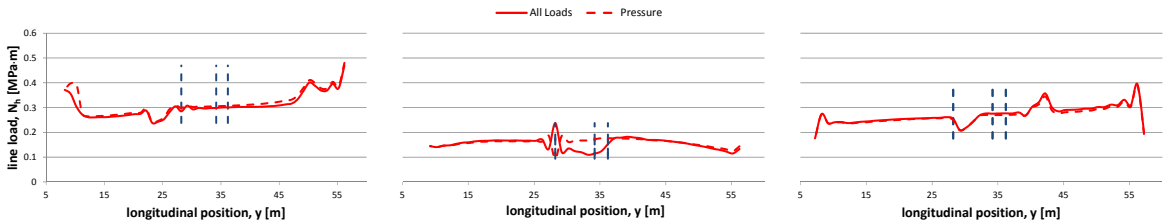
(a) Arc 1 **(b)** Arc 2 **(c)** Arc 3

Figure 4-3: Longitudinal line load comparison in Arc 1, Arc 2 and Arc 3. The vertical lines from left to right: front spar, center of gravity and aft spar location

In Figure 4-4 the lateral (i.e. hoop) loads present in the three circular arcs are plotted. In contrast to the longitudinal loads, these loads are almost exclusively dictated by the differential pressure as may be seen from the minimal difference between the dashed and solid lines in each graph.

Furthermore, when comparing the magnitudes of the longitudinal and lateral graphs due to the pressurization load, it may be seen that the magnitude in the longitudinal direction is approximately half of the lateral load, which follows the theory of pressurization of cylindrical shells as was explained in subsection 2-2-1. It should be noted that due to the presence of the trapezoidal box, the results of the theory and the actual results may differ seeing as this trapezoidal box provides additional resistance to the deformations in the longitudinal direction.

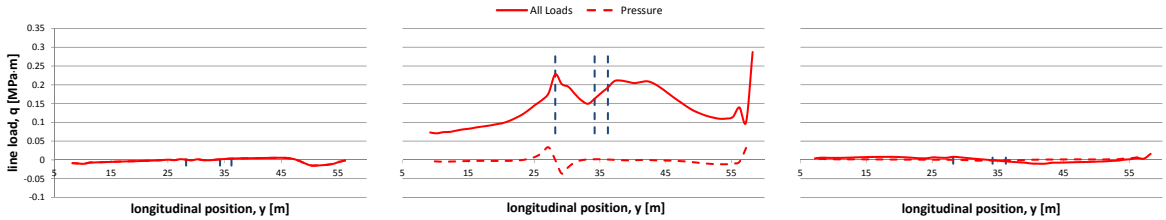
When considering the longitudinal and lateral line load plots it may be seen that a sudden increase/decrease in load is present in arc 2 near the location of the front spar. This is related to the higher proximity of the middle arc's nodes to the boundary condition at the front spar, which restrains the longitudinal and vertical motion of the fuselage and therefore results in higher line loads at these locations.



(a) Arc 1 **(b)** Arc 2 **(c)** Arc 3

Figure 4-4: Lateral line load comparison in Arc 1, Arc 2 and Arc 3. The vertical lines from left to right: front spar, center of gravity and aft spar location

For the investigation of the shear loads Figure 4-5 may be used. The first thing that may be seen is that the pressure has no significant contribution to the shear loads which is to be expected. Secondly, the results shown in these figures follow the theory related to shear flow in closed symmetrical sections. This states that based on to the symmetry and loading applied, the shear flow should be zero towards the top and bottom arc while reaching its maximum at the side arcs.



(a) Arc 1 **(b)** Arc 2 **(c)** Arc 3

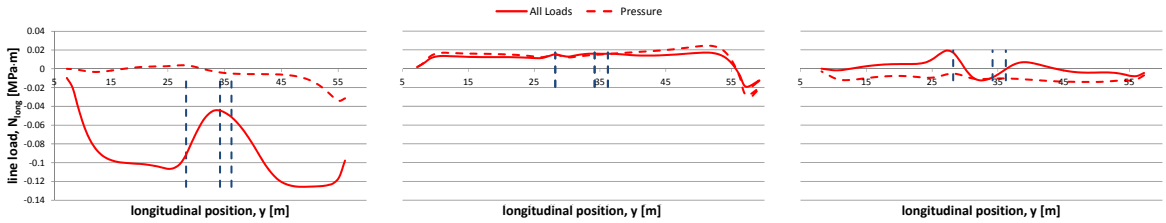
Figure 4-5: Shear line load comparison in Arc 1, Arc 2 and Arc 3. The vertical lines from left to right: front spar, center of gravity and aft spar location

4-2-2 Trapezoidal Box

The next structural component to be analyzed is the cabin area formed by the trapezoidal box structure. Within this structure, the floor, wall and ceiling are investigated which are all sandwich panels. It should be noted that all of these are sandwich panel constructions. The line loads shown in the plots below are those present in the bottom facing of the sandwich panel for the floor and ceiling and the inner facing for the wall.

Before analyzing the different components it should be mentioned that within the SAWE method a thickness of 1 nanometer was assigned to the wall's sandwich panel, i.e. the wall was not modeled as a sandwich structure. It is assumed that this was done because the calculations showed that the wall would be under purely tensile loading and therefore there would be no need for the sandwich panel to provide resistance against out of plane deformations. However, the FEA showed that these deformations were still present. Therefore the sandwich panel was given a higher thickness based on the thicknesses used for the floor and ceiling. This will result in a conservative value for the thickness of the wall's sandwich core but will ensure a more accurate FEA by limiting the influence of the deformations of the wall. For example, from the results of the FEA it could be seen that due to the deformation of the wall the integrity of the surrounding structural components, i.e. the floor and ceiling surface an the outer skin close to this area, is significantly affected.

From Figure 4-6 it may be seen that with respect to the longitudinal line loads the floor is most heavily loaded. In contrast to the wall and ceiling, which do not carry any additional loads, the floor is required to support the passengers, cabin utilities, etc. This explains the large difference between the pressure only and fully loaded analysis plots for the floor surface. The floor and ceiling are almost exclusively subjected to compressive loads whereas the wall is under a nearly constant tensile loading.



(a) Floor **(b)** Wall **(c)** Ceiling

Figure 4-6: Longitudinal line load comparison in the floor, wall and ceiling. The vertical lines from left to right: front spar, center of gravity and aft spar location

The reason for the compressive loading in the floor and ceiling surface is related to the bending of both these surfaces as shown in Figure 4-7. As may be seen these surfaces deform in the longitudinal direction with their ends being lower than the center, which causes the lower surface to be under compressive loading.

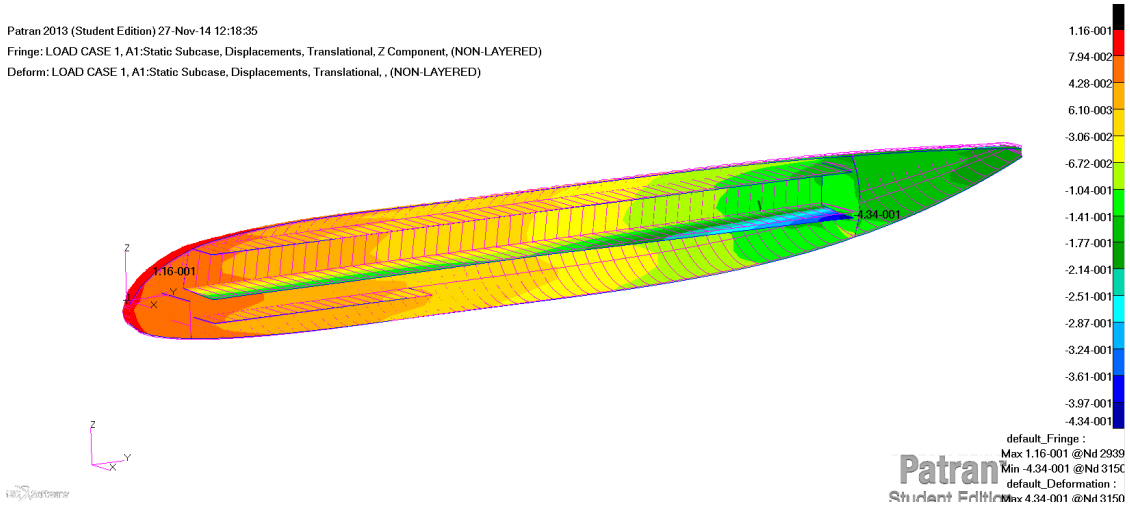
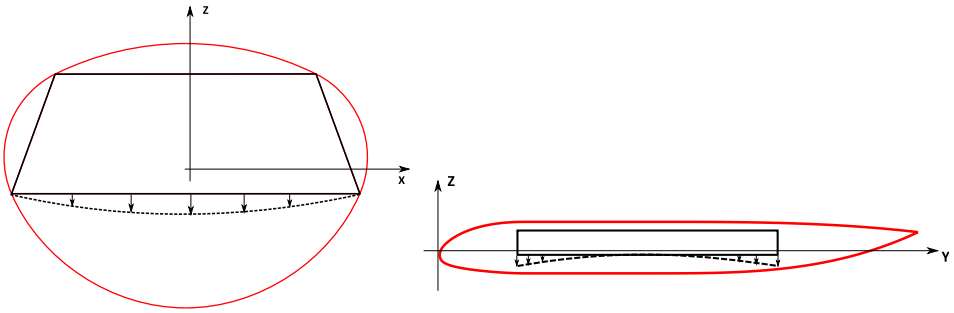


Figure 4-7: Vertical displacements of the oval fuselage

When considering the floor surface, a decrease in compressive loading may be seen around the center of gravity. This is believed to be related to the combination of lateral and longitudinal bending as illustrated in Figure 4-8a and 4-8b respectively. The deformation of the floor surface is limited due to the clamped condition of the floor at the intersection with the skin and the stiffness of the outer structure. However, as one moves away from the skin toward the middle of the floor surface, the vertical displacement increases. Similarly, as one moves away from the center of gravity along the longitudinal center line of the floor, the vertical displacement will increase. Therefore, as one moves towards the front and aft of the floor, the combination of compressive loads due to longitudinal bending and as a result of the Poisson effect become larger than when near the center of gravity.

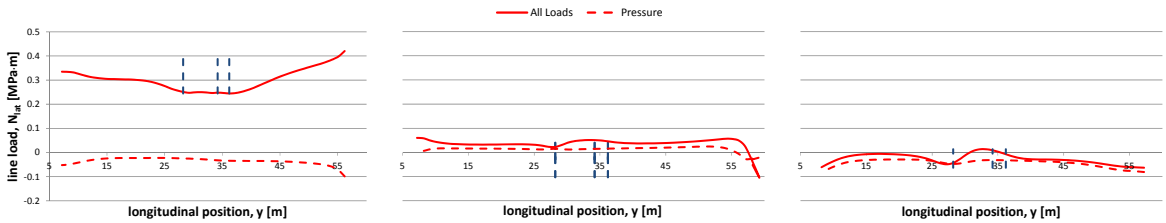


(a) Lateral bending floor **(b)** Longitudinal bending floor

Figure 4-8: Longitudinal and lateral bending of the floor surface

For the validation of the lateral loads, Figure 4-9 may be used. Similarly to the longitudinal case, the floor is most heavily loaded in the lateral direction. Seeing as the cabin width is larger than the cabin height, the floors and ceiling surfaces are subjected to compressive loads while the wall is subjected to tensile loading. This happens due to the reaction forces created by the circular arcs under pure differential pressurization seeing as the top and bottom arc's radius is larger than that of the middle arc which causes compressive forces to act on the horizontal members and tensile loads on the "vertical" components.

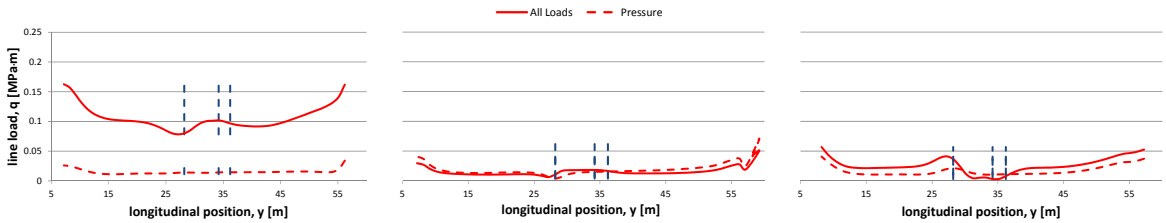
As seen from Figure 4-8a, due to the additional loading on the floor, the lateral bending will cause tensile loads to be present in the bottom facing. The magnitude of the loads increases as one moves towards the front and aft end of the floor due to the increased vertical displacement.



(a) Floor **(b)** Wall **(c)** Ceiling

Figure 4-9: Lateral line load comparison in the floor, wall and ceiling. The vertical lines from left to right: front spar, center of gravity and aft spar location

Finally, the shear loads have been plotted in Figure 4-10. As may be seen from this figure the shear loads in ceiling and wall are almost non-existent when compared to the floor surface. This is related to the additional loads the floor surface has to carry.



(a) Floor **(b)** Wall **(c)** Ceiling

Figure 4-10: Shear line load comparison in the floor, wall and ceiling. The vertical lines from left to right: front spar, center of gravity and aft spar location

4-2-3 Recalculated Components

In the paragraphs below, the different components for which the structural properties were recalculated as a result of the use of empirical calculations are investigated.

The nose and aft pressure bulkheads both deform in the longitudinal direction when the fuselage is subjected to the differential pressure. For the nose bulkhead, this deformation is relatively small ($\approx$ -4.5 cm) and does not influence the connected components. For the aft bulkhead, the deformation is larger ($\approx$ 30 cm) which causes a decrease in height of 1.5%.

For the nose and cargo surface no additional significant deformations exists besides the displacements due to the displacement of the complete fuselage. The tail surface on the other hand does experience additional deformation in the vertical direction at the point where the horizontal and vertical stabilizer loads are introduced onto the surface. The maximum vertical displacement in the tail surface is approximately 6% of the height of the oval cross-section at which this deformation takes place. This may be related to the fact that during the calculations the assumption was made that the sections under consideration were perfect ellipses rather than oval sections, which assumes a stronger structural shape than is actually present.

As mentioned earlier, the floor surface undergoes significant vertical displacements and therefore it was opted to investigate the use of floor struts as a solution to this problem. Through the placement of the floor struts, the deformation of the floor was significantly decreased.

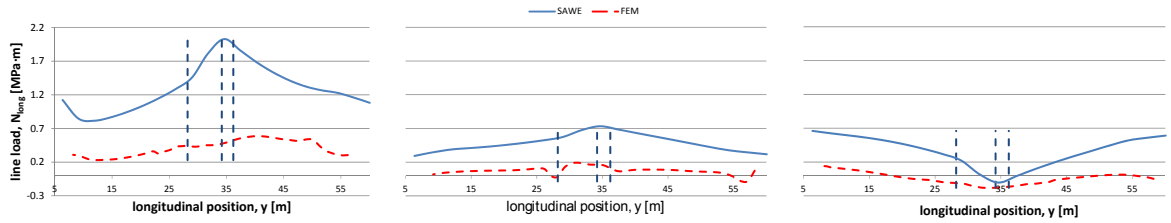
4-3 Load Comparison

Within the SAWE method not all the structural components are investigated using analytical methods. Therefore a complete validation of all of the fuselage's components is not possible. This limits the comparison of the two analysis methods to that of the three circular arcs and the members of the trapezoidal box, i.e. the floor, ceiling and vertical wall.

In all the graphs in this section, the blue line represents the results for the line load from the SAWE method while the dashed red line represents those from the FEA.

4-3-1 Longitudinal Line Loads

When comparing the longitudinal line loads in the circular arcs of the oval fuselage (Figure 4-11) it may be seen that although similar load behaviour is seen the SAWE method obtains significantly higher tensile loads than the FEA, especially for arc 2. This may be explained by considering the overall displacement/rotation of the oval fuselage structure.



(a) Arc 1 **(b)** Arc 2 **(c)** Arc 3

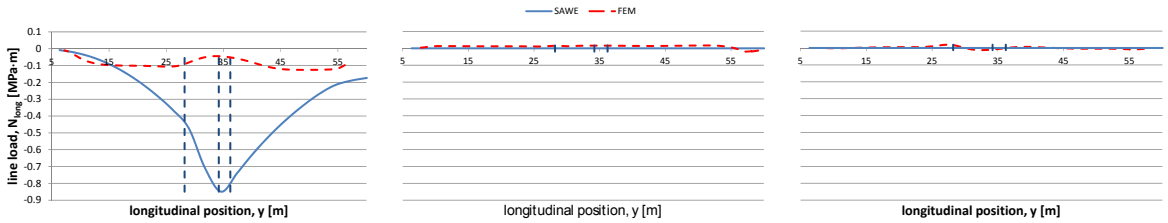
Figure 4-11: Longitudinal line load comparison in the outer skin. The vertical lines from left to right: front spar, center of gravity and aft spar location

From Figure 4-7 it may be seen that that rather than hanging from the front spar boundary condition, the oval fuselage rotates with an upward displacement of the nose and a downward displacement of the tail section. This explains why the stresses as found by the SAWE method, in which the fuselage hangs from the center of gravity, are significantly higher than for the FEA. The longitudinal stresses that were found in the FEA are related to the unequal rotation of the oval fuselage, which results in a non-linear displacement along the longitudinal axis and thus creates loads in this direction.

The reason as to why the oval fuselage rotates in the FEA can be related to the longitudinal balance of the oval fuselage. When the different loads were investigated, it was found that the aerodynamic loads that act on the oval fuselage are not accurate. This results in a longitudinal imbalance that causes the fuselage to rotate rather than allow for a simulation of the cruise flight situation. The reason for this can be related to the error in reference values provided by the Initiator modules to to the aerodynamic solver used within it. Furthermore, this aerodynamic solver (Athena Vortex Lattice (AVL) is known to provide inaccuracies with respect to the lift distribution data when it comes to the longitudinal direction of a lifting body.

With the aerodynamic loads on the fuselage of BWBs being significantly larger than for conventional aircraft, the interpretation of the results of a BWB fuselage analysis would be even more difficult. This difficulty is further increased due to the fact that for BWB aircraft, the aerodynamic loads on both the fuselage and the wings were found to be incorrect. This is again related to the fact that the wrong reference values are provided to the aerodynamic solver for aircraft with fuselages that act as lifting bodies.

For a comparison of the longitudinal loads in the trapezoidal box, Figure 4-12 may be used. From Figure 4-12b and 4-12c it may be seen that the SAWE method was correct in assuming that the contribution of the wall and ceiling surface in carrying the longitudinal bending loads is negligible [21]. However, the longitudinal loads present within the floor are once again much larger in the SAWE method than in the FEA.

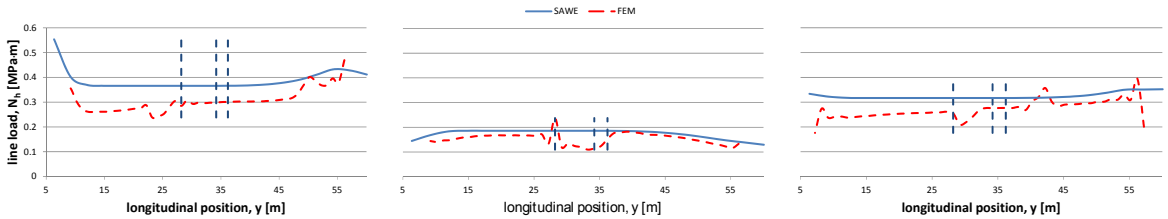


(a) Floor **(b)** Wall **(c)** Ceiling

Figure 4-12: Longitudinal line load comparison in the trapezoidal box. The vertical lines from left to right: front spar, center of gravity and aft spar location

4-3-2 Lateral Line Loads

The next loads to be compared are the lateral line loads. Figure 4-13 shows the results for the circular arcs from which may be seen that there is a very high correspondence between the two analysis methods. With the lateral loads consisting almost exclusively of the differential pressure for the circular arcs, the number of additional factors influencing the analysis are minimized which explains the similar results.

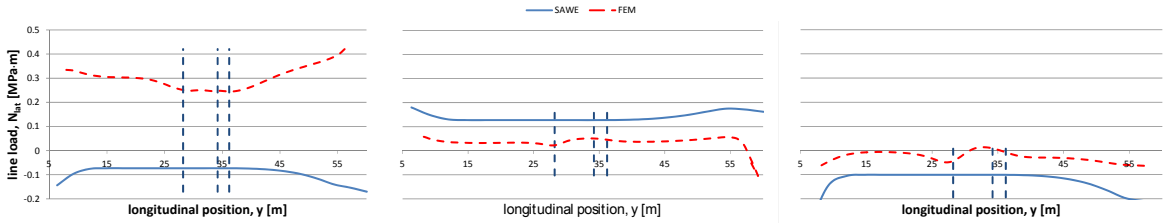


(a) Arc 1 **(b)** Arc 2 **(c)** Arc 3

Figure 4-13: Lateral line load comparison in the outer skin. The vertical lines from left to right: front spar, center of gravity and aft spar location

The lateral loads for the trapezoidal box are displayed in Figure 4-14. For the floor surface, an opposite lateral stress state is seen. The FEA shows a tensile loading while the SAWE

method shows a compressive loading. This difference can be explained due to the fact that within the SAWE method the extensive bending of the floor, as shown by the FEA, was not foreseen. Therefore, a significant tensile load was not added to the compressed floor surface. For the wall surface, the two analysis methods provide very similar results while for the ceiling surface a small difference exists between the two calculated compressive loads. This can also be related to the slight bending of the ceiling surface which will induce tensile loads in the lower facing.

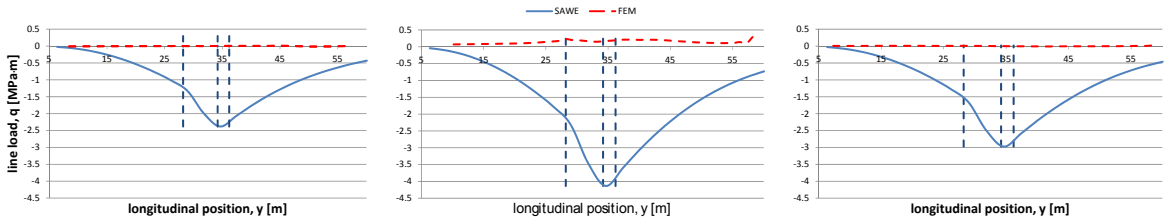


(a) Floor **(b)** Wall **(c)** Ceiling

Figure 4-14: Lateral line load comparison in the trapezoidal box. The vertical lines from left to right: front spar, center of gravity and aft spar location

4-3-3 Shear Line Loads

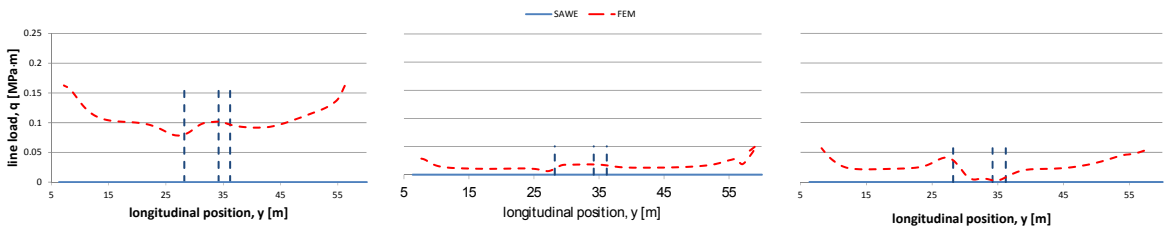
A significant difference exists between the shear flow calculated by the SAWE method and that from the FEA for the circular arcs. Unlike the FEA, the top and bottom surface show significant shear stresses which contradict the theory of shear flow in closed section beams. As was mentioned earlier, based on the symmetry and loading applied, the shear flow should be zero towards the top and bottom arc while reaching its maximum at the side arcs.



(a) Arc 1 **(b)** Arc 2 **(c)** Arc 3

Figure 4-15: Shear line load comparison in the outer skin. The vertical lines from left to right: front spar, center of gravity and aft spar location

Within the SAWE method, the assumption was made that the members of the trapezoidal box structure do not provide any shear load carrying capabilities. From Figure 4-16 it may be seen that this can be considered true for the wall and ceiling surfaces but not for the floor surfaces seeing as the magnitude of the loads is too high in this component.



(a) Floor **(b)** Wall **(c)** Ceiling

Figure 4-16: Shear line load comparison in the trapezoidal box. The vertical lines from left to right: front spar, center of gravity and aft spar location

4-4 Von Mises Loading

From the discussion above, it becomes clear that for some structural components a significant difference in magnitude exists between the calculated loads. This has a strong influence on the yield criterion used within the SAWE method and therefore makes it difficult to gain insight into whether or not the weight estimation carried out in this analysis method is accurate and whether the weight can be improved or not.

Chapter 5

Conclusion and Recommendation

5-1 Conclusions

Motivated by both the potential of the Blended Wing Body (BWB) aircraft to meet the industry's demands and the promising results the initial studies of the oval fuselage showed for its use on both conventional and unconventional aircraft, the goal of this thesis project was to analyze the oval fuselage through the use of a Finite Element Analysis (FEA). Concretely, this meant the validation of the results of the Semi-Analytical Weight Estimation (SAWE) method by comparing them to those obtained from a FEA.

At the start of this thesis project, it was soon realized that due to the heavy dependence on FEA for the validation process the number of Finite Element (FE) models that would need to be set up would be high. Therefore, in order to facilitate this validation process and provide an aid for future research a Knowledge-based Engineering (KBE) approach was used for this project. As a result the Automated Finite element model Generator (AFG) was created which translates the conceptual design data of aircraft fuselages into an analysis-ready FE model and overcomes the non-creative and time-consuming tasks related to the setup of FE models. To allow for the AFG to automate this complete process, four different processing modules were developed, i.e. the Initiator Controller module, the Geometry module, the FEA Code module and the FEA module. Through the use of these four processing modules the required conceptual design data was collected, the geometry of the fuselage redefined to allow for a direct import in the FEA pre-processor and the Patran Command Language (PCL) code required to automate the FE setup generated.

Before using the FE models created by the AFG a verification of the results of these models was carried out. Due to the lack of oval fuselage data, use was made of logical/analytical reasoning to show that the results produced by the FEA were as expected. However, the verification of the FE models should be extended to a comparison based on empirical data.

From the comparison of the results of both the SAWE method and the FEA a number of interesting conclusions can be drawn. For the circular arcs, a relatively good correspondence was seen between the loading behaviour in the longitudinal and lateral direction as determined

by the two methods. For the longitudinal direction the difference in magnitude of the results was significantly larger than for the lateral direction. This is related to the difficulties with obtaining a longitudinal balance for the FE model as a result of the inaccuracies in the aerodynamic loads acting on the fuselage as calculated by the Initiator. For the lateral direction, a near perfect match was found between the magnitudes of the loads calculated due to the fact that the differential pressure was the largest (and virtually sole) force acting in this direction for the outer skin. When comparing the results of the shear loading a better correspondence between the theory and analysis results was found for the FEA than for the SAWE method.

For the case of the trapezoidal box, the main differences were found in the results of the floor, which may be related to the extensive deformation of this surface which was not taken into account by the SAWE method. This deformation was resolved through the use of floor struts that reinforce the floor and decrease its vertical displacement.

The additional components that were recalculated, i.e. the nose surface, tail surface, cargo floor and pressure bulkheads, did not show excessive deformation and remained below the yield stresses. This shows that the calculations provide a good but very conservative initial estimation of the thicknesses required for these components.

With respect to the weight of the oval fuselage it is difficult to provide a final conclusion of the accuracy of the semi-analytical weight estimation method and whether or not the oval fuselage structure can be optimized. The reason for this is that the sizing criterion of the semi-analytical weight estimation is based on the Von-Mises stresses which are influenced by all the loads present. Due to the large differences between the results of both methods it is difficult to therefore indicate whether the structural components can be made lighter or not. Therefore, additional testing would be required to further validate the FE models and the differences with the semi-analytical weight estimation should be investigated and altered in its methods.

5-2 Recommendations

FEA Using Improved Aerodynamic Fuselage Loads

One of the most significant aspects that can be considered to improve the validation study carried out in this thesis project is the use of an improved aerodynamic solver for the determination of the aerodynamic loads. When better results are obtained for the aerodynamic force distribution in the longitudinal direction the cruise flight situation can be represented more accurately in the FE model.

Experimental FE Model Verification

Due to the lack of experimental data, the verification of the FE model is limited to the use of logical/analytical reasoning. Therefore, in order to gain better insight into the accuracy with which the FE model simulates the real-life situation some experiments could be carried out on a scale model of the oval fuselage to obtain comparable data that can be used for a more in-depth FE model verification.

Improved Geometric Redefinition

Currently, the geometric redefinition of the oval fuselage is based on the locations of the frames and cargo, floor and ceiling surfaces. This means that only at the intersections of these horizontal and vertical planes nodes will be created. This limits the application points of loads and other boundary conditions to these locations and consequently limits the accuracy with which these represent the real-life situation. Therefore, the use of virtual frames in combination with the real frame locations may provide even more accurate representations of the oval fuselage in the FE model.

Optimization Routine

Although not incorporated within this thesis project, but with an eye to future expansion possibilities, an Optimization module could be implemented into the architecture of the AFG as shown in Figure 5-1. This would require the automatic extraction and interpretation of the loads as generated by the FEA software. Once these are extracted, the results, in the form of internal loads present within the structural components of the fuselage, can be used within the Optimization module to allow for the optimized structural weight of the fuselage to be calculated. This module updates the properties of the oval fuselage based on the internal loads extracted from the FEA and recalculates the thicknesses required for each component. In this manner, the weight of the fuselage can be redetermined. The newly calculated weight is then compared to the SAWE weight in case it is the first iteration and otherwise to the weight calculated during the previous iteration of the loop. If the weight has not converged, the FE model is updated and the FEA is carried out again. This loop continues until the weight of the fuselage has converged.

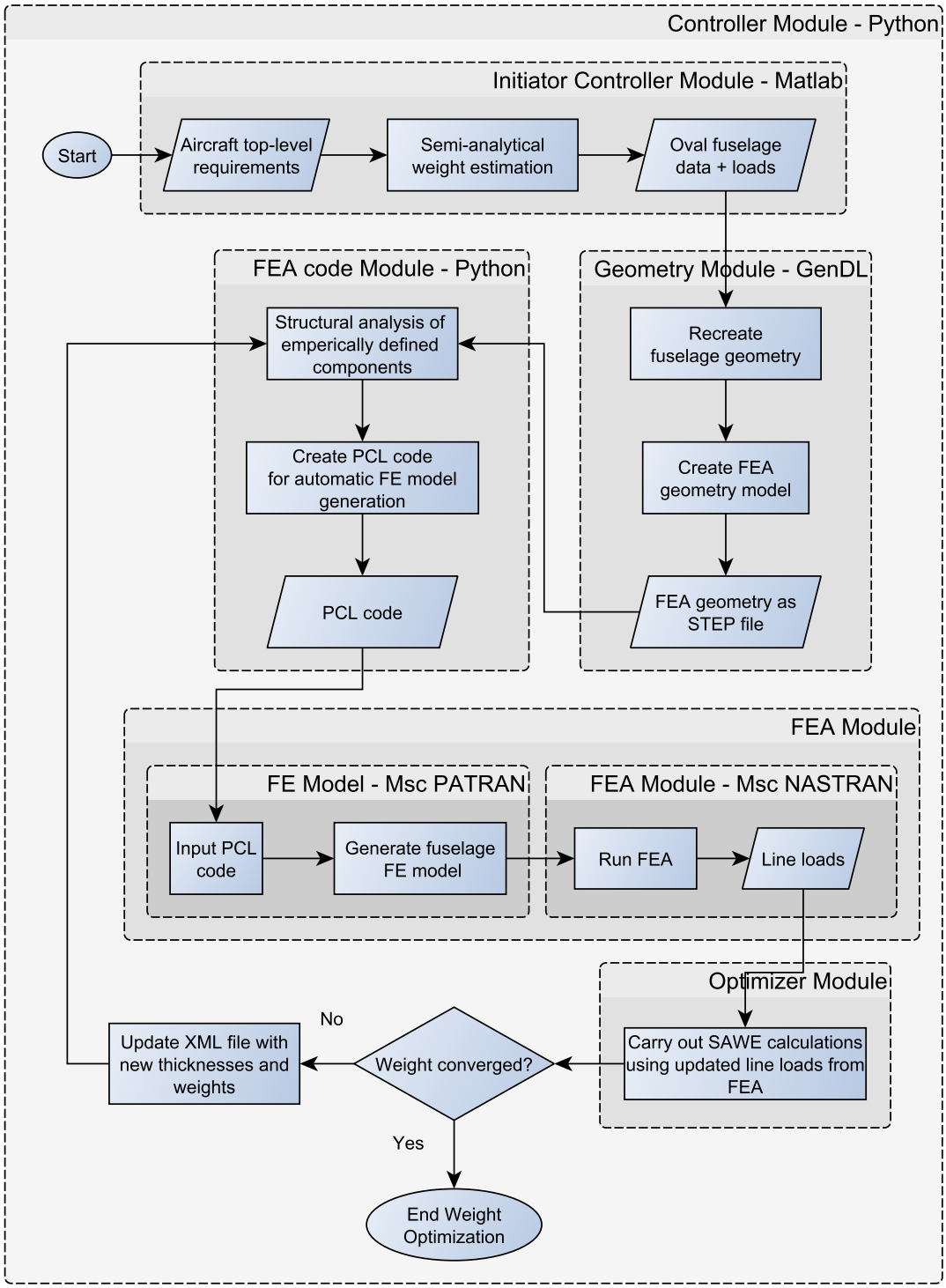


Figure 5-1: Flow of the AFG tool including Optimization module

Result Accessibility

With respect to the tool that was created it can be said that the accessibility of the results after processing is limited to running the scripts required for each module using the correct software and retrieving the information through the parameters/variables linked to the information.

This could be significantly improved by either creating new output files or appending/editing the data in the original Extensible Markup Language (XML) file written by the Initiator.

Part II

Operational Manual

Chapter 6

Manual Introduction

The AFG tool was developed as a KBE tool to assist in the validation of the oval fuselage concept for novel aircraft configurations. The validation method used in this project was based on a comparison of the loads present in the fuselage's structural components. The FE model used for the higher fidelity analysis was based on the conceptual design results as carried out by the SAWE.

To fully define the oval fuselage's FE model, the following actions need to be carried out:

- Recreation of the oval fuselage's geometry
- Meshing of the geometric model
- Definition and application of the materials and properties
- Definition and application of the loads and boundary conditions
- Setup of load cases

The above actions may be carried out manually using a FEA's pre-processor for simple models and a limited number of analysis. However, due to the novelty of the oval fuselage concept, the large number of tests required to ensure the accuracy of the FE model and the different configurations that needed to be tested to validate the SAWE method the manual setup of the FE models was considered a highly inefficient approach. Therefore, the AFG tool was developed to allow for the automatic translation of the results of the SAWE method into an analysis-ready FE model.

Within this report the workings of the AFG tool and the different modules it comprises are explained in great detail to provide the user with both the knowledge of the workings of the tool as well as the architecture behind it.

As a final note, it should be mentioned that the SAWE method was created as part of a larger project called the Initiator[26]. This design tool allows for the translation of top-level requirements into a conceptual aircraft design based on a number of configuration design variables.

The main output of the Initiator comes in the form of a Extensible Markup Language (XML) document containing the results of the modules that were run during an analysis. These results include amongst others the dimensions of the parameters used in the parametrization, the structural and properties, the aerodynamic loads, a weight breakdown, etc.

Chapter 7

AFG Operation

In the sections that follow, each of the different modules contained within the AFG tool is discussed in greater detail. For each of the modules the following aspects will be discussed:

- Module Operation: The way in which the module is controlled using it's respective programming software/language.
- Input/Output: Specification of the input required and the output generated by this module.
- Module Architecture: An overview of how the module is built up and operates.
- Known Errors, Limitations and Assumptions: Errors that have been encountered during the AFG tool development and have not yet been resolved and limitations of the module in question.

7-1 Controller Module

The complete process described in Section 3-1 is managed using the Controller module. Using this module the user is able to run specific processing modules and edit the settings used within the AFG tool.

The Controller module is housed in a Python based command shell that accepts text-based input from the user and is initialized by opening the "PCLWriterTool.py" file. The command-shell at initialization of the AFG tool is displayed in Figure 7-1.

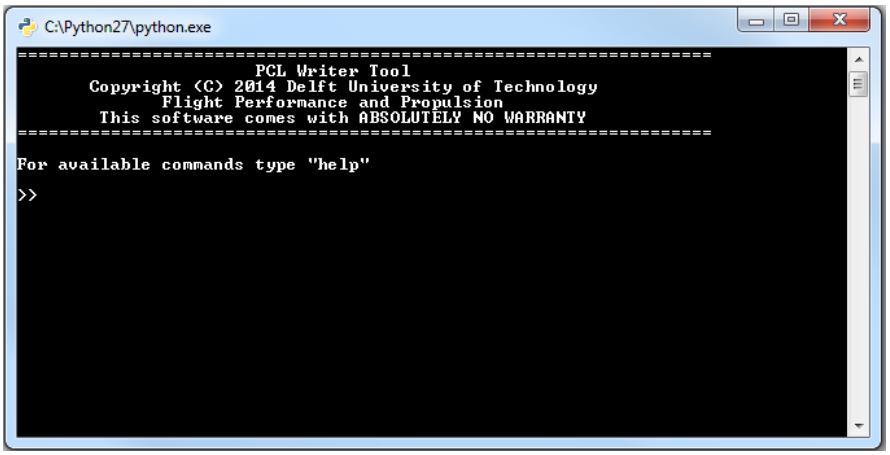


Figure 7-1: Command shell of the AFG Tool at start-up

Besides ensuring the correct program and data flow, the Controller module was also put in place to avoid the need for the user to understand all of the different programming languages and conventions used within each module. In theory, the user should only need the Controller module up until the point where the actual FEA needs to be carried out and the results need to be interpreted.

It should be noted that the Controller module is put in place as a means to facilitate the FE model generation. This means that all of the separate processing modules may also be operated individually by initializing and operating them using the software and programming language specific to it. Both methods of running the processing modules will be described in their respective sections.

7-1-1 Module Operation

To operate the Controller module, five basic commands are used which have been collected in Table 7-1 together with the function these fulfill.

Command	Function
edit	Opens the settings and directory files used by the processing modules for editing
help	Provides information on the different operational commands
location	Outputs the location of a certain folder or program to the command shell
quit	Quits the AFG tool
run	Initializes the requested processing module

Table 7-1: Commands used in the Controller module

For the "edit", "location" and "run" commands, the command itself needs to be followed up by a command option. In order to find out which command options can be used with each command, simply input "help" + the name of the command for which the options should be displayed. For example, when the different options for the "run" module should be displayed on the AFG command shell the user puts in "help run" (without the quotation marks) and the different options are provided as shown in the figure below.

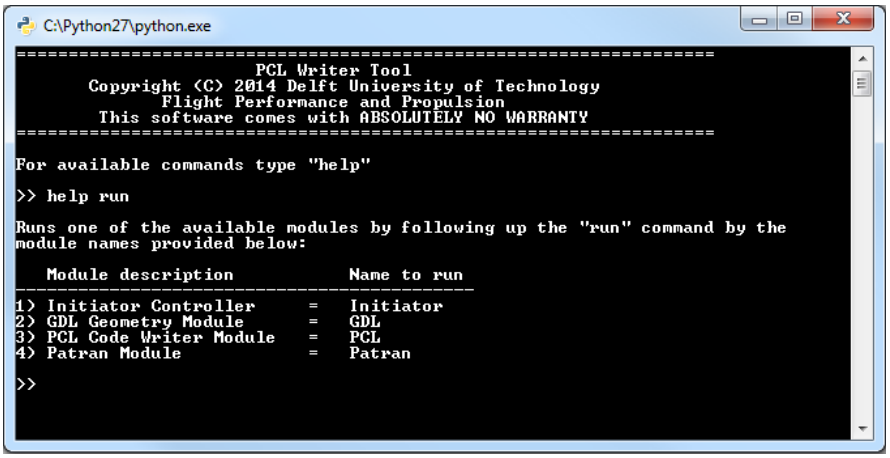


Figure 7-2: Command and option example for "run" command

7-1-2 Input/Output

As mentioned earlier, this module is operated through a text-based command-line input system. This requires the user to provide the available input commands to have the Controller module carry out a specific task. The output of the module, i.e. module information and error messages, are provided as text on the command shell screen.

Indirectly, this module also creates the output of the different modules seeing as this module initiates the processing modules. However, the output each of these processing modules creates are handled in more detail in the Input/Output subsection of these modules.

7-1-3 Module Architecture

The architecture of this module is relatively simple seeing as it's only function is to process the text-based inputs of the user and carry out the task related to it. Figure 7-3 provides a flow chart of the basic operation of the Controller module.

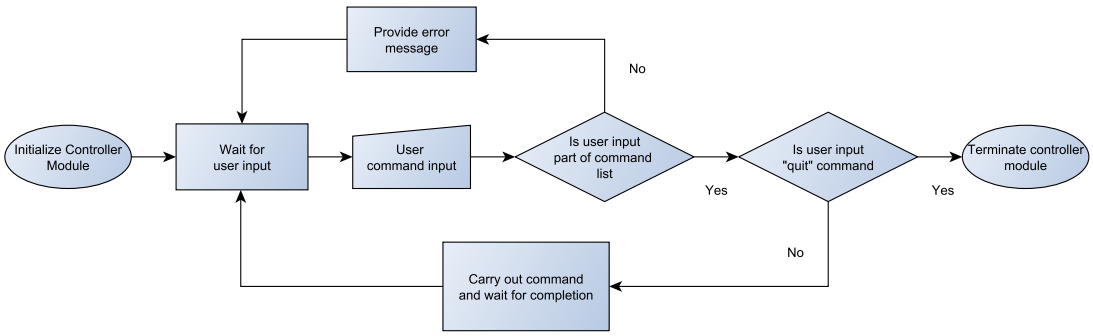


Figure 7-3: Flow chart of Controller module

7-1-4 Known Errors, Limitations and Assumptions

- When a "[Errno 13] Permission denied: ..." error occurs when trying to run the Geometry module a problem exists with the accessibility of the directory where the GenDL software is stored. The user should edit the security settings of this folder.
- One of the limitations of the module is that not all of the required error handling has been been put in place at the time of writing this manual. This means that for example not all of the checks have been put in place that verify whether every single parameter/file is present before a module runs. However, when executing a module, the error message is also displayed on the command-shell of the Controller module for Python based modules, the Matlab command window for the Initiator module and on the GenDL console window for the Geometry module.

7-2 Initiator Controller Module

As mentioned earlier, the Initiator is capable of translating a number of top-level requirements into a conceptual design of an aircraft of which the characteristics are stored in an XML file. Seeing as the results of this conceptual design are required to create the FE model the first step is to provide the SAWE method with the required input data to allow for it to run the design and analysis modules. This is achieved by executing the Initiator Controller module. With respect to the operation of the Initiator this module is capable of starting the Initiator, providing the correct input file and running the specific modules to obtain the required results.

However, part of the data required by the subsequent modules is not contained within the SAWE's XML file. Therefore, the Initiator Controller module writes a script in the programming language of the Initiator, i.e. Matlab programming language, that tells the Initiator to carry out the aforementioned tasks and in addition retrieves the required data that is only stored temporarily while the Initiator runs. This data is stored in a number of files to allow for future access by the modules that follow in the chain of operation.

7-2-1 Module Operation

The first mode of operation of this module is through the use of the Controller module. In order to use this way of operation, two steps are required. Firstly, the settings to be used within the Initiator should be set by opening the "programming_settings.py" file. This can be achieved by inputting "edit settings" (without quotation marks) into the command-line of the Controller module. This will open up the settings file and allow for the user to edit the required settings for the Initiator. These are the following:

Setting	Explanation
matlab_quit	A boolean setting to indicate whether Matlab should be quit once the Initiator analysis has been run.
module_to_run	Name of the module to be run by the Initiator. "Class25weightEstimation" is the advised module for the correct function of the AFG tool.
xml_file	Name of the XML file to be used by the Initiator and which is stored in the "data" folder of the AFG tool.

Table 7-2: Operational settings for the Initiator Controller module

As mentioned in Table 7-2 the advised module to run is the "Class25weightEstimation". The reason for this is to ensure that all of the required results are calculated by the Initiator to allow for the AFG module to run without complications. However, for analysis purposes, the Initiator Controller module can be used to run the Initiator with predefined settings.

Once the settings have been adjusted by the user, the Initiator Controller module can be run by passing "run Initiator" to the command-line of the Controller module. This will cause the Initiator Controller module to create a script written in the programming language of the Initiator and incorporates the settings provided by the user. When the script has been written the file containing it will be placed into the Initiator folder directory. Hereafter, Matlab will be initialized and instructed to run the aforementioned script. This will instruct the Initiator

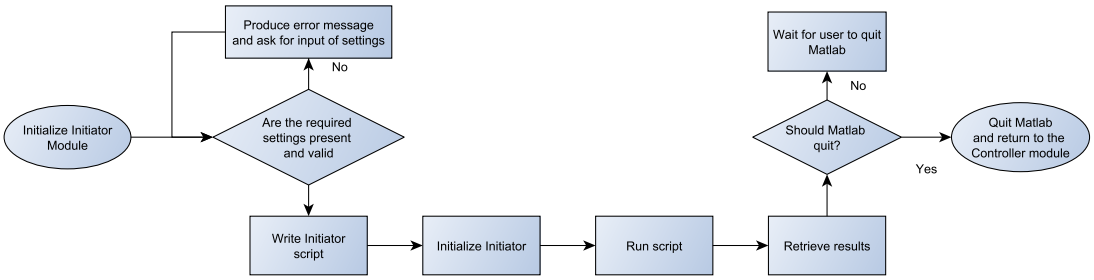


Figure 7-4: Flow chart of the Initiator Controller module

to run the requested module and store the results not available in the XML file into a number of external files. Depending on the "matlab_quit" setting, Matlab will either remain active or close once the script has been completely run.

In order to run the Initiator Controller module using the alternative method, the user should open the "Initiator_controller.py" and "program_settings.py" file into a suitable editor capable of running Python scripts. By then carrying out the tasks described above, the same results can be achieved. These files are contained within the "..\KBE_Tool\data \Patran_input\PCL_writer" folder.

7-2-2 Input/Output

The input required from the user by the Initiator Controller module are the settings mentioned above, i.e. the "matlab_quit". "module_to_run" and "xml_file" settings. By providing these the Initiator Controller module has all of the input it requires to carry out it's designated tasks.

The output that is generated as a result of the Initiator Controller module comes in the form of the XML file and the data files created by the script written by this module. The XML file will contain the data that generated and stored by the analysis modules run during the complete analysis. The second form of data output is the data contained within the files created after the analysis modules have been run and the script written by the Initiator Controller module sends the commands to the Initiator to retrieve the required data.

7-2-3 Module Architecture

Figure 7-4 provides the basic operational flow chart of the Initiator Controller module.

7-2-4 Known Errors, Limitations and Assumptions

- At the time of writing of this operational manual no errors are known and therefore no further information can be provided in this subsection.

7-3 Geometry Module

When considering the Initiator's output XML file, it may be seen that the geometrical data stored in this file is limited to the basic inputs required by the parametrization of the oval fuselage. In order to define this geometry within the FE model, two methods could be used. Firstly, the user could manually create the geometry of the oval fuselage. However, seeing as the geometry itself is a relatively complex to create for each new analysis and to increase the efficiency of the project approach, it was opted to use a KBE approach for this task and thus the fuselage geometry is recreated in a parametrized way.

The main issue however with importing the geometry (as it was defined by Schmidt and Vos [21]) directly into a FE model is that a lack of congruence between the different components' meshes would be present. For example, when meshing the outer skin and the floor, the nodes on these two components may not align. As a result of this, the FEA will not understand that a connection exists between the floor surface and the fuselage skin. Therefore, a geometrical redefinition is required in addition to the geometry reconstruction to ensure congruency between the nodes present on the different components making up the fuselage's model.

This redefinition is carried out through the use of a KBE platform. The choice for this project went to the General-purpose Declarative Language (General Descriptive Language (GenDL)) from Genworks International. GenDL is an object oriented KBE platform which uses the lisp programming language. The main reason for choosing this platform is related to the strong object-oriented Computer Aided Design (CAD) capabilities it offers, which allows for the geometrical reconstruction and redefinition of the oval fuselage concept in a parametrized way.

7-3-1 Module Operation

The operation of the Geometry module can be handled using two methods. The first is through the use of the Controller module in which the user simply has to input the "run GDL" option into the command-shell. This will initialize the Genworks GenDL console which reads in an initialization file that has been written by the Geometry module after it's initialization. In a similar manner to the Initiator Controller module's script, this script tells the GenDL console which tasks it needs to carry out and what data should be exported in order for the subsequent modules to run without any complications.

The second mode of operation differs significantly from the other modules seeing as this involves the use of the relatively more difficult programming language, i.e. GenDL, the conventions used to compile and process results and some knowledge of the architecture of the Geometry module. Therefore, it is advised to the users without any GenDL knowledge to use the Geometry Controller module.

For the operation of the Geometry module using a GenDL compatible editor, e.g. emacs with GenDL, firstly, the complete source folder needs to be loaded in through the *slime-repl gdl* buffer using the "(cl-lite "...\\KBE_Tool\\source\\") command. This will invoke GenDL to process and load-in all the required data to correctly run the Geometry module. It should be noted that the actual directory of the source folder needs to be provided to the command-line of the *slime-repl gdl* buffer in order to successfully load it in.

Once all of the data has been loaded, the user has the choice of which oval fuselage geometry should be created, i.e. the "oval fuselage" object or the "patran-geometry" object. The first geometry is a recreation of the geometry as defined in the SAWE method using a parametrization similar to the one used there. (It should be noted that a small number of adjustments were carried through to optimize the parametrization used in the Initiator of which the details are provided in the following paragraphs.) Although the second object also creates the oval fuselage geometry, the parametrization used for this object is aimed towards creating a geometry suitable for a direct import into the FEA pre-processor, i.e. Msc. PATRAN.

When the user has made the choice of which geometry should be created, two choices are once again available to the user depending on whether the data should be simply created or whether it should be created and displayed. In the latter case, the URL `http://localhost:9000/tasty` should be visited and either the "oval fuselage" object or the "patran-geometry" object should be entered (without quotation marks) into the "Class Package:Type" box. This will open up the GenDL interface and allow for the user to plot the geometry and retrieve all of the details related to it. It should be noted that in case the user would like to retrieve the geometrical properties related to the oval fuselage, e.g. floor widths, radii of the oval fuselage cross-sections, fuselage length using tasty, the "oval fuselage" object should be requested seeing as this geometry is based on the original parametrization used in the SAWE method.

In case only the data is required for inspection or in the case the subsequent modules are to be run, the GenDL editor is used. To create the "oval fuselage" object or the "patran-geometry" object the "(make-self 'oval fuselage)" or "(make-self 'patran-geometry)" command should be provided to the command-line of the *slime-repl gdl* buffer. Using these two commands and having knowledge of the Unified Modeling Language (UML) diagram of the Geometry module, which is shown in Appendix A, the user can request for all the data available through the "(the ...)" command. For example, if the user would like to retrieve the fuselage length when having compiled the "oval fuselage" object the "(the fus-geom l-fuselage)" command should be used. Care should be taken that with the previous commands only the geometry is created, which does not include the data required by the subsequent modules. This is achieved by the subsequent input of the following two commands: "(make-self 'PCL-data-writer)" and "(the PCL-data)". The first loads the object in which the data is created and the second is a variable that gives the actual task to create the data.

7-3-2 Input/Output

Seeing as the Geometry module carries out a purely geometry related task, this module can be used both with and without having run the Initiator Controller module. The reason for this is that the input of this module is limited to two ".dat" files. The first of these is the "result_check.dat" file which is used to verify whether the values of the input data are present within the Geometry data folder. When this file contains the line "Initiator_Complete" the Geometry module knows the data is available for the geometry creation as will be explained in the Module Architecture subsection. The second file is the "oval fuselage-data.dat" file which contains all of the parameters required for the generation of the oval fuselage geometry. An example of such an input is provided in Appendix chap:GeomModInput. This input file provides the data required for the creation of an oval fuselage based on the top-level requirements of an A380-800.

The output of the Geometry module that is used by the subsequent modules is discussed in the Table 7-3. For the complete directory locations, please refer to Figure 7-5.

Output	Description
PCL_STEP_data.dat	Information about the STEP folder directory and which STEP files are to be included by the FEA Code module.
STEP-bulkhead.stp	STEP file containing the data of a pressure bulkhead surface.
STEP-cargo.stp	STEP file containing the data of a cargo floor surface.
STEP-ceiling.stp	STEP file containing the data of a ceiling surface.
STEP-floor.stp	STEP file containing the data of a floor surface.
STEP-skin.stp	STEP file containing the data of a skin surface.
STEP-tail-surf.stp	STEP file containing the data of a tail surface.
STEP-wall.stp	STEP file containing the data of the wall surface.
Patran_geom_data.py	Data related to the geometry created, e.g. surface areas, frame positions, etc.
GDL_oval_section_data.py	Coordinates of the circular arcs making up the oval fuselage.

Table 7-3: Outputs of the Geometry module

7-3-3 Module Architecture

Due to the extent of the Geometry module, the flowchart of the module has been divided into a number of components to allow for some overview to be maintained. The main process, i.e. the creation of the output required by the FEA Code module, has been provided in Figure 7-5.

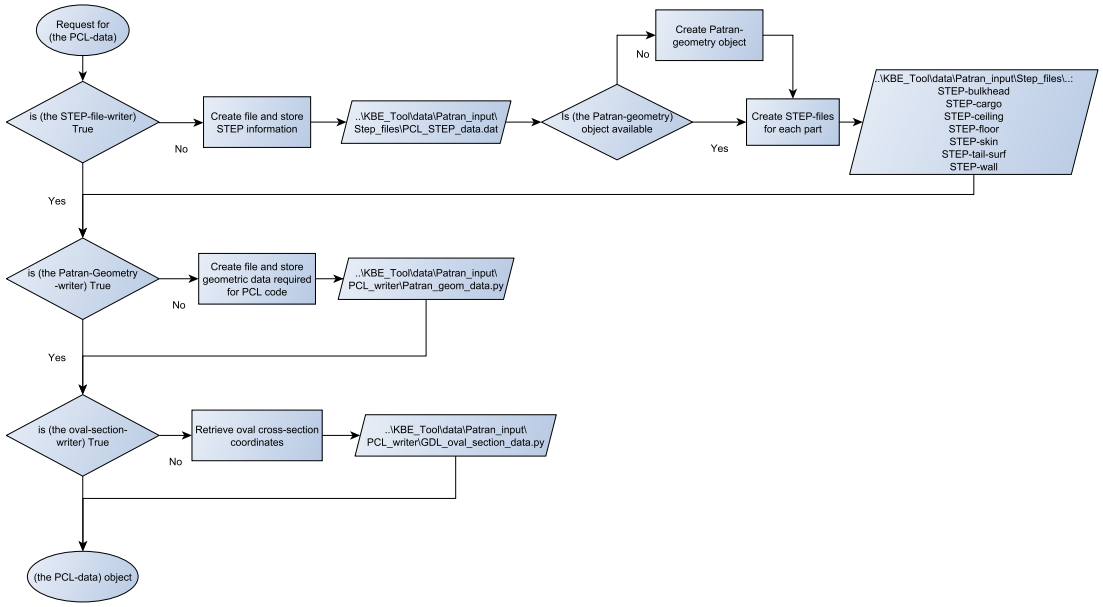


Figure 7-5: Flow chart of the Geometry module

Figure 7-6 illustrates the main steps of how the Patran-geometry object is generated within the Geometry module.

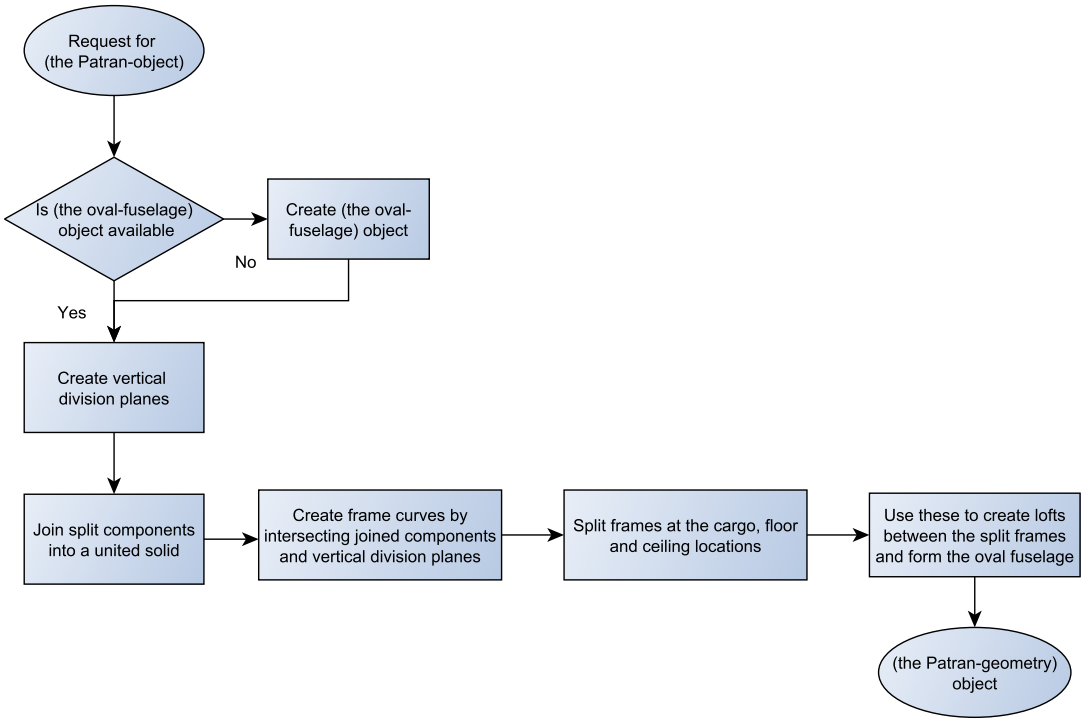


Figure 7-6: Flow chart of the creation of the Patran-geometry object

The last process of which the flowchart is provided is the creation of the oval fuselage object. This process is shown in Figure 7-7.

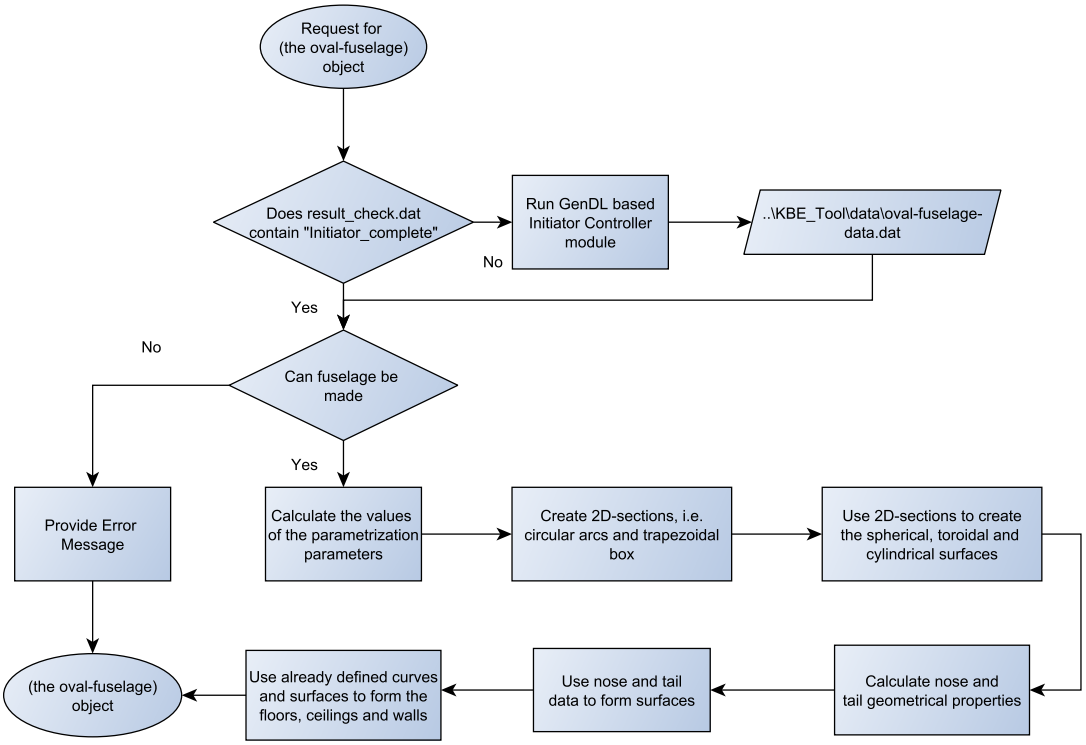


Figure 7-7: Flow chart of the creation of the oval fuselage object

In Appendix A the complete UML diagram of the oval fuselage optimizer may be found.

7-3-4 Known Errors, Limitations and Assumptions

There are a number of errors/limitations/assumptions with regard to the setup and operation of the Geometry module:

- A "Bummer, boolean operation failed" error may have been caused by a value of the approximation tolerance being too low for the "joined-skin" object within the "geometry-patran.lisp" file.
- The directories should be specified with either a single forward or a double back slash in order for GDL to correctly interpret the directory location.
- The division of "joined-skin" object into the patran-geometry sometimes fails due to a GenDL internal error. This can usually be solved by increasing the tolerances.

7-4 FEA Code Module

Based on the geometric data provided by both the Initiator Controller module and the Geometry module, the FEA Code module is capable of creating most of the code required for the automation of the FEA setup. However, due to the semi-analytical nature of the weight estimation method, a number of structural components have either only a mass or a mass and geometric shape related to them. This means that for these surfaces no thickness has been defined yet.Therefore, the role of the FEA Code Module is two-fold, i.e. firstly the calculation of the missing structural properties and secondly the creation of the FEA code. Seeing as Msc. PATRAN is used for the pre- and post-processing of the FE model, the FEA Code module writes the required code in the PCL

.

7-4-1 Module Operation

The operation of the FEA Code module can be carried out in two ways. The first way is through the Controller module by inputting the "run PCL" command into the command-line. This will initialize the "PCL_writer.py" file which in turn calls a number of separate files to assemble the complete FEA code. It should be noted that with respect to this module a number of settings may be set by the user through the "edit settings" command. These settings and their respective explanation are provided in Table 7-4. Through these settings, the user has some control over the meshing of the model and whether or not the floor struts should be included.

Setting	Explanation
floor_strut	A boolean setting to indicate whether floor struts should be created in the FE model.
perc_floor_strut_pos	Percentage expressing the position of the floor strut on the cabin floor surface.
perc_skin_strut_pos	Percentage expressing the position of the floor strut on the skin surface.
mesh_seed_floor	Number of elements per edge of the floor surface laterally.
mesh_seed_hor	Number of elements per surface edge along the longitudinally.
mesh_seed_vert	Number of elements per surface edge along the vertical direction.
skin_strut_surf	Can be set to "bot_cargo" or "cargo_floor" to indicate whether the skin surface connection should be located on the skin between the bottom and the cargo floor or between the cargo floor and cabin floor.

Table 7-4: Operational settings for the FEA Code module

The second method of operation of the FEA Code module is by running the "PCL_writer.py" file using a Python suitable editor such as Spyder. This file is contained within the "..\KBE_Tool\data \Patran_input\PCL_writer" folder.

7-4-2 Input/Output

Seeing as the FEA Code module depends solely on the data provided by the Initiator Controller module and the Geometry module, there is no need for user input. This does however imply that both the Geometry module and the Initiator Controller module need to be run before this module can be run.

The output of this module comes in the form of the PCL files created, i.e. the "PCL_file.pcl", "PCL_frame_assignment.pcl", "PCL_materials.pcl" and "PCL_pressure_fields.pcl". These files contain all of the data required to set up a complete analysis-ready FE model.

7-4-3 Module Architecture

The flow of the FEA Code module is illustrated in Figure 7-8. From this figure it may be seen that the largest part of the module is responsible for creating all of the required PCL code and a smaller part for the recalculation of the structural components to which no thickness had been assigned yet. The details of these structural calculations may be found in the next subsection.

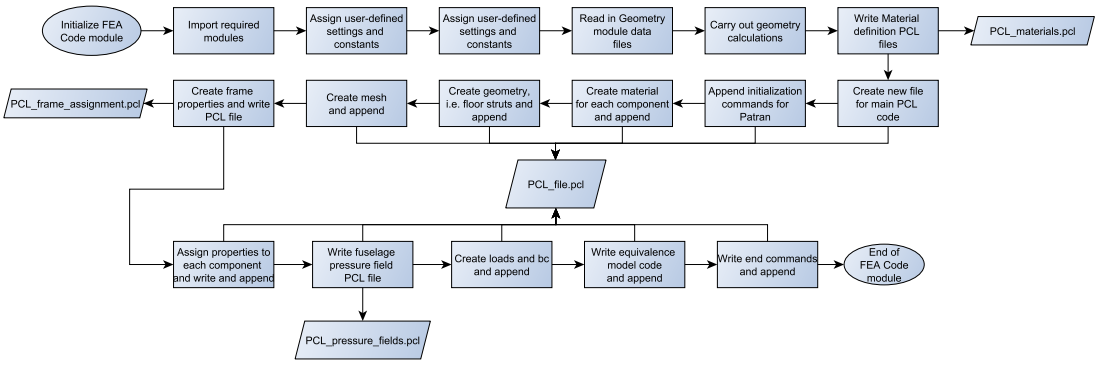


Figure 7-8: Flow chart of the FEA Code module

With respect to the writing of the PCL code, the basis of this is formed by the general functions that translate the input given to them into PCL code, which are all contained within the "PCL_writer_functions.py" file. The reason for keeping these functions rather general is related to the fact that it was attempted to keep the re-useability of these functions high so that these may also be used for future projects.

A short description of each of these functions is provided below together with the input that is required to use them. These are provided in a similar order as they are used within the FEA Code module.

header(main_pcl_file_dir)

Input

Input	Explanation
main_pcl_file_dir	Directory where the "PCL_file.py" is stored

Table 7-5: Inputs required by header()

Description
The purpose of this function is to create the header information at the start of the PCL file. This will include both the input code required by the FE model pre-processor, i.e Msc. PATRAN, as well as the date and time at which the file was created.

new_file(patran_folder_dir, file_destination_dir, file_name)

Input

Input	Explanation
patran_folder_dir	Directory where Msc. Patran may be found
file_destination_dir	Directory to indicate where the database should be created and the new file stored
file_name	Name of the FE model file

Table 7-6: Inputs required by new_file()

Description
This functions creates a new file in the "file_destination_dir", with the name "file_name" and sets the working directory of Msc. PATRAN equal to this location.

STEP_file_import(STEP_files, Patran_dir, KBE_Tool_dir)

Input

Input	Explanation
KBE_Tool_dir	Directory of the KBE_Tool folder
Patran_dir	Directory where Msc. Patran may be found
STEP_files	List containing the names of the STEP files present in the "..\KBE_Tool\data\Patran_input\STEP_files" directory.

Table 7-7: Inputs required by STEP_file_import()

Description
This function is capable of importing the geometry of the oval fuselage through the definition of the "STEP_files" list. Depending on the length of this list, this function will import the correct number of files.

coord_frame(name)

Input

Input	Explanation
name	Name of the coordinate system to be used for the analysis

Table 7-8: Inputs required by coord_frame()

Description
This functions creates a coordinate system similar to the one used by the GenDL KBE platform. This means that the y-axis is aligned with the longitudinal direction, the x-axis with the lateral direction and the z-axis with the vertical direction. This coordinate system will be used for all the subsequent definitions and analysis.

input_pcl_file(block_comment,pcl_dir)

Input

Input	Explanation
block_comment	String containing some information on which pcl file is being read in.
pcl_dir	Directory indicating the location of the PCL file being imported.

Table 7-9: Inputs required by input_pcl_file()

Description
Using this function, the user can import a PCL file into the FEA pre-processor and have it execute the contents of the file.

sandwich_materal(mat_name, face_mat, core_mat, face_thick, core_thick, skin_name)

Input

Input	Explanation
mat_name	List with PATRAN names of the materials to be created.
face_mat	List with names of the facing materials to be used for the sandwich structure.
core_mat	List with names of the core materials to be used for the sandwich structure.
face_thick	List with thicknesses of the facing to be used for the sandwich structure.
core_thick	List with thicknesses of the core to be used for the sandwich structure.
skin_name	A string indicating the name of the material for the comments within the "PCL_file.pcl".

Table 7-10: Inputs required by sandwich_materal()

Description
This function creates a number of sandwich materials based on the length of the lists provided. In order to create these sandwich structure, the materials referenced in the lists need to have been already defined within Msc. PATRAN.

mesh_seed_number(surf_list, n_seeds, edge_list, skin_name)

Input

Input	Explanation
surf_list	List containing strings in the form of "Surface #" where, # indicates the number of the surface under consideration.
n_seeds	List of lists that each contain the number of seeds for the edges of the surface under consideration.
edge_list	List of lists that each contain the number of the edge to be meshed.
skin_name	String representing the surface group to be meshed. Used for the comments in the "PCL_file.pcl".

Table 7-11: Required inputs mesh_seed_number()

Description
In order to place the mesh seed on the edge of a surface this function may be used. For this function, the number of seeds to be placed on each edge needs to be provided separately. This was done to provide the user with more control when creating the mesh on a surface.

mesh_surface(n_surfaces, global_edge_length, coord_name, surface_name)

Input

Input	Explanation
n_surfaces	A string in the form of "Surface #:#" indicating the range of surface to be meshed.
global_edge_length	The global edge length to be used for meshing the surfaces
coord_name	String representing the name of the coordinate system to be used for the element definitions.
surface_name	String representing the surface group to be meshed. Used for the comments in the "PCL_file.pcl".

Table 7-12: Inputs required by mesh_surface()

Description
This function meshes the surfaces of the FE model contained within the provided range given that the these have already been provided with a mesh seed.

mesh_curve(n_curves, global_edge_length, coord_name,curve_name)

Input

Input	Explanation
n_curves	A string in the form of "Curve #:#" indicating the range of curves to be meshed.
global_edge_length	The global edge length to be used for meshing the curves.
coord_name	String representing the name of the coordinate system to be used for the finite element defintions.
curve_name	String representing the curve group to be meshed. Used for the comments in the "PCL_file.pcl".

Table 7-13: Inputs required by mesh_curve()

Description
This function meshes the curves of the FE model contained within the provided range given that the these have already been provided with a mesh seed.

create_curve_nodes(curve_id, node1, node2, curve_name)

Input

Input	Explanation
curve_id	Curve id by which it may be identified later.
node1	Starting node of the curve.
node2	Ending node of the curve.
curve_name	String representing the curve group to be created. Used for the comments in the "PCL_file.pcl".

Table 7-14: Required inputs create_curve_nodes()

Description
In order to create the floor struts, this function was used. It allows for the creation of a curve by creating a curve in between two existing nodes, i.e. node1 and node2.

assign_homogenous_shell_prop(prop_set_name, surf_list, material_input, thickness, surf_name)

Input

Input	Explanation
prop_set_name	List of strings representing the name of the properties to be created.
surf_list	List containing strings in the form of "Surface #" where, # indicates the number of the surface under consideration.
material_input	List of materials to be assigned to each property.
thickness	List of thicknesses for each of the properties.
surf_name	String representing the name of the component to which the properties belong to. Used for the comments in the "PCL_file.pcl".

Table 7-15: Inputs required by assign_homogenous_shell_prop()

Description
Using this function, the user can assign homogenous properties to a specific surface and the corresponding elements that belong to that surface.

assign_laminate_shell_prop(prop_set_name, surf_list, material_input, thickness, surf_name)

Input

Input	Explanation
prop_set_name	List of strings representing the name of the properties to be created.
surf_list	List containing strings in the form of "Surface #" where, # indicates the number of the surface under consideration.
material_input	List of materials to be assigned to each property.
thickness	List of thicknesses for each of the properties.
surf_name	String representing the name of the component to which the properties belong to. Used for the comments in the "PCL_file.pcl".

Table 7-16: Inputs required by assign_laminate_shell_prop()

Description
Using this function, the user can assign laminate shell properties to a specific surface and the corresponding elements that belong to that surface.

create_C_beam(frame_section, width, height, thickness1, thickness2)

Input

Input	Explanation
frame_section	List containing strings that specify the name of the frame through which these may be referenced during the assignment.
width	List containing the width dimensions of the C-beam.
height	List containing the height dimensions of the C-beam.
thickness1	List containing the thickness dimensions of the flange of the C-beam.
thickness2	List containing the thickness dimensions of the web of the C-beam.

Table 7-17: Inputs required by create_C_beam()

Description
Through this function, a C-beam frame can be defined by the user that can be used later on to add the structural properties of these frames to a (or multiple) curve(s). It should be noted that this function can only be used to define C-beam frames.

assign_C_beam_prop(frame_name, frame_section, frame_mat, frame_orient, frame_offset, frame_numb)

Input

Input	Explanation
frame_name	List of strings representing the name of the frame to be created.
frame_section	List of strings representing the name of the previously defined frame properties.
frame_mat	List of strings representing the name of the materials to be used for a frame.
frame_orient	List containing lists in the form [#,#,#] which represent the orientation of each of the frames.
frame_offset	List containing lists in the form [#,#,#] which represent the offset of each of the frames.
frame_numb	List containing strings in the form "Element #:#" which indicate to which elements the properties need to be assigned.

Table 7-18: Inputs required by assign_C_beam_prop()

Description
This functions assigns the C-beam properties that were previously created to the elements linked to these curves. It should be noted that for the definition of these beams, elements are used rather than surface edges due to the fact that using the latter results in errors in Msc. PATRAN.

assign_1D_rod_prop(prop_set_name, rod_name, material_input, area, curve_id_list)

Input

Input	Explanation
prop_set_name	String representing the name of the component to which the properties belong to. Used for the comments in the "PCL_file.pcl".
rod_name	List of strings representing the name of the rod to be created.
material_input	String (or list of strings) representing the name of the material to be used for the 1D rod.
area	List of areas indicating the cross-sectional area of the 1D rods to be created.
curve_id_list	List with numbers representing the identifiers used to retrieve the curves at a later stage.

Table 7-19: Inputs required by assign_1D_rod_prop()

Description
In order to define the floor struts, this function may be used. It will create one-dimensional floor struts based on the definition of their cross-sectional area, which is circular in this case.

create_disp_bc_nodal(set_name, translations, rotations, coord_frame, surf_list)

Input

Input	Explanation
set_name	String representing the name of the boundary condition group. Used for the comments in the "PCL_file.pcl".
translations	A list in the form of [#,#,#] which represent the restrictions in translation for the longitudinal, lateral and vertical direction respectively.
rotations	A list in the form of [#,#,#] which represent the restrictions in orientation about the longitudinal, lateral and vertical direction respectively.
coord_frame	String representing the name of the coordinate system to be used.
surf_list	List of strings which represent which nodes should be included in the boundary condition group.

Table 7-20: Inputs required by create_disp_bc_nodal()

Description
With this function, the a newly defined boundary conditions can be applied to a set of or a single node(s).

create_disp_bc_elem(set_name, translations, rotations, coord_frame, surf_list)

Input

Input	Explanation
set_name	String representing the name of the boundary condition group. Used for the comments in the "PCL_file.pcl".
translations	A list in the form of [#,#,#] which represent the restrictions in translation for the longitudinal, lateral and vertical direction respectively.
rotations	A list in the form of [#,#,#] which represent the restrictions in orientation about the longitudinal, lateral and vertical axis respectively.
coord_frame	String representing the name of the coordinate system to be used.
surf_list	List of strings which represent which elements should be included in the boundary condition group.

Table 7-21: Inputs required by create_disp_bc_elem()

Description
With this function, the a newly defined boundary conditions can be applied to a set of or a single element(s).

create_inertial_load(set_name, coord_name, trans_acc, rot_vel, rot_acc)

Input

Input	Explanation
set_name	String representing the name of the load group. Used for the comments in the "PCL_file.pcl".
coord_name	String representing the name of the coordinate system to be used.
trans_acc	A list in the form of [#,#,#] which represent the translational acceleration in the longitudinal, lateral and vertical direction respectively.
rot_vel	A list in the form of [#,#,#] which represent the rotational velocity in the longitudinal, lateral and vertical direction respectively.
rot_acc	A list in the form of [#,#,#] which represent the rotational acceleration about the longitudinal, lateral and vertical axis respectively.

Table 7-22: Inputs required by create_inertial_load()

Description
The gravitational effect used in the FEA was modelled in the FE model using this function. It allows for the definition of an inertial load based on the translational acceleration and the rotational acceleration and velocity.

create_2D_press(set_name, surf_list, pressure, name)

Input

Input	Explanation
set_name	String representing the name of the load group which is used to reference it later.
surf_list	List of strings which represent the surface on which surface this pressure load act.
pressure	The magnitude of the two-dimensional pressure field.
name	String representing the name of the load group. Used for the comments in the "PCL_file.pcl".

Table 7-23: Inputs required by create_2D_press()

Description
Using this function, the user can define a two-dimensional pressure field that acts normal to the surfaces that have been provided to the function.

create_press_field_tabular(field_name, coord_name,n_ind_var, ind_var, X, Y, Press, name)

Input

Input	Explanation
field_name	String representing the name of the pressure field to be created, which is later used for referencing purposes.
coord_name	String representing the name of the coordinate system to be used.
n_ind_var	Number of independent variables used in the definition of the pressure field.
ind_var	The independent variables in the form of a list, e.g. ["X", "Y", ""].
X	List of the values of the first independent variable.
Y	List of the values of the second independent variable.
Press	List of lists containing the pressure values at the locations of X and Y.
name	String representing the name of the load group. Used for the comments in the "PCL_file.pcl".

Table 7-24: Inputs required by create_press_field_tabular()

Description
What this function does is provide the means to define an interpolation array that allows Msc. PATRAN to determine the values of the pressure at specific node/element locations. This is used in this project to determine the pressure forces on the oval fuselage skin as a result of its aerodynamic lifting characteristics.

create_total_load(set_name, surf, coord, surf_load, edge_load, name)

Input

Input	Explanation
set_name	String representing the name of the pressure field to be created, which is later used for referencing purposes.
surf_list	List of strings which represent the surfaces/curves on which the total load acts.
coord_name	String representing the name of the coordinate system to be used.
surf_load	The magnitude of the total load that is to be divided over the surface.
edge_load	The magnitude of the total load that is to be divided over the edge.
name	String representing the name of the load group. Used for the comments in the "PCL_file.pcl".

Table 7-25: Inputs required by create_total_load()

Description
The idea behind a total load is to create a load of a certain magnitude and equally divide it over an area in the case of surfaces or length in the case of edges.

create_point_load(set_name, force ,moment, coord_name, surf_list, name)

Input

Input	Explanation
set_name	String representing the name of the pressure field to be created, which is later used for referencing purposes.
force	A list in the form of [#,#,#] which represent the magnitude of the force that acts in the longitudinal, lateral and vertical direction respectively.
moment	A list in the form of [#,#,#] which represent the magnitude of the moment that acts about the longitudinal, lateral and vertical axis respectively.
coord_name	String representing the name of the coordinate system to be used.
surf_list	List of strings which represent the surfaces/curves on which the total load acts.
name	String representing the name of the load group. Used for the comments in the "PCL_file.pcl".

Table 7-26: Inputs required by create_point_load()

Description
For the creation of a point-load, i.e. a load that acts on a single surface corner or node, this

function may be used by specifying the magnitude of the force and moment that act on that surface corner or point.

create_load_case(name, description, loads, load_factor)

Input

Input	Explanation
name	A string representing the name of the load case that is to be created.
description	A string providing a description of the the load case that is to be created.
loads	A list containing strings that identify the loads and boundary conditions that should be included into this load case.
load_factor	A list containing the magnitudes of the load/scaling factor to be used for each load or boundary condition (BC) present in the load case.

Table 7-27: Inputs required by create_load_case()

Description
By providing this function with the loads and their respective loading factor, the user can have Msc. PATRAN automatically generate the load cases that should be used during the analysis.

equivalence(node_exclusion, tolerance)

Input

Input	Explanation
node_exclusion	A string representing the nodes to be excluded from the equivalencing process.
tolerance	The magnitude of the tolerance to be used during the equivalencing process.

Table 7-28: Inputs required by equivalence()

Description
The reason for equivalencing the FE model is to ensure that the FE solver understands that when two or more nodes are in close proximity of each other, a connection exists between these nodes. This allows for the establishment of for example the connection of the floor surface to the outer skin. Whether two or more nodes are in close proximity of each other

depends on the magnitude set for the tolerance. The higher it is set, the larger the distance between the nodes can be for them to be included in the equivalencing. In case a number of nodes can be excluded from the process, these can be specified using the "node_exclusion" string.par

footer()

Input
No input is required for the correct operation of this function.
Description
The footer is used to carry out a number of commands that have to do with "cleaning up" the FE model and ensuring a clear overview of the model is provided once the complete FE model has been generated.

7-4-4 Known Errors, Limitations and Assumptions

- One of the limitations of the current version of the FEA Code module is related to the numbering system used within Msc. PATRAN, more specifically, the use of nodes to assign properties, boundary conditions or loads in a more accurate manner. However, due to the combination of the numbering method of the nodes within Msc. PATRAN and the use of multiple STEP-files, the traceability of nodes was too complex to be included within this thesis project. Currently, the assignment of the properties, boundary conditions and loads is dependent on the frame-spacing and positions of the floors, which indirectly define the locations of the nodes seeing as these serve as the indicators of where the divisions of the surfaces should be made.
- The locations of the C-frames in the FE model do not represent the real-life situation a 100% accurate. The reason for this is that these were not give an offset to move the flange to be in contact with the skin. Instead, the center of the C-frames' web is located on the skin.
- Due to the relatively large variations in the dimensions of the components that make up the different oval fuselage, not all of the quadrilateral surface ratios are near 1.0, i.e. a perfect square. The use of these elements might have a small influence on the accuracy of the FEA results. However, this has not been investigated so far.
- The data calculated by the FEA Code module is only contained within the module itself and not exported into for example the Initiator's XML file which limits the ability to transfer the results to another module.

7-5 FEA Module

With the data and code files for the FEA software generated, the FEA module is started. This boots the FEA software and allows for the input of a single line of code which reads in the input file containing the PCL code. Hereafter a sequence of actions is started which creates the complete FE model.

- Setup and meshing of the geometric model
- Creation and assignment of the material and structural properties to the geometry
- Definition and application of the loads and boundary conditions (BC)
- Preparation of the analysis itself

Once these tasks have been completed the model is analyzed using Msc. Nastran and the results are extracted.

7-5-1 Module Operation

The two ways of operating the FEA module are very similar in the sense that by providing the "run Patran" command to the command shell of the Controller module only the FEA pre- and post-processor will be booted. The actual input of the command which tells the software to generate the FE model needs to be carried out manually by the user.The user could also simply boot the software manually and input the code in that way. It can therefore be said that the implementation of the "run Patran" command is purely for a sense of completeness within the AFG tool.

7-5-2 Input/Output

The only input required for the correct functioning of this module when using the Controller module to operate it is the directory specification of the Msc. PATRAN software. Once the software has been booted, the input required is the command which instructs MSc. PATRAN to read in the PCL file and generate the FE model. This line of code takes the form of:

!! input "C:/.../KBE_Tool/data/Patran_input/PCL_files/PCL_file.PCL"

This code may be retrieved in the header of the PCL file by inserting the "edit PCL" command into the command shell of the Controller module before initializing the FEA software.

7-5-3 Module Architecture

Due to the simplicity of this module, there is no need to discuss the module architecture in greater detail than done in the paragraphs above.

7-5-4 Known Errors, Limitations and Assumptions

- At the time of writing of this operational manual no errors are known and therefore no further information can be provided in this subsection.

Appendix A

Geometry Module UML

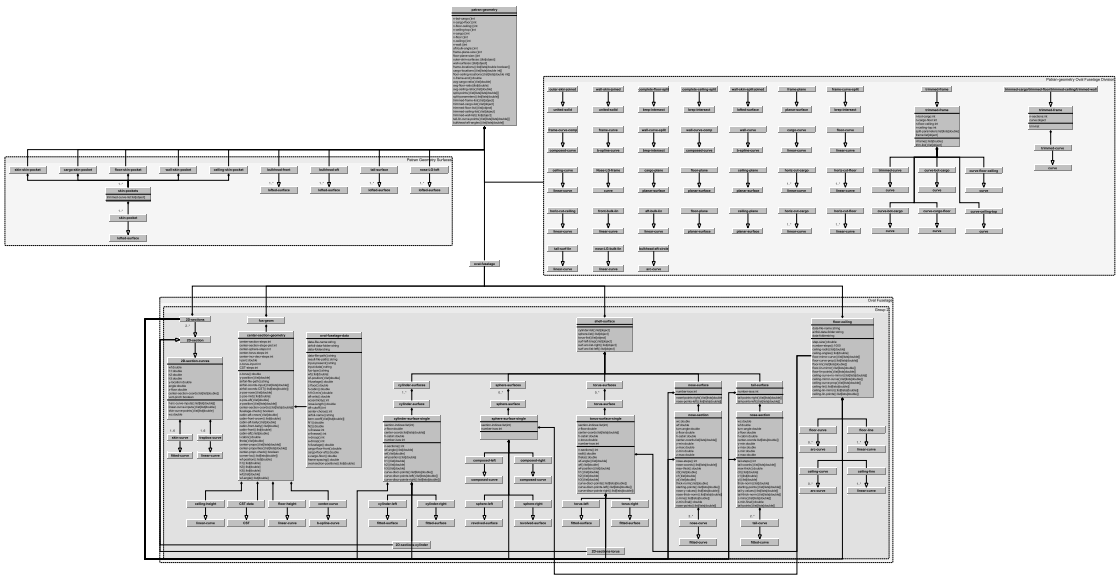


Figure A-1: UML diagram of the Patran Geometry

Bibliography

[1] "Airlines to Welcome 3.6 Billion Passengers in 2016," http://www.iata.org/pressroom/pr/pages/2012-12-06-01.aspx, December 2012, Institution: International Air Transport Association, Accessed: October 28, 2013.

[2] "Global Air Transport Outlook to 2030 and Trends to 2040 (ICAO Cir 333)," http://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=36&ved=0CEkQFjAFOB4&url=http%3A%2F%2Fwww.techstreet.com%2Fproducts%2Fpreview%2F1857404&ei=Mpl2UuLtBeeU0AWZ4YDAAg&usg=AFQjCNFJ6Z1qcalzYaMShmHDMaUE6uy9CQ&bvm=bv.55819444,d.d2k, 2013, Institution: International Civil Aviation Organization, Accessed: October 30, 2013.

[3] "Fact Sheet: Climate Change," http://www.iata.org/pressroom/facts_figures/fact_sheets/pages/environment.aspx, June 2013, Institution: International Air Transport Association, Accessed: October 28, 2013.

[4] Liebeck, R. H., "Design of the Blended Wing Body Subsonic Transport," *Journal of Aircraft*, Vol. 41, No. 1, Januray-February 2004, pp. 10–25.

[5] Pertuze, J. A., Sato, S., Spankovszky, Z., and Tan, C., "N + 3 Aircraft Concept Designs and Trade Studies, Final Report Volume 1," Report, NASA, December 2010.

[6] Mukhopadhyay, V., "Analysis, Design, and Optimization of Non-cylindrical Fuselage for Blended-Wing-Body Vehicle," *Journal of Aircraft*, Vol. 41, No. 4, July-August 2004, pp. 925–930.

[7] Geuskens, F. J. M. M., "Analysis of Conformable Pressure Vessels: Introducing the Mulitbubble," *AIAA Journal*, Vol. 49, No. 8, August 2011, pp. 1683–1692.

[8] Schmidt, K. and Vos, R., "A Semi-Analytical Weight Estimation Method for Oval Fuselages in Conventional and Novel Aircraft," *52nd Aerospace Sciences Meeting*, AIAA SciTech, American Institute of Aeronautics and Astronautics, Washington, DC, January 2014.

[9] Denisov, V., Bolsunovsky, A., Buzoverya, N., and Gurevich, B., "Recent Investigations of the very large passenger Blended-Wing-Body Aircraft," 21st ICAS Congress, April 1998.

[10] Ruijgrok, G., *Elements of Airplane Performance*, Delft University Press, 2004, ISBN 90-6275-608-5.

[11] Anderson, J. D., *Introduction to Flight - Fifth Edition*, McGraw-Hill, 2005, ISBN 007-123818-2.

[12] Qin, N., Vavalle, A., LeMoigne, A., Laban, M., Hackett, K., and Weinerfelt, P., "Aerodynamic Considerations of Blended Wing Body Aircraft," *Progress in Aerospace Sciences*, Vol. 40, 2004, pp. 321–343.

[13] Qin, N., Vavalle, A., and Moigne, A. L., "Spanwise Lift Distribution for Blended Wing Body Aircraft," *Journal of Aircraft*, Vol. 42, No. 2, March-April 2005, pp. 356–365.

[14] Bansal, R., *A Textbook of Strength of Materials Fourth Edition*, Laxmi, 2010, ISBN 81-7008-147-5.

[15] Harvey, J. F., *Theory And Design Of Pressure Vessels*, Springer, September 1991, ISBN 0412986515.

[16] Mukhopadhyay, V., "Structural Concepts Study of Non-circular Fuselage Configurations," SAE/AIAA World Aviation Congress, 22-24 October 1996.

[17] Mukhopadhyay, V., Sobieszczanski-Sobieski, J., Kosaka, I., Quinn, G., and Vanderplaats, G. N., "Analysis, Design, and Optimization of Noncylindrical Fuselage for Blended-Wing-Body Vehicle," *Journal of Aircraft*, Vol. 41, No. 4, July-August 2004, pp. 925–930.

[18] Mukhopadhyay, V., "Blended-Wing-Body (BWB) Fuselage Structural Design for Weight Reduction," 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 18-21 April 2005.

[19] Geuskens, F. J. J. M. M., Bergsma, O. K., Koussios, S., and Beukers, A., "Pressure Vessels & Pressure Cabins for Blended Wing Bodies," 17th International Conference on Composite Materials, 27-31 July 2009.

[20] ASC conference, *A New Structural Design Concept for Blended Wing Body Cabins*, April 2012.

[21] Schmidt, K., *A Semi-Analytical Weight Estimation Method for Oval Fuselages in Novel Aircraft Configurations*, Master's thesis, Technical University Delt, October 2013.

[22] Elmendorp, R., *Synthesis of Novel Aircraft Concepts for Future Air Travel*, Master's thesis.

[23] Young, W. and Budynas, R., *Roark's Formulas for Stress and Strain*, McGraw-Hill, seventh-edition ed., 2002.

[24] Megson, T., *Aircraft Structures for Engineering Students*, Elsevier, fourth-edition ed., 2007.

[25] “Selection Guide. Self-supporting double-skin metal-faced insulating panels. Specifications - Factory-made products.” Tech. rep., AAAMSA, March 2013.

[26] RJM Elmendorp, R Vos, G. L. R., “A Conceptual Design and Analysis Method for Convetnional and Unconventional Airplanes,” *29th Congress of the International Council of the Aeronautical Sciences*, ICAS, International Council of the Aeronautical Sciences, St. Petersburg, Russia, 2014.