IMPERIAL

Aerospace Vehicle Design

Preliminary Design Report

Academics responsible:	Dr. Demetrios V and Dr. Laura Mainini
Department:	Department of Aeronautics
Course:	MEng in Aeronautics
Module:	AERO60002 Aerospace Vehicle Design
Academic year:	2023/2024
Students:	Jude Mbroh - 01844648
	Henil Shah - 02024885
	Anthony Akinola - 01878401
	Himanshu Tyagi - 02072009
	Darshil Gajjar - 01866770
	Het Talati - 02054672
Group:	16
Date:	18/03/2024

This report details the comprehensive structural analysis and optimisation process for a 20 passenger business jet. The flight envelope was plotted using the constraining load cases of the aircraft to determine the ultimate loads acting on the aircraft. With accordance to these loads, the fuselage skin thickness was designed to be 0.001m with 87 Z-stringers of 0.021m x 0.0397m x 0.0026m. Similarly for the wing, 21 Z-stringers of 0.00372m x 0.00745m x 0.001m with 16 ribs. The structural optimisation process for the empennage followed closely to that of the wing. Ansys software was leveraged to perform the detailed component design of a flap extension, as well as finite element modeling and analysis techniques to predict structural behavior under various loading conditions, ensuring compliance with safety standards and performance criteria. The final design has a mass of 256 g and is expected to experience a maximum stress of 251 MPa at the limit load. With a structural failure predicted at 6.5 kN, a merit index of $\approx \eta = 25,400$ was expected to be achieved.

Department of Aeronautics
South Kensington Campus
Imperial College London
London SW7 2AZ
U.K.

Contents

List of Figures

List of Tables

Introduction

Following the conceptual design stage, this report details the preliminary design stage for a 20-passenger business jet. The jet features a low-mounted wing design with aft-mounted turbofan engines and a T-tail configuration. Fowler flaps are used for high-lift devices. This report outlines the overall structural layout of primary components which are the fuselage, wings and empennage for three critical load cases. This is carried out in stages including the optimal sizing of elements, such as spars, stringers and ribs, and the material selection for different components of the aircraft. Furthermore, this report also details the detailed design of a flap extension mechanism which was carried out using Topology Optimisation and Finite Element Analysis.

Literature Review

2.1 Fuselage

The central structure of an aircraft is known as the fuselage. It serves two main purposes - to house the payload and equipment of the aircraft and act as the structural connection point for the wings, engines, and tail. In fulfilling its purpose, the fuselage experiences loading in the form of bending, shear, torsion, and an outward stress due to cabin pressurisation. To effectively withstand these various loads, the fuselage structure is designed in one of four main types – truss, geodetic, monocoque, and semi-monocoque [2]. The truss structure utilises a frame of beams welded into an array of triangles, whereas the geodetic structure uses a frame wound spirally in the form of a basket-weave [2]. In a monocoque structure, the outer skin of the fuselage acts as the main frame of the structure, bearing the main part of the loads. The skin, however, is unable to bear compression loads effectively without being overly thick, making this design less favourable due to weight concerns [3]. The semi-monocoque structure utilises a subframe, with longerons to carry longitudinal stresses and stringers to stabilise the skin. The skin is attached to this subframe to reduce loading on it, reducing skin thickness in the process [3]. The subframe is also made up of light frames and heavy frames. Light frames exist to ensure the shape of the fuselage is maintained, and heavy frames are used as reinforcement for connection points to other aircraft structures [4]. Due to its construction, the semi-monocoque structure benefits from having increased structural integrity while being lightweight, leading to better fuel efficiency as well. For these reasons, the aircraft will be employing the semi-monocoque design for the fuselage.

The aircraft skin is the component attached to the fuselage frame, acting as a streamlined protective outer surface, and as a load-carrying component. It can be manufactured from either aluminium alloys or composites, depending on the aircraft, and is usually 3.175 mm thick. Aluminium alloys are the prevailing material used for fuselage construction as they are inexpensive and lightweight. However, there has been a shift towards composites due to their superior strength while weighing less than the alloys, and without the risk of corrosion or fatigue that plagues the alloys. Other than cost and weight, other factors such as fatigue and maintenance requirements also come into account when selecting the material [5]. The material for the skin will be selected in a later section.

The fuselage frame is the structural component of the fuselage that aids in the load-bearing capabilities of the skin and maintains the cross-sectional shape of the fuselage. Light frames, consisting of lightweight cross-sectional rings positioned at closely spaced intervals, serve the dual purpose of preserving the cross-sectional form and effectively dividing the shear-carrying skin into efficient panel sizes. These are designed to ensure that only local buckling of the skin occurs, and not global buckling or general instability. Heavy frames are heavy cross-section rings that are found at points of high concentrated load inputs. They serve as the main bearer of loads at these points of concentrated loads [4]. Furthermore, in a pressurised fuselage, there exists a partition between the pressurised and depressurised parts called

the bulkhead. These are found at the front and rear end of the cabin [3].

Stringers, integral to the structural integrity of the aircraft, assume the role of slender axial members strategically positioned to counteract significant deflections induced by aerodynamic forces and cabin pressurisation. Their primary function involves the efficient transfer of loads and stresses from the skin to the more robust frames, ensuring a well-distributed burden across various components. Widely embraced in aircraft design, the extruded Z and J section stringers emerge as popular choices [6].
The Z-section stringers stand out for their remarkable structural efficiency and ease of assembly, making them a preferred selection. Concurrently, the J-section stringers exhibit a fail-safe design, thanks to the incorporation of double-row fasteners linking them securely to the skin. This design choice ensures redundancy and enhances overall safety measures. Due to its ease of assembly and structural efficiency, however, the z-section stringer is the choice in the design

2.2 Wings

Wings are the main lifting surfaces of the aircraft and encounter aerodynamic and inertial loads. The primary structural components of aircraft wings, include spars, ribs, and stringers which help maintain the structural integrity and aerodynamic performance of the aircraft.
Spars are the primary load-bearing elements of an aircraft wing, running spanwise from the wing root to the tip. They support the wing by carrying the bending loads experienced during flight and landing. Typically, modern commercial FAR-25 compliant aircraft use a two-spar configuration to distribute loads efficiently. These are usually placed towards either end of the aerofoil section, and are referred to as the front and rear spar. There are multiple different cross-section designs for spars with the most common being an I-section. This is comprised of two sections, the spar cap and spar web.
The spar cap helps the attachment of the spar to the skin panels and also help counteract bending loads. The spar web is used to optimise the load-carrying capacity whilst minimising weight. The shape of the spar web is also something that has been researched and optimised. The most common configurations are a truss spar, sine wave spar and fail-safe spar. A fail-safe configuration splices two separate spar sections together with rivets to ensure that if either section breaks, the other can still carry the load. [7]

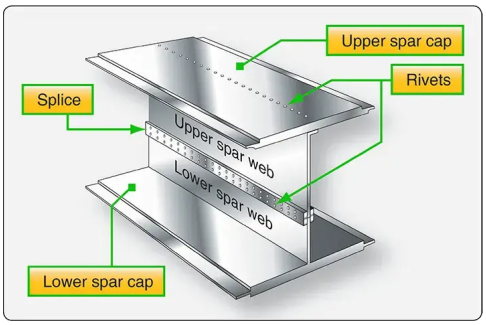


Figure 2.1: Spar web and Spar cap

Ribs are the structural components running perpendicular to the spars and stringers. They help maintain the aerodynamic shape of the wing by supporting the wing skin and transmitting transverse loads to the spars. They also increase the critical buckling load for the skin-panels by reducing the length of the section. Various different types of ribs are used based on load requirements and structural considerations, including form, plate-type, truss, and forged ribs[8]. Rib spacing depends on the load distribution on the wing and external factors such as the location of fuel tanks. The rib spacing tends to be smaller towards the root and larger towards the tip due to the higher concentration of load at the root.
Stringers are longitudinal members that run parallel to the spars and provide additional structural support to the wing skin. Together with the skin, they form skin-stringer panels which increase the overall stiffness and strength of the wing. There are many different stringer shapes with different advantages. Z and J shaped stringers are commonly used in commercial aircraft due to their structural efficiency and high buckling load capacity[9].

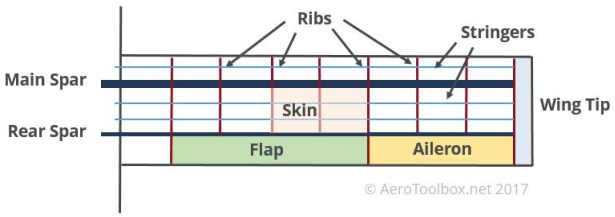


Figure 2.2: Simplified Wing Structure Example

The wing box structure is formed by the integration of spars, ribs, stringers, and wing skin. It acts as a torsion-resistant box, providing structural rigidity and support to the wing. Typically, the wing box configuration incorporates the two spars, forming a closed tube with the wing skin to distribute torsional and shear loads efficiently.

2.3 Empennage

The empennage, consisting of a horizontal and vertical stabiliser in a T-Tail configuration, has a similar composition to the wings but is designed to withstand lighter loads. The vertical stabiliser's structure consists of ribs and stringers in order to support its compressive loads due to housing the horizontal stabiliser above it. Similar to the wing, the most commonly used stringers are Z and J shaped and the same stringer shape as the wing is usually chosen. As per the brief, the horizontal stabiliser is designed using composite materials due to their high yield strength and low mass. The horizontal stabiliser was considered a better option due to the fact that, relative to the vertical stabiliser, the horizontal stabiliser can be designed to withstand more bending loads compared to shear loads, where better performance can be achieved through the usage of composites. There is also a 2-spar configuration for the horizontal stabiliser for additional structural support.

Fuselage Design

The fuselage is designed to be able to withstand the inertial loads, the thrust force from the engines and the reaction forces of the horizontal tail, vertical tail and the wing. These forces impose direct and shear stresses as well as bending moments.

3.1 Material Selection

Before proceeding further in the design process of the fuselage, the materials for the different components have to be selected. The material selection process will be conducted using ANSYS Granta EduPack, with its wide selection of aerospace grade materials. The components of the fuselage for which the material selection will be conducted are the skin, stringers, light frames and heavy frames. The six key material parameters taken into account are yield strength, compressional strength, tensile strength, stiffness, torsional strength and fatigue resistance. Appropriate material merit indexes will be used based on the loading experienced by the component, and the CES graphs on EduPack will help to narrow the material.

3.1.1 Skin

To begin with, the material selection was narrowed down to aluminium alloys and composites, as discussed in the literature review. This is because these materials are common in industry. In the semi-monocoque design, the skin undertakes the shear loading of the fuselage. It also undergoes some torsion during the OEI case. Lastly, fatigue needs to be considered for the skin due to the constantly varying pressurisation loads. As such, the material merit index selected were the ones for compression, tension, torsion and

fatigue, as seen below.

$$M_{compression} = \frac{\sigma_c^{\frac{2}{3}}}{\rho c} M_{tension} = \frac{\sigma_t^{\frac{2}{3}}}{\rho c} M_{torsion} = \frac{G}{\rho c} M_{fatigue} = \frac{K_{IC}}{\rho c} \tag{3.1}$$

This was plotted against density and price in the CES plots. To maximise for the y-axis parameters but minimise the density and price, a line with gradient of 1 was plotted, and the first material that intersected with the line was selected, in this case being Al 2024-T861. This was the material selected for the fuselage skin, which is the same as the A380 and Boeing 777 [10].

3.1.2 Stringer

As mentioned above, the stringers exist to provide support to the skin, helping it to resist bending and buckling, while providing stiffness. Furthermore, it is imperative that the stringers themselves do not buckle during service, and therefore is also taken into account. Thus, the material merit indexes for stiffness, yield strength and fatigue were selected, with the merit indexes for stiffness and yield shown below.

$$M_{stiffness} = \frac{\sigma_y^{\frac{2}{3}}}{\rho c} M_{yield} = \frac{E^{\frac{1}{3}}}{\rho c} \tag{3.2}$$

Repeating the same process as with the skin yielded the Al 7055-T77511, the final material for the stringers.

3.1.3 Light and Heavy Frames

The fuselage light frames play a crucial role in the structural integrity of the fuselage, ensuring there is not global panel buckling or instability. It does so by allowing the onset of local panel buckling instead, albeit at high loads only. As such, the material indexes for compression, tension, torsion and fatigue were selected, yielding in the same material as selected for the skin – Al 2024-T861.
Heavy frames are used as reinforcements at locations where the wings and empennage connect to the fuselage, which is where the loading from these components is transferred. As such, they undergo higher peak stresses in its loading, therefore requiring a higher fatigue resistance. As such, apart from compression, tension and yield, the stiffness and fatigue resistance material indexes were chosen for the heavy frames. This yielded a material of Al 7068 -T6511 for them.

3.1.4 Summary of Materials

Presented below in Table 1 is a summary of the material properties of the fuselage components

Material Property	Al 2024 T861	Al 7055 T77511	Al 7068 T6511
Yield Strength (MPa)	431	595	702
Tensile Strength (MPa)	469	635	740
Compressive strength (MPa)	442	605	710
Shear Strength (MPa)	310	330	365
Young's Modulus (GPa)	74	69	73
Fatigue Strength at 10^7 cycles (MPa)	136	340	229
Fracture Toughness	39	39	33

Table 3.1: Table of summary of material properties

3.2 Fuselage Loading

During this design process, the fuselage is modelled as a cantilever beam that is simply-supported at the front and rear spars of the wing. This is a useful approach as the undercarriage is attached to the

wings and will transfer their inertial and dynamic loads through the spars to the fuselage. During steady-level-flight the inertial loads are balanced by the spars of the wing and the trim load from the horizontal stabiliser. The nose and the tail of the fuselage have been modelled as linearly tapering.

Component	Mass (kg)	Range of Positions (m)
Avionics	954	0.50
Flight Control System	906	1.48
Nose Landing Gear	129	3.24
Crew	395	3.54
Electrical Systems	581	0 - 27.08
Fuselage Nose	624	0 - 5.50
Main Fuselage	3635	5.50 - 22
Fuselage Tail	541	22 - 27.08
Fuel	6329	5.94 - 16.85
Fuel System	627	16.85
Furnishing	2365	3.54 - 23.5
Passengers	1700	10.47 - 20.14
Air Con	937	13.54
Main Landing Gear	498	16.72
Nacelle and Engines	4754	22.14
Cargo	600	22.84
APU	383	25.31
Vertical Tail	544	23.50 - 26.60
Horizontal Tail	397	27.08

Table 3.2: Fuselage Component Layout

The fuselage has been discretised along its length to allocate point and distributed loads as seen in table 3.2. These loads are then counteracted by equal and opposite aerodynamic loads to ensure zero shear force and bending moment at the free ends of the fuselage. These aerodynamic loads are calculated using equations 3.3 and 3.4.

$$\Sigma mg = R_F + R_R + R_T \tag{3.3}$$

$$\Sigma mxg = x_F R_F + x_R R_R + x_T R_T \tag{3.4}$$

Since there are more unknowns than equations, this arrangement was valid for a range of trim load values. In addition, these variables are also dependent on the load factor experienced by the aircraft. The calculated reaction forces and their trim loads according to the various load cases can be seen in table 3.3.

Load Case	$\mathbf{R}_F$ (N)	$\mathbf{R}_R$ (N)	$\mathbf{R}_T$ (N)
3.75g	428,602.3	346,937.9	-19,548.7
Landing	342,881.8	277,550.3	-15,941.2
OEI	114,293.9	92,516.8	-8,173.1
-1.5g	-171,440.9	-138,775.2	10,134.8

Table 3.3: Wing spar reaction forces and their trim load

3.2.1 Load Case 1: Symmetric flight at the ultimate load factor

Following FAR-25 regulations, symmetric flight at ultimate load factor was considered for the first load case. Referring to the flight envelope in figure 2.1, the maximum and minimum limit loads were found to be 2.5 and -1 respectively. A safety factor of 1.5 was added to the limit load to reach the ultimate load factor of 3.75g and -1.5g. This was done to ensure that the forces experienced on an aircraft would not exceed the ultimate load factor during its life cycle, preventing total failure of the fuselage.

3.2.2 Load Case 2: OEI condition

The second load case was the One Engine Inoperative case (OEI). For this asymmetric loading case, the load factor was assumed to be 1g. Additionally, due to the asymmetric thrust from the engine, there is a yaw motion experienced by the aircraft, which has to be corrected by a rudder deflection on the vertical tailplane. This rudder deflection causes a force on the horizontal tailplane, which is transmitted as a torque on the fuselage.

Furthermore, the deflection causes a moment imbalance on the aircraft, causing the aircraft to roll, as seen in figure 3.1. To prevent this rolling motion, equal but opposite aileron deflection are required on each wing, generating a lift imbalance. The lift imbalance also generates a torque on the fuselage, equal but opposite in value to that of the tailplane. As such, the overall torque in the fuselage is twice that of the torque generated either by the wing or the vertical tailplane.

To find the torque generated, the maximum thrust of one engine was used, which was 61,100 N. Using equilibrium, the force caused by the rudder deflection was found to be 12,140 N, acting through the middle of the rudder. From this force, the torque on the fuselage, which acts between the wing and vertical tailplane, was found to be 77,832 N.

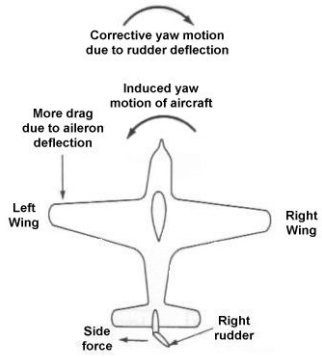


Figure 3.1: Forces Experienced due to OEI [1]

3.2.3 Load Case 3: Landing with main undercarriage only

The third load case is the nose-off landing case, in which the aircraft has landed on its main gears only. With the main gears being positioned in the wings, the load of landing is transferred to the fuselage through the front and rear spars in the wing. As this load case was assumed to be on impact with the ground, the aerodynamic forces of the wing were neglected, as they were assumed to be 0. The landing load factor was assumed to be 3g, the same as the gear factor when designing the gear in the conceptual design report. The overall Shear and Bending moment graphs can be seen below in figure 5 and 6.

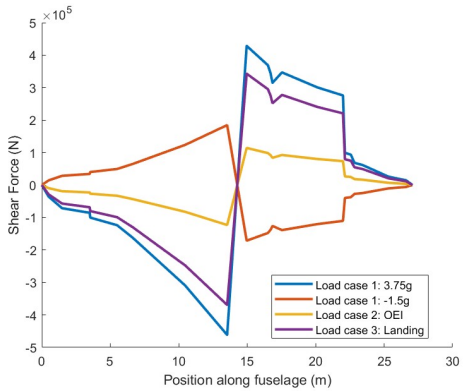


Figure 3.2: Shear Force Distribution

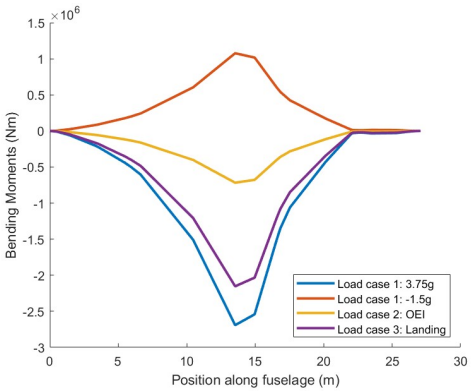


Figure 3.3: Bending Moment Distribution

3.3 Cabin Pressurisation Loads

An additional factor to consider for the fuselage is the pressurisation loading that occurs during a flight and over an aircraft's lifecycle. The fuselage will experience both an instantaneous pressure differential while cruising, and a cyclic pressure loading during climb and descent. This is because for passenger comfort and safety, the cabin is pressurised to 8,000 ft, while the absolute ceiling of the aircraft is at 48,000 ft. As such, there is a net 40,000 ft of pressure differential causing bulkhead stresses (σ_s) in the hemispherical caps, and hoop (σ_H) and longitudinal (σ_L) stresses in the cylindrical portion of the fuselage. This equates to about 75.3 Kpa of outward acting load. The stresses were calculated using equations (4.5)

$$\frac{\sigma_s}{P} = \frac{d}{4t}\frac{\sigma_H}{P} = \frac{d}{2t}\frac{\sigma_L}{P} = \frac{d}{4t} \tag{3.5}$$

From these stresses, the max skin thickness to withhold the pressurisation loads, $t_{fus,pressure}$, was found to be 0.27 mm. This skin thickness does not however include the total shear stresses that the skin will experience, which will be calculated and finalised in the section below.

3.4 Optimisation Method

The fuselage structure can be optimised using many techniques. Since this optimisation problem is non-linear, one method is to discretise the variables and iterate through every possible combination to find when the objective function is at a minimum. However, this method is quite inefficient and the computational cost increases exponentially with every additional variable or finer discretisation. This process was made more efficient using the $fmincon$ function in the MATLAB optimisation toolbox. The steps below show how this method was implemented for each structure:

1. An objective mass function was defined and was calculated using the variables for the dimensions which were to be optimised. This is the function which the system aimed to minimise.

2. Constraint functions were defined to ensure that certain criteria are met by the optimised solution. This consisted of buckling and other material failure equations to ensure structural integrity. Constraints are also required to ensure the optimisation does not converge at zero. These functions were stored in a function: *combinedconstraints* to allow them to be implemented simultaneously.

3. The upper and lower bounds of each variable were then set. The lower bound was dictated by the minimum manufacturable limit of 1mm and the upper bound was set to ensure a physically plausible solution.

4. An initial guess for each variable was set to guide the optimiser towards a local minimum. The nature of the outcome depended greatly on the initial guess as some variables would be kept to a minimum for each iteration based on the guess.

5. Using the *optimoptions* function, settings for $fmincon$ are configured. This includes specifying the algorithm type, termination tolerance, maximum iterations, and other parameters crucial for the optimisation process. These settings allow fine-tuning of the optimisation algorithm to suit the problem at hand, ensuring efficient and effective optimisation.

6. $fmincon$ as well as its various inputs as defined above were called using the $createOptimProblem$ function.

7. Since this process is heavily dependent on the initial guess, it was only capable of finding the local minima. The optimisation could be iterated through every possible initial guess between its defined bounds. This was a computationally expensive method and was automated using *GlobalSearch* to ensure that the problem's global minimum was found.

8. Finally, a sensitivity analysis was performed by using *grad* to understand the importance of each variable. This helps in identifying which variables have the most significant impact on the objective function and constraints. This information can be invaluable for refining the design and making informed decisions about which design parameters to prioritise or adjust in future iterations.

3.5 Skin and Stringer Design

With the skin thickness being designed with the consideration of cabin pressurisation loads, the thickness also needs to be designed by taking into account the maximum shear force, moment and torque on the fuselage as seen in table 3.4.

Maximum shear force (N)	Maximum bending moment (Nm)	Maximum torque (N)
461,530	2,714,900	77,832

Table 3.4: Most constraining loads acting on the fuselage

Symmetric loading at 3.75g produced the highest total shear flow. Therefore, using this load case, the minimum thickness required for the shear force was calculated as $t_{fus,shear} = 0.73$mm using equation 3.6. This ensures that the skin does not yield under the applied loads, Φ is the angle along the fuselage and β is the angle where the load is applied.

$$t_{fus,shear} = \frac{q_{max,total}}{\sigma_y} = \frac{1}{\sigma_y}(\frac{T + Pr}{2\pi r^2} + \frac{Pcos(\Phi - \beta)}{\pi r} + \frac{Qsin(\Phi - \beta)}{\pi r}) \tag{3.6}$$

When taking into account the minimum manufacture-able thickness of 1mm, both approaches of designing the skin thickness did not meet this requirement. Therefore the new skin thickness was set at 1mm. Stringers are introduced to carry direct stresses, hinder buckling and stiffen the fuselage. They span from the nose to the tail of the fuselage. The stringers were represented as concentrations of load-carrying points and therefore were evaluated using boom shear panel idealisation. The skin is idealised as individual shear panels smeared together.

A various selection of stringer cross sections were considered. Z stringers were chosen due to a number of reasons. They provide structural reinforcement while adding minimal weight to the overall aircraft structure. This is crucial as weight savings directly translate to fuel efficiency and performance improvements. Z-stringers can be efficiently manufactured using automated processes, such as roll forming or extrusion, which makes them cost-effective and scale-able for large-scale aircraft production. Additionally, in the case of repairs, damaged Z-stringers can be replaced or repaired relatively easily compared to more complex structural components. Z stringers are also known to protect against corrosion due to moisture. [11]

To optimise for the stringer's dimensions and placement, the mass of the skin and stringer was minimised while imposing three constraints: stress due to bending on the stringer, stringer buckling and skin buckling. Assuming a uniform stringer profile, stringer pitch and stringer width, the mass of the fuselage could be minimised. This was performed by calculating the stress limits of various combinations of skin thickness, spar width, spar thickness and number of stringers, and finding a local minimum for mass within feasible bounds. The stringers were modelled as simply-supported components in order to prevent Euler or global buckling throughout the design process, this constraint was imposed using equation 3.7. Since some clamping force is applied at the stringer end during real stringer assembly in the fuselage, this method produced a conservative estimate of the stringers' buckling strength. Similarly, the curving skin panels were modelled as flat plates to avoid local buckling under design load circumstances. The considerably wide radius of curvature of the skin panels made this approach suitable. It was corroborated by ESDU charts, which show that curved plates have an intrinsic safety margin because they have a larger critical local buckling stress than flat plates. [12]

$$\sigma_{Euler} = \frac{\pi^2 EI}{L^2 A_s} \tag{3.7}$$

In this load case, the shear distribution in the y axis was 0. With the dimensions of the stringer known with each iteration, for the symmetric loading case, the maximum direct stress was calculated using equation 3.8.

$$\sigma_{direct,max} = \frac{M_{max} z_{max}}{I_{yy}} \tag{3.8}$$

A normalised shear flow distribution at maximum shear force can be seen in figure 3.4. A normalised direct stress distribution can be seen in figure 3.5, which displays that in the symmetric load case the top half of the fuselage was in tension and the bottom half in compression as expected.

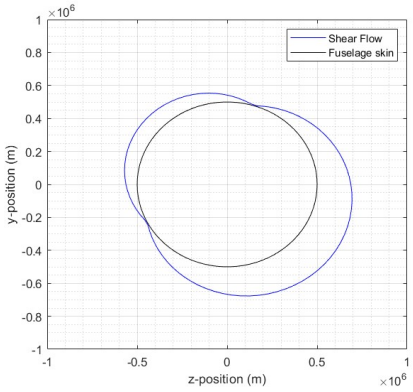


Figure 3.4: Normalised shear flow distribution

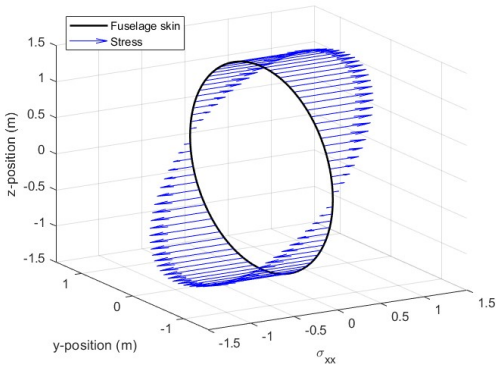


Figure 3.5: Normalised stringer direct stress

Here, the moment of area of the discretised booms and the skin were calculated at each iteration using equations 3.9 and 3.10.

$$I_{yy,stringer} = \Sigma y^2 A \tag{3.9}$$

$$I_{yy,skin} = \frac{\pi}{4}(r^4 - (r - t_{skin})^4) \tag{3.10}$$

For the skin, the shear force will be carried through it so it is likely based on bending theory [13], that it will fail through localised panel buckling rather than yield. Equation 3.11 was used to calculate shear stress and was compared with the material property. The ESDU-02.03.18 data sheet [13] was used to find the value of shear buckling coefficient K_s.

$$\tau = K_s E_{skin} (\frac{t}{b})^2 \tag{3.11}$$

With all of the constraints defined, the optimisation process was run using the *fmincon* function as described in section 3.4 and its code can be seen in [APPENDIX]. The number of stringers was rounded to be an integer and is assumed to be sufficiently large in number that I_{xx} could be considered to vary linearly with the number of stringers (providing they have the same area). The final optimised parameters are shown in table 3.5.

Number	Spacing (m)	Flange length (m)	Thickness (m)	Height (m)
87	0.091	0.021	0.0026	0.0397

Table 3.5: Stringer Parameters

3.6 Light Frames

In order to ensure panel stability for the overall strength and rigidity of the fuselage structure, the fuselage light frames were then introduced to the design. They maintain the fuselage cross-section and disperse the shear loads and stress sustained at specific points of the fuselage. For light frame design, a variety of

cross-sections are used, including C, tapered, and rectangular cross sections. Although the production of a tapered cross-section is costly and its resistance to bending and twisting is diminished in the design's thinner sections, it can help save weight and enhance aerodynamic performance. The cost-effective and straightforward rectangular cross-section can collapse prematurely due to stress concentrations caused by its sharp corners. Because of its comparatively high moment of inertia to cross-sectional area ratio and capacity to withstand bending and torsional loads, which increases its overall strength and stiffness, the C section was used in the light frame design.
Equation 3.12 was utilised to size the light frame's initial stiffness. This gave the frame the necessary second moment of area to stop general instability that spreads over several fuselage frames, as well as panel and skin instability. Equation 3.12 makes use of the maximum bending moment (M) acting on the fuselage across the diameter (d) and the frame spacing, as well as C_{fr}, an empirical correction factor of value 1/16000. In order to make sure there is adequate room between frames for windows, which has a typical value of 0.5 m.

$$I_{fr} = \frac{C_{fr} M d^2}{E L_{pitch}} = \frac{t_{fr} h_{fr}^3}{12} + \frac{w_{fr} t_{fr} h_{fr}^2}{2} \tag{3.12}$$

In addition, a buckling constraint was required which ensured that the skin can withstand the expected loading conditions. Equation 3.13 shows the transverse Euler buckling equated with the skin panel buckling.

$$K(\frac{t}{b})^2 = \frac{\pi^2 I_f}{A_{str} L_f^2} \tag{3.13}$$

With the number of stringers optimised for in section 3.5, the spacing of the light frames can be selected without over-compensating for the loads. With all of the constraints defined, the optimisation process was run using the *fmincon* function as described in section 3.4 and its code can be seen in [APPENDIX].

Number	Spacing (m)	Flange length (m)	Thickness (m)	Height (m)
54	0.507	0.014	0.002	0.047

Table 3.6: Light frame parameters

3.7 Heavy Frames

High concentrated stresses at particular points along the fuselage are sustained by heavy frames. Every heavy frame has distinct specifications because of the variations in loads at every location. These frames were placed at the front and rear spars of the wing and the front and rear spars of the tail. Using I-beams for heavy frames in the fuselage has various benefits. Firstly, because of their excellent strength-to-weight ratio, they are perfect for uses where reducing weight without sacrificing structural integrity is essential. I-beam design also guarantees uniform load distribution along the length of the beam, preventing localised stress concentrations that might cause failure. I-beams are very simple to install and manufacture. I-beams are a popular option for large frames in the fuselage because of their cost-effectiveness and exceptional performance qualities.
According to figure 5.8, the most limited load case—symmetric flight at n=2.5—determines the largest vertical shear force. P and T are taken to be zero for the symmetric flight and nose-off landing case since, in each case, only the landing gear reaction force and the wing spar's reaction are acting perpendicular to the frame. The main tailplane frames was designed with the influence of the OEI case, which affects the stabiliser torque (see figure 3.1). With the frame placements and radius predetermined from the conceptual design report, the resolved radial, tangential, and moment loads—taking downhill as positive—are given in Table 3.7.

Location	Radius (m)	P (N)	Q (N)	T (N)
Wing FS	1.405	0	491,220	0
Wing RS	1.405	0	341,598	0
Tail FS	0.727	-105,616	43,104	445,168
Tail RS	0.491	36,869	13,872	148,834

Table 3.7: Location of heavy frames and their corresponding resolved forces

Since the ring is cylindrical, WISE curves are used to reduce the redundancy problem and the equations needed to calculate the loads around the frame are provided in [14]. By linearly superposing these distributions around the frame, it was possible to calculate the total normal and shear force, moments, and other quantities. figure 3.6 provides an illustration of the WISE curves. To determine the smallest area needed to withstand these stresses, the maximum values for these distributions are extracted. Table 3.8 contains the values that were determined.

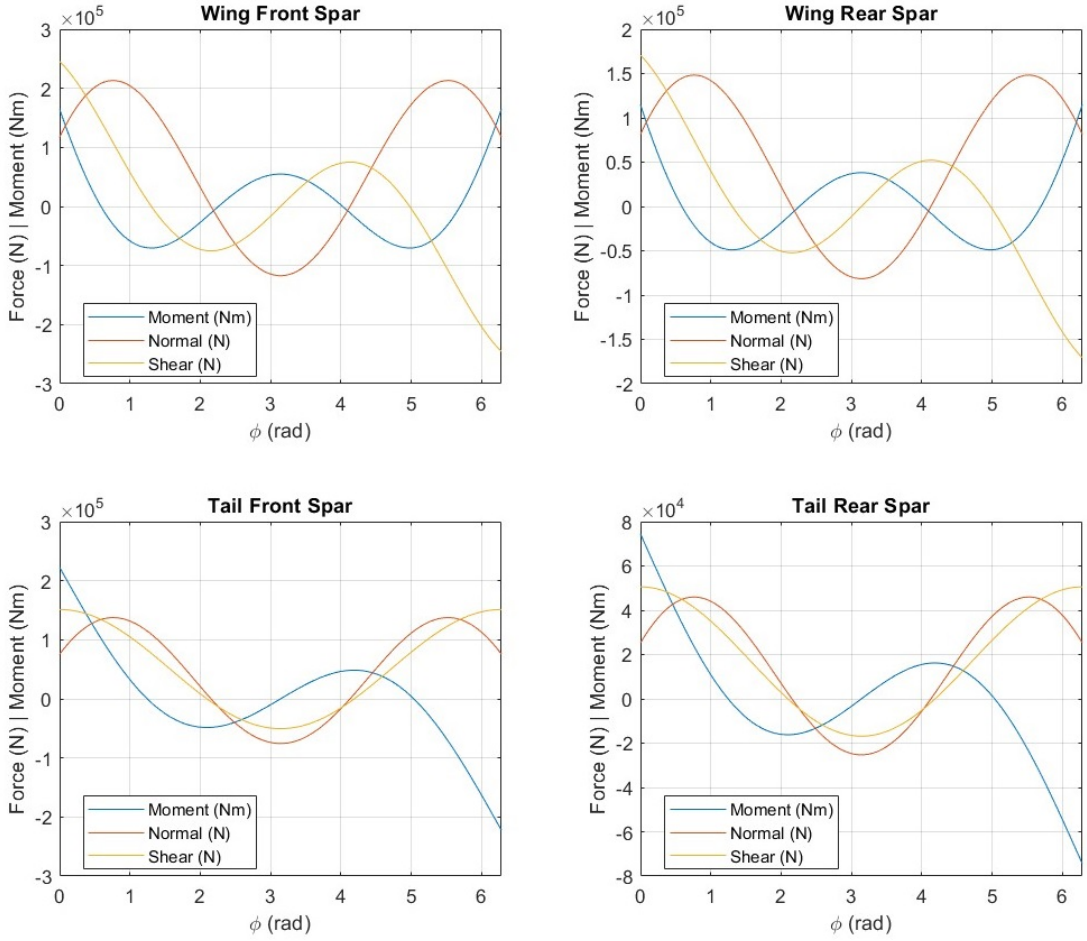


Figure 3.6: WISE Plots

Calculating the constraining loads allowed for optimisation of the lightest frames to prevent failure. Equations 3.14 and 3.15 from the stress definition are readily used to compute the area of frames needed to resist normal and shear stresses. However, they do not account for failure caused by bending moments. To get the minimum area due to moments, equation 3.12 was used which considers the second moment of area of the frame. The most constraining area out of the three was due to the bending moment and its required area is shown in table 3.8.

$$A_{normal} = \frac{N_{max}}{\sigma_{y,tensile}} \qquad (3.14) \qquad A_{shear} = \frac{S_{max}}{\sigma_{y,shear}} \qquad (3.15)$$

Location	M_{max} (Nm)	N_{max} (N)	S_{max} (N)	A_{min} (m^2)
Wing FS	164,760	213,270	245,610	0.003418
Wing RS	114,580	148,310	170,800	0.002197
Tail FS	222,584	137,570	151,280	0.002807
Tail RS	74,417	45,993	50,579	0.000784

Table 3.8: Location of heavy frames and their corresponding moment, shear and normal forces and area

With the maximum forces calculated, the optimisation process can be performed. A stiffness constraint was applied to ensure the integrity of the frame and to ensure that it does not fail under bending, its equation is given in 3.16. The optimisation was implemented as described in 3.4.

$$I_{req} = \frac{M_{max} h_{fr}}{2\sigma_{y,tensile}} \tag{3.16}$$

The optimised parameters for the heavy frames at their respective locations can be seen in 3.9 and its diagram alongside the light frames can be seen in figure 3.7

Location	Flange length (m)	Thickness (m)	Height (m)
Wing FS	0.0892	0.0104	0.0798
Wing RS	0.0887	0.0078	0.0805
Tail FS	0.1563	0.0099	0.0459
Tail RS	0.0622	0.0064	0.0492

Table 3.9: Heavy frame parameters

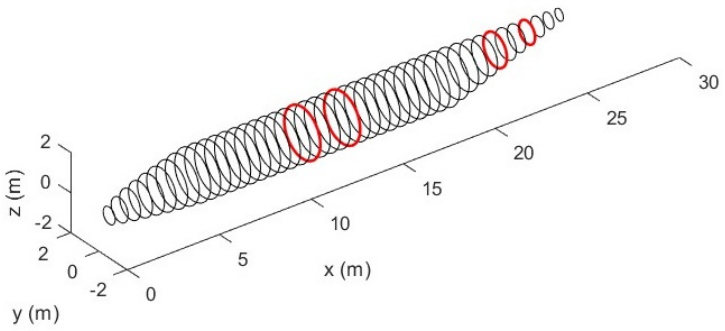


Figure 3.7: Light (black) and heavy (red) frame placement

3.8 Fatigue Analysis

3.9 Summary

Table 4.3 shows the mass breakdown of the fuselage. This is definitely not close to the mass of the fuselage estimated in table 3.2, hence this is an underestimate. The discrepancy is due to secondary structures such as rivets and joints and other components such as windows and doors not being accounted for in this analysis.

Component	Stringers	Light Frames	Heavy Frames	Skin	**Total**
Mass (kg)	597	243	193	816	**1849**

Table 3.10: Fuselage mass breakdown

The sensitivity analysis conducted sheds light on the influence of each variable on the overall objective function and constraints. By examining the gradient values presented in table 3.11, it becomes evident that certain variables exert a more pronounced effect on the fuselage structure's performance metrics compared to others. This insight enables engineers to focus their attention and resources on optimising the most influential parameters to achieve superior design outcomes.
In addition, the sensitivity analysis helps to identify potential bottlenecks or areas for improvement within the design process. Variables with high sensitivity levels may suggest crucial features that require additional research or refining to improve overall performance. As a result, this knowledge enables designers to efficiently deploy resources, prioritising efforts to solve the most important design aspects.
In conclusion, the sensitivity analysis is an invaluable tool for designers, providing critical insights into the importance of individual factors in fuselage structural design. Engineers can use this information to strategically optimise design parameters, refine the overall structure, and make educated decisions that will drive ongoing progress in aircraft performance, safety, and efficiency.

Parameter	Stringer	Light Frame	Heavy Frame
Thickness	0.0293	2.2644	0.0798
Height	-0.0006	1.1322	0.0805
Flange length	2.2644	0.0099	0.0459
Spacing	2.5159	-0.2547	0.0492

Table 3.11: Sensitivity analysis

Wing Design

4.1 Load Case Analysis

Every aircraft structural design starts with considering the loads it will be subjected to during usage. A working design is one that can withstand all loads reasonably possible during the aircraft's mission profile. There are many load contributions ranging from Aerodynamic, Inertia and even Propulsive origins. Typically, the starting approach taken when determining loads is considering the extremities of symmetric loading on the wings displayed on a V-n Diagram.

4.1.1 Flight Envelope

Figure 4.1 Displays the overall Flight Envelope mission profile details for the aircraft. All flight load factors and flight speeds should reside within the boundaries. It was generated considering the stall performance of the aircraft, the limit load of general business jets with a safety factor of 1.5 applied, the design cruise speed, and the design dive speed. The effect of gust perturbations were also considered, adjusting the positive limit load to a value of 2.71. The code used to generate the figure, and the construction lines, can be found in the Appendix. The method to generate the V-n Diagram followed the material from the lecture slides and from [15].

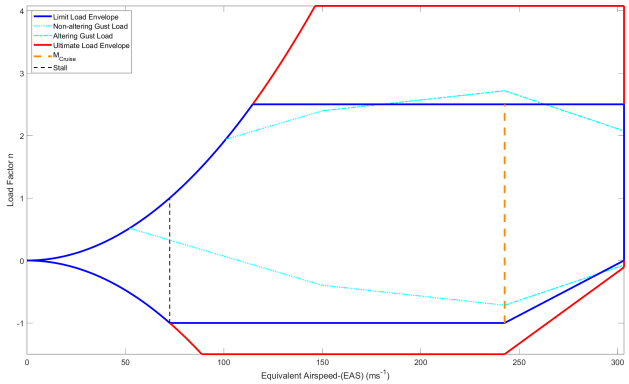


Figure 4.1: n-V Diagram during flight

4.1.2 Load Cases

Usage of the Flight Envelope and the detailing of the design cases from the design brief, three constricting design cases were determined. These will be responsible for setting the maximum loads the aircraft should be able to withstand provided an emergency, or even series of, occurs and said procedure is taken. These were defined as follows:

- Symmetric flight at the ultimate load factor, evaluated at manoeuvre V_A and dive V_D speeds, to determine the structural loading of the wing, fuselage and tail structures at failure.
- One Engine Inoperative (OEI) condition assuming a load acting at the aerodynamic centre of the vertical stabiliser such that the aircraft is directionally trimmed in case of single critical engine failure.
- Landing with the full aircraft load supported by the main undercarriage only.

4.2 Wings Loading

The V-n diagram is most useful in directly predicting the lifting loads experienced by the wings. In reality, and aircraft's total lift would be comprised of all lifting surfaces individual contribution, but designing the wing assuming it is solely responsible provides a margin of safety in design.

Calculation of Loads

Although the magnitude of lift generated by the wings is known, it is important to know how this lift is distributed across the wing. This requires an assumption of the span-wise distribution, and for further analysis, an assumption in the chord-wise distribution. These were taken as an Elliptical and Triangular distribution respectively. The triangular distribution was used to visualise the effect of a chord-wise distribution and hence was normalised at each chord length to have minimal impact on overall 3-D lift calculations.

$$L'(y) = \begin{cases} L'_0\sqrt{1-\left(\frac{y}{b/2}\right)^2}, & f_* \leq |y| \leq b/2 \\ 0, & otherwise \end{cases}, \quad \& \quad L'(x) = \begin{cases} \frac{8}{c}\left(\frac{x}{c}\right), & 0 \leq x/c \leq 0.25 \\ \frac{8}{3c^2}\left(c-x\right), & 0.25 < x/c \leq 1 \\ 0, & otherwise \end{cases} \tag{4.1}$$

Where, f_* refers to the half width of the fuselage in this case.
Formulation of the lift distribution was taken from the Excel File provided, but also regenerated using a MATLAB Script provided in the Appendix.
Figure 4.2 shows the general configuration of the wings. Powerplant loads were not necessary due to the aft podded design choice, hence the only other contributions are inertia loads due to unique mass distribution along the wing from fuel tanks and control surfaces, and loads applied through the landing gear. In reality, a large contributor would be from Aero-Elastic phenomena, however this requires a

dynamic analysis and development this early in the design cycle is best conducted on solely a static analysis.

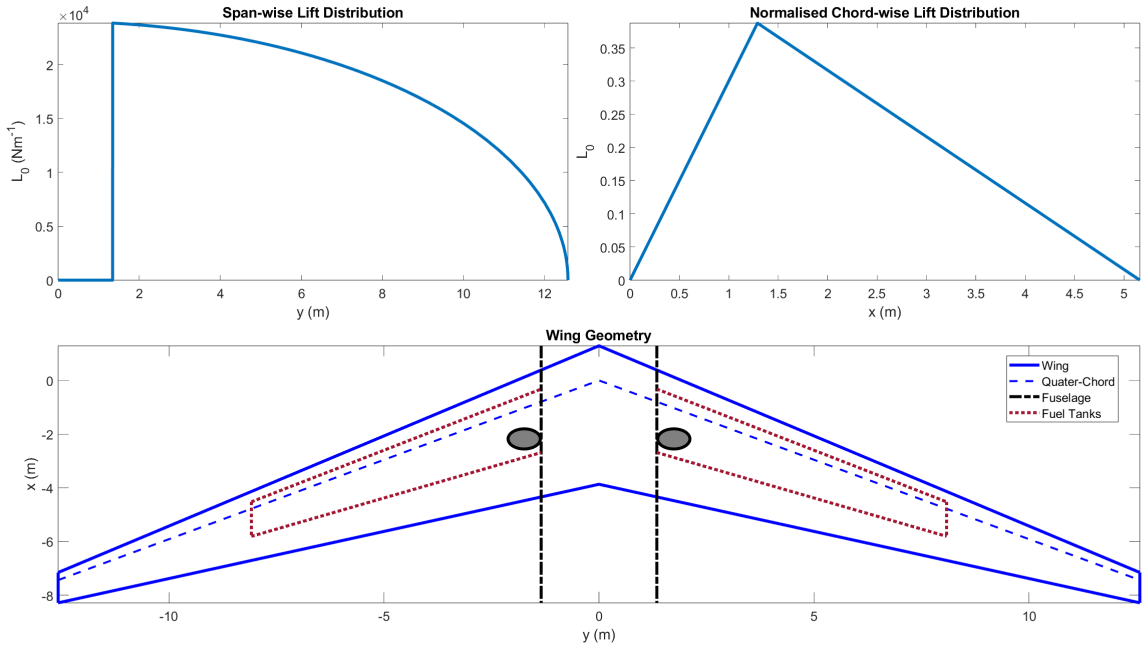


Figure 4.2: Lift layout.

Considering the inertia distribution and aerodynamic load, the shear force was evaluated to be the equilibrium force contributor to all span-wise load from the free edge. Furthermore, the Bending Moment was found to be the integral of the Shear force distribution. Lastly, the Torsional Moment was found as the individual moment contributor of each force, added to the sectional zero-lift Aerodynamic moment. An example of results using these relations can be seen in Figure A.1.

$$\begin{aligned} SF(y') &= \int_{y'}^{b/2} \sum F_v(y)\, dy, \qquad BM(y') = \int_{y'}^{b/2} \sum SF(y)\, dy, \ \& \\ TF(y') &= \int_{y'}^{b/2} \sum_{i=1}^{n} F_i(y) \cdot p_i + M_0'(y)\, dy \end{aligned} \tag{4.2}$$

For ease of computation, the analysis was further done on the 2-D Code (also provided in the appendix). However, the 3-D code revealed some topics to be later discussed.

Load cases 1 & 3

At, V_A, the ultimate load factors n_+ and n_- where found to be 4.07 and -1.50 respectively. In addition, the maximum gear load factor was found to be 3, highlighted in the conceptual design phase. This was paired with an assumption that the aircraft is descending at a constant rate, hence L = W. Evaluating the wing load distribution at these conditions yields Figure 4.3.

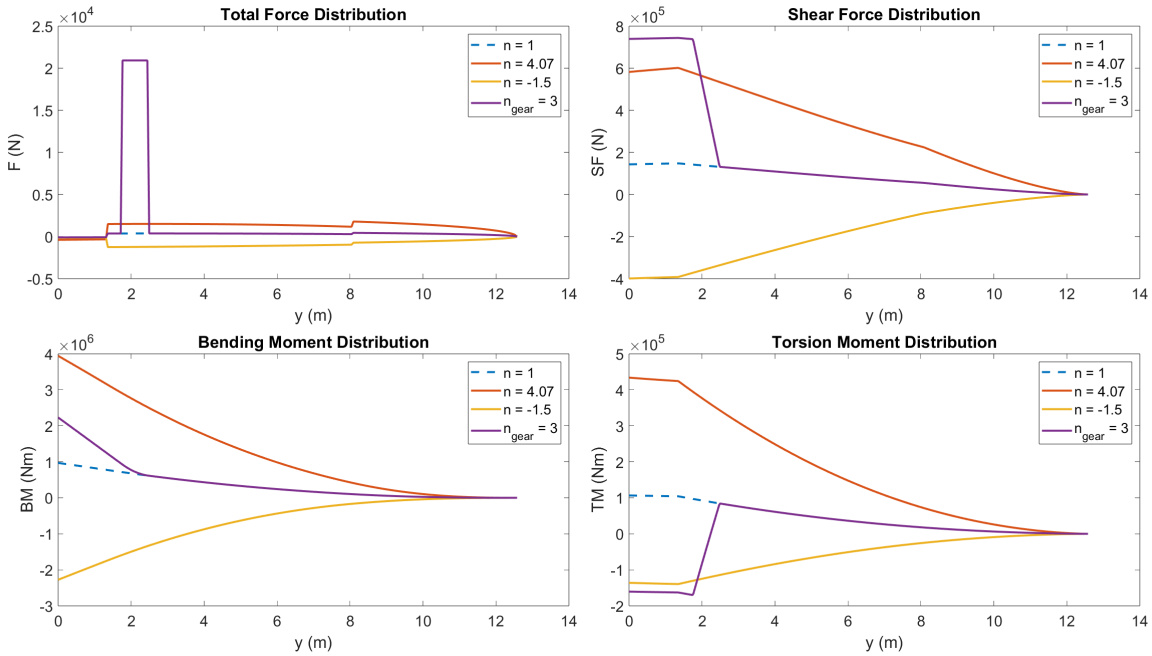


Figure 4.3: Wing Load Distribution

4.3 Material Selection

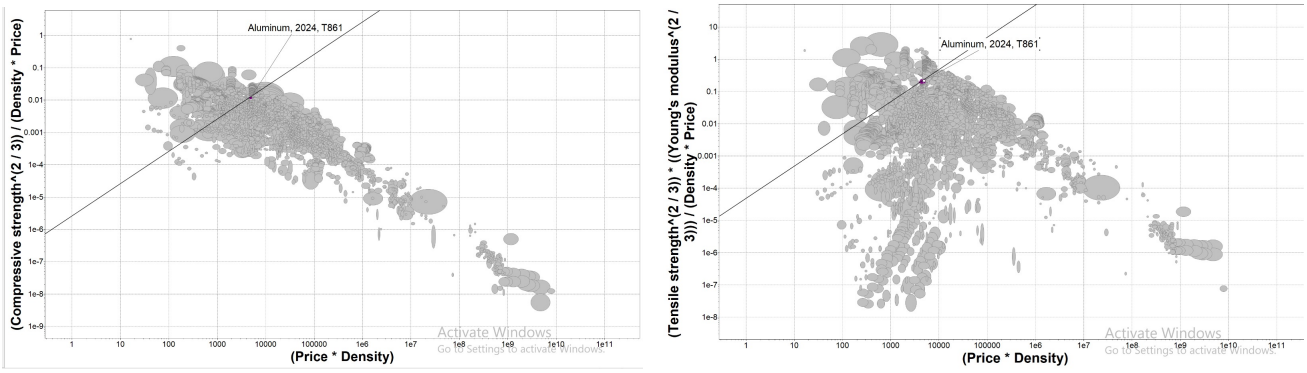
For the material selection of the wing, different parts of the wing were considered as well as what kind of properties a material should have for that specific part. The key considerations were that the wing would encounter a high shear force and bending moments. Furthermore, we want the wing to maintain its shape during the flight to maintain optimal aerodynamic performance. This is because if the wing flexes, the lift vector would be oriented in a different direction meaning the total lift would decrease. Hence, deformation and local buckling criteria were also considered. From CES, various merit indexes were used. During a flight, one could expect the upper skin on the wing to be under compression and the lower skin to be under tension. Furthermore, as the aircraft performs different maneuvers in the mission profile, the lift force can be modelled as cyclic loading, meaning the role of fatigue would also have to be considered. These were the main considerations taken into account for material selection. As mentioned in the literature review for materials, the most commonly used materials are aluminium alloys due to their high strength to weight ratio as well as their corrosion resistance properties.
Material indexes :

$$M_{stiffness} = \frac{E^{1/3}}{\rho c} \tag{4.3}$$

$$M_{compression} = \frac{\sigma^{2/3}}{\rho c} \tag{4.4}$$

$$M_{buckling} = \frac{\sigma^{1/2}}{\rho c} \tag{4.5}$$

For the upper skin, the factors considered in the selection process were compression strength, stiffness, buckling and fatigue. For the lower skin, it was tensile strength, buckling, stiffness and fatigue. For both of these merit indices, the material best suited came out to be Aluminium 2024-T861. Similar to the upper skin, the ribs also experience compressive loads and hence Aluminium 2024-T861 was the best choice for ribs as well. Finally, the for the spars, the spar caps experience compressive loading as well whilst the spar web carries the shear force. Aluminium 2024-T861 was once again the best material for this.



(a) Compressive Strength Merit Index **(b)** Tensile Strength Merit Index

Figure 4.4: Material Selection

4.4 Wing Box

The wing-box geometry was idealised from a DSMA-523B airfoil shape to a perfectly rectangular box in order to simplify the structural analysis. This was done by using the front and rear spars, located at 15% and 65% chord respectively, as the end points for the box. The height of the box was taken to be the mean height of the wing at the front and rear spars and was calculated to be 0.4543m at the root. The length of the box was taken to be the distance between the spars, 50% chord, and was calculated to be 2.58m at the root. Due to the geometry of the wing being tapered, the dimensions of the wing-box were also linearly tapered such that the height tapers from 0.454m to 0.099m and length from 2.58m to 0.565m from root to tip.

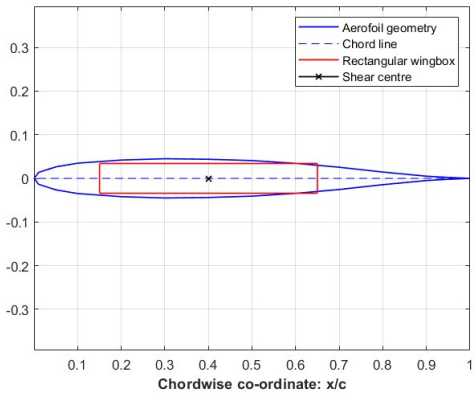


Figure 4.5: Wingbox Idealisation

Although this idealisation does not accurately replicate the shape of the airfoil, it serves as a conservative design approach as the flat panels used in this idealisation have a lower buckling resistance relative to curved panels. [16]

4.5 Skin-Stringer Panel Design

A similar approach to the fuselage section was taken to design the skin-stringer panels for the wing. Z-shaped stringers were used due to their high strength-to-weight ratio, better manufacturability and higher corrosion resistance.

The initial attempt at stringer optimisation involved designing an optimisation script from scratch. However, this approach was swiftly revised in order to vary a larger selection of parameters using the powerful $fmincon$ function on MATLAB which cycles through the different combinations of input variables and finds the optimal solution for a specified design parameter. The constraints applied for the optimisation

problem was buckling. The critical buckling stress was calculated using the following equation :

$$\sigma_{crit} = (\frac{\sigma_{crit}}{\sigma_0})KE(\frac{t}{b})^2 \tag{4.6}$$

The $\frac{\sigma_{crit}}{\sigma_0}$ term is a correction factor used for Z-shaped stringers as given in Farrar's 1949 paper[17]. K is the buckling stress factor and was taken to be 3.62 as a simply supported boundary condition was applied in order to consider a conservative design approach.
In the optimisation problem, a few assumptions were made. Firstly, the stringer orientation was set to be parallel to the flexural axis, the position of a distributed set of loads applied simultaneously along a wing that will result in zero twist along the entire wing.[18]. This axis was taken to be along the midspan of the wingbox with accordance to the guidance given in the lectures [19]. Furthermore, the stringer pitch was assumed to be constant in order to make the optimisation problem simpler as there is no real benefit to having a varied stringer pitch.
Alongside, the stringer pitch and number of stringers, the stringer dimensions and skin thickness were also varied in the optimisation process. This was done to ensure the minimum mass combination was chosen considering the mass of stringers and mass of skin together.
The optimisation approach was to vary these parameters using the *fmincon* function and specifying mass as the design parameter to be minimised. Limits were set for the values of each of the parameters due to manufacturing constraints and this resulted in the following optimal parameters.

Number	Stringer Spacing (mm)	Flange length (mm)	Thickness (mm)	Height (mm)	Skin Thickness
21	246.0	1.000	3.720	7.450	1.000

Table 4.1: Wing Skin-Stringer Panel Optimisation Parameters

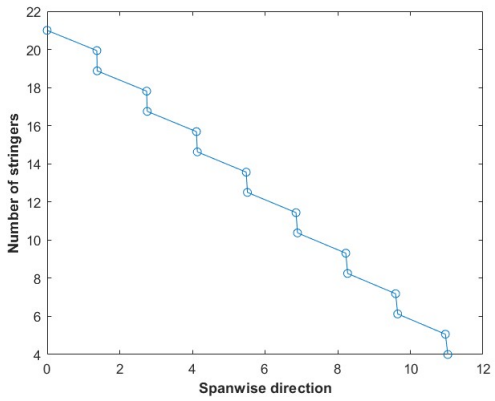


Figure 4.6: Spanwise variation of stringers

It was decided to couple the rib spacing parameter with the upper and lower panel design process - this entailed placing ribs at the known concentrated load locations and considering the wing section between them as independent panels and applying Farrar's correction factor to obtain optimal rib spacing at individual points in the panel.

4.6 Rib Design

The assumptions made regarding rib design are as follows. They were to be placed perpendicular to the flexural axis in order to unload the shear from the skin. Furthermore, ribs were to be as spaced out as possible whilst ensuring that the skin does not buckle. The first ribs were placed at known locations of concentrated load i.e. landing gear, flaps and aileron junctions. This further will aid in unloading the shear in the skin and reduces the tendency for the skin to buckle whilst also providing attachment points.

The approach following this was to consider the part of the wing from the root to the first rib as a panel and model the load acting on that panel as a point load. The optimiser increased the number of ribs in the panel in question whilst comparing the critical buckling load against the load that panel is required to sustain. Once this load was sustained, the optimiser would output the minimum number of ribs required and consequently the optimal rib spacing for that panel. In the interest of minimising mass and cost, this minimum number of ribs was used. These ribs were spaced out equally using this optimal rib spacing, such that the aspect ratio between the ribs and the skin remains close to 1. This is to aid in unloading the shear from the skin and further reduces the tendency for the skin to buckle [20]. The optimisation process was then repeated for all the panels along the span of the wing to obtain the optimal rib spacing for each panel.

Rib Station	**Rib Thickness,** mm	**Rib Spacing,** m	**Spanwise Distance,** m
0	1.000	0.000	0.000
1	1.000	0.562	0.562
2	1.000	0.562	1.124
3	1.000	0.562	1.686
4	1.000	0.562	2.248
5	1.000	0.562	2.810
6	1.000	0.562	3.372
7	1.000	0.562	3.934
8	1.000	0.562	4.496
9	1.000	0.562	5.058
10	1.000	0.562	5.620
11	1.000	0.562	6.182
12	1.000	0.562	6.744
13	1.000	0.562	7.306
14	1.000	0.562	7.868
15	2.000	1.124	8.992
16	2.000	1.124	10.116
17	2.000	1.124	11.240

Table 4.2: Wing Rib Stations

4.7 Spar Design

From the conceptual design phase of the report, the location of the spars were at 15% and 65% chord for the front and rear spar respectively.
The spar cross-section was chosen to be an I-section as discussed in the literature review. Hence, the key design parameter to be optimised was the spar web thickness. This was done by considering the web to carry only the shear force. Using the following equation, the thickness was calculated along the span of the wing:

$$t = \sqrt[3]{\frac{qH^2}{8.1E}} \tag{4.7}$$

where H is the height of the wingbox and q is the shear flow through the front and rear spar calculated using:

$$q = q_0 \pm q_b \tag{4.8}$$

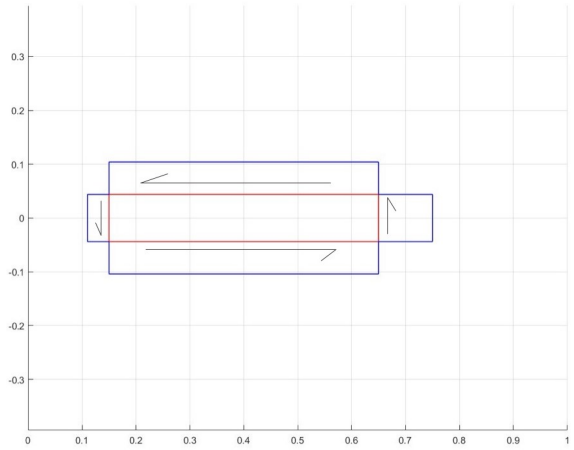
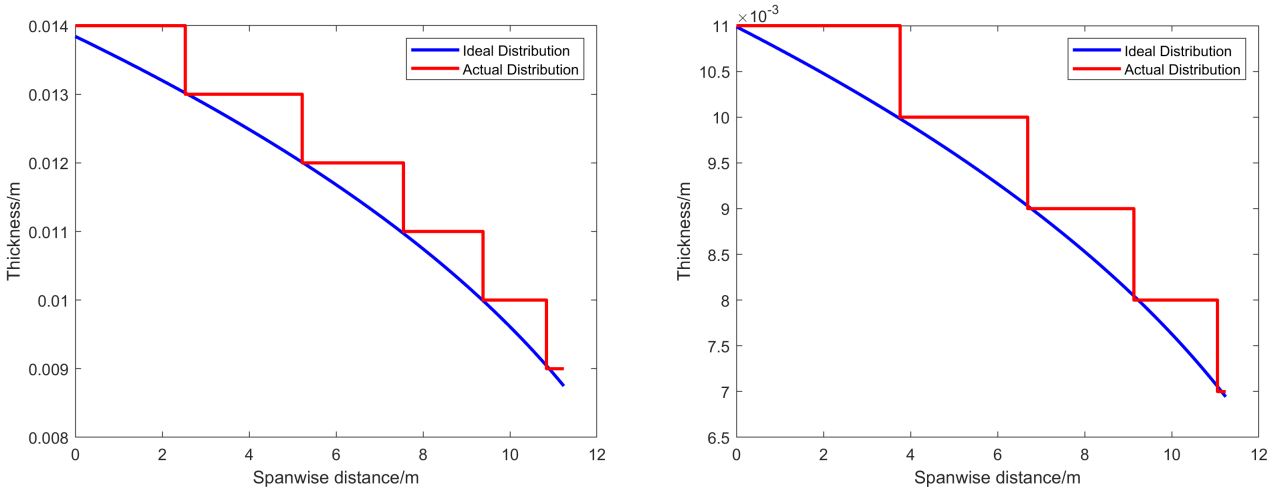


Figure 4.7: Shearflow around the wingbox

This method resulted in an optimal spar web thickness distribution that decreases along the span of the wing due to the lower load towards the tip. However, due to manufacturing reasons, the spar web thickness cannot be continually varied. Hence, a stepped approach was taken to minimise mass whilst still ensuring manufacturability.



(a) Front Spar Web thickness Distribution

(b) Rear Spar Web thickness Distribution

Figure 4.8: Spar Thickness Distributions

4.8 Final Summary

Component	Ribs	Skin	Spars	Stringers	**Total**
Mass (kg)	96.8	1378.8	452.2	357.1	**2284.9**

Table 4.3: Wing mass breakdown

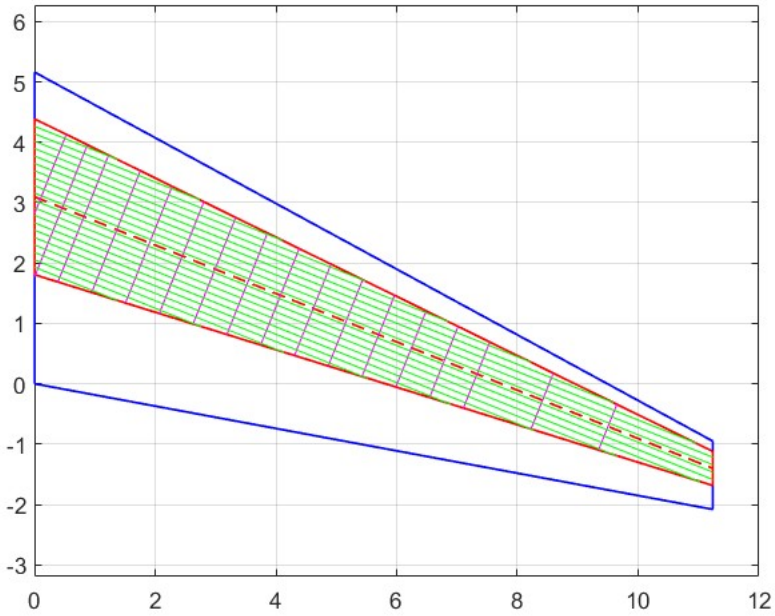


Figure 4.9: Final Wing Design with optimised ribs and stringers configurations

Empennage Design

5.1 Empennage Loading

The loading on the Empennage was mainly split into two contributions. This was the shear loading on the Horizontal Tail-plane, and the Vertical Tail-plane.

5.1.1 Calculation of loads

The Horizontal Tail-plane, first had its aerodynamic loads evaluated by considering the total lift required from the tail to trim the aircraft at the current flight conditions. This was estimated using equation 5.1. These loads were assumed to be distributed elliptically across the span of the HT.

$$L_H = \frac{nW\left(x_{cg} - x_{np}\right)}{\left(x_{ht} - x_{np}\right)} \tag{5.1}$$

The inertia loading was considered to only be dependent on the HT weight distribution as the nature of a T-Tail leads to a similar relationship to the Wings. The effect of pitching angle was also considered, but very quickly revealed the maximum load cases were for level flight.
The Vertical Tail-plane, had its aerodynamic loads evaluated by considering the required lift to counter act any side-slip moments. These become important when considering Load case 2, as the engines provide a consistent torque on the aircraft.
This was more complex than the HT case, as simply modelling the lift on the VT as an individual sum results in an under-defined moments equation. This appears to be a problem as it results in the calculated lifting load required at the Tail, appearing to change as the point where moments are taken about varies along the length of the plane.
The FAA [21], present different flying conditions and how an aircraft deals with OEI. For Wings Level flight, the load was modelled as a pair of load points, one generated by the Vertical Tail-plane solely, and the other generated by the Rudder. This however requires large forces on the VT, due to the large reduction in effective moment arm.

The more ideal case, is flight with a slight bank angle. This produces just the right weight component to counter act the force produced by the tail-plane. Plane designers usually define a V_{MC} which dictates the sea-level equilibrium speed the aircraft can fly at before needing larger than a 5°bank during OEI. We can find the Lift required by now taking moments from the CoG.

$$L_V = F_{Thrust} * \frac{(y_{eng} - y_{CG})}{(x_{HT} - x_{CG})} \tag{5.2}$$

For predicting bank angle, we can find the angle that satisfies $W \sin \alpha = L_V$. Lastly, this can be used to find the Lift force required for flying at equilibrium, $L = W \cos \alpha$. The code to produce these values can be found in the Appendix.
The Lift distribution for the VT was considered to be an elliptical distribution along the span. This situation is slightly different to the Wing and Horizontal Tail-plane, as neither end is exposed to the free-stream. Regardless, a decision to treat the x-component of Shear force produced by the HT as negligible was made. This is a fair assumption, relying on no major side-slip analysis such as flat spins, but the effect from the HT considering an uneven lift production between fins, will produce a Torque on the VT. The weight of each component was assumed to be 1% of MTOW for each as shown by NASA [22] highlighting Raymer, Roskam and FLOPs approximate methods for predicting Vertical Tailplane weight. Raymer's provided the largest weight and hence was taken.
Wrong! With the shear force being considered chord-wise as the summed magnitude of both Rudder and VT contributions. This is assuming, due to the Rudder being attached the VT, the shear force experienced in the VT will have to account for any extra load generated by the Rudder. The weight of the VT can be ignored for the calculations, as the weight does not contribute to any no axial force, i.e. the weight is purely a compressive or tensile stress (assumption in S.L.F.). The Control Surfaces' layouts can be seen in Figure 5.1 and 5.2. An enlarged version can be seen in the appendix A.6 & A.7.

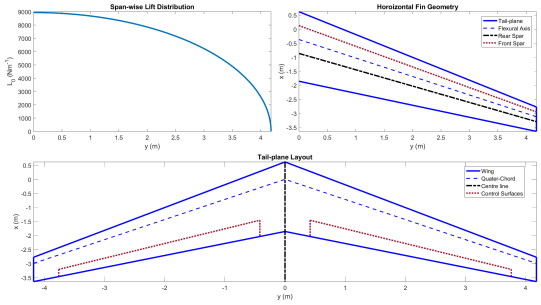


Figure 5.1: Horizontal Tail-plane

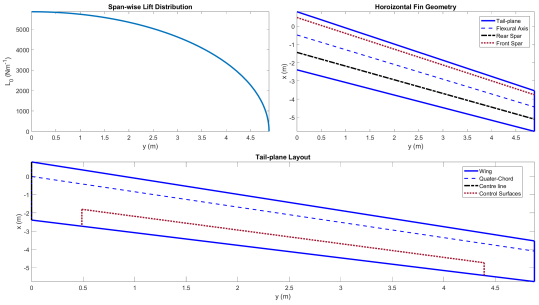


Figure 5.2: Vertical Tail-plane

Load case 1 & 3, HT

At, V_A the load generated at the Tail-plane was taken to be the trim condition required for the lift loads calculated earlier. In addition, the maximum gear load case was added to the Lift force at n = 1. Evaluating the HT load distribution at these conditions yields Figure 5.3.

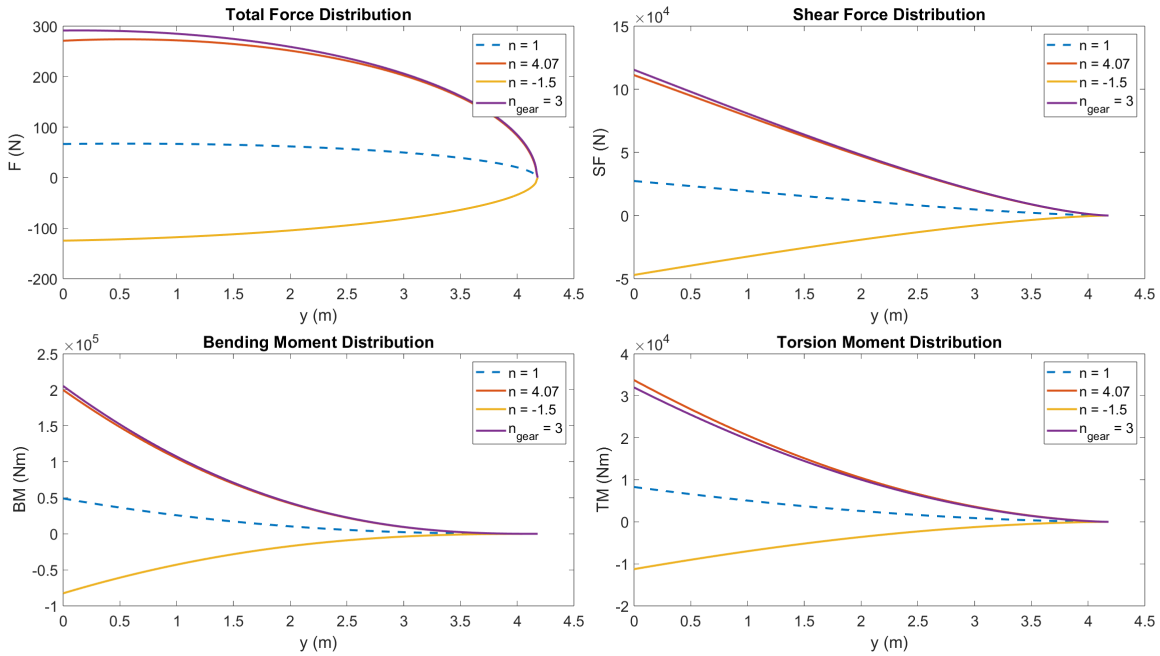


Figure 5.3: Horizontal Tail Load Distribution

Load case 2, VT

Evaluating Equilibrium about the aircraft centre of gravity provided the value for L_V at max Thrust OEI. This was multiplied by the previous safety factor of 1.5 as FAR highlights this should apply to all limit loads.

The situation of constant bank to account was assumed, and the magnitude for bank angle was found to be 3.1572°. This is well within the 5° limit highlighted in the earlier source. However, the torque was calculated slight differently. To remain consistent with Airplane Flying Handbook, the lifting force was assumed to be produced at the quarter chord of the rudder, instead of the entire VT.

For redundancy, the matching conditions for which Wings Level OEI Flight could be sustained was evaluated. This occurs at around 30% Thrust. In this case, the Shear Force is eliminated, but there is a far larger contribution to Torsional Moment. The forces were modelled at the quarter chord of the Vertical Stabiliser (excluding Rudder) and Rudder respectively. These results are shown in Figure 5.4

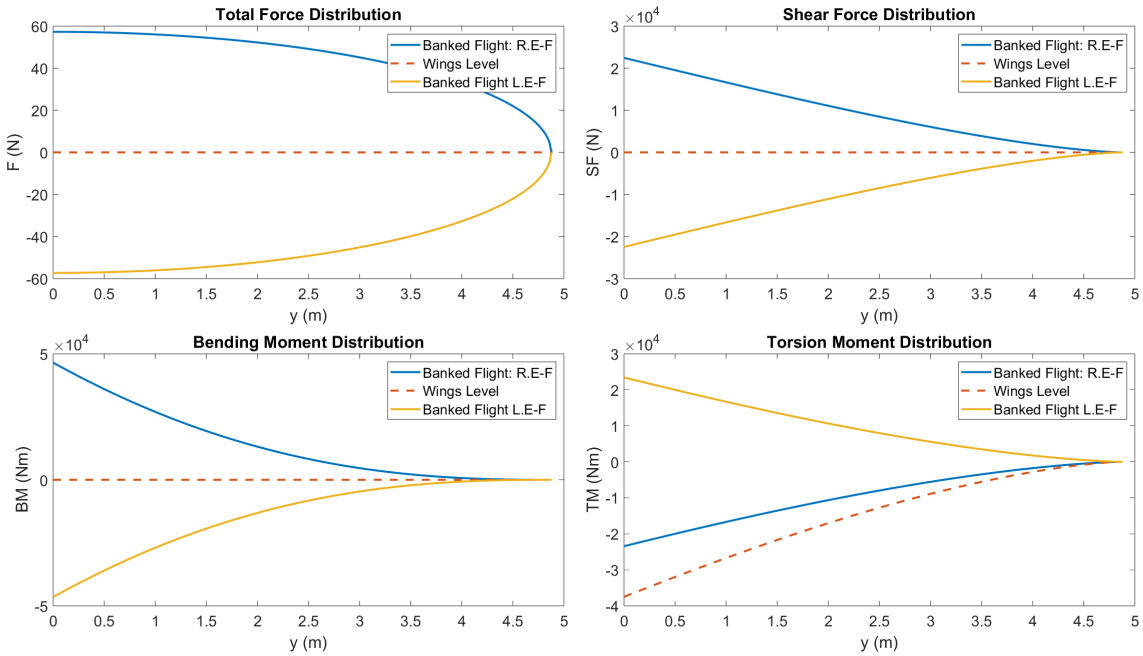


Figure 5.4: Vertical Tail Load Distribution

In reality, the dynamics are far more complex as the individual drag on the wings should be accounted for, especially the variation between wings due to either dihedral or a coupled roll motion, but this quickly becomes an intractable problem.
T-tails even add the influence of elevator loads to the vertical tailplane. However, this is largely mitigated by the swept-back nature of the vertical tailplane.

5.2 Vertical Tailplane

5.2.1 Material Selection

Due to similar considerations to the main wing, the same materials were chosen for the vertical tail skin-stringer panels, spars and ribs.

5.2.2 Wing Box

The wing-box geometry was idealised from an EPPLER EA 6(-1)-009 airfoil shape to a perfectly rectangular box in order to simplify the structural analysis. This was done in a similar manner to the process used on the wing by using the front and rear spars, located at 15% and 65% chord respectively, as the end points for the box. The width of the box was taken to be the mean width of the tailplane at the front and rear spars and was calculated to be 0.2158m at the root. The length of the box was taken to be the distance between the spars, 50% chord, and was calculated to be 1.5775m at the root. Due to the geometry of the wing being tapered, the dimensions of the wing-box were also linearly tapered such that the width tapers from 0.2158m to 0.1511m and length from 1.5775m to 1.1044m from root to tip. Note that the same wingbox idealisation was performed for the horizontal tailplane also.

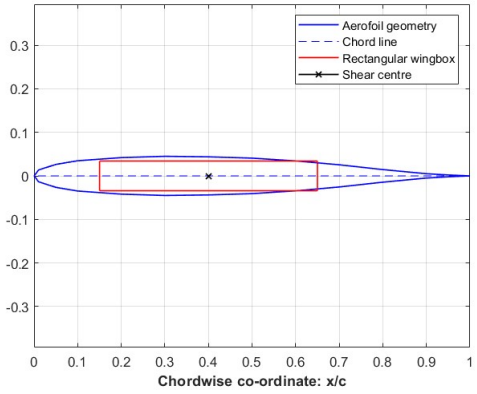


Figure 5.5: Visualisation of wingbox on Eppler EA 6(-1)-009 Aerofoil

5.2.3 Skin and Stringer Design

As the loading on the vertical tailplane is symmetric and has the same constraining load case as the wing, the same approach to designing the Skin-Stringer panels was taken. The panels were sufficiently supported such that they wouldn't buckle at ultimate load factor. The fmincon function on MATLAB was used to find the optimum stringer dimensions and stringer pitch. The final dimensions were:

Number	Stringer Spacing (mm)	Flange length (mm)	Thickness (mm)	Height (mm)	Skin Thickness
6	526	1.000	2.510	5.030	1.000

Table 5.1: Wing Skin-Stringer Panel Optimisation Parameters

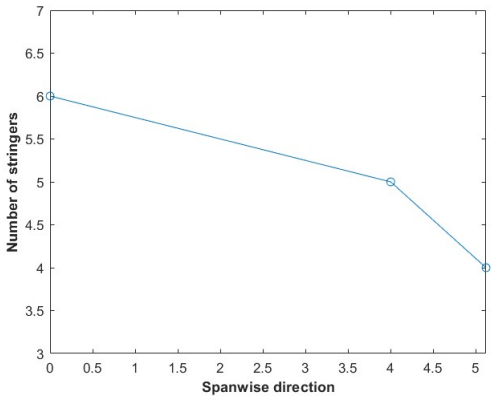


Figure 5.6: Spanwise variation of stringers

5.2.4 Rib Design

The method for calculating the optimal rib spacing for the vertical tailplane was slightly different to the main wings. Instead of splitting the span into panels depending on concentrated load points, only one panel was considered between the root and tip of the vertical tailplane and an equi-spaced rib configuration was chosen based on the minimum number of ribs required to ensure it doesn't buckle which would minimise mass whilst ensuring structual integrity. The final rib spacing was as follows:

Rib Station	Rib Spacing, m	Spanwise Distance, m
0	0.000	0.000
1	2.440	2.440
2	2.440	4.880

Table 5.2: Table overivew of Al 2024-T861 material properties

The rib spacing here is much larger than the rib spacing in the wing, which is in line with prior predictions as the vertical tailplane is having to sustain much less force than the wings, thus requiring less structural support.

5.2.5 Spar Design

For the spar design, a very similar method was adopted to the wing spar design where the shear flow was used to calculate the thickness distribution of the front and rear spars. However, after performing the calculation, the thickness for both the front and rear spars only varied by around 0.3mm over the entire span of the VT. Hence, for ease of manufacturing, a decision was made to have constant thickness on both front and rear spars.

Front Spar Thickness (mm)	Rear Spar Thickness (mm)
3.0	2.4

Table 5.3: Wing Skin-Stringer Panel Optimisation Parameters

5.2.6 Final Design and Layout

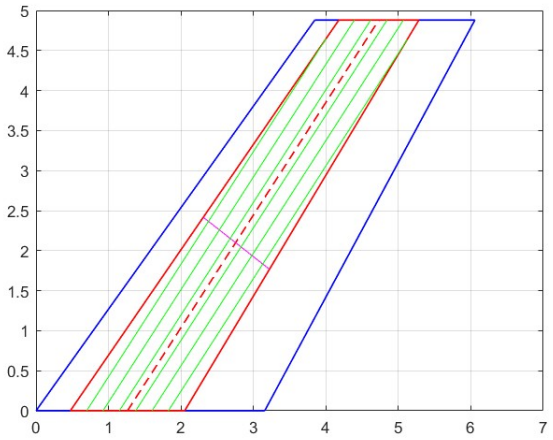


Figure 5.7: Final Vertical Tailplane Structure

5.3 Horizontal Tailplane

5.3.1 Composite Design

The horizontal tailplane (HT) was designed to be made with a composite design, as per brief. The HT is expected to experience similar loading cases to the main wings and, hence, similar constraints were considered when designing the structure of it. This mainly revolved around optimising the HT to prevent buckling as composite material's tend to have a much lower compressive strength relative to tensile strength.
For the composite material, a unidirectional prepreg tape was chosen which consists of an epoxy matrix and carbon fibres which was optimised to ensure the best material properties were obtained. The thickness

of plies were set to an industry standard of 0.125mm with a manufacturing constraint of minimum laminate thickness of 1mm enforced additionally.
Plies of orientation 0°, ±45° and 90° were used to construct the final laminates. Each of the different orientations are used for different purposes. Specifically, the 0° ply helps the column stability and carries the tensile and compressive loads. The ±45° ply helps with buckling stability and carries the shear loads. The 90° ply carries the transverse loads and reduces Poisson effects. Hence, these plies can be arranged in a way that achieves the desired performance in different structures.
The number and distribution of the different orientations of plies follow a few rules. There must be a minimum of 10% of each orientation of ply and a maximum of 70% in a well-rounded laminate. Furthermore, stacking multiple plies in the same orientation should be avoided if possible. The number of +45° and -45° plies should be equal for a balanced laminate, which can be implemented using ply pairing. It is also useful to have the outermost plies oriented at ± 45° to improve resistance to potential impact. [23] For the laminates, constant ply thickness was used alongside a symmetric laminate about it's mid-plane.
The number of plies required orientated in the 0° direction, n_{0°, was calculated using the following formula:

$$n_{0^\circ} = \frac{M}{\sigma_{xx} htb} \tag{5.3}$$

where M is the bending moment, h is the height, b is the width and t is the thickness.

This number was then multiplied by 0.1 to obtain the number of 45° and 90° plies. This process was done at the root to obtain the maximum number of plies required.
As the load along the span of the wing decreases, a lower number of plies are required. Hence, the laminates have been designed with an interleaving pattern. However, this must be done in a specific manner for a well-designed wing such that multiple adjacent plies don't terminate as this could hinder the desired performance.

Figure 5.8: Front Spar Web thickness dsitribution

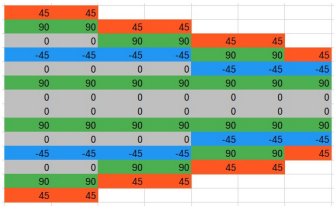


Figure 5.9: Rear Spar Web thickness dsitribution

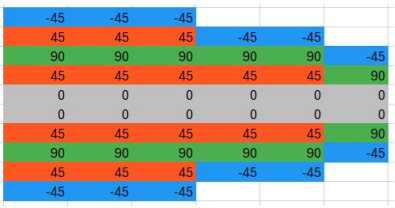


Figure 5.10: Ply Layup for Spars

The distribution of plies was done iteratively by considering the critical buckling stress of the laminate using the following equation:
where the D_{11}, D_{12}, D_{22} and D_{66} values were obtained from the ABD matrix. After multiple iterations, the following ply distribution was chosen for the rib, skin and spar sections:

5.3.2 Skin and Stringer Design

The design of the skin-stringer panels followed a very similar procedure to the main wings. This included the use of the fmincon function to optimise the stringer pitch and dimensions alongside the skin thickness. The key difference in this process was that the material chosen was different but the rest of the process was identical. This yielded the following optimal results:

Number	Stringer Spacing (mm)	Flange length (mm)	Thickness (mm)	Height (mm)	Skin Thickness
6	412.8	1.000	1.000	4.320	1.000

Table 5.4: HT Skin-Stringer Panel Optimisation Parameters

For simplicity in manufacturing the skin thickness was kept constant on both the upper and lower skin panels.

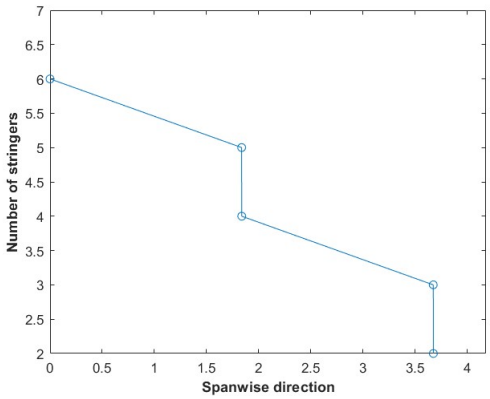


Figure 5.11: Spanwise variation of stringers

5.3.3 Final Design and Layout

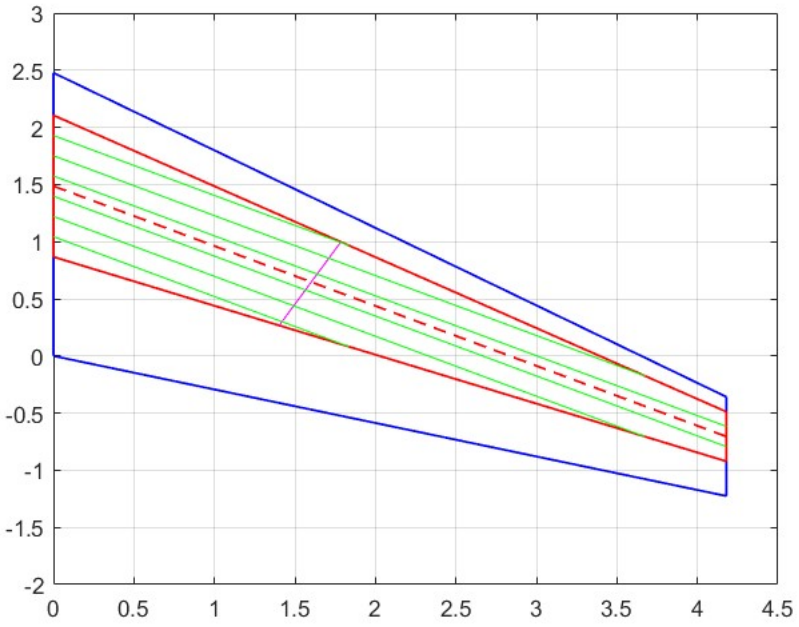


Figure 5.12: Final Tailplane Design with optimised ribs and stringers configurations

Fatigue Analysis

An important consideration for failure of structural component is fatigue. Fatigue occurs when the aircraft structure is loaded and unloaded repeatedly with a cyclic load. In this analysis, a high-cycle fatigue analysis was preferred. This is because with the range of the aircraft, it is expected to carry out one ground-air-ground cycle per day, resulting in a low amplitude but high frequency loading. The lifespan of this aircraft is expected to be on par with average business jets, about 15,000 to 20,000 flight cycles.

For the fatigue analysis, it was important to find the maximum stress on the aircraft, which was found to be at the base of the wing in the 1g case. This stress of 105MPa was assumed to act through the wing spars at the base, which is made of Al 2024-T861. The fatigue strength of this material is 136 MPa, and therefore, the stress is in the safe zone of fatigue. Furthermore, this also falls in the safe zone of the the notched S-N curve for this material, and therefore fatigue is not expected for the wing. Additionally, with Al 2024-T861 having the lowest fatigue strength of the materials used in this aircraft, the plane is designed against fatigue failure.

That being said, crack propagation for our aircraft may still occur, as the values used in the analysis are just estimates. As such, a fail-safe approach is preferred for our aircraft, and regular inspections and maintenance should be conducted to ensure that the structure is safe against fatigue [24].

Secondary Structures

Secondary structures are defined as the structural or non-structural parts of an aircraft whose failure would not affect the structural integrity of the aircraft, but may affect the performance of said aircraft. That being said, these structures still need to be designed with normal operations in mind, and must be able to withstand the limit and ultimate loads of the aircraft.

7.1 Cutouts

As the name suggests, cutouts are area in the structure that are removed, either for strategical weight-saving purposes, or to fit in commodities such as windows and doors. However, cutouts lead to a local increase in the stress concentration, and therefore require thicker material to accommodate this increased stress concentration [25].

An example of a cutout are windows. These come in a standard oval shape, as the ovular shape helps to ensure an even stress distribution through the lack of sharp edges, which would otherwise carry about 70% of the stress concentration. Furthermore, to reinforce the windows to deal with the constant pressurisation and the cyclic loading, aircraft windows come in three layers made of stretch acrylic – the thick, outer layer that deals with the pressure and the elements, a middle layer with the bleed hole to equalise the pressure, and a thin inner layer called the scratch pane, that helps reduce damage to the other 2 layers. Lastly, doublers are added to the windows to reinforce them structurally, preventing the shear forces and bending moments from affecting them [26].

Cutouts for doors are much larger than that of windows, resulting in higher stress concentrations in the skin surrounding this cutout. As such, this necessitates a thicker skin distribution around a door, which is achieved through bonding of metal sheets called bear straps around the door cutout. Additionally, the corners of doors are rounded to prevent crack growth and propagation through unnecessarily high stress concentrations [25].

Cutouts are also seen in wing ribs and spars for weight saving purposes, as it results in fewer material

being used to manufacture these parts. Careful care and consideration needs to be taken when designing these cutouts, ensuring that the structural member is still able to perform its duties without compromise, which includes the use of FEA. Similar to doublers, flanged holes can be used as reinforcement for the weight saving cutout [27].

7.2 Joints and Fittings

Aircraft structures incorporate a variety of joints and fittings essential for structural integrity and load distribution. These include rivets, threaded fasteners, blind fasteners, and nails. Rivets are permanent fasteners that transfer loads primarily in shear, ensuring structural stability. Threaded fasteners, commonly known as bolts, allow for disassembly and re-installation, offering versatility in maintenance. Blind fasteners are used in confined spaces where access is limited. These joints are meticulously designed to withstand different types of loads, including tension and shear forces, crucial for maintaining the stability and safety of the aircraft structure. Failure modes of these fasteners include sheet failure, bearing strength failure, and shear out of the sheet. Mechanical fastening in composites involves considerations of the orthotropic nature of composite panels, where strength depends on fibre orientation. Welded joints in metals provide integral structures but may lack natural barriers against crack propagation. Adhesive joints, while offering high strength, require pre-treatment for optimal bonding and are not typically used as single load path solutions in primary structures due to concerns about premature failure. Proper installation, regular inspection, and adherence to stringent maintenance protocols are imperative to uphold the structural integrity and airworthiness of the aircraft [28].

7.3 Engine Mounts

The engines on the aircraft will be mounted at the rear of the fuselage using pylons. These pylons are a vital structural component of the engine-fuselage connection as they serve a multitude of roles. Apart from being the structural connection of the engine, they also serve to transfer all the load generated from the engines to the fuselage. This includes shear, bending, torque, and the vibrations generated by the engines. This is done using spherical roller bearings, due to their load carrying capacity, self-aligning capabilities and resilience to shock loads. The engine pylons also have a role in emergencies, designed to breakaway to prevent the fuel tanks from catching fire [29].

7.4 Undercarriage

Due to similar considerations as those for the engine mounts, it is essential for the undercarriage to have the capability to detach in case of an emergency landing. As outlined in the conceptual design concept, the undercarriage will be affixed to the rear spar of the wing, leading to heightened stress on this component, particularly during landing manoeuvres. To address this, there are two primary approaches for reinforcement: either utilizing an auxiliary spar to distribute some of the extra loads or strengthening the rib structure. The latter option, involving reinforced ribs, offers direct support to the attachment point on the rear spar and is deemed superior, especially given the utilisation of high lift devices at the wing's trailing edge.

Detailed Component Design

Detailed design is a further step in the development process of an aircraft, where structural performance is considered at an individual component level. The team had been tasked with designing an efficient structural member for a flap mechanism according to a specification laid out in the brief [30]. The metric by which the various designs were ranked was according to following merit index.

$$\eta = \frac{LOAD(N)}{MASS(KG)} \tag{8.1}$$

8.1 Design Considerations

8.1.1 Material Properties

The material used for this design was Al 6061-T6. The structure was cut from a blank of 6.3mm thickness, as seen in figure 8.1, and all holes and cut-outs were through the full thickness as prescribed in the brief. Table 8.1 summarises the relevant mechanical properties of the material [31].

Table 8.1: Mechanical Properties of Aluminium 6061-T6

Density (kgm^{-3})	**Young's Modulus (GPa)**	**Poisson's Ratio**	**Yield Stress (MPa)**	**UTS (MPa)**
2,700	70	0.33	276	310

8.1.2 Plate Pre-processing

An initial shape optimisation study for the entire aluminium blank was carried out in Fusion 360 to determine the load paths, and consequently to ascertain the locations where material would need to be retained. From this study, much of the aluminium blank's initial mass was deemed superfluous so the decision was made to remove it before starting the topology optimisation process. This rationale was followed due to the prescribed merit index given to judge structural performance, rewarding large reductions in mass, and isn't an explicitly necessary step in the design process. Figures 8.1 and 8.2 illustrate the typical pre-processing step, showing a reduction in mass from 2.01 kg to 1.08 kg before further optimisation.

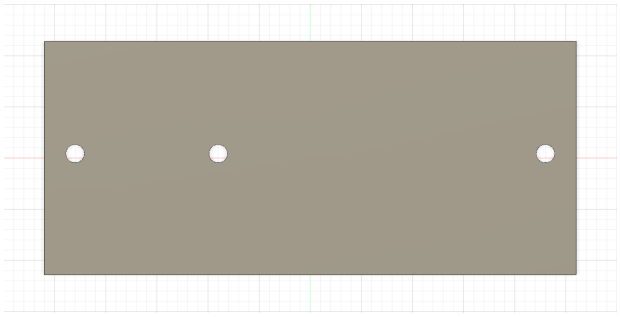


Figure 8.1: Initial Aluminium blank geometry

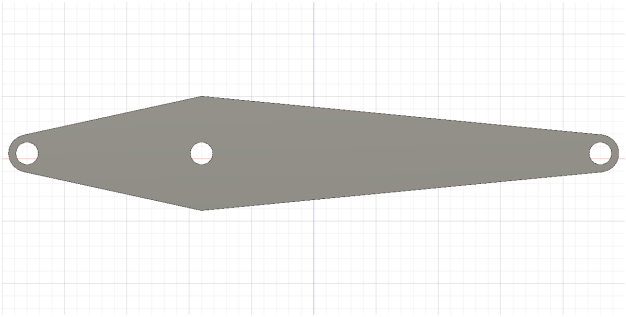


Figure 8.2: Pre-optimisation geometry

8.1.3 Loading Case

The slender nature of the structure made it more susceptible to failure via buckling. The idea of having a primarily compression-based or tension-based design was explored. These designs would be achieved by distributing the structure's mass predominantly above or below the loading point, respectively. This strategy was enacted by producing various 'starting geometries' that became the basis for further optimisation via FEA and CAD software, as illustrated in figure 8.2.
The particular combination of member geometry, loading (since it is in the plane of the structure) and boundary conditions of the test cases can be likened to a 1-dimensional analysis of a simply supported beam. From theoretical solutions, it is known that the deflection of the structure, due to a vertical load, at any point is inversely proportional to the second moment of area about the axis normal to the plane of the beam.
Therefore, it was concluded to take an approach to design similar to that of pin-jointed frameworks to decrease mass significantly whilst retaining the flexibility to have internal struts with a sufficiently large height to meaningfully increase the second moment of area.

8.2 FE Modelling

Careful attention is required when creating finite element (FE) models of physical parts. There will inevitably be inaccuracy that is introduced into the results as a consequence of the approximations applied throughout the FE method.

8.2.1 Simulation Setup

Upon consideration of the particular material and loading, it was elected to perform a linear static structural analysis, and the following assumptions regarding the design's mechanical properties were deemed to be valid for the particular testing case. The structural member was assumed to be homogeneous and isotropic. Furthermore, since it was designed against yielding before the limit load, the structure would be operating within its linear-elastic regime.
The boundary conditions that were enforced during the analysis were determined by examining the configuration of the test rig that was prescribed in the briefing document. The central hole through which the structure would be loaded was modelled as free. The prescribed loading at this hole was implemented using a 'bearing load' in **ANSYS**. The near supporting hole boundary condition was specified to be a 'cylindrical support' to allow slipping along an axis normal to the plane of the structure. The far supporting hole boundary condition was modelled as a remote displacement such that there could be translation along the longitudinal axis of the member and also permitting out-of-plane rotation.
Subsequently, the static analysis data was used to run an eigen buckling analysis. The implications of this decision are expounded upon in the discussion.

8.2.2 Mesh Generation and Element Selection

Second-order brick elements were used throughout the body and for every mesh base size. This choice was made to more accurately capture the actual geometry of the structure. Additionally, element topology also influences how loads are introduced to the structure at the nodes as to preserve more accuracy with this approach rather than opting for linear elements.
A full integration scheme was used by the solver to avoid the possible introduction of instabilities into the model. This would manifest in the presence of spurious deformation modes [32].

8.3 Design Methodology

8.3.1 Initial Design

Initially a 'bulkier' near symmetric design was chosen to observe how difficult it would be to predict the failure mode with a particular geometric configuration. The deflection and mass were undesirable and different starting geometries were thereon explored.

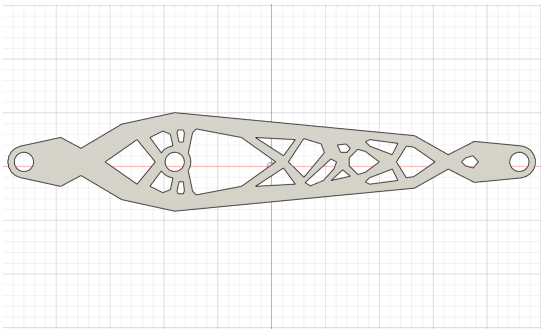


Figure 8.3: First design iteration

8.3.2 Refinement Process

Through examination of the stress and deformation results from FEA simulations, a systematic approach was followed to update the design at each iteration.

For each shape optimisation study, a vertical symmetry plane condition was enforced for the output to ensure that when the final CAD model was sketched, it would closely match the optimal geometry that was generated according to the optimisation criteria. The implementation of the structural optimisation algorithm in Fusion 360 was derived from the Simplified Isotropic Material with Penalisation (SIMP) method. The objective was to minimise the strain energy of the structure, which results in maximising stiffness. A target was set to achieve a 60% reduction in mass from a given starting geometry.

8.3.3 Final Design

The final design has a mass of 256 g and is expected to experience a maximum stress of 251 MPa at the limit load. The strut lengths were increased to increase the second moment of area which is linked to a higher bending stiffness. With a structural failure predicted at 6.5 kN, a merit index of $\approx \eta = 25,400$ was expected to be achieved.

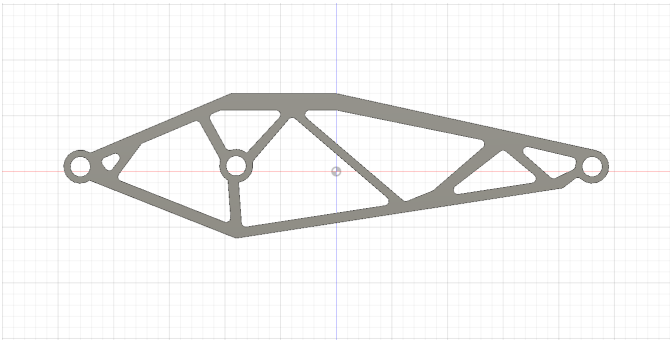


Figure 8.4: Final design iteration

8.4 Test Case Performance

8.4.1 Limit Load

The designed part is predicted to be able to withstand the limit load within the specified deflection criteria. Simulation results estimate a vertical deflection of 0.6 mm at 5kN, with a maximum (Von-Mises) stress of 251 MPa. No regions of the structure are expected to yield. It is notable that the maximum stress is close to the material's yield stress which suggests that the structure doesn't have much redundant material. The eigen buckling analysis was done to identify the various possible buckling modes. The structure is expected to experience local buckling around the top strut or at the strut connecting the loading hole to the bottom of the structure. The eigenvalues were similar in magnitude, 1.25 which is less than the 1.5 aerospace safety factor. This is illustrated in figures A.10 and A.11.

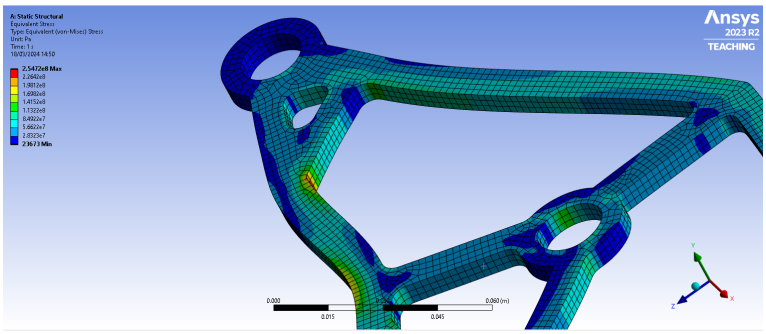


Figure 8.5: Limit load yielding

8.4.2 Ultimate Load

The failure criterion had been relaxed with regards to the maximum allowable displacement of 2mm. The results from FEA suggested a maximum deflection of 0.8mm. However, as can be seen from the mesh convergence plot for the ultimate load in figure A.9 , the lack of convergence suggests that the

structure would fail at a smaller load. Static linear simulations were ran by gradually incrementing the bearing load and monitoring the maximum stress, for comparison with the ultimate tensile strength. The analysis suggests that the primary failure mode of the structure would be yielding occurring at a load of ≈ 6.5 kN. The expected location of yielding is illustrated in figure A.13.

Discussion

9.1 Wings

Usage of the 3-D Load plot code quickly revealed a potential flaw with the design of the wing. For a constant chord-wise Lift distribution, the wing may suffer from severe tip stall.
This happens regardless of the uniform chord distribution when attempting to enforce an elliptic distribution across the wing. This was validated with an even with a more 'conventional Lift distribution' generated using a Gamma function, and manipulated to place the aerodynamic centre near the quarter chord as shown in A.4.
This is due to the aggressive taper ratio used, as explained in [33]. This could be solved by either lowering the taper ratio, which would resize the entire structure of the wing, or a similar approach could be taken to that of many similar aircraft. This would involve applying a geometric twist to the wing, hence a different tip chord shape and root chord shape. This requires a complex analysis of entire wing lift distribution, which would occur further along in the design cycle.
It would also have the effect of changing the wing box layout, but with clever design, could be less intrusive.
For the wing design, some improvements could be made to the optimiser code used output the skin thickness as a single parameter and did not consider variable skin thickness as an option. This is something that could be improved in a future iteration as varying skin thickness may be able to achieve a similar structural performance whilst reducing the mass of the wing. Additionally, more adventurous mass optimisation techniques could have been leveraged, such as topology optimisation, in order to make the structural members in the wing as mass efficient as possible. The ribs would be the most likely candidate for this in order to maximise the mass reduction. However, this could also have implications on the manufacturing complexity and thus cost. Furthermore, the implication of having a fuel box housed in the wing was not considered during this design process and some structural components may need to be slightly altered to accomodate for this.

9.2 Fuselage

Optimising the structural design of an aircraft fuselage is a difficult task that necessitates a multidisciplinary approach and close coordination among designers, engineers, and manufacturers. The fuselage design process given in Section 3 was a concise approach to the initial design iteration. However, this can be modified to enhance the design process. Due to scheduling constraints, only the initial fuselage optimisation cycle was carried out. Longer timescales allow for consideration of additional characteristics, such as aerodynamic performance and manufacturability, to enhance optimisation. A thorough investigation, including stress and vibration measurements, can confirm performance.
Further iterations can be performed with a more accurate load distribution instead of point or linearly varying masses. During flight, each load's centre of gravity may shift, which should be considered. A mixture of materials could have been selected for one structure, such as the skin. Composites could have been used in places where there is no pressurisation such as the connection with the empennage.
The optimisation can be improved by incorporating manufacturing and maintenance cost functions therefore drawing light to the sensitivity of each parameter with regards to cost as well as mass.
The stringer pitch should vary between bays in the fuselage to accommodate for the varying direct stresses and shear stresses. The variations can occur due to many reasons such as change in the diameter and only certain areas being pressurised. Curved plate buckling analysis should be conducted, as it is a more robust approach compared to flat plate buckling analysis. When designing the heavy frames, the maximum loads P, Q and T were superposed from various load cases. In reality, this would not occur,

therefore the load cases should have been analysed separately. Dynamic and static loads from the nose gear should have been accounted for which would have likely resulted in the need for a heavy frame at the nose landing gear.
Many additional structural components could have been designed such as the pressure bulkhead, nose cone, floor beams, windows and ventilation systems. Finite element analysis should be used to validate current results, with cutouts and these other secondary structures in the fuselage which were not accounted for.
Fatigue analysis for a fuselage involves an evaluation of the loads, stresses, and materials involved to predict the estimated fatigue life of the structure. This is a crucial part of designing and maintaining aeroplane fuselages. The process evaluates the structural integrity of a fuselage over its operating life, taking into consideration the varied loads and stresses encountered during flight. To enhance, specify the loads during flight, including takeoff, landing, turbulence, and pressurisation cycles. The load range is based on historical data and predicts the number of cycles the fuselage will experience over its lifetime.
Finally, one of the best methods to analyse the design is to include physical testing, even if it is conducted on a smaller scale. This will give a realistic and detailed overview which can be used in future iterations.

9.3 Empennage

The horizontal tailplane was assumed to be one force, when in reality it is a combination of elevator and horizontal stabiliser. This generally is not a bad assumption as any major mismatch between their lifts would be due to a large pitch rate enforcement. Furthermore, for most equilibrium cases they should average out quite nicely
However, despite splitting the force on the Vertical Tailplane, it is important to consider the required elevator deflection for trimming the aircraft in OEI. This changes at different velocities and Rudders have a maximum deflection angle they can apply. Compensating for larger angles would require additional plane yaw angles, which links back to the region between SLF and perfectly Banked flight. Furthermore, these angles can quite quickly grow in magnitude until the Tailplane has stalled.
Overall, the restriction on the flight envelope due to OEI is very important but complex to analyse.

9.4 Detailed Design

In a linear structural analysis, the structure's displacement field is approximated by interpolating (polynomial) shape functions and corresponding nodal displacements [32]. As a consequence, the true displacement, strain, and stress within the body aren't necessarily what the body is actually experiencing. Therefore, there exists a discrepancy between the simulated and actual loading on the structure. However, this has been adequately mitigated through a mesh convergence study, this can been seen in figure A.8.
Another source of error would be that the stress and strain fields near the member's holes couldn't have an exact model due to the lack of information about the bearings' material properties. To ensure an appropriate safety margin, more material was kept near the holes during the optimisation process to move the areas of maximum stress into the bulk of the structure and away from the supports.
A key assumption made during simulation was homogeneity of the material. Since the material would likely be sourced from a high-quality manufacturer and thereby contain relatively few impurities, allowing this assumption to be made with confidence.
The buckling resistance of the structure was not sufficiently high for the ultimate load. This issue could have potentially been fixed by minimising the strut lengths and thereby reducing their slenderness ratios. It is worth noting that since a linear buckling analysis was carried out, this didn't include nonlinear effects that the real strtucture would experience. Therefore, the buckling load cannot be precisely determined. Overall, the structure met the test objective for the limit load.

Bibliography

[1] Aerospaceweb.org — Ask Us - Propeller Torque Effect;. Available from: https://aerospaceweb.org/question/dynamics/q0015a.shtml.

[2] Fuselage — Paramount Business Jets;. Available from: https://www.paramountbusinessjets.com/aviation-terminology/fuselage.

[3] Types of Aircraft Construction;. Available from: https://www.flight-study.com/2021/01/types-of-aircraft-construction.html.

[4] Venetsanos D. AERO60002 Airframe Design; 2024. .

[5] What Is Airplane Skin?;. Available from: https://monroeaerospace.com/blog/what-is-airplane-skin/.

[6] Vidyapeeth G, Kaur R. Spars and Stringers-Function and Designing Ambri. International Journal of Aerospace and Mechanical Engineering. 2014;1(1).

[7] Tech AS. Wings - Aircraft Structures;. Https://www.aircraftsystemstech.com/p/wings-wing-configurations-wings-are.html.

[8] Aerospace M. What Are the 'Ribs' of an Airplane?;. Https://monroeaerospace.com/blog/what-are-the-ribs-of-an-airplane/.

[9] R Kaur A. Spars and Stringers- Function and Designing. International Journal of Aerospace and Mechanical Engineering; 2014.

[10] 2024/2524/7075 Aircraft Aluminum for Fuselage Skins;. Available from: https://www.aircraftaluminium.com/application/202425247075-aircraft-aluminum-for-fuselage-skins.html.

[11] Abdo b M, Sedaghati R, Elsayed MSA, Chintapalli S. The development of a preliminary structural design optimization method of an aircraft wing-box skin-stringer panels. 2009 12.

[12] Tran KL, Douthe C, Sab K, Dallot J, Davaine L. Buckling of stiffened curved panels under uniform axial compression. Journal of Constructional Steel Research. 2014 12;103:140-7. Available from: https://hal.science/hal-01073382/document.

[13] Accuris. ESDU: Home; 2024. Available from: https://www.esdu.com/cgi-bin/ps.pl?sess=unlicensed&t=gen&p=home.

[14] Wise JA. Analysis of Circular Rings for Monocoque Fuselages. Journal of the Aeronautical Sciences (Institute of the Aeronautical Sciences). 1939 09;6:460-3.

[15] Glizde N. Plotting the Flight Envelope of an Unmanned Aircraft System Air Vehicle. Transport and Aerospace Engineering. 2017 08;4. Available from: https://www.researchgate.net/publication/319872276_Plotting_the_Flight_Envelope_of_an_Unmanned_Aircraft_System_Air_Vehicle.

[16] K L Tran KSJDLD C Douthe. Buckling of stiffened curved panels under uniform axial compression. Universit´e Paris-Est; 2014.

[17] D J FARRAR MAAFRAeS. THE DESIGN OF COMPRESSION STRUCTURES FOR MINIMUM WEIGHT. Aircraft Division, Bristol Aeroplane Co. Ltd.; 1949.

[18] O Stodieck PMW J E Cooper. On the Interpretation of Bending-Torsion Coupling for Swept, Non-Homogenous Wings;.

[19] Venetsanos D. AERO60002 Wing Structural Design; 2024. .

[20] ENGINEERING DOA. AIRFRAME STRUCTURAL DESIGN. MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY; 2018.

[21] Administration FA. Chapter 13: Transition to Multiengine Airplanes. In: Airplane Flying Handbook. faa-h-8083-3c ed. U.S. Department of Transportation; 2021. Available from: https://www.faa.gov/sites/faa.gov/files/regulations_policies/handbooks_manuals/aviation/airplane_handbook/14_afh_ch13.pdf.

[22] "Horvath BL. Comparison of Aircraft Conceptual Design Weight Estimation Methods to the Flight Optimization System. NASA Langley Research Center; 2022. Available from: https://ntrs.nasa.gov/api/citations/20190000431/downloads/20190000431.pdf.

[23] Niu MCY. Airframe Structural Design;. Https://air.flyingway.com/books/Airframe-Stuctural-Design.pdf.

[24] L I. AirFrame Design Fatigue design.

[25] Vellaichamy S, Prakash BG, Brun S. Optimum design of cutouts in laminated composite structures. Computers Structures. 1990 January;37(3):241-6.

[26] Why Aircraft Have Small, Round Windows;. Available from: https://simpleflying.com/why-aircraft-have-small-round-windows/.

[27] Wang CH, Duong CN. Introduction and overview. Bonded Joints and Repairs to Composite Airframe Structures. 2016 January:3-19. Available from: https://linkinghub.elsevier.com/retrieve/pii/B9780124171534000013.

[28] Alderliesten. Introduction to Aerospace Structures and Materials.

[29] 10. Engine Mounts [Aeroengine Safety];. Available from: https://aeroenginesafety.tugraz.at/doku.php?id=10:10.

[30] Levis E. AERO60002 Aerospace Vehicle Design - Coursework Briefing Sheet; 2023. .

[31] ASM. The Aluminum Association I, editor. ASM material data sheet references. Aerospace Specification Metals Inc.; 2023. Available from: https://asm.matweb.com/search/GetReference.asp?bassnum=ma6061t6.

[32] Panesar A. AERO70010 Finite Elements; 2021. .

[33] Gudmundsson S. Chapter 9 - The Anatomy of the Wing. In: Gudmundsson S, editor. General Aviation Aircraft Design. Boston: Butterworth-Heinemann; 2014. p. 299-399. Available from: https://www.sciencedirect.com/science/article/pii/B978012397308500009X.

Appendix

A.1 Wings

A.1.1 Wing Load Distribution

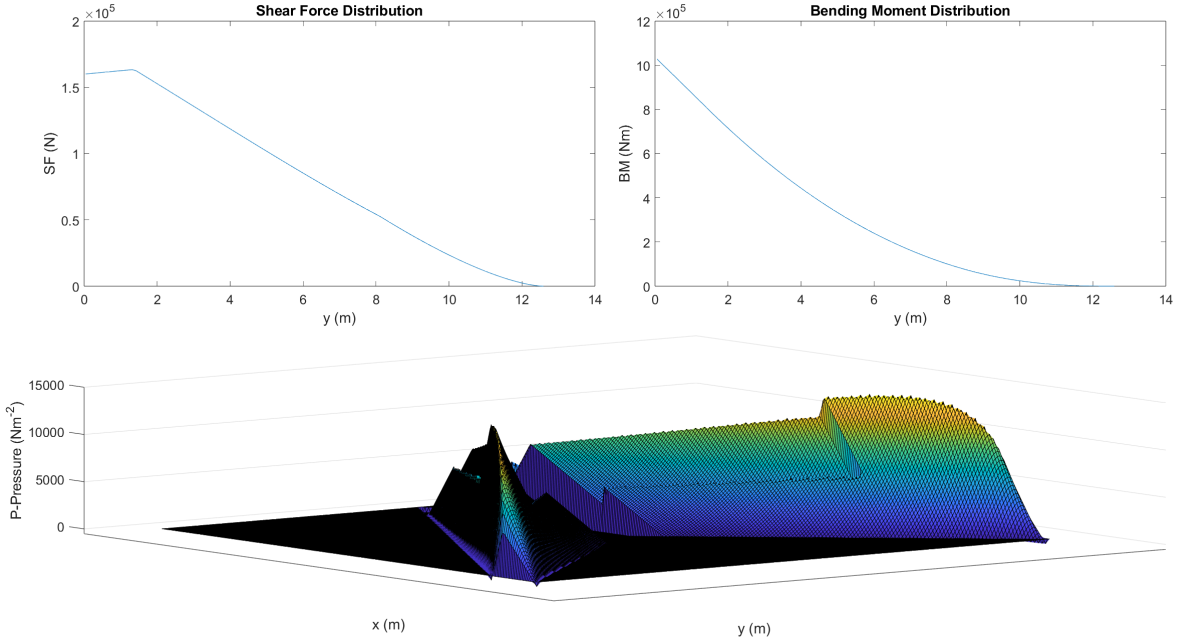


Figure A.1: Wing Load Distribution

A.1.2 V-n Diagram Code

```
%% V-n Diagram Construction Lines
clear
clc

% Constants
n_max = 2.5;
n_min = -1;
SF = 1.5;

% Remember L = nW hence n = L/W = 1/2(rho)(V^2)(Cl)/(W/Sref) For our n at
% stall or Cl = CLmax
WSRatio = 5150;
rho_0 = 1.225;
Cl = 1.6;
%FAR-25 says Design dive speed should be 1.25*Design Cruise speed or at
%least 0.05 Mach away from design Cruise Mach if compressiibility is an
%issue
M_Cruise = 0.81;
M_dive = 1.25*M_Cruise;
alt = 10000;
```

```
[~, a_cr, ~, rho_cr] = atmosisa(alt);
V_Cruise = M_Cruise * a_cr;
V_dive = M_dive * a_cr;
Vs = sqrt( (2*WSRatio) / (rho_0 * Cl) );

n = @(V) ( ( (1/2)*(rho_0)*(V^2)*(Cl) / (WSRatio) ) ); % Positive m limit
n_minus = @(V) ( ( (-1/2)*(rho_0)*(V^2)*(Cl) / (WSRatio) ) ); % Negative m limit

%% Gust Perturbations
g = 9.81;
c_bar = 3.14;
a_polar_sect = 6.039;
AR = 8;
a_polar = ((1 / a_polar_sect) + (1 / (pi*AR) ))^(-1);
Cl_alpha = 0.34;


mu = ( ((2)*(WSRatio)) / ((rho_0)*(g)*(c_bar)*(a_polar)) );
K = 0.88*mu / (5.3 + mu);

n_gustp = @(V, u) (1 + (0.5)*(1/WSRatio)*(rho_0)*(V)*(a_polar)*(K*u) );
n_gustm = @(V, u) (1 - (0.5)*(1/WSRatio)*(rho_0)*(V)*(a_polar)*(K*u) );

%% Constuction Lines
% Help build the final V-n Diagram
fplot(n, [0 150]); %120
hold on;
fplot(n_minus, [0 90]); %80
yline(n_max);
yline(n_min);
%yline(n_max*SF);
yline(n_min*SF);
xline(V_Cruise);
xline(V_dive);
line([V_Cruise V_dive], [n_min*SF 0]);
ylim([-2 4.3]);
xlim([0 343]);

ngustp1 = @(V) n_gustp(V, 20);
ngustm1 = @(V) n_gustm(V, 20);

ngustp2 = @(V) n_gustp(V, 15.2);
ngustm2 = @(V) n_gustm(V, 15.2);

ngustp3 = @(V) n_gustp(V, 7.6);
ngustm3 = @(V) n_gustm(V, 7.6);

fplot(ngustp1, [0 150], '.r');
fplot(ngustm1, [0 150], '.r');
fplot(ngustp2, [0 V_Cruise], '--b');
fplot(ngustm2, [0 V_Cruise], '--b');
fplot(ngustp3, [0 V_dive], 'pg');
fplot(ngustm3, [0 V_dive], 'pg');

yline(ngustp2(V_Cruise));
```

```
yline(ngustp2(V_Cruise)*SF);

line([0 150], [1 ngustp1(150)], 'Color', 'cyan');
line([0 150], [1 ngustm1(150)], 'Color', 'cyan');

line([150, V_Cruise], [ngustp1(150), ngustp2(V_Cruise)], 'Color', 'cyan');
line([V_Cruise, V_dive], [ngustp2(V_Cruise), ngustp3(V_dive)], 'Color', 'cyan');
line([V_dive, V_dive], [ngustp3(V_dive) ngustm3(V_dive)], 'Color', 'cyan' ...
    , 'LineWidth', 3)
line([150, V_Cruise], [ngustm1(150), ngustm2(V_Cruise)], 'Color', 'cyan');
line([V_Cruise, V_dive], [ngustm2(V_Cruise), ngustm3(V_dive)], 'Color', 'cyan');

hold off


Vm = sqrt( (n_max)*WSRatio / (Cl*rho_0*0.5) );
Vmm = sqrt( (abs(n_min))*WSRatio / (Cl*rho_0*0.5) );


VmU = sqrt( (SF*ngustp2(V_Cruise))*WSRatio / (Cl*rho_0*0.5) );
VmUm = sqrt( (abs(SF*n_min))*WSRatio / (Cl*rho_0*0.5) );

fun1 = @(V) (n(V) - ngustp1(V));
fun2 = @(V) (n(V) - ngustm1(V));

Vgp = fzero(fun1,100);
Vgm = fzero(fun2,70);


%% Final V-n Diagram
fplot(n, [0 Vm], 'b', 'LineWidth', 3);
hold on

% Gust Lines ------------------------------------------------------------------
line([Vgp 150], [ngustp1(Vgp) ngustp1(150)], 'Color', 'cyan', 'LineWidth' ...
    , 2, 'LineStyle', ':');
line([Vgm 150], [ngustm1(Vgm) ngustm1(150)], 'Color', 'cyan', 'LineWidth' ...
    , 2, 'LineStyle', ':');
line([150, V_Cruise], [ngustp1(150), ngustp2(V_Cruise)], 'Color', 'cyan' ...
    , 'LineWidth', 2, 'LineStyle', '-.');
line([V_Cruise, V_dive], [ngustp2(V_Cruise), ngustp3(V_dive)], 'Color' ...
    , 'cyan', 'LineWidth', 2, 'LineStyle', '-.');
line([V_dive, V_dive], [ngustp3(V_dive) ngustm3(V_dive)], 'Color', 'cyan' ...
    , 'LineWidth', 3);
line([150, V_Cruise], [ngustm1(150), ngustm2(V_Cruise)], 'Color', 'cyan' ...
    , 'LineWidth', 2, 'LineStyle', ':');
line([V_Cruise, V_dive], [ngustm2(V_Cruise), ngustm3(V_dive)], 'Color' ...
    , 'cyan', 'LineWidth', 2, 'LineStyle', '-.');

% Other Curves ----------------------------------------------------------------
fplot(n_minus, [0 Vmm], 'b', 'LineWidth', 3);
line([Vm V_dive], [n_max n_max], 'Color', 'b', 'LineWidth', 3);
line([Vmm V_Cruise], [n_min n_min], 'Color', 'b', 'LineWidth', 3);
line([V_Cruise V_dive], [n_min 0], 'Color', 'b', 'LineWidth', 3);
line([V_dive V_dive], [n_max 0], 'Color', 'b', 'LineWidth', 3);
```

```
line([V_dive V_dive], [0 SF*ngustm3(V_dive)], 'Color', 'r', 'LineWidth', 3);
line([V_dive V_dive], [n_max SF*ngustp2(V_Cruise)], 'Color', 'red' ...
    , 'LineWidth', 3);

fplot(n, [Vm VmU], 'r', 'LineWidth', 3);
fplot(n_minus, [Vmm VmUm], 'r', 'LineWidth', 3);
line([VmU V_dive], [SF*ngustp2(V_Cruise) SF*ngustp2(V_Cruise)], 'Color' ...
    , 'red', 'LineWidth', 3);
line([VmUm V_Cruise], [SF*n_min SF*n_min], 'Color', 'red', 'LineWidth', 3);
line([V_Cruise V_dive], [SF*n_min SF*ngustm3(V_dive)], 'Color', 'red' ...
    , 'LineWidth', 3);

line([Vs Vs], [n_minus(Vs) n(Vs)], 'Color', 'k', 'LineWidth', 1.5 ...
    , 'LineStyle', '--');
line([V_Cruise V_Cruise], [n_min n_max], 'Color', '#f80', 'LineWidth' ...
    , 3, 'LineStyle', '--');


% Custom legend ---------------------------------------------------------
qw{1} = plot(nan, 'b', LineWidth=3);
qw{2} = plot(nan, 'c:', LineWidth=2);
qw{3} = plot(nan, 'c-.', LineWidth=2);
qw{4} = plot(nan, 'r', LineWidth=3);
qw{5} = plot(nan, '--', LineWidth=3, Color="#f80");
qw{6} = plot(nan, 'k--', LineWidth=1.5);
legend([qw{:}], {'Limit Load Envelope','Non-altering Gust Load' ...
    ,'Altering Gust Load', 'Ultimate Load Envelope', 'M_{Cruise}', 'Stall'} ...
    , 'location', 'northwest')
hold off;

xlabel('Equivalent Airspeed-(EAS) (ms^{-1})');
ylabel('Load Factor n');
fontsize(gcf, scale=1.3)
% Vert landing velocity of 10ft/sec as calcuation yeilded 14.15ft/sec
```

A.1.3 V-n Construction Lines

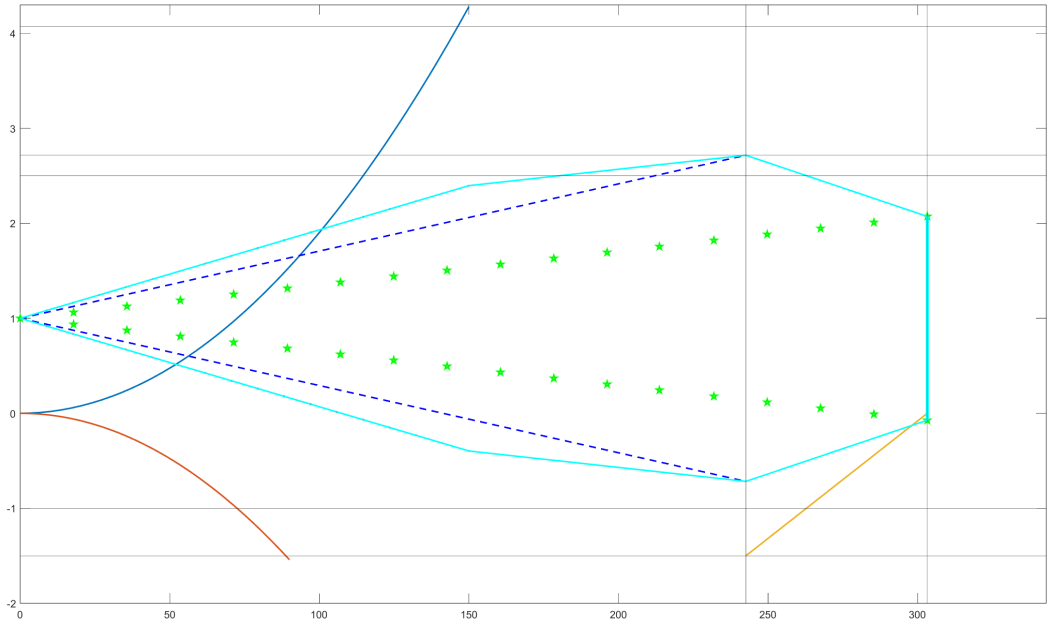


Figure A.2: Flight Envelope

A.1.4 Wings 3-D Load Distribution Code

```
%% Wing Loading
clear
clc

%Set some constants here I guess
W0 = 79.15*5150;
n = 1;
%L0 = n * W0/19.75; %This is for no fuselage width %W0/6.4; %approx this
L0 = n * W0/17; %/17.6; % For width of 1
%L0 = 20020; %910;
b = 25.16;
Chord_root = 5.16;
Chord_tip = 1.13;
Sweep_c4 = 30.6;
fw = 2.69/2; %1; % Fuselage width
Wing_weight = 0.1*W0; %Both wings sun to approx 10% if MTOW
spar_f = 0.15;
spar_b = 0.65;
tank_length = 6.74;
Fuel_tanks = [spar_f spar_b -fw-tank_length -fw; %Positions
              spar_f spar_b  fw  fw+tank_length];
% 15% chord to 65% for spars hence fuel tanks.
Tank_density = 9.81*720*5740/1000;
gw = 1.74; % Displacement from centreline.
gLength = 0.75; % Size of square side length to distribute gear force across.
g_n = 0;
```

```
Gear_leg = [0.5  0.5 -gw-gLength -gw; % Pos similar format to fuel tanks
            0.5  0.5   gw gw+gLength];
G_Area = gLength^2;
Gear_load_factor = g_n/2/G_Area; % Prep for distributed force about gLength.

%% Lift Distribution
% Span-wise Lift distribution
L_span = @(y) (Lift_Span(y, fw, L0, b));
% Chord-wise Lift distribution
L_chord = @(x, chord) (Lift_Chord(x, chord));
%Normalized to integrate to a total value of 1 per section

%% Wing Shape
Wing_chord = @(y) (Chord_root * (1 - (1 - (Chord_tip/Chord_root)) ...
    *(abs(y)/(b/2) ) ) );
Wing_Sweep = @(y) (-abs(y)*tand(Sweep_c4));

Wing_LE = @(y) Wing_Sweep(y) + 0.25*Wing_chord(y);
Wing_TE = @(y) Wing_Sweep(y) - 0.75*Wing_chord(y);

% Correction for tank density in 3-D
Tank_density = Tank_density/(0.5*((spar_b-spar_f)*Wing_chord(fw) ...
    + (spar_b-spar_f)*Wing_chord(fw+tank_length))*tank_length);

%% Overal Lift Distribution about Wing domain
Lift = @(x, y) ( L_span(y) * L_chord(Wing_LE(y) - x, Wing_chord(y)) );


%% Plotting
%%% Fig1
fig1 = figure(1);
tcl = tiledlayout(2,2,'TileSpacing','compact');
nexttile
% Span-wise Lift Distribution
fplot(L_span, [0 b/2], LineWidth=3)
title("Span-wise Lift Distribution")
ylabel("L_0 (Nm^{-1})")
xlabel("y (m)")

nexttile
% Chord-wise Lift Distribution (at root)
L_chord_root = @(x) L_chord(x, Wing_chord(0));
fplot(L_chord_root, [0 Wing_chord(0)], LineWidth=3)

title("Normalised Chord-wise Lift Distribution")
ylabel("L_0")
xlabel("x (m)")

nexttile([1 2])
% Wing Shape

fplot(Wing_LE, [-b/2 b/2], LineWidth=3, Color='b')
hold on
fplot(Wing_TE, [-b/2 b/2], LineWidth=3, Color='b')
% Quarter chord sweep
```

```
fplot(Wing_Sweep, [-b/2 b/2], LineStyle="--", LineWidth=2, Color='b')
line([b/2 b/2], [Wing_LE(b/2) Wing_TE(b/2)], 'LineWidth', 3, 'Color' ...
    , 'b')
line([-b/2 -b/2], [Wing_LE(-b/2) Wing_TE(-b/2)], 'LineWidth', 3, ...
    'Color', 'b')

% Fuselage
line([-fw -fw], [Wing_LE(0) Wing_TE(-b/2)], 'LineWidth', 3, 'Color' ...
    , 'black', 'LineStyle', '-.')
line([fw fw], [Wing_LE(0) Wing_TE(-b/2)], 'LineWidth', 3, 'Color' ...
    , 'black', 'LineStyle', '-.')

% Fuel tanks
fplot(@(y) Wing_LE(y) - spar_f*Wing_chord(y), [-fw-tank_length, -fw] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_f*Wing_chord(y), [fw, fw+tank_length] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_b*Wing_chord(y), [-fw-tank_length, -fw] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_b*Wing_chord(y), [fw, fw+tank_length] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')

line([-fw-tank_length -fw-tank_length], ...
    [Wing_LE(-fw-tank_length)-spar_f*Wing_chord(-fw-tank_length) ...
    , Wing_LE(-fw-tank_length)-spar_b*Wing_chord(-fw-tank_length)] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')

line([fw+tank_length fw+tank_length], ...
    [Wing_LE(fw+tank_length)-spar_f*Wing_chord(fw+tank_length) ...
    , Wing_LE(fw+tank_length)-spar_b*Wing_chord(fw+tank_length)] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')

%Gear legs
rectangle('Position', ...
    [+gw-gLength/2 ...
    , Wing_LE(-gw)-Gear_leg(1, 1)*Wing_chord(-gw)-gLength/2 ...
    , gLength, gLength], LineWidth=3, LineStyle='-' ...
    , FaceColor='#808080', Curvature=1)
rectangle('Position',[-gw-gLength/2 ...
    , Wing_LE(-gw)-Gear_leg(1, 1)*Wing_chord(-gw)-gLength/2 ...
    , gLength, gLength], LineWidth=3, LineStyle='-' ...
    , FaceColor='#808080', Curvature=1)

% Custom legend
qw{1} = plot(nan, 'b', LineWidth=3);
qw{2} = plot(nan, 'b--', LineWidth=2);
qw{3} = plot(nan, 'k-.', LineWidth=3);
qw{4} = plot(nan, ':', LineWidth=3, Color="#A2142F");
legend([qw{:}], {'Wing','Quater-Chord','Fuselage', 'Fuel Tanks'} ...
    , 'location', 'best')
hold off

title("Wing Geometry")
xlabel("y (m)")
ylabel("x (m)")
```

```
fontsize(gcf, scale=1.3)

%% Sectional Lift on Wing Shape
%%% Fig 2
fig2 = figure(2);

xmax = Wing_LE(0);
if Wing_TE(b/2) <= Wing_TE(0)
    xmin = Wing_TE(b/2);
else
    xmin = Wing_TE(0);
end

fsurf(Lift, [xmin xmax -b/2 b/2])

%% Approximate Integral for Total Lift
X = xmin:0.02:xmax;
Y = linspace(-b/2, b/2, length(X));
Z = zeros(length(X), length(Y)); % Wing Lift
Z2 = zeros(length(X), length(Y)); % Wing Weight
Z3 = zeros(length(X), length(Y)); % Fuel tanks
Z4 = zeros(length(X), length(Y)); % Gear leg


for k = 1:size(Gear_leg)
    g_pos_y1 = Wing_LE(-gw) - Gear_leg(k, 1)*Wing_chord(-gw);
    g_pos_y2 = Wing_LE(-gw) - Gear_leg(k, 2)*Wing_chord(-gw);
    Gear_leg(k, :) = [g_pos_y1+gLength/2, g_pos_y2-gLength/2 ...
        , Gear_leg(k, 3:4)];
end

clear("g_pos_y1", "g_pos_y2")

for i = 1:length(X)
    for j = 1:length(Y)
        Z(i, j) = Lift(X(i), Y(j));
        Z2(i, j) = inStruct(X(i), Y(j), Wing_LE(Y(j)), Wing_TE(Y(j)) ...
            , -b/2, b/2);

        for k = 1:size(Fuel_tanks, 1)
            Z3(i, j) = Z3(i, j) + abs(n)*Tank_density*inStruct(X(i) ...
                , Y(j) ...
                , Wing_LE(Y(j)) - Fuel_tanks(k, 1)*Wing_chord(Y(j)) ...
                , Wing_LE(Y(j)) - Fuel_tanks(k, 2)*Wing_chord(Y(j)) ...
                , Fuel_tanks(k, 3), Fuel_tanks(k, 4));

            Z4(i, j) = Z4(i, j) + inStruct(X(i), Y(j), Gear_leg(k, 1) ...
                , Gear_leg(k, 2), Gear_leg(k, 3), Gear_leg(k, 4));
        end % Only works if Z4 has the same rows as Z3.
    end% Should be seperated for different analysis.
end

%% Inertia loads
Wing_density = Wing_weight / (0.5*b*(Chord_root+Chord_tip));
```

```
Z2 = abs(n)*Wing_density*Z2;
%sum(Z3,"all")
%Z3 = Tank_density*Z3; %Already done in loop
Z4 = Gear_load_factor*W0*Z4;

Total = trapz(X, trapz(Y, Z, 2)) % Check if Lift = n*W0
Total2 = trapz(X, trapz(Y, Z2, 2)) % Check if weight of wing hence 10% W0
Total3 = trapz(X, trapz(Y, Z3, 2)) % Check Fuel Weight
Total4 = trapz(X, trapz(Y, Z4, 2)) % Check gear force

Z_Sum = Z - Z2 - Z3 + Z4; % Used to comment out Z3
Z_Sum_Span = trapz(X, Z_Sum, 1);

size_Y = floor(length(Y)/2);
rem_Y = mod(length(Y), 2);
DSF = zeros(size_Y, 1);
%Z_Sum_Span(2)
for i = 1:size_Y
    if i == size_Y
        DSF(i) = 0.5*(Z_Sum_Span(size_Y + rem_Y + i))*(Y(end)-Y(end-1));
    else
        DSF(i) = ( 0.5*(Z_Sum_Span(size_Y + rem_Y + i) ...
            + Z_Sum_Span(size_Y + rem_Y + i+1)) ...
            * (Y(size_Y + rem_Y + i+1)-Y(size_Y + rem_Y + i)) );
    end
end

SF = flip(cumsum(flip(DSF(1:end))));

DBM = zeros(length(SF), 1);
for i = 1:length(SF)
    if i == length(SF)
        DBM(i) = 0.5*(SF(i))*(Y(end)-Y(end-1));
    else
        DBM(i) = ( 0.5*(SF(i) + SF(i+1)) * (Y(size_Y + rem_Y + i+1) ...
            -Y(size_Y + rem_Y + i)) );
    end
end

BM = flip(cumsum(flip(DBM)));

%Total2 = trapz(X, Z, 1)
%arrayfun(L_span, Y) %Are equivalent!

%% Plotting
fig3 = figure(3);
tcl2 = tiledlayout(2,2,'TileSpacing','compact');
%nexttile
nexttile([2 1])
%length(Y(length(Y)/2:end))
plot(Y(end/2 + 1:end), SF)

title("Shear Force Distribution")
xlabel("y (m)")
ylabel("SF (N)")
```

```
%nexttile
nexttile([2 1])
plot(Y(end/2 + 1:end), BM)

title("Bending Moment Distribution")
xlabel("y (m)")
ylabel("BM (Nm)")

fontsize(gcf, scale=1.3)
%nexttile([1, 2])
%%surf(X, Y, Z2)
fig4 = figure(4);
surf(Y, X, Z_Sum)
xlabel("y (m)")
ylabel("x (m)")
zlabel("P-Pressure (Nm^{-2})")
set(gca, 'XTick', [], 'YTick', [])

fontsize(gcf, scale=1.3)

%save("plot" + num2str(floor(n)) + num2str(floor(g_n)), 'Y', 'SF', 'BM')


%% Function definitions

% Control case to ensure Lift always sums to 1 chordwise. #Used this to
% Calculate loads.

% function [L] = Lift_Chord(x, chord)
%     if x/chord >= 0 && x/chord <= 1
%         L = 1/chord;
%     else
%         L = 0;
%     end
% end

% Simple case: 1. Usage of a triangular distribution maximising at quarter
% chord. However, this does not create a quarter chord centre of pressure.
% Regardless, this was when tip stall was first observed.

% function [L] = Lift_Chord(x, chord)
%     if x/chord <= 0.25 && x/chord >= 0
%         L = 8/(chord^2) * x;
%     elseif x/chord > 0.25 && x/chord <= 1
%         L = 8/(chord^2) * (1/3)*(chord - (x));
%     else
%         L = 0;
%     end
% end



% More intuitive case: 2. Usage of the gamma probability function to model
% a distribution that appears more familiar with an aerofoils. This was
```

```
% also tweaked to ensure the centre of pressure lies around the quarter
% chord. This was to confirm for constant chordwise pressure distribution,
% the aircraft may experience tipstall due to the high taper ratio.

function [L] = Lift_Chord(x, chord)
    beta = 1;
    alpha = 1.8;
    if x/chord <= 1 && x/chord >= 0
        L = 7/chord*(beta^alpha /(gamma(alpha))) * (7/chord*x)^(alpha-1) ...
        *exp(-beta*(7/chord*x));
    else
        L = 0;
    end
end

% Corrected Lift Span to compensate fuselage.
function [L] = Lift_Span(y, f_width, L0, b)
    if abs(y) <= f_width
        L = 0;
    else
        L = L0 * sqrt(1 - (y / (b/2)).^2 );
    end
end

% 3-D check if within boundaries.
function [D] = inStruct(x, y, c1, c2, b1, b2)
    if y > b1 && y < b2
        if x < c1 && x > c2
            D = 1;
        else
            D = 0;
        end
    else
        D = 0;
    end
end
```

A.1.5 More Traditional Wing Lift Layout

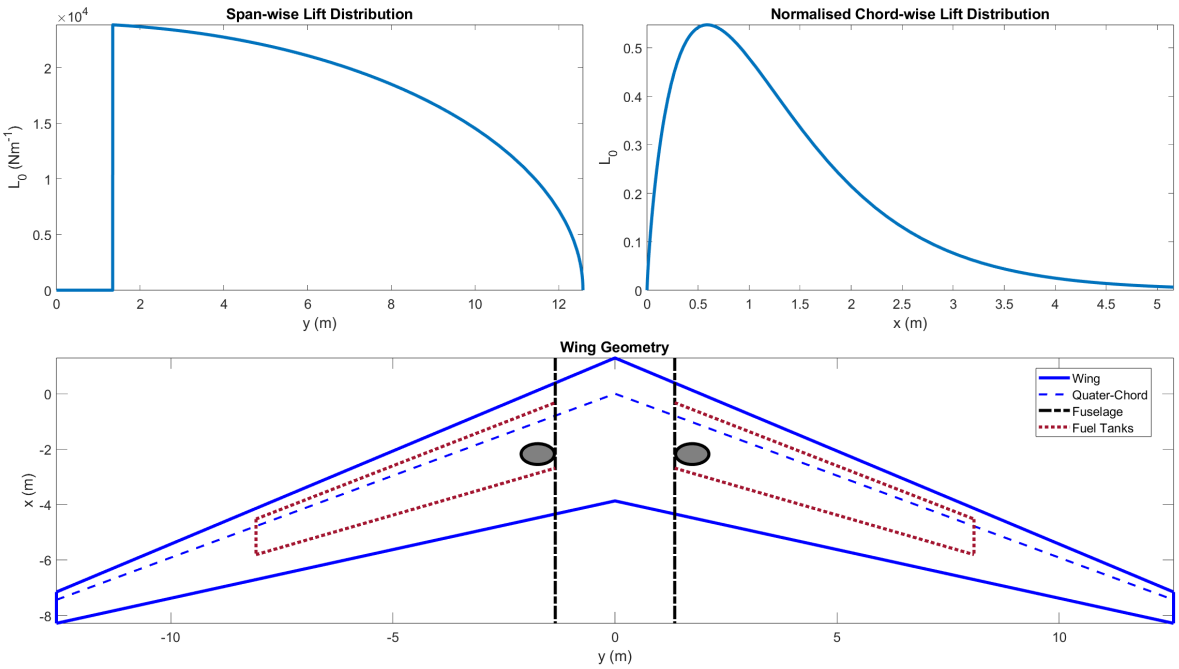


Figure A.3: Wing layout

A.1.6 Pure 3-D Lift

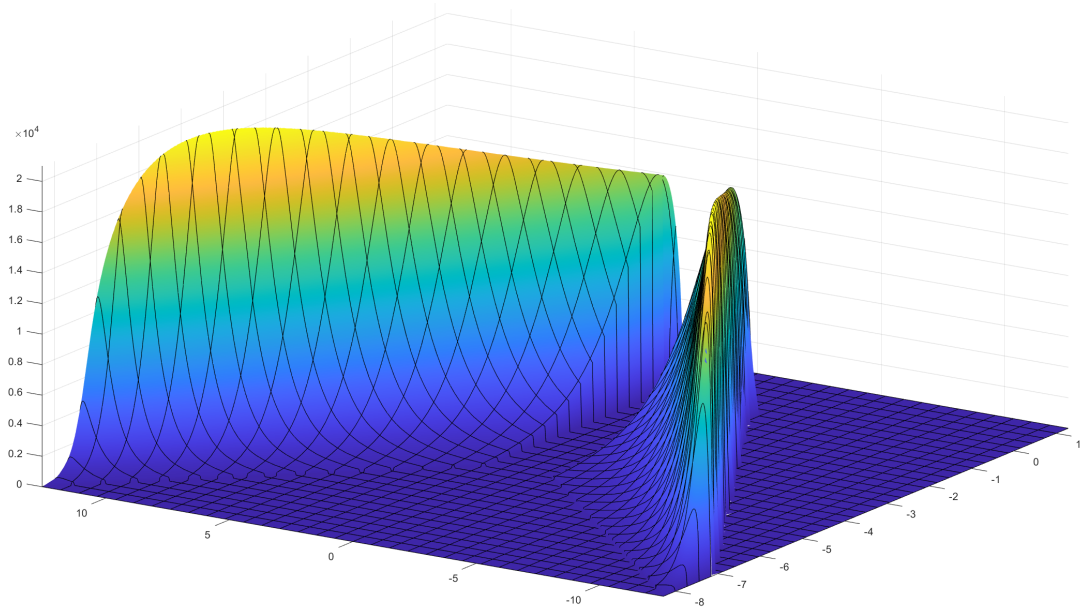


Figure A.4: Lift layout

A.1.7 Summed Load 3-D

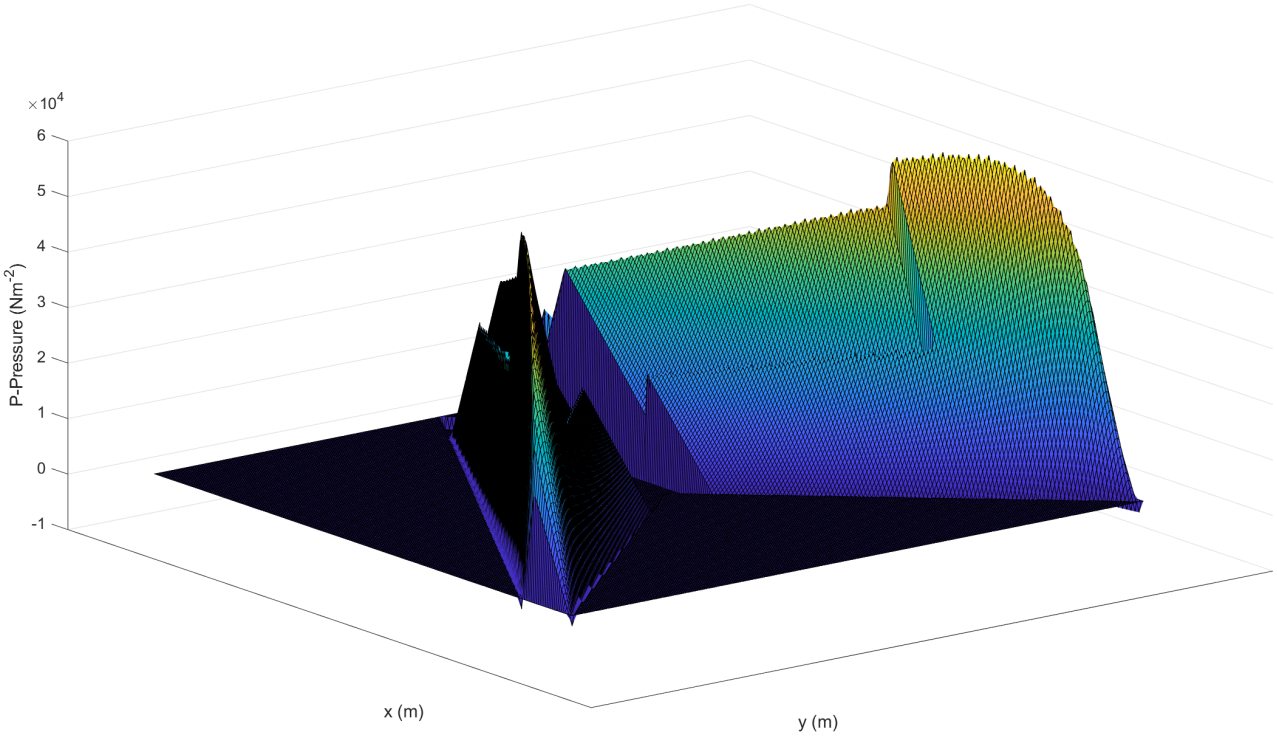


Figure A.5: Load layout

A.1.8 Wings 2-D Load Code

```
%% Wing Loading
clear
clc

%Set some constants here
W0 = 79.15*5150;
n = 1;
L0 = n * W0/17;
b = 25.16;
Chord_root = 5.16;
Chord_tip = 1.13;
Sweep_c4 = 30.6;
fw = 2.69/2; %1; % Fuselage width
Wing_weight = 0.1*W0; %Both wings sum to approx 10% if MTOW
%%% Chord ratios
spar_f = 0.15;
spar_b = 0.65;
chord_ac = 0.25;
chord_cg = 0.5;
chord_ge = 0.5;
%%% Fuel tank and Gear values
tank_length = 6.74;
Fuel_tanks = [spar_f spar_b -fw-tank_length -fw; %Positions
              spar_f spar_b  fw  fw+tank_length];
Tank_density = 9.81*720*5840;
gw = 1.74; % Displacement from centreline.
gLength = 0.75; % Size of square side length to distribute gear force across
```

```
g_n = 0;

Gear_leg = [chord_ge chord_ge -gw-gLength -gw; % Same format to fuel tanks
            chord_ge chord_ge  gw gw+gLength];

G_Area = gLength;
Gear_load_factor = g_n/2/G_Area; % Prep for distributed force about gLength.

%% Lift Distribution
%Span-wise Lift distribution
minY = -b/2;
maxY = b/2;
Ny = 500;

L_span = @(y) (Lift_Span(y, fw, L0, b));
%% Wing Shape
Wing_chord = @(y) (Chord_root ...
    * (1 - (1 - (Chord_tip/Chord_root))*(abs(y)/(b/2) ) ) );
Wing_Sweep = @(y) (-abs(y)*tand(Sweep_c4));

Wing_LE = @(y) Wing_Sweep(y) + 0.25*Wing_chord(y);
Wing_TE = @(y) Wing_Sweep(y) - 0.75*Wing_chord(y);

% Correction for tank density in 2-D

Tank_density = Tank_density/tank_length/1000;
%% Plotting
%%% Fig1
fig1 = figure(1);
tcl = tiledlayout(2,2,'TileSpacing','compact');
nexttile
% Span-wise Lift Distribution
fplot(L_span, [0 b/2], LineWidth=3)
title("Span-wise Lift Distribution")
ylabel("L_0 (Nm^{-1})")
xlabel("y (m)")

nexttile

% Wing Spar and flexiral axis
fplot(Wing_LE, [0 b/2], LineWidth=3, Color='b')
hold on
fplot(Wing_TE, [0 b/2], LineWidth=3, Color='b')
line([b/2 b/2], [Wing_LE(b/2) Wing_TE(b/2)], 'LineWidth', 3, 'Color', 'b')
fplot(@(y) Wing_LE(y) - spar_f*Wing_chord(y), [0, b/2], 'LineWidth', 3 ...
    , 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_b*Wing_chord(y), [0, b/2], 'LineWidth', 3 ...
    , 'Color', 'k', 'LineStyle', '-.')
fplot(@(y) Wing_LE(y) - 0.5*(spar_f+spar_b)*Wing_chord(y), [0, b/2] ...
    , 'LineWidth', 2, 'Color', 'b', 'LineStyle', '--')

qw{1} = plot(nan, 'b', LineWidth=3);
qw{2} = plot(nan, 'b--', LineWidth=2);
qw{3} = plot(nan, 'k-.', LineWidth=3);
qw{4} = plot(nan, ':', LineWidth=3, Color="#A2142F");
```

```
legend([qw{:}], {'Wing','Flexural Axis','Rear Spar', 'Front Spar'} ...
    , 'location', 'best')
hold off

title("Wing Geometry")
xlabel("y (m)")
ylabel("x (m)")

nexttile([1 2])

%%% Wing Shape

fplot(Wing_LE, [-b/2 b/2], LineWidth=3, Color='b')
hold on
fplot(Wing_TE, [-b/2 b/2], LineWidth=3, Color='b')
% Quarter chord sweep
fplot(Wing_Sweep, [-b/2 b/2], LineStyle="--", LineWidth=2, Color='b')
line([b/2 b/2], [Wing_LE(b/2) Wing_TE(b/2)], 'LineWidth', 3, 'Color', 'b')
line([-b/2 -b/2], [Wing_LE(-b/2) Wing_TE(-b/2)], 'LineWidth', 3 ...
    , 'Color', 'b')

% Fuselage
line([-fw -fw], [Wing_LE(0) Wing_TE(-b/2)], 'LineWidth', 3, 'Color' ...
    , 'black', 'LineStyle', '-.')
line([fw fw], [Wing_LE(0) Wing_TE(-b/2)], 'LineWidth', 3, 'Color' ...
    , 'black', 'LineStyle', '-.')

% Fuel tanks
fplot(@(y) Wing_LE(y) - spar_f*Wing_chord(y), [-fw-tank_length, -fw] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_f*Wing_chord(y), [fw, fw+tank_length] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_b*Wing_chord(y), [-fw-tank_length, -fw] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_b*Wing_chord(y), [fw, fw+tank_length] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')

line([-fw-tank_length -fw-tank_length], ...
    [Wing_LE(-fw-tank_length)-spar_f*Wing_chord(-fw-tank_length) ...
    , Wing_LE(-fw-tank_length)-spar_b*Wing_chord(-fw-tank_length)] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')

line([fw+tank_length fw+tank_length], ...
    [Wing_LE(fw+tank_length)-spar_f*Wing_chord(fw+tank_length) ...
    , Wing_LE(fw+tank_length)-spar_b*Wing_chord(fw+tank_length)] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')

%Gear legs
rectangle('Position',[+gw-gLength/2 ...
    , Wing_LE(-gw)-Gear_leg(1, 1)*Wing_chord(-gw)-gLength/2 ...
    , gLength, gLength], LineWidth=3, LineStyle='-' ...
    , FaceColor='#808080', Curvature=1)
rectangle('Position',[-gw-gLength/2, ...
    Wing_LE(-gw)-Gear_leg(1, 1)*Wing_chord(-gw)-gLength/2 ...
    , gLength, gLength], LineWidth=3, LineStyle='-' ...
```

```
    , FaceColor='#808080', Curvature=1)

% Custom legend
qw{1} = plot(nan, 'b', LineWidth=3);
qw{2} = plot(nan, 'b--', LineWidth=2);
qw{3} = plot(nan, 'k-.', LineWidth=3);
qw{4} = plot(nan, ':', LineWidth=3, Color="#A2142F");
legend([qw{:}], {'Wing','Quater-Chord','Fuselage', 'Fuel Tanks'} ...
    , 'location', 'best')
hold off

title("Wing Geometry")
xlabel("y (m)")
ylabel("x (m)")

fontsize(gcf, scale=1.3)

Gear_leg = [-gw-gLength -gw;
             gw gw+gLength];
Fuel_tanks = [-fw-tank_length -fw;
               fw fw+tank_length];

%% 2-D
Y = linspace(0, b/2, Ny);
Y = [-flip(Y(2:end)), Y];
Z = zeros(1, length(Y));  % Wing Lift
Z2 = zeros(1, length(Y)); % Wing Weight
Z3 = zeros(1, length(Y)); % Fuel tanks
Z4 = zeros(1, length(Y)); % Gear leg

Wing_area = arrayfun(@(y) (Wing_chord(y) / (0.5*(Chord_root+Chord_tip)))^2, Y);
Wing_density = Wing_weight / trapz(Y, Wing_area);


%%% Find moment arms for Torque Calculations
p_lift = arrayfun(@(y) (0.5*(spar_f+spar_b) - chord_ac)*Wing_chord(y), Y);
% From Aerodynamic centre
p_weight = arrayfun(@(y) (0.5*(spar_f+spar_b) - chord_cg)*Wing_chord(y) , Y);
% From Wing CG
p_gear = arrayfun(@(y) (0.5*(spar_f+spar_b) - chord_ge)*Wing_chord(y), Y);
% From Gear strut placement


for i = 1:length(Y)
    Z(i) = L_span(Y(i));
    Z2(i) = Wing_density*Wing_area(i)*inStruct2D(Y(i), minY, maxY);
    for k = 1:size(Fuel_tanks, 1)
        Z3(i) = Z3(i) + abs(n)*Tank_density*inStruct2D(Y(i) ...
            , Fuel_tanks(k, 1), Fuel_tanks(k, 2));
        Z4(i) = Z4(i) + inStruct2D(Y(i), Gear_leg(k, 1), Gear_leg(k, 2));
    end
end


Z2 = abs(n)*Z2;
```

```
Z4 = Gear_load_factor*W0*Z4;

Total_Lift = trapz(Y, Z);              % Check if Lift = n*W0
Total_Wing_Weight = trapz(Y, Z2);      % Check Weight of Wing
Total_Fuel_Weight = trapz(Y, Z3);      % Check Fuel weight
Total_Gear_Force = trapz(Y, Z4);       % Check Gear Resultant force

Z_Sum_Span = Z - Z2 - Z3 + Z4; % Used to comment out Z3
Z_Tor_Span = p_lift.*Z - p_weight.*Z2 + p_gear.*Z4; %- dM_0;
% Total Sectional Torque


%% Quick test
%fplot(@(y) (Wing_chord(y) / (0.5*(Chord_root+Chord_tip)))^2, [minY, maxY])

%% Inertia loads


size_Y = floor(length(Y)/2);
rem_Y = mod(length(Y), 2);
DSF = zeros(Ny, 1);
DT = zeros(Ny, 1);

for i = 1:Ny
    if i == Ny
        DSF(i) = 0.5*(Z_Sum_Span(size_Y + i))*(Y(end)-Y(end-1));
        DT(i) = 0.5*(Z_Tor_Span(size_Y + i))*(Y(end)-Y(end-1));
    else
        DSF(i) = ( 0.5*(Z_Sum_Span(size_Y + i) + Z_Sum_Span(size_Y + i+1)) ...
            * (Y(size_Y + i+1)-Y(size_Y + i)) );
        DT(i) = ( 0.5*(Z_Tor_Span(size_Y + i) + Z_Tor_Span(size_Y + i+1)) ...
            * (Y(size_Y + i+1)-Y(size_Y + i)) );
    end
end

SF = flip(cumsum(flip(DSF(1:end))));
T = flip(cumsum(flip(DT)));

DBM = zeros(length(SF), 1);
for i = 1:Ny
    if i == Ny
        DBM(i) = 0.5*(SF(i))*(Y(end)-Y(end-1));
    else
        DBM(i) = ( 0.5*(SF(i) + SF(i+1)) * (Y(size_Y + i+1)-Y(size_Y + i)) );
    end
end

BM = flip(cumsum(flip(DBM)));

%Total2 = trapz(X, Z, 1)
%arrayfun(L_span, Y) %Are equivalent!

%% Plotting
fig3 = figure(3);
tcl2 = tiledlayout(2,2,'TileSpacing','compact');
```

```
nexttile
plot(Y(Ny:end), SF)

title("Shear Force Distribution")
xlabel("y (m)")
ylabel("SF (N)")

nexttile
plot(Y(floor(end/2)+1:end), BM)

title("Bending Moment Distribution")
xlabel("y (m)")
ylabel("BM (Nm)")

nexttile
plot(Y(Ny:end), T)

title("Torsion Moment Distribution")
xlabel("y (m)")
ylabel("TM (Nm)")

nexttile
plot(Y(Ny:end), DSF)

title("Force Distribution")
xlabel("y (m)")
ylabel("DF (N)")

fontsize(gcf, scale=1.3)

% save("plot" + num2str(floor(n)) + num2str(floor(g_n)), 'Y', 'SF', 'BM' ...
% , 'T', 'DSF')


%% Function definitions

% Control case to ensure Lift always sums to 1 chordwise. #Used this to
% Calculate loads.
function [L] = Lift_Chord(x, chord)
    if x/chord >= 0 && x/chord <= 1
        L = 1/chord;
    else
        L = 0;
    end
end

% Simple case: 1. Usage of a triangular distribution maximising at quarter
% chord. However, this does not create a quarter chord centre of pressure.
% Regardless, this was when tip stall was first observed.

% function [L] = Lift_Chord(x, chord)
%     if x/chord <= 0.25 && x/chord >= 0
%         L = 8/(chord^2) * x;
%     elseif x/chord > 0.25 && x/chord <= 1
%         L = 8/(chord^2) * (1/3)*(chord - (x));
```

```
%     else
%         L = 0;
%     end
% end



% More intuitive case: 2. Usage of the gamma probability function to model
% a distribution that appears more familiar with an aerofoils. This was
% also tweaked to ensure the centre of pressure lies around the quarter
% chord. This was to confirm for constant chordwise pressure distribution,
% the aircraft may experience tipstall due to the high taper ratio.

% function [L] = Lift_Chord(x, chord)
%     beta = 1;
%     alpha = 1.8;
%     if x/chord <= 1 && x/chord >= 0
%         L = 7/chord*(beta^alpha /(gamma(alpha))) ...
%           * (7/chord*x)^(alpha-1)*exp(-beta*(7/chord*x));
%     else
%         L = 0;
%     end
% end

% Corrected Lift Span to compensate fuselage.
function [L] = Lift_Span(y, f_width, L0, b)
    if abs(y) <= f_width
        L = 0;
    else
        L = L0 * sqrt(1 - (y / (b/2)).^2 );
    end
end

% 3-D check if within boundaries.
function [D] = inStruct(x, y, c1, c2, b1, b2)
    if y > b1 && y < b2
        if x < c1 && x > c2
            D = 1;
        else
            D = 0;
        end
    else
        D = 0;
    end
end

% 2-D check if within boundaries.
function [D] = inStruct2D(y, b1, b2)
    if y > b1 && y < b2
        D = 1;
    else
        D = 0;
    end
end
```

A.1.9 Wings Plotting Code

```
%% Wing Loading Plot all on one fig
clear
clc

case1 = load("plot10.mat"); % SLF (control line)
case2 = load("plot-20.mat"); % -1.5 case
case3 = load("plot13.mat"); % Landing gear inpact 3, lift 1
case4 = load("plot40.mat"); % 4.07 case

tcl = tiledlayout(2,2,'TileSpacing','compact');

nexttile
plot(case1.Y(floor(end/2) + 1:end), case1.DSF, "--", LineWidth=2)
hold on
plot(case4.Y(floor(end/2) + 1:end), case4.DSF, LineWidth=2)
plot(case2.Y(floor(end/2) + 1:end), case2.DSF, LineWidth=2)
plot(case3.Y(floor(end/2) + 1:end), case3.DSF, LineWidth=2)
hold off

title("Total Force Distribution")
xlabel("y (m)")
ylabel("F (N)")
legend({"n = 1",  "n = 4.07", "n = -1.5", "n_{gear} = 3"})

nexttile
%length(Y(length(Y)/2:end))
plot(case1.Y(floor(end/2) + 1:end), case1.SF, "--", LineWidth=2)
hold on
plot(case4.Y(floor(end/2) + 1:end), case4.SF, LineWidth=2)
plot(case2.Y(floor(end/2) + 1:end), case2.SF, LineWidth=2)
plot(case3.Y(floor(end/2) + 1:end), case3.SF, LineWidth=2)
hold off

title("Shear Force Distribution")
xlabel("y (m)")
ylabel("SF (N)")
legend({"n = 1",  "n = 4.07", "n = -1.5", "n_{gear} = 3"})

nexttile
plot(case1.Y(floor(end/2) + 1:end), case1.BM, "--", LineWidth=2)
hold on
plot(case4.Y(floor(end/2) + 1:end), case4.BM, LineWidth=2)
plot(case2.Y(floor(end/2) + 1:end), case2.BM, LineWidth=2)
plot(case3.Y(floor(end/2) + 1:end), case3.BM, LineWidth=2)
hold off

title("Bending Moment Distribution")
xlabel("y (m)")
ylabel("BM (Nm)")
legend({"n = 1",  "n = 4.07", "n = -1.5", "n_{gear} = 3"})

nexttile
plot(case1.Y(floor(end/2) + 1:end), case1.T, "--", LineWidth=2)
```

```
hold on
plot(case4.Y(floor(end/2) + 1:end), case4.T, LineWidth=2)
plot(case2.Y(floor(end/2) + 1:end), case2.T, LineWidth=2)
plot(case3.Y(floor(end/2) + 1:end), case3.T, LineWidth=2)
hold off

title("Torsion Moment Distribution")
xlabel("y (m)")
ylabel("TM (Nm)")
legend({"n = 1",  "n = 4.07", "n = -1.5", "n_{gear} = 3"})

fontsize(gcf, scale=1.5)
```

A.1.10 Wings Optimisation and Plotting Code

```
clear
close all
clc

% SI units used

% place ribs at known locations, treat space between locations as "section"
% and iterate through placing different of ribs in order to withstand
% buckling

% find industry standard rib thickness, mass = density x thickness x CSA since our
% will be aerofoil shape, idealise our aerofoil as kite, chord x
% characteristic height =  CSA, look into indsutry scale factor for
% manufacturing optimisation, ribs w holes etc

%% Rib optimsiation
close all
clc

% place ribs at known locations, treat space between locations as "section"
% and iterate through placing different of ribs in order to withstand
% buckling

shear_loads = [512035.7
506577.7
501120.4
495664.4
490210.4
484759.1
479311.1
473867.0
468427.6
462993.5
457565.5
452144.0
446729.9
441323.8
435926.4
430538.3
425160.2
```

419792.9
414436.9
409093.1
403762.0
398444.4
393140.9
387852.4
382579.4
377322.6
372082.9
366860.9
361657.3
356472.9
351308.4
346164.5
341042.0
335941.6
330864.1
325810.3
320780.8
315776.5
310798.3
305846.7
300922.8
296027.2
291160.8
286324.4
281518.8
276745.0
272003.7
267295.8
262622.2
257983.8
253381.5
248816.2
244288.8
239800.3
235351.6
230943.8
226577.8
222254.5
217975.1
213740.7
209552.1
205410.6
201317.2
197273.1
193279.5
189337.4
185448.2
181613.0
177833.2
172962.4
167002.2
161101.2

```
155260.9
149482.8
143768.4
138119.1
132536.7
127022.6
121578.6
116206.4
110907.9
105684.9
100539.4
95473.4
90489.0
85588.5
80774.2
76048.4
71413.8
66873.0
62428.9
58084.5
53842.9
49707.7
45682.3
41770.7
37977.1
34306.0
30762.4
27351.7
24080.0
20953.7
17980.4
15168.6
12528.0
10069.9
7807.9
5758.6
3942.9
2388.8
1136.3
251.1
81.5
0.0];

disc = linspace(0,11.24,1124);

lfunc = @(x) -0.06792.*x.^6 + 12.82.*x.^5 - 295.1.*x.^4 + 2510.*x.^3 - ...
    7610.*x.^2 - 7.804e4.*x + 8.308e5;

ldisc = lfunc(disc);

figure()
plot(disc,ldisc);

ribs_known = [0, 1.124, 6.744, 7.868, 8.992, 10.116];
% ribs placed at known concentrated loads
```

```
E = 72e9;
k = 3.62;
wlength = 11.24;
r_chord = 5.16;
t_chord = 1.13;
density = 2780;

pcrit = @(a) k*E*(a)^2;
% "a" is equal to t/b here

optrib = [];

for panels = 1:5
    t_skin = 1e-3;
    % optimal skin thickness obtained using fmincon function
            for nribs = 1:30
                ribspacing = (ribs_known(panels + 1) - ...
                    (ribs_known(panels)))/nribs;
                a = t_skin/ribspacing;
                averageload = (lfunc(ribs_known(panels)) + ...
                    lfunc(ribs_known(panels)+ribspacing))/2;
                if pcrit(a) < averageload
                    nribs = nribs + 1;
                else
                    disp(nribs)
                    break
                end
            end
        optrib(panels) = ribspacing;
end
disp(optrib)
ribs_known = [0, 1.124, 6.744, 7.868, 8.992, 10.116];
rib_locations = [0.562, 1.124, 1.686, 2.248, 2.810, 3.372, 3.934, 4.496,...
    5.058, 5.620, 6.182, 6.744, 7.306, 7.868, 8.992, 10.116];
% note these values are spanwise, along the quarter-chord line

%% Stringer optimisation

% setup variables
% CHECK THESE AFTER DOING MATERIAL SELECTION
aluminium_density = 2.78e+03;
w_length = 11.24;

% Initial values of the optimisation variables
skin_thickness = 0.003;
number = 30;
thickness = 0.01;
height = 0.01;
width = 0.01;

x_initial = [thickness, height, width, number, skin_thickness];

% The mass function which is what is reduced
mass = @(x) (((x(2)-2*x(1))*x(1)) + (x(3)*x(1)*2)*aluminium_density)...
    *x(4) + (w_length*3.5*x(5)*aluminium_density);
```

```
% The upper and lower bounds of each variable
lb = [0.001, 0.001, 0.0010, 21, 0.0010];
ub = [0.1, 0.1, 0.1, 40, 0.005];

nonlcon = @combinedconstraints;

options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm',...
    'interior-point');

% Define the problem as a GlobalSearch problem
problem = createOptimProblem('fmincon', 'objective', mass, 'x0',...
    x_initial, 'lb', lb, 'ub', ub, 'nonlcon', nonlcon, 'options', options);

% Create a GlobalSearch object
gs = GlobalSearch;

% Run the optimization using GlobalSearch
[x_final, final_mass, exitflag] = run(gs, problem);

% Analyze the results
if exitflag > 0
    disp('Optimal Solution to resist Stress');
    disp(x_final);
    disp('Optimal Mass Value');
    disp(final_mass);
else
    disp('Optimization did not converge to a solution');
end

%% Aerofoil plot with wingbox
clc
close all

geom=table2array(readtable("dsma523b.dat"));
x = geom(:,1);
y = geom(:,2);

figure()
plot(x,y,'Color','b','LineWidth',1);
grid on;
hold on;

chord = table2array(readtable("XYZc1.txt"));
xc = chord(:,1);
yc = chord(:,2);
plot(xc,yc,'--','Color','b');

htop = ((0.04778 - -0.049365) + (0.050276 - -0.028665))/4;
hbot = -htop;

v = [hbot, htop];
xl = [0.15,0.15];
plot(xl,v,'LineWidth',1,'Color','r');
```

```
xr = [0.65,0.65];
plot(xr,v,'LineWidth',1,'Color','r');

h = [0.15,0.65];
ytop = [htop, htop];
plot(h,ytop,'LineWidth',1,'Color','r');

ybot = [hbot, hbot];
plot(h,ybot,'LineWidth',1,'Color','r');

plot(0.4,0,'LineWidth',1,'Marker','x','Color','k')

xlabel('Chordwise co-ordinate: x/c','FontWeight','bold')
legend('Aerofoil geometry', 'Chord line', '','','',...
    'Rectangular wingbox', 'Shear centre')
axis equal;

%% Wingbox shear
clc
close all

figure()
grid on;
hold on;

htop = ((0.04778 - -0.049365) + (0.050276 - -0.028665))/4;
hbot = -htop;

v = [hbot, htop];
xl = [0.15,0.15];
plot(xl,v,'LineWidth',1,'Color','r');

xr = [0.65,0.65];
plot(xr,v,'LineWidth',1,'Color','r');

h = [0.15,0.65];
ytop = [htop, htop];
plot(h,ytop,'LineWidth',1,'Color','r');

ybot = [hbot, hbot];
plot(h,ybot,'LineWidth',1,'Color','r');

% right to 0.75

htop = ((0.04778 - -0.049365) + (0.050276 - -0.028665))/4;
hbot = -htop;
h = [0.65,0.75];
ytop = [htop, htop];
plot(h,ytop,'LineWidth',1,'Color','b');

v = [hbot, htop];
xr = [0.75,0.75];
plot(xr,v,'LineWidth',1,'Color','b');

ybot = [hbot, hbot];
```

```
plot(h,ybot,'LineWidth',1,'Color','b');


% top

htop = ((0.04778 - -0.049365) + (0.050276 - -0.028665))/4 + 0.06;
hbot = ((0.04778 - -0.049365) + (0.050276 - -0.028665))/4;
h = [0.15,0.65];
ytop = [htop, htop];
plot(h,ytop,'LineWidth',1,'Color','b');

v = [hbot, htop];
xr = [0.65,0.65];
plot(xr,v,'LineWidth',1,'Color','b');

v = [hbot, htop];
xl = [0.15,0.15];
plot(xl,v,'LineWidth',1,'Color','b');

% bottom

htop = -((0.04778 - -0.049365) + (0.050276 - -0.028665))/4 - 0.06;
hbot = -((0.04778 - -0.049365) + (0.050276 - -0.028665))/4;
h = [0.15,0.65];
ytop = [htop, htop];
plot(h,ytop,'LineWidth',1,'Color','b');

v = [hbot, htop];
xr = [0.65,0.65];
plot(xr,v,'LineWidth',1,'Color','b');

v = [hbot, htop];
xl = [0.15,0.15];
plot(xl,v,'LineWidth',1,'Color','b');

% left 0.04

htop = ((0.04778 - -0.049365) + (0.050276 - -0.028665))/4;
hbot = -htop;
h = [0.11,0.15];
ytop = [htop, htop];
plot(h,ytop,'LineWidth',1,'Color','b');

v = [hbot, htop];
xl = [0.11,0.11];
plot(xl,v,'LineWidth',1,'Color','b');

ybot = [hbot, hbot];
plot(h,ybot,'LineWidth',1,'Color','b');

xlim([0 1])
ylim([-0.3 0.3])
axis equal;

%% Plan view of wing (ribs and stringers)
```

```
clc
close all

% root chord
h_rc = [0,5.16];
x_rc = [0,0];

figure()

plot(x_rc,h_rc,'LineWidth',1,'Color','b')
xlim([0 12]);

grid on;
hold on;

% flexural axis
plot([0, 11.24],[3.096,-1.40235],'LineWidth',1,'Color','r', ...
    'LineStyle','--')
m_qc = (-1.40235-3.096)/11.24;
qc_eq = @(x) m_qc*x + 3.096;

% leading edge
plot([0, 11.24],[5.16, 5.16*0.75 + (11.24*sind(-27)) + 1.13*0.25],...
    'LineWidth',1,'Color','b')
% trailing edge
plot([0, 11.24],[0, 5.16*0.75 + (11.24*sind(-27)) - 1.13*0.75],...
    'LineWidth',1,'Color','b')

% tip chord
h_tc = [5.16*0.75 + (11.24*sind(-27)) - 1.13*0.75,  5.16*0.75 + ...
    (11.24*sind(-27)) + 1.13*0.25];
x_tc = [11.24,11.24];
plot(x_tc,h_tc,'LineWidth',1,'Color','b')

% wing box
plot([0, 11.24],[0.35*5.16, 5.16*0.75 + (11.24*sind(-27)) - 1.13*0.4],...
    'LineWidth',1,'Color','r')
plot([0, 11.24],[0.85*5.16, 5.16*0.75 + (11.24*sind(-27)) + 1.13*0.1],...
    'LineWidth',1,'Color','r')
plot([0, 0],[0.35*5.16, 0.85*5.16],'LineWidth',1,'Color','r')
plot([11.24, 11.24],[5.16*0.75 + (11.24*sind(-27)) - 1.13*0.4, 5.16*0.75...
    + (11.24*sind(-27)) + 1.13*0.1],'LineWidth',1,'Color','r')

% ribs
rib_locations = [0.3913, 0.7826, 1.174, 1.761, 2.348, 2.935, 3.522,...
    4.109, 4.696, 5.283, 5.870, 6.457, 7.044, 7.631, 8.218, 9.392, 10.566];

m_ribs = -1/m_qc;

y_ribs = [];
x_ribs = [];
x_ribs = [];


for i = 1:length(rib_locations)
```

```
    y_ribs(i) = 5.16*0.75 + (rib_locations(i).*sind(-27));
    x_ribs(i) = rib_locations(i).*cosd(-27);
    c_ribs(i) = y_ribs(i) - m_ribs.*x_ribs(i);

    ribs_eq = @(x) c_ribs(i) + m_ribs*x;

    m_box_upper = (-1.11985-4.386)/11.24;
    box_upper = @(x) m_box_upper*(x) + 4.386;

    m_box_lower = (-1.69085 - 1.806)/11.24;
    box_lower = @(x) m_box_lower*(x) + 4.386;

    if i < 3
        x_inter = (4.386 - c_ribs(i))/(m_ribs-m_box_upper);
        x_ribrange = linspace(0,x_inter,2);
        plot(x_ribrange,ribs_eq(x_ribrange),'Color','magenta');
    else
        x_inter_upper = (4.386 - c_ribs(i))/(m_ribs-m_box_upper);
        x_inter_lower = (1.806 - c_ribs(i))/(m_ribs-m_box_lower);
        x_ribrange = linspace(x_inter_lower,x_inter_upper,2);
        plot(x_ribrange,ribs_eq(x_ribrange),'Color','magenta');
    end
end

x_inter1 = [];

for i = 1:8
    c_str(i) = 4.386 - (2.58/21)*(i);
    x_inter1(i) = (4.386 - c_str(i))/(m_qc-m_box_upper);
    x_strrange = linspace(0,x_inter1(i),2);
    str_eq = @(x) m_qc*x_strrange + c_str(i);
    plot(x_strrange,str_eq(x_strrange),'Color','green');
end

for i = 9:12
    c_str(i) = 4.386 - (2.58/21)*(i);
    x_strrange = linspace(0,11.24,2);
    str_eq = @(x) m_qc*x_strrange + c_str(i);
    plot(x_strrange,str_eq(x_strrange),'Color','green');
end

x_inter2 = [];

for i = 13:21
    c_str(i) = 4.386 - (2.58/21)*(i);
    x_inter2(i-12) = (1.806 - c_str(i))./(m_qc-m_box_lower);
    x_strrange = linspace(0,x_inter2(i-12),2);
    str_eq = @(x) m_qc*x_strrange + c_str(i);
    plot(x_strrange,str_eq(x_strrange), 'Color','green');
end


axis equal;
%% Stringers x span plot
```

```
close all
clc

span = [0,1.37063452676356,1.37884601668925,2.74126905352712,...
    2.75769203337850,4.11190358029067,4.13653805006776,5.48253810705424,...
    5.51538406675701,6.85317263381779,6.89423008344626,8.22380716058135,...
    8.27307610013551,9.59444168734491,9.65192211682477,10.9650762141085,...
    11.0307681335140];
stringers = linspace(21,4,17);

figure()
plot(span, stringers,'Marker','o')
xlabel('Spanwise direction','FontWeight','bold')
ylabel('Number of stringers','FontWeight','bold')

%%

function [c, ceq] = combinedconstraints(x)

% setup variables
% CHECK THESE AFTER DOING MATERIAL SELECTION
M_yy = 6646942.7;
direct_yield_stress = 4.31*10^8;
comp_yield_stress = 4.42e+8;
shear_yield_stress = 3.1e+08;
aluminium_ym = 7.2e+10;
D = 2.81;
K = 3.62;
L = 0.5;
h_box = 0.454;
LD = 35;
M_xx = M_yy/LD;
w_length = 11.24;

I_xx = 1/6 * ((x(2)-2*x(1))*x(1)) * (x(3)*x(1)*2) * x(4)*(x(4)+1)*...
    (x(4)*2 + 1)*(5.16/x(4)) + x(5)*w_length^3/4;

%Stress due to bending constraint inequality:
c(1) = direct_yield_stress - (M_yy*(h_box/2))/((x(2)-2*x(1))*x(1)) +...
    (x(3)*x(1)*2) * x(4)*(h_box/2)^2 * 2 - M_xx*5.16/(1/6 * ((x(2)-2*...
    x(1))*x(1)) * (x(3)*x(1)*2) * x(4)*(x(4)+1)*(x(4)*2 + 1)*(5.16/x(4))...
    + x(5)*w_length^3/4);

%Buckling constraint inequality:
c(2) = comp_yield_stress - (pi^2*aluminium_ym * ((2*((x(2)-x(1))/2)^2)...
    *(x(3)*x(1)) + x(1)/12*(x(2)-2*x(1))^3 )) / ((x(3)*x(1)*2+((x(2)-...
    2*x(1))*x(1)))*L^2);
c(3) = shear_yield_stress - K*aluminium_ym*(x(5)/(((5.16/x(4))))^2);

ceq = [];

end
```

A.2 Fuselage

A.2.1 Stringer-Skin Optimiser Code

```
clear; clc;

% setup variables
density = 2.85e+03;
D = 2.81;

% Initial values of the optimisation variables which includes the skin
% and stringer dimensions
skin_thickness = 0.003;
number = 50;
thickness = 0.01;
height = 0.01;
flange_length = 0.01;

x_initial = [thickness, height, flange_length, number, skin_thickness];

% The mass function which is what is reduced
mass = @(x) (((x(2)-2*x(1))*x(1)) + (x(3)*x(1)*2) *density)*x(4)...
+ (D*pi*x(5)*density);

% The upper and lower bounds of each variable
lb = [0.001, 0.001, 0.001, 1, 0.001];
ub = [0.1, 0.1, 0.1, 100, 0.01];

nonlcon = @combinedconstraints;

options = optimoptions('fmincon', 'Display', 'iter', ...
    'Algorithm', 'interior-point');

% Define the gradient of the mass function
gradmass = @(x) [2*(x(2)-2*x(1))*density*x(4) + 2*x(1)*density*x(4) +...
    2*x(3)*x(1)*density;
                2*x(1)*density*x(4) - 4*x(1)*density*x(4);
                2*x(3)*x(1)*density;
                (((x(2)-2*x(1))^2 + 2*x(1)^2 + 2*x(3)*x(1))*density);
                D*pi*density];

% Define the problem as a GlobalSearch problem
problem = createOptimProblem('fmincon', 'objective', mass, ...
    'x0', x_initial, 'lb', lb, 'ub', ub, ...
    'nonlcon', nonlcon, 'options', options);

% Create a GlobalSearch object
gs = GlobalSearch;

% Run the optimization using GlobalSearch
[x_final, final_mass, exitflag, output] = run(gs, problem);

% Analyze the results
if exitflag > 0
    disp('Optimal Solution to resist Stress');
```

```
    disp(x_final);
    disp('Optimal Mass Value');
    disp(final_mass);
    disp('Sensitivity Analysis')
    disp(gradmass(x_final))
else
    disp('Optimization did not converge to a solution');
end
%%
function [c, ceq] = combinedconstraints(x)

% setup variables
M_yy = 2714900;
direct_yield_stress = 4.31e+8;
comp_yield_stress = 4.42e+8;
shear_yield_stress = 3.1e+08;
youngs_mod = 7.3e+10;
D = 2.81;
K = 3.62;
L = 0.5;

% Stringer Euler buckling constraint
c(1) = comp_yield_stress - (pi^2*youngs_mod * ((2*((x(2)-x(1))/2)^2)...
    *(x(3)*x(1) + x(1)/12*(x(2)-2*x(1))^3 )) /...
    ((x(3)*x(1)*2+((x(2)-2*x(1))*x(1)))*L^2);

%Stress due to bending constraint
c(2) = direct_yield_stress - (M_yy*(D/2)/...
    ( sum((((D/2)*sind(0:360/x(4):360)).^2)*(x(3)*x(1)*2 ...
    +((x(2)-2*x(1))*x(1)))) + (pi*((D/2)^4 - (D-x(5))^4))/4 ));

%Skin buckling constraint :
c(3) = shear_yield_stress - K*youngs_mod*((x(5)/((D*pi)/x(4)))^2);

ceq = [];
end
```

A.2.2 Light Frame Optimiser Code

```
clear; clc;

% Defining constants
aluminium_density = 2.85e+03;
aluminium_ym = 7.4e+10;
D = 2.81;
fuselage_length = 27.08;
C_fr = 1/16000;
max_moment = 2714900;

% Defining intial variables that will be optimised
flange_length = 0.05;
flange_thickness = 0.005;
base_length = 0.05;
base_thickness = 0.005;
frame_space = 0.5;
```

```
x_initial = [flange_length, flange_thickness, base_length,...
    base_thickness, frame_space];

% Mass function which is minimised
mass = @(x) (2*x(1)*x(2) + x(3)*x(4))*pi*D*aluminium_density...
    *floor(fuselage_length/x(5));

% Stiffness constraint
stiffness = @(x) deal([], ((x(4)*(x(3)^3)/12 + x(1)*x(2)*(x(3)^2)/2) ...
    - C_fr*max_moment*(D^2)/(x(5)*aluminium_ym)));

% Define the gradient of the mass function
gradmass = @(x) [2 * x(2) * pi * D * aluminium_density *...
    floor(fuselage_length / x(5))
                2 * x(1) * pi * D * aluminium_density *...
                floor(fuselage_length / x(5))
                x(4) * pi * D * aluminium_density *...
                floor(fuselage_length / x(5))
                x(3) * pi * D * aluminium_density *...
                floor(fuselage_length / x(5))
                -(2 * x(1) * x(2) + x(3) * x(4)) * pi * D *...
                aluminium_density * floor(fuselage_length /...
                x(5))^2 / x(5)];



Aeq = [];
beq = [];

% The upper and lower bounds of each variable
lb = [0.001,0.001,0.001,0.001,0.1];
ub = [0.1,0.01,0.1,0.01,0.6];

options = optimoptions('fmincon', 'Display', 'iter',...
    'Algorithm', 'interior-point');

% Define the problem as a GlobalSearch problem
problem = createOptimProblem('fmincon', 'objective', mass,...
    'x0', x_initial, 'lb', lb, 'ub', ub, 'nonlcon',...
    stiffness, 'options', options);

% Create a GlobalSearch object
gs = GlobalSearch;

% Run the optimization using GlobalSearch
[x_final, final_mass, exitflag] = run(gs, problem);

% Analyze the results
if exitflag > 0
    disp('Optimal Solution to resist Stress');
    disp(x_final);
    disp('Optimal Mass Value');
    disp(final_mass);
    disp('Sensitivity Analysis')
```

```
    disp(gradmass(x_final))
else
    disp('Optimization did not converge to a solution');
end
```

A.2.3 Heavy Frame Optimiser Code

```
clear; clc;

% Defining constants
aluminium_density = 2.85e+03;
aluminium_ym = 7.3e+10;
D = 2.81;
fuselage_length = 27.08;
max_moment = 2714912.9;
tensile_yield_stress = 4.31e+8;

% Initial values of the optimisation variables which includes the
% skin and stringer dimensions
thickness = 0.01;
height = 0.01;
flange_length= 0.01;

x_initial = [thickness, height, flange_length];

% The mass function which is what is reduced
mass = @(x) (((x(2)-2*x(1))*x(1)) + (x(3)*x(1)*2) *pi*D*aluminium_density);

% The upper and lower bounds of each variable
lb = [0.001, 0.001, 0.001];
ub = [0.1, 0.1, 0.1];

% Stiffness constraint
stiffness = @(x) deal([], ((x(1)*(x(2)^3)/12 + x(3)*x(1)*(x(2)^2)/2)...
    - max_moment*x(2)/(2*tensile_yield_stress)));

% Define the gradient of the mass function
gradmass = @(x) [(2 * x(2) - 4 * x(1) + 4 * x(3) * pi * D...
    * aluminium_density)
                (x(2) - 2 * x(1)) * pi * D * aluminium_density
                2 * x(1) * pi * D * aluminium_density];

Aeq = [];
beq = [];

options = optimoptions('fmincon', 'Display', 'iter',...
    'Algorithm', 'interior-point');

% Define the problem as a GlobalSearch problem
problem = createOptimProblem('fmincon', 'objective', mass,...
    'x0', x_initial, 'lb', lb, 'ub', ub,...
    'nonlcon', stiffness, 'options', options);

% Create a GlobalSearch object
gs = GlobalSearch;
```

```
% Run the optimization using GlobalSearch
[x_final, final_mass, exitflag] = run(gs, problem);

% Analyze the results
if exitflag > 0
    disp('Optimal Solution to resist Stress');
    disp(x_final);
    disp('Optimal Mass Value');
    disp(final_mass);
    disp('Sensitivity Analysis')
    disp(gradmass(x_final))
else
    disp('Optimization did not converge to a solution');
end
```

A.3 Empennage

A.3.1 Tail Force Calculations Code

```
%% Tailplane Loads
clear
clc

Plane_Length = 27.18;
x_cg = Plane_Length/2 + 2.28;
%W0 = 42000*9.81;
W0 = 79.15*5150; %n=1: 5.8706e+04 n =4; 2.3482e+05 n=-1.5: -8.8058e+04
%n = 3.75    g_n=3: 2.3482e+05
n = 1;
g_n = 3;

Lift = n*W0;
Gear_F = g_n*W0;

Weights(1, :) = [x_cg 0 0 W0]; % Aircraft centre of gravity

Wing_NP = [14.1 0 -1.02];

HT_NP = [26.39 0 3.86];

rear = [30, 0, 0];

alpha = 0;

%% Horizontal Tailplane

fig1 = figure(1);

Weights(1, 1:3) = atIncidence(alpha, Weights(1, 1:3));
Wing_NP = atIncidence(alpha, Wing_NP);
HT_NP = atIncidence(alpha, HT_NP);
rear = atIncidence(alpha, rear);
```

```
scatter([0 Weights(1, 1) Wing_NP(1), HT_NP(1) rear(1)] , ...
    [0 Weights(1, 3) Wing_NP(3), HT_NP(3) rear(3)])
hold on
line([0, rear(1)], [0 rear(3)], 'LineWidth', 1, 'LineStyle', '--', 'Color', 'b')
line([Wing_NP(1) Wing_NP(1)], [Wing_NP(3) 0], 'LineWidth', 1, 'LineStyle' ...
    , '--', 'Color', 'm')
line([HT_NP(1) HT_NP(1)], [HT_NP(3) 0], 'LineWidth', 1, 'LineStyle', '--' ...
    , 'Color', 'r')
hold off

% Mom from cg: (x_np-x_cg)*(Lift_W+Gear_F) + (x_ht-x_cg)*Lift_H = 0.
% Lift_W = Lift-Lift_H = nW - Lift_H.
% (x_np-x_cg)(nW - Lift_H + Gear_F) + (x_ht-x_cg)*Lift_H = 0.
% Lift_H(x_np-x_cg) - Lift_H(x_ht-x_cg) = (nW+Gear_F)(x_np-x_cg)
% Lift_H = (nW+Gear_F)(x_np-x_cg) / (x_np-x_ht).
disp("L_H force")
Lift_H = (Lift+Gear_F)*(Weights(1, 1) - Wing_NP(1)) / (HT_NP(1) - Wing_NP(1))
(Lift-Lift_H+Gear_F)*(Wing_NP(1) - x_cg) + Lift_H*(HT_NP(1) - x_cg)
disp("L_W force")
Lift_W = Lift-Lift_H

disp("Does L_W exceed L? : <- " + (abs(Lift_W) > abs(Lift)))

disp("Use these if L_W exceeds L") % Finding max Lift such that L_W <= nW
% Mom from cg: (x_np-x_cg)*(Lift_W+Gear_F) + (x_ht-x_cg)*Lift_H = 0.
% Lift_W = nW
% (x_np-x_cg)(nW+Gear_F) + (x_ht-x_cg)*Lift_H = 0.
% - Lift_H(x_ht-x_cg) = (nW+Gear_F)(x_np-x_cg)
% Lift_H = (nW+Gear_F)(x_np-x_cg) / (x_cg-x_ht).
disp("L_H force")
Lift_H = (Lift+Gear_F)*(x_cg-Wing_NP(1)) / (HT_NP(1) - x_cg)
(Lift+Gear_F)*(Wing_NP(1) - x_cg) + Lift_H*(HT_NP(1) - x_cg)
disp("L force")
Lift_W = Lift+Lift_H

ylim([-10 10])
xlim([0 30])

%% Vertical Tailplane
disp("Vertical Tailplane ----------")
AR_v = 1.8;
a_v_sect = 4.198;
a_curve = ((1/a_v_sect) + 1/(AR_v*pi))^(-1);
Sv = 13.23;
V = 1.3*80; % FAR regulation
hmm = 0.5*1.225*(V^2)*Sv*a_curve*deg2rad(5);


%%% Wings Level Flight
Eng_Span_dist = 2.5; % Engine Moment Arm
VT_x = 25.32; % Vertical Tailplane distance
Eng_Thrust = 56575;
R_x = 27.32; % Rudder distance
% L_R* R_x = L_VT* VT_x + T_OEI * T_y
% By equilibrium, L_VT = L_R
```

```
% L_VT(R_x - VT_x) = T_OEI * T_y
L_VT = (0.3*Eng_Thrust*Eng_Span_dist) / (R_x - VT_x) % 30% Throttle

%%% Slight Bank angle
% L_VT * VT_x = T_OEI * T_y
disp("Lv force")
L_VT = 1.5*(Eng_Thrust*Eng_Span_dist) / (VT_x-x_cg) % Safety factor
disp("Bank angle")
asind(L_VT / W0)

%% Function defs
% Rotates vector by alpha about Y axis.
function [rotated] = atIncidence(alpha, vector)
    rotated(1) = cosd(alpha)*vector(1) + sind(alpha)*vector(3);
    rotated(2) = 0;
    rotated(3) = cosd(alpha)*vector(3) - sind(alpha)*vector(1);
end
```

A.3.2 Horizontal Tail Distribution Code

```
%% Wing Loading
clear
clc

%Set some constants here I guess
W0 = 79.15*5150;
%%% Load factors (these are used for file outnames and weight of lifting
%%% surface
n = 1;
g_n = 3;
%%% This lift
Lv = 5.8706e+04; %Taken from Tail_Calcs.
%%% Ref area = b*cbar, AR = b/cbar so b^2 - S_ref*AR
Ref_area = 13.98; %13.23
AR = 5; %1.8
b = sqrt(Ref_area*AR);
L0 = 2*(n+g_n)*Lv / (pi*0.5*b); % Elliptic Equation
c_bar = b/AR;
Taper_Ratio = 0.35; %0.7
%%% 0.5(Chord_root+Chord_tip)=c_bar, Chord_tip/Chord_root=Taper_Ratio ->
%%% 0.5(Chord_root(1+taper_ratio) = c_bar so Chord_root = 2c_bar/(1+taper)
Chord_root = 2*c_bar/(1+Taper_Ratio);
Chord_tip = Taper_Ratio*Chord_root;
Sweep_c4 = 35.6; %40
fw = 0; %2.69/2; %1; % Fuselage width
Wing_weight = 0.01*W0; % Raymer
%%% Chord ratios
spar_f = 0.2; %10
spar_b = 0.6; %70
chord_ac = 0.25;
chord_cg = 0.5;
chord_ge = 0.5;
chord_cont = 0.75; % Control surfaces
span_cont = [0.10 0.90]; % First is start of semi-span, second end.
```

```
%% Lift Distribution
%Span-wise Lift distribution
minY = -b/2;
maxY = b/2;
Ny = 500;

L_span = @(y) (Lift_Span(y, fw, L0, b));

%% Wing Shape
Wing_chord = @(y) (Chord_root * (1 - (1 - (Chord_tip/Chord_root))*(abs(y)/(b/2) ) )
Wing_Sweep = @(y) (-abs(y)*tand(Sweep_c4));

Wing_LE = @(y) Wing_Sweep(y) + 0.25*Wing_chord(y);
Wing_TE = @(y) Wing_Sweep(y) - 0.75*Wing_chord(y);

%% Plotting
%%% Fig1
fig1 = figure(1);
tcl = tiledlayout(2,2,'TileSpacing','compact');
nexttile
% Span-wise Lift Distribution
fplot(L_span, [0 b/2], LineWidth=3)
title("Span-wise Lift Distribution")
ylabel("L_0 (Nm^{-1})")
xlabel("y (m)")

nexttile

% Wing Spar and flexural axis
fplot(Wing_LE, [0 b/2], LineWidth=3, Color='b')
hold on
fplot(Wing_TE, [0 b/2], LineWidth=3, Color='b')
line([b/2 b/2], [Wing_LE(b/2) Wing_TE(b/2)], 'LineWidth', 3, 'Color', 'b')
fplot(@(y) Wing_LE(y) - spar_f*Wing_chord(y), [0, b/2], 'LineWidth', 3 ...
    , 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_b*Wing_chord(y), [0, b/2], 'LineWidth', 3 ...
    , 'Color', 'k', 'LineStyle', '-.')
fplot(@(y) Wing_LE(y) - 0.5*(spar_f+spar_b)*Wing_chord(y), [0, b/2] ...
    , 'LineWidth', 2, 'Color', 'b', 'LineStyle', '--')

qw{1} = plot(nan, 'b', LineWidth=3);
qw{2} = plot(nan, 'b--', LineWidth=2);
qw{3} = plot(nan, 'k-.', LineWidth=3);
qw{4} = plot(nan, ':', LineWidth=3, Color="#A2142F");
legend([qw{:}], {'Tail-plane','Flexural Axis','Rear Spar', 'Front Spar'} ...
    , 'location', 'best')
hold off

title("Horoizontal Fin Geometry")
xlabel("y (m)")
ylabel("x (m)")

nexttile([1 2])

%%% Wing Shape
```

```
fplot(Wing_LE, [-b/2 b/2], LineWidth=3, Color='b')
hold on
fplot(Wing_TE, [-b/2 b/2], LineWidth=3, Color='b')
% Quarter chord sweep
fplot(Wing_Sweep, [-b/2 b/2], LineStyle="--", LineWidth=2, Color='b')
line([b/2 b/2], [Wing_LE(b/2) Wing_TE(b/2)], 'LineWidth', 3, 'Color', 'b')
line([-b/2 -b/2], [Wing_LE(-b/2) Wing_TE(-b/2)], 'LineWidth', 3, 'Color' ...
    , 'b')

% Vertical Attach point
line([-fw -fw], [Wing_LE(0) Wing_TE(-b/2)], 'LineWidth', 3, 'Color' ...
    , 'black', 'LineStyle', '-.')
line([fw fw], [Wing_LE(0) Wing_TE(-b/2)], 'LineWidth', 3, 'Color' ...
    , 'black', 'LineStyle', '-.')

% Control surfaces
fplot(@(y) Wing_LE(y) - chord_cont*Wing_chord(y), [-0.5*b*span_cont(2) ...
    , -0.5*b*span_cont(1)], 'LineWidth', 3, 'Color', '#A2142F' ...
    , 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - chord_cont*Wing_chord(y), [0.5*b*span_cont(1) ...
    , 0.5*b*span_cont(2)], 'LineWidth', 3, 'Color', '#A2142F' ...
    , 'LineStyle', ':')

line([-0.5*b*span_cont(2) -0.5*b*span_cont(2)], ...
    [Wing_LE(-0.5*b*span_cont(2))-chord_cont*Wing_chord(-0.5*b*span_cont(2)) ...
    , Wing_LE(-0.5*b*span_cont(2))-Wing_chord(-0.5*b*span_cont(2))] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')
line([-0.5*b*span_cont(1) -0.5*b*span_cont(1)], ...
    [Wing_LE(-0.5*b*span_cont(1))-chord_cont*Wing_chord(-0.5*b*span_cont(1)) ...
    , Wing_LE(-0.5*b*span_cont(1))-Wing_chord(-0.5*b*span_cont(1))] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')

line([0.5*b*span_cont(2) 0.5*b*span_cont(2)], ...
    [Wing_LE(0.5*b*span_cont(2))-chord_cont*Wing_chord(0.5*b*span_cont(2)) ...
    , Wing_LE(0.5*b*span_cont(2))-Wing_chord(0.5*b*span_cont(2))] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')
line([0.5*b*span_cont(1) 0.5*b*span_cont(1)], ...
    [Wing_LE(0.5*b*span_cont(1))-chord_cont*Wing_chord(0.5*b*span_cont(1)) ...
    , Wing_LE(0.5*b*span_cont(1))-Wing_chord(0.5*b*span_cont(1))] ...
    , 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')


% Custom legend
qw{1} = plot(nan, 'b', LineWidth=3);
qw{2} = plot(nan, 'b--', LineWidth=2);
qw{3} = plot(nan, 'k-.', LineWidth=3);
qw{4} = plot(nan, ':', LineWidth=3, Color="#A2142F");
legend([qw{:}], {'Wing','Quater-Chord','Centre line', 'Control Surfaces'} ...
    , 'location', 'best')
hold off

title("Tail-plane Layout")
xlabel("y (m)")
ylabel("x (m)")
```

```
fontsize(gcf, scale=1.3)

%% 2-D
Y = linspace(0, b/2, Ny);
Y = [-flip(Y(2:end)), Y];
Z = zeros(1, length(Y));  % Wing Lift
Z2 = zeros(1, length(Y)); % Wing Weight

Wing_area = arrayfun(@(y) (Wing_chord(y) / (0.5*(Chord_root+Chord_tip)))^2, Y);
Wing_density = Wing_weight / trapz(Y, Wing_area);

%%% Find moment arms for Torque Calculations
p_lift = arrayfun(@(y) (0.5*(spar_f+spar_b) - chord_ac)*Wing_chord(y), Y);
% From Aerodynamic centre
p_weight = arrayfun(@(y) (0.5*(spar_f+spar_b) - chord_cg)*Wing_chord(y) , Y);
% From Wing CG

for i = 1:length(Y)
    Z(i) = L_span(Y(i));
    Z2(i) = Wing_density*Wing_area(i)*inStruct2D(Y(i), minY, maxY);
end



Total_Lift = trapz(Y, Z);           % Check if Lift = n*W0
Total_Wing_Weight = trapz(Y, Z2);   % Check Weight of Wing

Z_Sum_Span = Z - Z2;
Z_Tor_Span = p_lift.*Z - p_weight.*Z2;
% Total Sectional Torque

%% Quick test
%fplot(@(y) (Wing_chord(y) / (0.5*(Chord_root+Chord_tip)))^2, [minY, maxY])

%% Shear Loads

size_Y = floor(length(Y)/2);
rem_Y = mod(length(Y), 2);
DSF = zeros(Ny, 1);
DT = zeros(Ny, 1);

for i = 1:Ny
    if i == Ny
        DSF(i) = 0.5*(Z_Sum_Span(size_Y + i))*(Y(end)-Y(end-1));
        DT(i) = 0.5*(Z_Tor_Span(size_Y + i))*(Y(end)-Y(end-1));
    else
        DSF(i) = ( 0.5*(Z_Sum_Span(size_Y + i) + Z_Sum_Span(size_Y + i+1) ...
            ) * (Y(size_Y + i+1)-Y(size_Y + i)) );
        DT(i) = ( 0.5*(Z_Tor_Span(size_Y + i) + Z_Tor_Span(size_Y + i+1)) ...
            * (Y(size_Y + i+1)-Y(size_Y + i)) );
    end
end

SF = flip(cumsum(flip(DSF(1:end))));
```

```
T = flip(cumsum(flip(DT)));

DBM = zeros(length(SF), 1);
for i = 1:Ny
    if i == Ny
        DBM(i) = 0.5*(SF(i))*(Y(end)-Y(end-1));
    else
        DBM(i) = ( 0.5*(SF(i) + SF(i+1)) * (Y(size_Y + i+1)-Y(size_Y + i)) );
    end
end

BM = flip(cumsum(flip(DBM)));

%% Plotting
fig3 = figure(3);
tcl2 = tiledlayout(2,2,'TileSpacing','compact');
nexttile
plot(Y(Ny:end), SF)

title("Shear Force Distribution")
xlabel("y (m)")
ylabel("SF (N)")

nexttile
plot(Y(floor(end/2)+1:end), BM)

title("Bending Moment Distribution")
xlabel("y (m)")
ylabel("BM (Nm)")

nexttile
plot(Y(Ny:end), T)

title("Torsion Moment Distribution")
xlabel("y (m)")
ylabel("TM (Nm)")

nexttile
plot(Y(Ny:end), DSF)

title("Force Distribution")
xlabel("y (m)")
ylabel("DF (N)")

fontsize(gcf, scale=1.3)

% save("plotHT" + num2str(floor(n)) + num2str(floor(g_n)), 'Y', 'SF' ...
% , 'BM', 'T', 'DSF')


%% Function definitions
% Corrected Lift Span to compensate fuselage.
function [L] = Lift_Span(y, f_width, L0, b)
    if abs(y) < f_width
        L = 0;
```

```
    else
        L = L0 * sqrt(1 - (y / (b/2)).^2 );
    end
end

% 2-D check if within boundaries.
function [D] = inStruct2D(y, b1, b2)
    if y > b1 && y < b2
        D = 1;
    else
        D = 0;
    end
end
```

A.3.3 Enlarged Horizontal Tailplane Layout

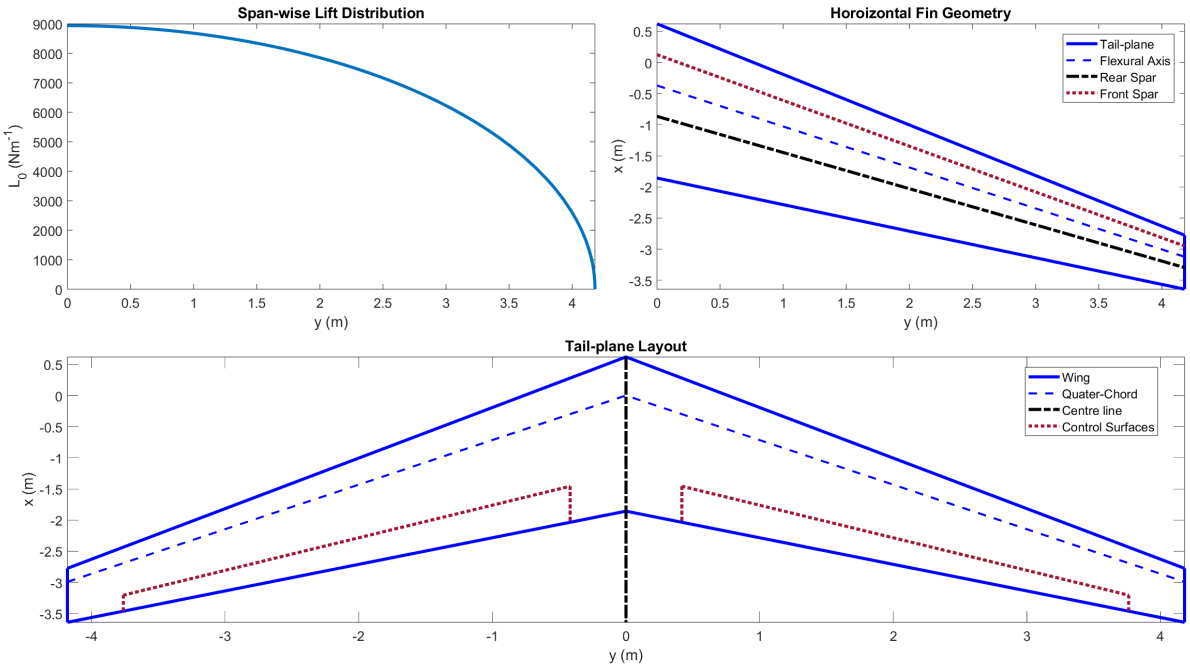


Figure A.6: Horizontal Tail-plane

A.3.4 Horizontal Tail Plotting Code

```
%% Wing Loading Plot all on one fig
clear
clc

case1 = load("plotHT10.mat"); % SLF (control line)
case2 = load("plotHT-20.mat"); % -1.5 case
case3 = load("plotHT13.mat"); % Landing gear inpact 3, lift 1
case4 = load("plotHT40.mat"); % 4.07 case

tcl = tiledlayout(2,2,'TileSpacing','compact');

nexttile
plot(case1.Y(floor(end/2) + 1:end), case1.DSF, "--", LineWidth=2)
hold on
```

```
plot(case4.Y(floor(end/2) + 1:end), case4.DSF, LineWidth=2)
plot(case2.Y(floor(end/2) + 1:end), case2.DSF, LineWidth=2)
plot(case3.Y(floor(end/2) + 1:end), case3.DSF, LineWidth=2)
hold off

title("Total Force Distribution")
xlabel("y (m)")
ylabel("F (N)")
legend({"n = 1",  "n = 4.07", "n = -1.5", "n_{gear} = 3"})

nexttile
%length(Y(length(Y)/2:end))
plot(case1.Y(floor(end/2) + 1:end), case1.SF, "--", LineWidth=2)
hold on
plot(case4.Y(floor(end/2) + 1:end), case4.SF, LineWidth=2)
plot(case2.Y(floor(end/2) + 1:end), case2.SF, LineWidth=2)
plot(case3.Y(floor(end/2) + 1:end), case3.SF, LineWidth=2)
hold off

title("Shear Force Distribution")
xlabel("y (m)")
ylabel("SF (N)")
legend({"n = 1",  "n = 4.07", "n = -1.5", "n_{gear} = 3"})

nexttile
plot(case1.Y(floor(end/2) + 1:end), case1.BM, "--", LineWidth=2)
hold on
plot(case4.Y(floor(end/2) + 1:end), case4.BM, LineWidth=2)
plot(case2.Y(floor(end/2) + 1:end), case2.BM, LineWidth=2)
plot(case3.Y(floor(end/2) + 1:end), case3.BM, LineWidth=2)
hold off

title("Bending Moment Distribution")
xlabel("y (m)")
ylabel("BM (Nm)")
legend({"n = 1",  "n = 4.07", "n = -1.5", "n_{gear} = 3"})

nexttile
plot(case1.Y(floor(end/2) + 1:end), case1.T, "--", LineWidth=2)
hold on
plot(case4.Y(floor(end/2) + 1:end), case4.T, LineWidth=2)
plot(case2.Y(floor(end/2) + 1:end), case2.T, LineWidth=2)
plot(case3.Y(floor(end/2) + 1:end), case3.T, LineWidth=2)
hold off

title("Torsion Moment Distribution")
xlabel("y (m)")
ylabel("TM (Nm)")
legend({"n = 1",  "n = 4.07", "n = -1.5", "n_{gear} = 3"})

fontsize(gcf, scale=1.5)
```

A.3.5 Vertical Tail Optimisation and Plotting Code

```
%% Rib optimsiation
```

```
close all
clc

% place ribs at known locations, treat space between locations as "section"
% and iterate through placing different of ribs in order to withstand
% buckling

shear_loads = [115370.454553919
115079.178987524
114787.882973683
114496.567743807
114205.234529336
113913.884561755
113622.519072606
113330.9293503
113039.746456148
112748.341792344
112456.926534011
112165.501913197
111874.069162098
111582.629513068
111291.184198637
110999.734451520
110708.281504640
110416.826591135
110125.370944375
109833.915797981
109542.462385831
109251.011942082
108959.565701183
108668.124897886
108376.690767265
108085.264544730
107793.847466040
107502.440767318
107211.045685067
106919.663456186
106628.295317980
106336.942508180
106045.606264955
105754.287826929
105462.988433193
105171.709323323
104880.451737393
104589.216915989
104298.006100230
104006.820531775
103715.661452843
103424.530106227
103133.427735309
102842.355584078
102551.314897138
102260.306919733
101969.332897754
101678.394077759
```

101387.491706987
101096.627033374
100805.801305567
100515.015772943
100224.271685620
99933.5702944769
99642.9128511652
99352.3006081283
99061.7348186153
98771.2167366974
98480.7476172835
98190.3287161367
97899.9612898896
97609.6465960609
97319.3858930713
97029.1804402597
96739.0314978992
96448.9403272138
96158.9081903945
95868.9363506156
95579.0260720514
95289.1786198925
94999.3952603627
94709.6772607354
94420.0258893504
94130.4424156307
93840.9281100993
93551.4842443965
93262.1120912963
92972.8129247239
92683.5880197730
92394.4386527225
92105.3661010543
91816.3716434704
91527.4565599106
91238.6221315699
90949.8696409161
90661.2003717077
90372.6156090114
90084.1166392205
89795.7047500723
89507.3812306665
89219.1473714834
88931.0044644018
88642.9538027179
88354.9966811632
88067.1343959233
87779.3682446566
87491.6995265129
87204.1295421522
86916.6595937638
86629.2909850852
86342.0250214213
86054.8630096637
85767.8062583100

85480.8560774834
85194.0137789519
84907.2806761485
84620.6580841910
84334.1473199013
84047.7497018263
83761.4665502573
83475.2991872511
83189.2489366496
82903.3171241009
82617.5050770798
82331.8141249088
82046.2455987784
81760.8008317692
81475.4811588721
81190.2879170103
80905.2224450604
80620.2860838743
80335.4801763010
80050.8060672080
79766.2651035040
79481.8586341609
79197.5880102358
78913.4545848941
78629.4597134319
78345.6047532986
78061.8910641203
77778.3200077228
77494.8929481547
77211.6112517113
76928.4762869578
76645.4894247537
76362.6520382762
76079.9655030447
75797.4311969452
75515.0505002547
75232.8247956660
74950.7554683127
74668.8439057939
74387.0914982002
74105.4996381386
73824.0697207583
73542.8031437769
73261.7013075061
72980.7656148782
72699.9974714728
72419.3982855431
72138.9694680430
71858.7124326546
71578.6285958151
71298.7193767445
71018.9861974739
70739.4304828729
70460.0536606782
70180.8571615223

69901.8424189620
69623.0108695076
69344.3639526521
69065.9031109007
68787.6297898005
68509.5454379708
68231.6515071334
67953.9494521426
67676.4407310167
67399.1268049688
67122.0091384383
66845.0891991225
66568.3684580084
66291.8483894053
66015.5304709771
65739.4161837751
65463.5070122714
65187.8044443921
64912.3099715511
64637.0250886842
64361.9512942836
64087.0900904326
63812.4429828404
63538.0114808778
63263.7970976130
62989.8013498472
62716.0257581514
62442.4718469033
62169.1411443239
61896.0351825157
61623.1554975002
61350.5036292559
61078.0811217576
60805.8895230148
60533.9303851117
60262.2052642464
59990.7157207718
59719.4633192356
59448.4496284218
59177.6762213921
58907.1446755278
58636.8565725721
58366.8134986729
58097.0170444263
57827.4688049199
57558.1703797774
57289.1233732031
57020.3293940268
56751.7900557499
56483.5069765910
56215.4817795332
55947.7160923704
55680.2115477557
55412.9697832492
55145.9924413668

54879.2811696294
54612.8376206130
54346.6634519987
54080.7603266240
53815.1299125341
53549.7738830342
53284.6939167419
53019.8916976411
52755.3689151355
52491.1272641032
52227.1684449524
51963.4941636765
51700.1061319114
51437.0060669922
51174.1956920114
50911.6767358772
50649.4509333730
50387.5200252174
50125.8857581251
49864.5498848678
49603.5141643370
49342.7803616071
49082.3502479985
48822.2256011429
48562.4082050483
48302.8998501653
48043.7023334543
47784.8174584530
47526.2470353457
47267.9928810325
47010.0568192002
46752.4406803933
46495.1463020869
46238.1755287597
45981.5302119683
45725.2122104225
45469.2233900618
45213.5656241322
44958.2407932650
44703.2507855558
44448.5974966451
44194.2828297996
43940.3086959953
43686.6770140008
43433.3897104622
43180.4487199898
42927.8559852443
42675.6134570264
42423.7230943658
42172.1868646124
41921.0067435288
41670.1847153834
41419.7227730459
41169.6229180830
40919.8871608566

40670.5175206225
40421.5160256312
40172.8847132297
39924.6256299652
39676.7408316898
39429.2323836674
39182.1023606818
38935.3528471464
38688.9859372158
38443.0037348986
38197.4083541728
37952.2019191019
37707.3865639536
37462.9644333198
37218.9376822390
36975.3084763200
36732.0789918680
36489.2514160125
36246.8279468373
36004.8107935123
35763.2021764279
35522.0043273311
35281.2194894642
35040.8499177051
34800.8978787108
34561.3656510627
34322.2555254140
34083.5698046404
33845.3108039925
33607.4808512513
33370.0822868861
33133.1174642147
32896.5887495672
32660.4985224519
32424.8491757243
32189.6431157588
31954.8827626240
31720.5705502601
31486.7089266604
31253.3003540554
31020.3473091000
30787.8522830652
30555.8177820316
30324.2463270877
30093.1404545310
29862.5027160737
29632.3356790507
29402.6419266331
29173.4240580444
28944.6846887817
28716.4264508400
28488.6519929420
28261.3639807710
28034.5650972092
27808.2580425802

27582.4455348962
27357.1303101102
27132.3151223726
26908.0027442936
26684.1959672106
26460.8976014600
26238.1104766563
26015.8374419748
25794.0813664416
25572.8451392287
25352.1316699555
25131.9438889963
24912.2847477942
24693.1572191822
24474.5642977099
24256.5089999785
24038.9943649816
23822.0234544544
23605.5993532298
23389.7251696026
23174.4040357013
22959.6391078687
22745.4335670505
22531.7906191922
22318.7134956461
22106.2054535864
21894.2697764341
21682.9097742921
21472.1287843895
21261.9301715373
21052.3173285934
20843.2936769403
20634.8626669724
20427.0277785962
20219.7925217422
20013.1604368887
19807.1350955995
19601.7201010735
19396.9190887091
19192.7357266822
18989.1737165387
18786.2367938019
18583.9287285960
18382.2533262852
18181.2144281291
17980.8159119562
17781.0616928540
17581.9557238783
17383.5019967810
17185.7045427578
16988.5674332159
16792.0947805630
16596.2907390184
16401.1595054457
16206.7053202100

16012.9324680589
15819.8452790282
15627.4481293747
15435.7454425350
15244.7416901135
15054.4413928981
14864.8491219080
14675.9694994716
14487.8072003372
14300.3669528188
14113.6535399758
13927.6718008305
13742.4266316236
13557.9229871091
13374.1658818915
13191.1603918054
13008.9116553406
12827.4248751144
12646.7053193921
12466.7583236591
12287.5892922466
12109.2037000125
11931.6070940805
11754.8050956404
11578.8034018117
11403.6077875738
11229.2241077658
11055.6582991592
10882.9163826069
10711.0044652723
10539.9287429423
10369.6955024284
10200.3111240599
10031.7820842746
9864.11495831123
9697.31642300887
9531.39325971971
9366.35235734010
9202.20071546673
9038.94544768468
8876.59378499429
8715.15307938485
8554.63080756286
8395.03457484404
8236.37211921826
8078.65131559745
7921.88018025727
7766.06687548407
7611.21971443961
7457.34716625680
7304.45786138076
7152.56059717092
7001.66434378036
6851.77825033083
6702.91165140258

6555.07407386017
6408.27524403687
6262.52509530251
6117.83377604133
5974.21165806916
5831.66934552146
5690.21768424672
5549.86777174297
5410.63096767848
5272.51890504173
5135.54350196994
4999.71697431037
4865.05184897389
4731.56097814657
4599.25755443143
4468.15512700042
4338.26761884526
4209.60934522522
4082.19503342105
3956.03984391685
3831.15939314537
3707.56977794898
3585.28760192639
3464.33000385695
3344.71468841811
3226.45995944036
3109.58475597592
2994.10869149568
2880.05209657269
2767.43606546205
2656.28250704782
2546.61420069864
2438.45485765874
2331.82918870131
2226.76297889182
2123.28317045366
2021.41795490294
1921.19687583219
1822.65094398232
1725.81276656182
1630.71669316877
1537.39898116529
1445.89798397714
1356.25436658013
1268.51135344726
1182.71501553810
1098.91460462276
1017.16294549481
937.516899658473
860.037918193170
784.792707179078
711.854037040319
641.301738566237
573.223945034100
507.718664798841

```
444.895807087451
384.879844665143
327.813397408826
273.862193490046
223.222177788941
176.130141622815
132.880513223506
93.8538905170426
59.5708038303160
30.8105966033902
8.96500572215552
0];

disc = linspace(0,tlength,1124);

lfunc = @(x) -0.06792.*x.^6 + 12.82.*x.^5 - 295.1.*x.^4 + 2510.*x.^3 - 7610.*x.^2 -

ldisc = lfunc(disc);

figure()
plot(disc,ldisc);

ribs_known = [0, 1.124, 6.744, 7.868, 8.992, 10.116];
% ribs placed at known concentrated loads
E = 72e9;
k = 3.62;
tlength = 4.18;
r_chord = 2.4773;
t_chord = 0.35*r_chord;
density = 2780;

pcrit = @(a) k*E*(a)^2;
% "a" is equal to t/b here

optrib = [];

for panels = 1:1
    t_skin = 1e-3;
    % optimal skin thickness obtained using fmincon function
            for nribs = 1:30
                ribspacing = (ribs_known(panels + 1) - ...
                    (ribs_known(panels)))/nribs;
                a = t_skin/ribspacing;
                averageload = (lfunc(ribs_known(panels)) + ...
                    lfunc(ribs_known(panels)+ribspacing))/2;
                if pcrit(a) < averageload
                    nribs = nribs + 1;
                else
                    disp(nribs)
                    break
                end
            end
        optrib(panels) = ribspacing;
end
disp(optrib)
```

```
% note these values are spanwise, along the quarter-chord line

%% Stringer optimisation

% setup variables
% CHECK THESE AFTER DOING MATERIAL SELECTION
aluminium_density = 2.78e+03;
w_length = 4.18;

% Initial values of the optimisation variables
skin_thickness = 0.003;
number = 30;
thickness = 0.01;
height = 0.01;
width = 0.01;

x_initial = [thickness, height, width, number, skin_thickness];

% The mass function which is what is reduced
mass = @(x) (((x(2)-2*x(1))*x(1)) + (x(3)*x(1)*2) ...
    *aluminium_density)*x(4) + (w_length*1.67*x(5)*aluminium_density);

% The upper and lower bounds of each variable
lb = [0.001, 0.001, 0.0010, 6, 0.0010];
ub = [0.1, 0.1, 0.1, 40, 0.005];

nonlcon = @combinedconstraints;

options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm', ...
    'interior-point');

% Define the problem as a GlobalSearch problem
problem = createOptimProblem('fmincon', 'objective', mass, 'x0', ...
    x_initial, 'lb', lb, 'ub', ub, 'nonlcon', nonlcon, 'options', options);

% Create a GlobalSearch object
gs = GlobalSearch;

% Run the optimization using GlobalSearch
[x_final, final_mass, exitflag] = run(gs, problem);

% Analyze the results
if exitflag > 0
    disp('Optimal Solution to resist Stress');
    disp(x_final);
    disp('Optimal Mass Value');
    disp(final_mass);
else
    disp('Optimization did not converge to a solution');
end

%% Plan view of HT (ribs and stringers)
clc
close all
```

```
tlength = 4.18;
r_chord = 2.4773;
t_chord = 0.35*r_chord;

% root chord
h_rc = [0,r_chord];
x_rc = [0,0];

figure()

plot(x_rc,h_rc,'LineWidth',1,'Color','b')
xlim([0  6]);

grid on;
hold on;

% flexural axis
plot([0, tlength],[1.4864,-0.7054],'LineWidth',1,'Color','r', ...
    'LineStyle','--')
m_qc = (-0.7054-1.4864)/tlength;
qc_eq = @(x) m_qc*x + 1.4864;

% leading edge
plot([0, tlength],[r_chord, r_chord*0.75 + (tlength*sind(-35.6)) + ...
    t_chord*0.25],'LineWidth',1,'Color','b')

% trailing edge
plot([0, tlength],[0, r_chord*0.75 + (tlength*sind(-35.6)) - ...
    t_chord*0.75],'LineWidth',1,'Color','b')

% tip chord
h_tc = [r_chord*0.75 + (tlength*sind(-35.6)) - t_chord*0.75,  ...
    r_chord*0.75 + (tlength*sind(-35.6)) + t_chord*0.25];
x_tc = [tlength,tlength];
plot(x_tc,h_tc,'LineWidth',1,'Color','b')

% wing box
plot([0, tlength],[0.35*r_chord, r_chord*0.75 + (tlength*sind(-35.6)) ...
    - t_chord*0.4],'LineWidth',1,'Color','r')
plot([0, tlength],[0.85*r_chord, r_chord*0.75 + (tlength*sind(-35.6)) + ...
    t_chord*0.1],'LineWidth',1,'Color','r')
plot([0, 0],[0.35*r_chord, 0.85*r_chord],'LineWidth',1,'Color','r')
plot([tlength, tlength],[r_chord*0.75 + (tlength*sind(-35.6)) - ...
    t_chord*0.4, r_chord*0.75 + (tlength*sind(-35.6)) + t_chord*0.1],...
    'LineWidth',1,'Color','r')

% ribs
rib_location = 2;

m_ribs = -1/m_qc;

y_ribs = r_chord*0.75 + (rib_location.*sind(-35.6));
x_ribs = rib_location.*cosd(-35.6);
c_ribs = y_ribs - m_ribs.*x_ribs;
```

```
ribs_eq = @(x) c_ribs + m_ribs*x;

m_box_upper = (-0.488594-2.1057)/tlength;
box_upper = @(x) m_box_upper*(x) + 2.1057;

m_box_lower = (-0.922121 - 0.867055)/tlength;
box_lower = @(x) m_box_lower*(x) + 0.867055;

x_inter_upper = (2.1057 - c_ribs)/(m_ribs-m_box_upper);
x_inter_lower = (0.867055 - c_ribs)/(m_ribs-m_box_lower);
x_ribrange = linspace(x_inter_lower,x_inter_upper,2);
plot(x_ribrange,ribs_eq(x_ribrange),'Color','magenta');

x_inter1 = [];

for i = 1:2
    c_str(i) = 2.1057 - ((2.1057 - 0.867055)/7)*(i);
    x_inter1(i) = (2.1057 - c_str(i))/(m_qc-m_box_upper);
    x_strrange = linspace(0,x_inter1(i),2);
    str_eq = @(x) m_qc*x_strrange + c_str(i);
    plot(x_strrange,str_eq(x_strrange),'Color','green');
end

for i = 3:4
    c_str(i) = 2.1057 - ((2.1057 - 0.867055)/7)*(i);
    x_strrange = linspace(0,4.18,2);
    str_eq = @(x) m_qc*x_strrange + c_str(i);
    plot(x_strrange,str_eq(x_strrange),'Color','green');
end

x_inter2 = [];

for i = 5:6
    c_str(i) = 2.1057 - ((2.1057 - 0.867055)/7)*(i);
    x_inter2(i-4) = (0.867055 - c_str(i))./(m_qc-m_box_lower);
    x_strrange = linspace(0,x_inter2(i-4),2);
    str_eq = @(x) m_qc*x_strrange + c_str(i);
    plot(x_strrange,str_eq(x_strrange), 'Color','green');
end

ylim([-2 3])
xlim([0 4.5])

%% Stringers x span plot
close all
clc

span = [0, 1.8371,1.8377,3.6741,3.6753];
stringers = [6, 5, 4, 3, 2];

figure()
plot(span, stringers,'Marker','o')
xlabel('Spanwise direction','FontWeight','bold')
ylabel('Number of stringers','FontWeight','bold')
```

```
xlim([0  4.18])
ylim([2  7])

%%
function [c, ceq] = combinedconstraints(x)

% setup variables
% CHECK THESE AFTER DOING MATERIAL SELECTION
M_yy = 6646942.7;
direct_yield_stress = 4.31*10^8;
comp_yield_stress = 4.42e+8;
shear_yield_stress = 3.1e+08;
aluminium_ym = 7.2e+10;
D = 2.81;
K = 3.62;
L = 0.5;
h_box = 0.454;
LD = 35;
M_xx = M_yy/LD;
w_length = 4.18;

I_xx = 1/6 * ((x(2)-2*x(1))*x(1)) * (x(3)*x(1)*2) * x(4)*(x(4)+1)*...
    (x(4)*2 + 1)*(r_chord/x(4)) + x(5)*w_length^3/4;

%Stress due to bending constraint inequality:
c(1) = direct_yield_stress - (M_yy*(h_box/2))/((x(2)-2*x(1))*x(1)) + ...
    (x(3)*x(1)*2) * x(4)*(h_box/2)^2 * 2 - M_xx*r_chord/(1/6 * ...
    ((x(2)-2*x(1))*x(1)) * (x(3)*x(1)*2) * x(4)*(x(4)+1)*(x(4)*2 + 1)*...
    (r_chord/x(4)) + x(5)*w_length^3/4);

%Buckling constraint inequality:
c(2) = comp_yield_stress - (pi^2*aluminium_ym * ((2*((x(2)-x(1))/2)^2)*...
    (x(3)*x(1)) + x(1)/12*(x(2)-2*x(1))^3 )) / ((x(3)*x(1)*2+((x(2)-2*...
    x(1))*x(1)))*L^2);
c(3) = shear_yield_stress - K*aluminium_ym*(x(5)/(((r_chord/x(4))))^2);

ceq = [];

end
```

A.3.6 Vertical Tail Optimisation and Plotting Code

```
clear
close all
clc

% SI units used

% place ribs at known locations, treat space between locations as "section"
% and iterate through placing different of ribs in order to withstand
% buckling

%% Rib optimsiation
close all
clc
```

```
% place ribs at known locations, treat space between locations as "section"
% and iterate through placing different of ribs in order to withstand
% buckling

shear_loads = [22449.2461417641
22391.9631776752
22334.6804436386
22277.3981697094
22220.1165859481
22162.8359224237
22105.5564092162
22048.2782764196
21991.0017541442
21933.7270725201
21876.4544616993
21819.1841518590
21761.9163732039
21704.6513559694
21647.3893304242
21590.1305268730
21532.8751756594
21475.6235071688
21418.3757518310
21361.1321401229
21303.8929025716
21246.6582697571
21189.4284723149
21132.2037409391
21074.9843063851
21017.7703994722
20960.5622510867
20903.3600921849
20846.1641537951
20788.9746670216
20731.7918630464
20674.6159731328
20617.4472286281
20560.2858609663
20503.1321016708
20445.9861823577
20388.8483347385
20331.7187906226
20274.5977819209
20217.4855406479
20160.3822989252
20103.2882889841
20046.2037431686
19989.1288939383
19932.0639738711
19875.0092156667
19817.9648521487
19760.9311162685
19703.9082411072
19646.8964598797
```

19589.8960059365
19532.9071127677
19475.9300140053
19418.9649434265
19362.0121349564
19305.0718226716
19248.1442408026
19191.2296237372
19134.3282060231
19077.4402223718
19020.5659076606
18963.7054969366
18906.8592254191
18850.0273285031
18793.2100417621
18736.4076009516
18679.6202420119
18622.8482010713
18566.0917144493
18509.3510186598
18452.6263504141
18395.9179466244
18339.2260444067
18282.5508810840
18225.8926941899
18169.2517214713
18112.6282008922
18056.0223706365
17999.4344691116
17942.8647349515
17886.3134070203
17829.7807244152
17773.2669264703
17716.7722527595
17660.2969431003
17603.8412375567
17547.4053764432
17490.9896003276
17434.5941500349
17378.2192666505
17321.8651915239
17265.5321662718
17209.2204327821
17152.9302332171
17096.6618100171
17040.4154059039
16984.1912638848
16927.9896272555
16871.8107396043
16815.6548448154
16759.5221870730
16703.4130108643
16647.3275609838
16591.2660825367
16535.2288209429

16479.2160219405
16423.2279315897
16367.2647962767
16311.3268627174
16255.4143779614
16199.5275893956
16143.6667447485
16087.8320920938
16032.0238798546
15976.2423568071
15920.4877720849
15864.7603751828
15809.0604159610
15753.3881446492
15697.7438118504
15642.1276685455
15586.5399660972
15530.9809562541
15475.4508911550
15419.9500233336
15364.4786057218
15309.0368916549
15253.6251348756
15198.2435895382
15142.8925102134
15087.5721518923
15032.2827699910
14977.0246203555
14921.7979592655
14866.6030434394
14811.4401300389
14756.3094766735
14701.2113414053
14646.1459827533
14591.1136596987
14536.1146316892
14481.1491586442
14426.2175009591
14371.3199195108
14316.4566756621
14261.6280312669
14206.8342486752
14152.0755907381
14097.3523208126
14042.6647027674
13988.0130009871
13933.3974803783
13878.8184063741
13824.2760449401
13769.7706625789
13715.3025263363
13660.8719038059
13606.4790631354
13552.1242730314
13497.8078027652

13443.5299221786
13389.2909016891
13335.0910122961
13280.9305255863
13226.8097137396
13172.7288495348
13118.6882063560
13064.6880581976
13010.7286796715
12956.8103460121
12902.9333330831
12849.0979173831
12795.3043760525
12741.5529868793
12687.8440283053
12634.1777794331
12580.5545200319
12526.9745305446
12473.4380920939
12419.9454864892
12366.4969962331
12313.0929045286
12259.7334952854
12206.4190531270
12153.1498633978
12099.9262121701
12046.7483862510
11993.6166731898
11940.5313612850
11887.4927395920
11834.5010979301
11781.5567268900
11728.6599178417
11675.8109629417
11623.0101551410
11570.2577881927
11517.5541566597
11464.8995559230
11412.2942821895
11359.7386325002
11307.2329047382
11254.7773976372
11202.3724107896
11150.0182446552
11097.7152005698
11045.4635807533
10993.2636883192
10941.1158272827
10889.0203025700
10836.9774200273
10784.9874864300
10733.0508094915
10681.1676978728
10629.3384611922
10577.5634100342

10525.8428559597
10474.1771115154
10422.5664902437
10371.0113066929
10319.5118764270
10268.0685160361
10216.6815431465
10165.3512764314
10114.0780356212
10062.8621415145
10011.7039159885
9960.60368201029
9909.56176364781
9858.57848608096
9807.65417561304
9756.78915968220
9705.98376687305
9655.23832692840
9604.55317076120
9553.92863046654
9503.36503933390
9452.86273185943
9402.42204375854
9352.04331197849
9301.72687471121
9251.47307140635
9201.28224278436
9151.15473084983
9101.09087890500
9051.09103156339
9001.15553476363
8951.28473578354
8901.47898325428
8851.73862717474
8802.06401892618
8752.45551128692
8702.91345844740
8653.43821602526
8604.03014108082
8554.68959213259
8505.41692917310
8456.21251368490
8407.07670865682
8358.00987860040
8309.01238956660
8260.08460916271
8211.22690656948
8162.43965255859
8113.72321951022
8065.07798143098
8016.50431397203
7968.00259444756
7919.57320185338
7871.21651688593
7822.93292196145

7774.72280123555
7726.58654062292
7678.52452781747
7630.53715231269
7582.62480542234
7534.78788030143
7487.02677196753
7439.34187732243
7391.73359517407
7344.20232625883
7296.74847326423
7249.37244085182
7202.07463568063
7154.85546643080
7107.71534382771
7060.65468066638
7013.67389183641
6966.77339434714
6919.95360735332
6873.21495218119
6826.55785235494
6779.98273362362
6733.49002398850
6687.08015373083
6640.75355544016
6594.51066404301
6548.35191683209
6502.27775349601
6456.28861614943
6410.38494936381
6364.56720019861
6318.83581823305
6273.19125559844
6227.63396701100
6182.16440980536
6136.78304396852
6091.49033217451
6046.28673981962
6001.17273505824
5956.14878883935
5911.21537494372
5866.37297002171
5821.62205363176
5776.96310827965
5732.39661945842
5687.92307568903
5643.54296856182
5599.25679277872
5555.06504619621
5510.96822986917
5466.96684809552
5423.06140846173
5379.25242188914
5335.54040268134
5291.92586857224

5248.40934077531
5204.99134403363
5161.67240667100
5118.45306064408
5075.33384159551
5032.31528890818
4989.39794576056
4946.58235918317
4903.86908011619
4861.25866346829
4818.75166817669
4776.34865726845
4734.05019792304
4691.85686153630
4649.76922378571
4607.78786469706
4565.91336871261
4524.14632476069
4482.48732632681
4440.93697152642
4399.49586317919
4358.16460888498
4316.94382110157
4275.83411722407
4234.83611966620
4193.95045594338
4153.17775875783
4112.51866608554
4071.97382126535
4031.54387309013
3991.22947590008
3951.03128967831
3910.94998014869
3870.98621887610
3831.14068336914
3791.41405718530
3751.80703003884
3712.32029791129
3672.95456316473
3633.71053465803
3594.58892786594
3555.59046500136
3516.71587514066
3477.96589435241
3439.34126582939
3400.84274002417
3362.47107478830
3324.22703551523
3286.11139528717
3248.12493502591
3210.26844364788
3172.54271822343
3134.94856414071
3097.48679527406
3060.15823415732

3022.96371216205
2985.90406968095
2948.98015631672
2912.19283107633
2875.54296257129
2839.03142922378
2802.65911947909
2766.42693202460
2730.33577601541
2694.38657130712
2658.58024869589
2622.91775016603
2587.40002914569
2552.02805077065
2516.80279215682
2481.72524268168
2446.79640427513
2412.01729172002
2377.38893296298
2342.91236943582
2308.58865638814
2274.41886323144
2240.40407389556
2206.54538719767
2172.84391722468
2139.30079372955
2105.91716254221
2072.69418599576
2039.63304336873
2006.73493134418
1974.00106448646
1941.43267573653
1909.03101692676
1876.79735931625
1844.73299414775
1812.83923322717
1781.11740952713
1749.56887781563
1718.19501531129
1686.99722236666
1655.97692318112
1625.13556654502
1594.47462661694
1563.99560373592
1533.70002527075
1503.58944650848
1473.66545158461
1443.92965445740
1414.38369992917
1385.02926471739
1355.86805857894
1326.90182549072
1298.13234489064
1269.56143298268
1241.19094411068

1213.02277220530
1185.05885230948
1157.30116218786
1129.75172402619
1102.41260622748
1075.28592531191
1048.37384792843
1021.67859298675
995.202433919016
968.947701081632
942.916784308557
917.112135628638
891.536272160857
866.191779202669
841.081313528503
816.207606917102
791.573469928609
767.181795954699
743.035565567631
719.137851197362
695.491822169229
672.100750138864
648.968014965603
626.097111071043
603.491654335656
581.155389593501
559.092198793609
537.306109906395
515.801306665055
494.582139245559
473.653136005043
453.019016417605
432.684705369589
412.655349004062
392.936332337644
373.533298913480
354.452172803796
335.699183336651
317.280892997237
299.204229048652
281.476519536099
264.105534489404
247.099533332251
230.467319756579
214.218305647658
198.362586077977
182.911027967481
167.875375795171
153.268378833135
139.103945898673
125.397335800561
112.165394841640
99.4268575090491
87.2027338203353
75.5168184439694

```
64.3963759037779
53.8730891920111
43.9844189480793
34.7756359373904
26.3030315087001
18.6393732622043
11.8841848571007
6.18647683971004
1.81235368932768
0];

% temp = linspace(0,11.24,113);
%
% plot(temp, loads);
taillength = 4.88;
disc = linspace(0,taillength,500);

lfunc = @(x) 1.371.*x.^6 - 16.26.*x.^5 + 77.07.*x.^4 - 128.6.*x.^3 + ...
    174.5.*x.^2 - 5928.*x + 2.246e4;

ldisc = lfunc(disc);

figure()
plot(disc,ldisc);

% ribs placed at known concentrated loads
E = 72e9;
density = 2780;
k = 3.62;

r_chord = 3.15548;
t_chord = r_chord*0.7;

ribs_known = [0, taillength];

pcrit = @(a) k*E*(a)^2;
% "a" is equal to t/b here

optrib = [];

for panels = 1:1
    t_skin = 1e-3;
    % optimal skin thickness obtained using fmincon function
            for nribs = 1:30
                ribspacing = (ribs_known(panels + 1) - ...
                    (ribs_known(panels)))/nribs;
                a = t_skin/ribspacing;
                averageload = (lfunc(ribs_known(panels)) + ...
                    lfunc(ribs_known(panels)+ribspacing))/2;
                if pcrit(a) < averageload
                    nribs = nribs + 1;
                else
                    disp(nribs)
                    break
                end
```

```
            end
        optrib(panels) = ribspacing;
end
disp(optrib)
rib_location = 2;
% note these values are spanwise, along the quarter-chord line

%% Stringer optimisation

% setup variables
% CHECK THESE AFTER DOING MATERIAL SELECTION
aluminium_density = 2.78e+03;
taillength = 4.88;

% Initial values of the optimisation variables
skin_thickness = 0.05;
number = 15;
thickness = 0.01;
height = 0.01;
width = 0.01;

x_initial = [thickness, height, width, number, skin_thickness];

% The mass function which is what is reduced
mass = @(x) (((x(2)-2*x(1))*x(1)) + (x(3)*x(1)*2) ...
    *aluminium_density)*x(4) + (taillength*2.71*x(5)*aluminium_density);

% The upper and lower bounds of each variable
lb = [0.001, 0.001, 0.0010, 6, 0.0010];
ub = [0.1, 0.1, 0.1, 20, 0.005];

nonlcon = @combinedconstraints;

options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm',...
    'interior-point');

% Define the problem as a GlobalSearch problem
problem = createOptimProblem('fmincon', 'objective', mass, 'x0',...
    x_initial, 'lb', lb, 'ub', ub, 'nonlcon', nonlcon, 'options', options);

% Create a GlobalSearch object
gs = GlobalSearch;

% Run the optimization using GlobalSearch
[x_final, final_mass, exitflag] = run(gs, problem);

% Analyze the results
if exitflag > 0
    disp('Optimal Solution to resist Stress');
    disp(x_final);
    disp('Optimal Mass Value');
    disp(final_mass);
else
    disp('Optimization did not converge to a solution');
end
```

```
%% Aerofoil plot with wingbox
clc
close all

geom=table2array(readtable("eppler.dat"));
x = geom(:,1);
y = geom(:,2);

figure()
plot(x,y,'Color','b','LineWidth',1);
grid on;
hold on;

chord = table2array(readtable("XYZc1.txt"));
xc = chord(:,1);
yc = chord(:,2);
plot(xc,yc,'--','Color','b');

htop = ((0.0384 - -0.0384) + (0.03 - -0.03))/4;
hbot = -htop;

v = [hbot, htop];
xl = [0.15,0.15];
plot(xl,v,'LineWidth',1,'Color','r');

xr = [0.65,0.65];
plot(xr,v,'LineWidth',1,'Color','r');

h = [0.15,0.65];
ytop = [htop, htop];
plot(h,ytop,'LineWidth',1,'Color','r');

ybot = [hbot, hbot];
plot(h,ybot,'LineWidth',1,'Color','r');

plot(0.4,0,'LineWidth',1,'Marker','x','Color','k')

xlabel('Chordwise co-ordinate: x/c','FontWeight','bold')
legend('Aerofoil geometry', 'Chord line', '','','', ...
    'Rectangular wingbox', 'Shear centre')
axis equal;

%% side view of VT (with ribs and stringers)
clc
close all

r_chord = 3.15548;
t_chord = r_chord*0.7;
% root chord
h_rc = [0,0];
x_rc = [0,r_chord];

figure()
```

```
plot(x_rc,h_rc,'LineWidth',1,'Color','b')
xlim([0 10]);
hold on;
% tip chord

% leading edge
plot([0,r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75],[0,4.88],...
    'LineWidth',1,'Color','b')
% trailing edge
plot([r_chord, r_chord*0.75 + (4.88*sind(40)) + t_chord*0.25],[0,4.88],...
    'LineWidth',1,'Color','b')
% tip chord
h_tc = [r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75,  r_chord*0.75 + ...
    (4.88*sind(40)) + t_chord*0.25];
y_tc = [4.88,4.88];
plot(h_tc,y_tc,'LineWidth',1,'Color','b')

% wing box
plot([0.15*r_chord, r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.15*t_chord],[0, 4.88],'LineWidth',1,'Color','r')
plot([0.65*r_chord, r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.65*t_chord],[0,4.88],'LineWidth',1,'Color','r')
plot([0.15*r_chord, 0.65*r_chord],[0,0],'LineWidth',1,'Color','r')
plot([r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + 0.15*t_chord, ...
    r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + 0.65*t_chord],...
    [4.88,4.88],'LineWidth',1,'Color','r')

% flexural axis
mid_box_tip = ((r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.15*t_chord) + (r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.65*t_chord))/2;
plot([0.4*r_chord,mid_box_tip],[0,4.88],'LineWidth',1,'Color','r', ...
    'LineStyle','--')
m_qc = 4.88/(mid_box_tip-0.4*r_chord);

rib_location = 2;

m_box_left = 4.88/(r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.15*t_chord - 0.15*r_chord);
box_left = @(x) m_box_left*(x - 0.15*r_chord);
m_box_right = 4.88/(r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.65*t_chord - 0.65*r_chord);
box_right = @(x) m_box_upper*(x - 0.65*r_chord);

m_ribs = -1/m_qc;

x_ribs = r_chord*0.75 + (rib_location.*cosd(40));
y_ribs = rib_location.*sind(40);
c_ribs = y_ribs - m_ribs.*x_ribs;

ribs_eq = @(x) c_ribs + m_ribs*x;

x_inter_upper = (-0.15*m_box_left*r_chord - c_ribs)/(m_ribs-m_box_left);
x_inter_lower = (-0.65*m_box_right*r_chord - c_ribs)/(m_ribs-m_box_right);
x_ribrange = linspace(x_inter_lower,x_inter_upper,2);
```

```
plot(x_ribrange,ribs_eq(x_ribrange),'Color','magenta');

grid on;
xlim([0 7]);

x_start = 0.473322 + (2.0516-0.473322)/7;
x_inter1 = ((m_qc*x_start) - (0.15*r_chord*m_box_left))/(m_qc-m_box_left);
x_strrange = linspace(x_start,x_inter1,2);
str_eq = @(x) m_qc*(x - x_start);
plot(x_strrange,str_eq(x_strrange),'Color','green');

x_inter2 = [];

for i = 2:5
    x_start(i) = 0.473322 + (2.0516-0.473322)*i/7;
    x_inter2(i-1) = (4.88/m_qc) + x_start(i);
    x_strrange = linspace(x_start(i),x_inter2(i-1),2);
    str_eq = @(x) m_qc*(x - x_start(i));
    plot(x_strrange,str_eq(x_strrange),'Color','green');
end

x_start = 0.473322 + (2.0516-0.473322)*6/7;
x_inter3 = ((m_qc*x_start) - (0.65*r_chord*m_box_right))...
    /(m_qc-m_box_right);
x_strrange = linspace(x_start,x_inter3,2);
str_eq = @(x) m_qc*(x - x_start);
plot(x_strrange,str_eq(x_strrange),'Color','green');

%% Stringers x span plot
close all
clc

span = [0, 4.0029,5.1224];
stringers = [6, 5, 4];

figure()
plot(span, stringers,'Marker','o')
xlabel('Spanwise direction','FontWeight','bold')
ylabel('Number of stringers','FontWeight','bold')

xlim([0 5.1224])
ylim([3 7])

%%

function [c, ceq] = combinedconstraints(x)

% setup variables
% CHECK THESE AFTER DOING MATERIAL SELECTION
M_yy = 6646942.7;
direct_yield_stress = 4.31*10^8;
comp_yield_stress = 4.42e+8;
shear_yield_stress = 3.1e+08;
aluminium_ym = 7.2e+10;
D = 2.81;
```

```
K = 3.62;
L = 0.5;
h_box = 0.454;
LD = 35;
M_xx = M_yy/LD;
w_length = 4.88;

I_xx = 1/6 * ((x(2)-2*x(1))*x(1)) * (x(3)*x(1)*2) * x(4)*(x(4)+1)*...
    (x(4)*2 + 1)*(5.16/x(4)) + x(5)*w_length^3/4;

%Stress due to bending constraint inequality:
c(1) = direct_yield_stress - (M_yy*(h_box/2))/((x(2)-2*x(1))*x(1)) + ...
    (x(3)*x(1)*2) * x(4)*(h_box/2)^2 * 2 - M_xx*5.16/(1/6 * ...
    ((x(2)-2*x(1))*x(1)) * (x(3)*x(1)*2) * x(4)*(x(4)+1)*(x(4)*2 + 1)...
    *(5.16/x(4)) + x(5)*w_length^3/4);

%Buckling constraint inequality:
c(2) = comp_yield_stress - (pi^2*aluminium_ym * ((2*((x(2)-x(1))/2)^2)*...
    (x(3)*x(1)) + x(1)/12*(x(2)-2*x(1))^3 )) / ...
    ((x(3)*x(1)*2+((x(2)-2*x(1))*x(1)))*L^2);
c(3) = shear_yield_stress - K*alumiSnium_ym*(x(5)/(((5.16/x(4))))^2);

ceq = [];

end
```

A.3.7 Vertical Tail Distribution Code

```
%% Wing Loading
clear
clc

%Set some constants here
W0 = 79.15*5150;
n = 1;
%%% Determines magnitude of Vertical Tailplane force seperate to Rudder.
%%% i.e. g_n = 1: SLF, g_n = 0: Banked Flight.
g_n = 0;
Lv = 2.2450e+04; %Taken from Tail_Calcs
%%% Ref area = b*cbar, AR = b/cbar so b^2 - S_ref*AR
Ref_area = 13.23;
AR = 1.8;
b = sqrt(Ref_area*AR);
L0 = 2*(n)*Lv / (pi*0.5*b); %/17.6; % For width of 1
c_bar = b/AR;
Taper_Ratio = 0.7;
%%% 0.5(Chord_root+Chord_tip)=c_bar, Chord_tip/Chord_root=Taper_Ratio ->
%%% 0.5(Chord_root(1+taper_ratio) = c_bar so Chord_root = 2c_bar/(1+taper)
Chord_root = 2*c_bar/(1+Taper_Ratio);
Chord_tip = Taper_Ratio*Chord_root;
Sweep_c4 = 40;
fw = 0;
%Wing_weight = 0.01*W0; %Raymer
Wing_weight = 0; % Does not contribute to Shear here
%%% Chord ratios
```

```
spar_f = 0.1;
spar_b = 0.7;
chord_ac = spar_b + 0.25*(1-spar_b);
chord_cg = 0.5;
chord_fin = 0.25*spar_b;
chord_cont = 0.7; % Control surfaces
span_cont = [0.10 0.90]; % First is start of semi-span, second end.



%% Lift Distribution
%Span-wise Lift distribution
minY = 0;
maxY = b;
Ny = 500;

L_span = @(y) (Lift_Span(y, fw, L0, 2*b));

%% Wing Shape
Wing_chord = @(y) (Chord_root * (1 - (1 - (Chord_tip/Chord_root)) ...
    *(abs(y)/(b) ) ) );
Wing_Sweep = @(y) (-abs(y)*tand(Sweep_c4));

Wing_LE = @(y) Wing_Sweep(y) + 0.25*Wing_chord(y);
Wing_TE = @(y) Wing_Sweep(y) - 0.75*Wing_chord(y);


%% Plotting
%%% Fig1
fig1 = figure(1);
tcl = tiledlayout(2,2,'TileSpacing','compact');
nexttile
% Span-wise Lift Distribution
fplot(L_span, [minY maxY], LineWidth=3)
title("Span-wise Lift Distribution")
ylabel("L_0 (Nm^{-1})")
xlabel("y (m)")

nexttile

% Wing Spar and flexural axis
fplot(Wing_LE, [minY maxY], LineWidth=3, Color='b')
hold on
fplot(Wing_TE, [minY maxY], LineWidth=3, Color='b')
line([maxY maxY], [Wing_LE(maxY) Wing_TE(maxY)], 'LineWidth', 3, 'Color' ...
    , 'b')
fplot(@(y) Wing_LE(y) - spar_f*Wing_chord(y), [0, maxY], 'LineWidth', 3 ...
    , 'Color', '#A2142F', 'LineStyle', ':')
fplot(@(y) Wing_LE(y) - spar_b*Wing_chord(y), [0, maxY], 'LineWidth', 3 ...
    , 'Color', 'k', 'LineStyle', '-.')
fplot(@(y) Wing_LE(y) - 0.5*(spar_f+spar_b)*Wing_chord(y), [0, maxY] ...
    , 'LineWidth', 2, 'Color', 'b', 'LineStyle', '--')

qw{1} = plot(nan, 'b', LineWidth=3);
qw{2} = plot(nan, 'b--', LineWidth=2);
```

```
qw{3} = plot(nan, 'k-.', LineWidth=3);
qw{4} = plot(nan, ':', LineWidth=3, Color="#A2142F");
legend([qw{:}], {'Tail-plane','Flexural Axis','Rear Spar', 'Front Spar'} ...
    , 'location', 'best')
hold off

title("Horoizontal Fin Geometry")
xlabel("y (m)")
ylabel("x (m)")

nexttile([1 2])

%%% Wing Shape

fplot(Wing_LE, [minY maxY], LineWidth=3, Color='b')
hold on
fplot(Wing_TE, [minY maxY], LineWidth=3, Color='b')
% Quarter chord sweep
fplot(Wing_Sweep, [minY maxY], LineStyle="--", LineWidth=2, Color='b')
line([maxY maxY], [Wing_LE(maxY) Wing_TE(maxY)], 'LineWidth', 3, 'Color' ...
    , 'b')
line([minY minY], [Wing_LE(minY) Wing_TE(minY)], 'LineWidth', 3, 'Color' ...
    , 'b')

% Vertical Attach point
line([-fw -fw], [Wing_LE(0) Wing_TE(0)], 'LineWidth', 3, 'Color', 'black' ...
    , 'LineStyle', '-.')
line([fw fw], [Wing_LE(0) Wing_TE(0)], 'LineWidth', 3, 'Color', 'black' ...
    , 'LineStyle', '-.')

% Control surfaces
fplot(@(y) Wing_LE(y) - chord_cont*Wing_chord(y), [b*span_cont(1) ...
    , b*span_cont(2)], 'LineWidth', 3, 'Color', '#A2142F', 'LineStyle', ':')

line([b*span_cont(2) b*span_cont(2)], ...
    [Wing_LE(b*span_cont(2))-chord_cont*Wing_chord(b*span_cont(2)) ...
    , Wing_LE(b*span_cont(2))-Wing_chord(b*span_cont(2))], 'LineWidth', 3 ...
    , 'Color', '#A2142F', 'LineStyle', ':')
line([b*span_cont(1) b*span_cont(1)], ...
    [Wing_LE(b*span_cont(1))-chord_cont*Wing_chord(b*span_cont(1)) ...
    , Wing_LE(b*span_cont(1))-Wing_chord(b*span_cont(1))], 'LineWidth', 3 ...
    , 'Color', '#A2142F', 'LineStyle', ':')


% Custom legend
qw{1} = plot(nan, 'b', LineWidth=3);
qw{2} = plot(nan, 'b--', LineWidth=2);
qw{3} = plot(nan, 'k-.', LineWidth=3);
qw{4} = plot(nan, ':', LineWidth=3, Color="#A2142F");
legend([qw{:}], {'Wing','Quater-Chord','Centre line', 'Control Surfaces'} ...
    , 'location', 'best')
hold off

title("Tail-plane Layout")
xlabel("y (m)")
```

```
ylabel("x (m)")

fontsize(gcf, scale=1.3)

%% 2-D Lift Integral
Y = linspace(minY, maxY, Ny);
Z = zeros(1, length(Y));  % Wing Lift
Z2 = zeros(1, length(Y)); % Wing Weight

Wing_area = arrayfun(@(y) (Wing_chord(y) / (0.5*(Chord_root+Chord_tip)))^2, Y);
Wing_density = Wing_weight / trapz(Y, Wing_area);

%%% Find moment arms for Torque Calculations
p_lift = arrayfun(@(y) (0.5*(spar_f+spar_b) - chord_ac)*Wing_chord(y), Y);
% From Aerodynamic centre
p_weight = arrayfun(@(y) (0.5*(spar_f+spar_b) - chord_cg)*Wing_chord(y) , Y);
% From Wing CG
p_fin = arrayfun(@(y) (0.5*(spar_f+spar_b) - chord_fin)*Wing_chord(y), Y);
% From Gear strut placement


for i = 1:length(Y)
    Z(i) = L_span(Y(i));
    Z2(i) = Wing_density*Wing_area(i)*inStruct2D(Y(i), minY-1, maxY);
end


Z2 = abs(n)*Z2;
Z3 = g_n*Z;

Total_Rudder_Lift = trapz(Y, Z);     % Check if Lift = n*W0
Total_Wing_Weight = trapz(Y, Z2);    % Check Weight of Wing
Total_Fin_Lift = trapz(Y, Z3);       % Check Fuel weight

Z_Sum_Span = Z - Z2 - Z3;
Z_Tor_Span = p_lift.*Z - p_weight.*Z2 - p_fin.*Z3;
%

%% Quick test
%fplot(@(y) (Wing_chord(y) / (0.5*(Chord_root+Chord_tip)))^2, [minY, maxY])

%% Shear loads

size_Y = 0;
DSF = zeros(Ny, 1);
DT = zeros(Ny, 1);
for i = 1:Ny
    if i == Ny
        DSF(i) = 0.5*(Z_Sum_Span(size_Y + i))*(Y(end)-Y(end-1));
        DT(i) = 0.5*(Z_Tor_Span(size_Y + i))*(Y(end)-Y(end-1));
    else
        DSF(i) = ( 0.5*(Z_Sum_Span(size_Y + i) + Z_Sum_Span(size_Y + i+1)) ...
            * (Y(size_Y + i+1)-Y(size_Y + i)) );
        DT(i) = ( 0.5*(Z_Tor_Span(size_Y + i) + Z_Tor_Span(size_Y + i+1)) ...
            * (Y(size_Y + i+1)-Y(size_Y + i)) );
```

```
    end
end

SF = flip(cumsum(flip(DSF(1:end))));
T = flip(cumsum(flip(DT)));

DBM = zeros(length(SF), 1);
for i = 1:Ny
    if i == Ny
        DBM(i) = 0.5*(SF(i))*(Y(end)-Y(end-1));
    else
        DBM(i) = ( 0.5*(SF(i) + SF(i+1)) * (Y(size_Y + i+1)-Y(size_Y + i)) );
    end
end

BM = flip(cumsum(flip(DBM)));

%% Plotting
fig3 = figure(3);
tcl2 = tiledlayout(2,2,'TileSpacing','compact');
nexttile
plot(Y, SF)

title("Shear Force Distribution")
xlabel("y (m)")
ylabel("SF (N)")

nexttile
plot(Y, BM)

title("Bending Moment Distribution")
xlabel("y (m)")
ylabel("BM (Nm)")

nexttile
plot(Y, T)

title("Torsion Moment Distribution")
xlabel("y (m)")
ylabel("TM (Nm)")

nexttile
plot(Y, DSF)

title("Force Distribution")
xlabel("y (m)")
ylabel("DF (N)")

fontsize(gcf, scale=1.3)

% save("plotVT" + num2str(floor(n)) + num2str(floor(g_n)), 'Y', 'SF', 'BM' ...
%       , 'T', 'DSF')


%% Function definitions
```

```
% Corrected Lift Span to compensate fuselage.
function [L] = Lift_Span(y, f_width, L0, b)
    if abs(y) < f_width
        L = 0;
    else
        L = L0 * sqrt(1 - (y / (b/2)).^2 );
    end
end

% 2-D check if within boundaries.
function [D] = inStruct2D(y, b1, b2)
    if y > b1 && y < b2
        D = 1;
    else
        D = 0;
    end
end
```

A.3.8 Enlarged Vertical Tailplane Layout

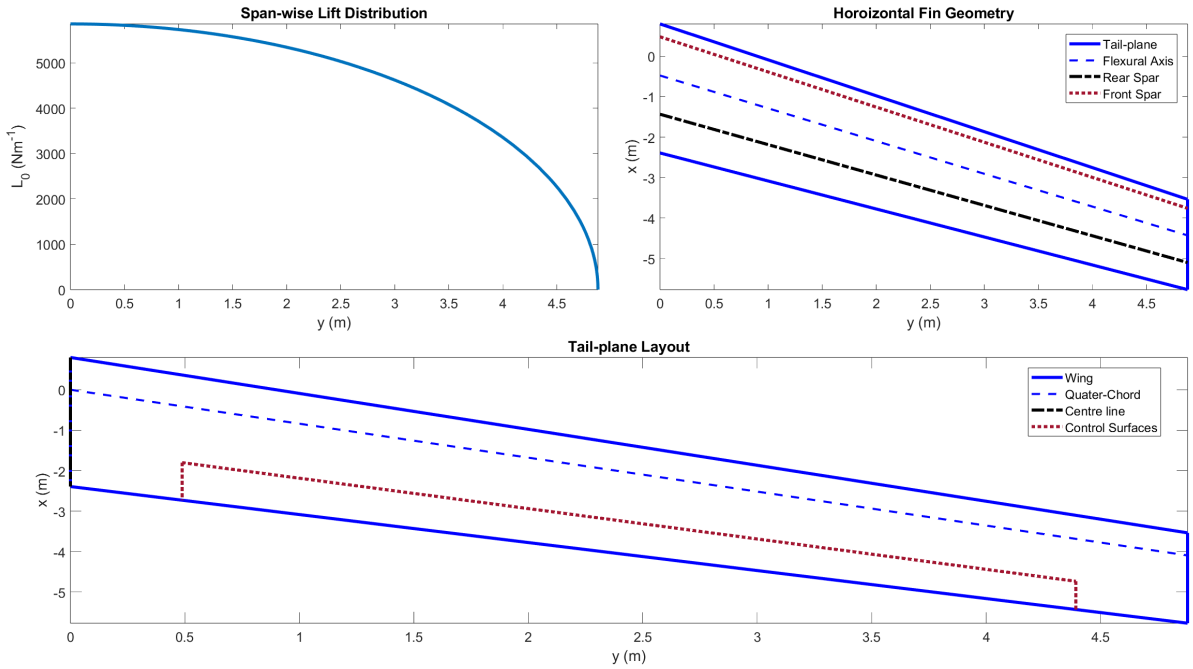


Figure A.7: Vertical Tail-plane

A.3.9 Vertical Tail Plotting Code

```
%% Wing Loading Plot all on one fig
clear
clc

case1 = load("plotVT10.mat");    % Banked Right Engine Failure
case2 = load("plotVT-10.mat");   % Banked Left Engine Failure
case4 = load("plotVT11.mat");    % Steady Level Flight

tcl = tiledlayout(2,2,'TileSpacing','compact');
```

```
nexttile
plot(case1.Y, case1.DSF, LineWidth=2)
hold on
plot(case4.Y, case4.DSF, "--", LineWidth=2)
plot(case2.Y, case2.DSF, LineWidth=2)
hold off

title("Total Force Distribution")
xlabel("y (m)")
ylabel("F (N)")
legend({"Banked Flight: R.E–F", "Wings Level", "Banked Flight L.E–F"})

nexttile
plot(case1.Y, case1.SF, LineWidth=2)
hold on
plot(case4.Y, case4.SF, "--", LineWidth=2)
plot(case2.Y, case2.SF, LineWidth=2)
hold off

title("Shear Force Distribution")
xlabel("y (m)")
ylabel("SF (N)")
legend({"Banked Flight: R.E–F", "Wings Level", "Banked Flight L.E–F"})

nexttile
plot(case1.Y, case1.BM, LineWidth=2)
hold on
plot(case4.Y, case4.BM, "--", LineWidth=2)
plot(case2.Y, case2.BM, LineWidth=2)
hold off

title("Bending Moment Distribution")
xlabel("y (m)")
ylabel("BM (Nm)")
legend({"Banked Flight: R.E–F", "Wings Level", "Banked Flight L.E–F"})

nexttile
plot(case1.Y, case1.T, LineWidth=2)
hold on
plot(case4.Y, case4.T, "--", LineWidth=2)
plot(case2.Y, case2.T, LineWidth=2)
hold off

title("Torsion Moment Distribution")
xlabel("y (m)")
ylabel("TM (Nm)")
legend({"Banked Flight: R.E–F", "Wings Level", "Banked Flight L.E–F"})

fontsize(gcf, scale=1.5)
```

A.3.10 Vertical Tail Optimisation and Plotting Code

```
clear
close all
clc
```

```
% SI units used

% place ribs at known locations, treat space between locations as "section"
% and iterate through placing different of ribs in order to withstand
% buckling

%% Rib optimsiation
close all
clc

% place ribs at known locations, treat space between locations as "section"
% and iterate through placing different of ribs in order to withstand
% buckling

shear_loads = [22449.2461417641
22391.9631776752
22334.6804436386
22277.3981697094
22220.1165859481
22162.8359224237
22105.5564092162
22048.2782764196
21991.0017541442
21933.7270725201
21876.4544616993
21819.1841518590
21761.9163732039
21704.6513559694
21647.3893304242
21590.1305268730
21532.8751756594
21475.6235071688
21418.3757518310
21361.1321401229
21303.8929025716
21246.6582697571
21189.4284723149
21132.2037409391
21074.9843063851
21017.7703994722
20960.5622510867
20903.3600921849
20846.1641537951
20788.9746670216
20731.7918630464
20674.6159731328
20617.4472286281
20560.2858609663
20503.1321016708
20445.9861823577
20388.8483347385
20331.7187906226
20274.5977819209
20217.4855406479
```

20160.3822989252
20103.2882889841
20046.2037431686
19989.1288939383
19932.0639738711
19875.0092156667
19817.9648521487
19760.9311162685
19703.9082411072
19646.8964598797
19589.8960059365
19532.9071127677
19475.9300140053
19418.9649434265
19362.0121349564
19305.0718226716
19248.1442408026
19191.2296237372
19134.3282060231
19077.4402223718
19020.5659076606
18963.7054969366
18906.8592254191
18850.0273285031
18793.2100417621
18736.4076009516
18679.6202420119
18622.8482010713
18566.0917144493
18509.3510186598
18452.6263504141
18395.9179466244
18339.2260444067
18282.5508810840
18225.8926941899
18169.2517214713
18112.6282008922
18056.0223706365
17999.4344691116
17942.8647349515
17886.3134070203
17829.7807244152
17773.2669264703
17716.7722527595
17660.2969431003
17603.8412375567
17547.4053764432
17490.9896003276
17434.5941500349
17378.2192666505
17321.8651915239
17265.5321662718
17209.2204327821
17152.9302332171
17096.6618100171

17040.4154059039
16984.1912638848
16927.9896272555
16871.8107396043
16815.6548448154
16759.5221870730
16703.4130108643
16647.3275609838
16591.2660825367
16535.2288209429
16479.2160219405
16423.2279315897
16367.2647962767
16311.3268627174
16255.4143779614
16199.5275893956
16143.6667447485
16087.8320920938
16032.0238798546
15976.2423568071
15920.4877720849
15864.7603751828
15809.0604159610
15753.3881446492
15697.7438118504
15642.1276685455
15586.5399660972
15530.9809562541
15475.4508911550
15419.9500233336
15364.4786057218
15309.0368916549
15253.6251348756
15198.2435895382
15142.8925102134
15087.5721518923
15032.2827699910
14977.0246203555
14921.7979592655
14866.6030434394
14811.4401300389
14756.3094766735
14701.2113414053
14646.1459827533
14591.1136596987
14536.1146316892
14481.1491586442
14426.2175009591
14371.3199195108
14316.4566756621
14261.6280312669
14206.8342486752
14152.0755907381
14097.3523208126
14042.6647027674

13988.0130009871
13933.3974803783
13878.8184063741
13824.2760449401
13769.7706625789
13715.3025263363
13660.8719038059
13606.4790631354
13552.1242730314
13497.8078027652
13443.5299221786
13389.2909016891
13335.0910122961
13280.9305255863
13226.8097137396
13172.7288495348
13118.6882063560
13064.6880581976
13010.7286796715
12956.8103460121
12902.9333330831
12849.0979173831
12795.3043760525
12741.5529868793
12687.8440283053
12634.1777794331
12580.5545200319
12526.9745305446
12473.4380920939
12419.9454864892
12366.4969962331
12313.0929045286
12259.7334952854
12206.4190531270
12153.1498633978
12099.9262121701
12046.7483862510
11993.6166731898
11940.5313612850
11887.4927395920
11834.5010979301
11781.5567268900
11728.6599178417
11675.8109629417
11623.0101551410
11570.2577881927
11517.5541566597
11464.8995559230
11412.2942821895
11359.7386325002
11307.2329047382
11254.7773976372
11202.3724107896
11150.0182446552
11097.7152005698

11045.4635807533
10993.2636883192
10941.1158272827
10889.0203025700
10836.9774200273
10784.9874864300
10733.0508094915
10681.1676978728
10629.3384611922
10577.5634100342
10525.8428559597
10474.1771115154
10422.5664902437
10371.0113066929
10319.5118764270
10268.0685160361
10216.6815431465
10165.3512764314
10114.0780356212
10062.8621415145
10011.7039159885
9960.60368201029
9909.56176364781
9858.57848608096
9807.65417561304
9756.78915968220
9705.98376687305
9655.23832692840
9604.55317076120
9553.92863046654
9503.36503933390
9452.86273185943
9402.42204375854
9352.04331197849
9301.72687471121
9251.47307140635
9201.28224278436
9151.15473084983
9101.09087890500
9051.09103156339
9001.15553476363
8951.28473578354
8901.47898325428
8851.73862717474
8802.06401892618
8752.45551128692
8702.91345844740
8653.43821602526
8604.03014108082
8554.68959213259
8505.41692917310
8456.21251368490
8407.07670865682
8358.00987860040
8309.01238956660

8260.08460916271
8211.22690656948
8162.43965255859
8113.72321951022
8065.07798143098
8016.50431397203
7968.00259444756
7919.57320185338
7871.21651688593
7822.93292196145
7774.72280123555
7726.58654062292
7678.52452781747
7630.53715231269
7582.62480542234
7534.78788030143
7487.02677196753
7439.34187732243
7391.73359517407
7344.20232625883
7296.74847326423
7249.37244085182
7202.07463568063
7154.85546643080
7107.71534382771
7060.65468066638
7013.67389183641
6966.77339434714
6919.95360735332
6873.21495218119
6826.55785235494
6779.98273362362
6733.49002398850
6687.08015373083
6640.75355544016
6594.51066404301
6548.35191683209
6502.27775349601
6456.28861614943
6410.38494936381
6364.56720019861
6318.83581823305
6273.19125559844
6227.63396701100
6182.16440980536
6136.78304396852
6091.49033217451
6046.28673981962
6001.17273505824
5956.14878883935
5911.21537494372
5866.37297002171
5821.62205363176
5776.96310827965
5732.39661945842

5687.92307568903
5643.54296856182
5599.25679277872
5555.06504619621
5510.96822986917
5466.96684809552
5423.06140846173
5379.25242188914
5335.54040268134
5291.92586857224
5248.40934077531
5204.99134403363
5161.67240667100
5118.45306064408
5075.33384159551
5032.31528890818
4989.39794576056
4946.58235918317
4903.86908011619
4861.25866346829
4818.75166817669
4776.34865726845
4734.05019792304
4691.85686153630
4649.76922378571
4607.78786469706
4565.91336871261
4524.14632476069
4482.48732632681
4440.93697152642
4399.49586317919
4358.16460888498
4316.94382110157
4275.83411722407
4234.83611966620
4193.95045594338
4153.17775875783
4112.51866608554
4071.97382126535
4031.54387309013
3991.22947590008
3951.03128967831
3910.94998014869
3870.98621887610
3831.14068336914
3791.41405718530
3751.80703003884
3712.32029791129
3672.95456316473
3633.71053465803
3594.58892786594
3555.59046500136
3516.71587514066
3477.96589435241
3439.34126582939

3400.84274002417
3362.47107478830
3324.22703551523
3286.11139528717
3248.12493502591
3210.26844364788
3172.54271822343
3134.94856414071
3097.48679527406
3060.15823415732
3022.96371216205
2985.90406968095
2948.98015631672
2912.19283107633
2875.54296257129
2839.03142922378
2802.65911947909
2766.42693202460
2730.33577601541
2694.38657130712
2658.58024869589
2622.91775016603
2587.40002914569
2552.02805077065
2516.80279215682
2481.72524268168
2446.79640427513
2412.01729172002
2377.38893296298
2342.91236943582
2308.58865638814
2274.41886323144
2240.40407389556
2206.54538719767
2172.84391722468
2139.30079372955
2105.91716254221
2072.69418599576
2039.63304336873
2006.73493134418
1974.00106448646
1941.43267573653
1909.03101692676
1876.79735931625
1844.73299414775
1812.83923322717
1781.11740952713
1749.56887781563
1718.19501531129
1686.99722236666
1655.97692318112
1625.13556654502
1594.47462661694
1563.99560373592
1533.70002527075

1503.58944650848
1473.66545158461
1443.92965445740
1414.38369992917
1385.02926471739
1355.86805857894
1326.90182549072
1298.13234489064
1269.56143298268
1241.19094411068
1213.02277220530
1185.05885230948
1157.30116218786
1129.75172402619
1102.41260622748
1075.28592531191
1048.37384792843
1021.67859298675
995.202433919016
968.947701081632
942.916784308557
917.112135628638
891.536272160857
866.191779202669
841.081313528503
816.207606917102
791.573469928609
767.181795954699
743.035565567631
719.137851197362
695.491822169229
672.100750138864
648.968014965603
626.097111071043
603.491654335656
581.155389593501
559.092198793609
537.306109906395
515.801306665055
494.582139245559
473.653136005043
453.019016417605
432.684705369589
412.655349004062
392.936332337644
373.533298913480
354.452172803796
335.699183336651
317.280892997237
299.204229048652
281.476519536099
264.105534489404
247.099533332251
230.467319756579
214.218305647658

```
198.362586077977
182.911027967481
167.875375795171
153.268378833135
139.103945898673
125.397335800561
112.165394841640
99.4268575090491
87.2027338203353
75.5168184439694
64.3963759037779
53.8730891920111
43.9844189480793
34.7756359373904
26.3030315087001
18.6393732622043
11.8841848571007
6.18647683971004
1.81235368932768
0];

% temp = linspace(0,11.24,113);
%
% plot(temp, loads);
taillength = 4.88;
disc = linspace(0,taillength,500);

lfunc = @(x) 1.371.*x.^6 - 16.26.*x.^5 + 77.07.*x.^4 - 128.6.*x.^3 + ...
    174.5.*x.^2 - 5928.*x + 2.246e4;

ldisc = lfunc(disc);

figure()
plot(disc,ldisc);

% ribs placed at known concentrated loads
E = 72e9;
density = 2780;
k = 3.62;

r_chord = 3.15548;
t_chord = r_chord*0.7;

ribs_known = [0, taillength];

pcrit = @(a) k*E*(a)^2;
% "a" is equal to t/b here

optrib = [];

for panels = 1:1
    t_skin = 1e-3;
    % optimal skin thickness obtained using fmincon function
            for nribs = 1:30
                ribspacing = (ribs_known(panels + 1) - ...
```

```
                    (ribs_known(panels)))/nribs;
                a = t_skin/ribspacing;
                averageload = (lfunc(ribs_known(panels)) + ...
                    lfunc(ribs_known(panels)+ribspacing))/2;
                if pcrit(a) < averageload
                    nribs = nribs + 1;
                else
                    disp(nribs)
                    break
                end
            end
        optrib(panels) = ribspacing;
end
disp(optrib)
rib_location = 2;
% note these values are spanwise, along the quarter-chord line

%% Stringer optimisation

% setup variables
% CHECK THESE AFTER DOING MATERIAL SELECTION
aluminium_density = 2.78e+03;
taillength = 4.88;

% Initial values of the optimisation variables
skin_thickness = 0.05;
number = 15;
thickness = 0.01;
height = 0.01;
width = 0.01;

x_initial = [thickness, height, width, number, skin_thickness];

% The mass function which is what is reduced
mass = @(x) (((x(2)-2*x(1))*x(1)) + (x(3)*x(1)*2) ...
    *aluminium_density)*x(4) + (taillength*2.71*x(5)*aluminium_density);

% The upper and lower bounds of each variable
lb = [0.001, 0.001, 0.0010, 6, 0.0010];
ub = [0.1, 0.1, 0.1, 20, 0.005];

nonlcon = @combinedconstraints;

options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm',...
    'interior-point');

% Define the problem as a GlobalSearch problem
problem = createOptimProblem('fmincon', 'objective', mass, 'x0',...
    x_initial, 'lb', lb, 'ub', ub, 'nonlcon', nonlcon, 'options', options);

% Create a GlobalSearch object
gs = GlobalSearch;

% Run the optimization using GlobalSearch
[x_final, final_mass, exitflag] = run(gs, problem);
```

```
% Analyze the results
if exitflag > 0
    disp('Optimal Solution to resist Stress');
    disp(x_final);
    disp('Optimal Mass Value');
    disp(final_mass);
else
    disp('Optimization did not converge to a solution');
end

%% Aerofoil plot with wingbox
clc
close all

geom=table2array(readtable("eppler.dat"));
x = geom(:,1);
y = geom(:,2);

figure()
plot(x,y,'Color','b','LineWidth',1);
grid on;
hold on;

chord = table2array(readtable("XYZc1.txt"));
xc = chord(:,1);
yc = chord(:,2);
plot(xc,yc,'--','Color','b');

htop = ((0.0384 - -0.0384) + (0.03 - -0.03))/4;
hbot = -htop;

v = [hbot, htop];
xl = [0.15,0.15];
plot(xl,v,'LineWidth',1,'Color','r');

xr = [0.65,0.65];
plot(xr,v,'LineWidth',1,'Color','r');

h = [0.15,0.65];
ytop = [htop, htop];
plot(h,ytop,'LineWidth',1,'Color','r');

ybot = [hbot, hbot];
plot(h,ybot,'LineWidth',1,'Color','r');

plot(0.4,0,'LineWidth',1,'Marker','x','Color','k')

xlabel('Chordwise co-ordinate: x/c','FontWeight','bold')
legend('Aerofoil geometry', 'Chord line', '','','', ...
    'Rectangular wingbox', 'Shear centre')
axis equal;

%% side view of VT (with ribs and stringers)
clc
```

```
close all

r_chord = 3.15548;
t_chord = r_chord*0.7;
% root chord
h_rc = [0,0];
x_rc = [0,r_chord];

figure()

plot(x_rc,h_rc,'LineWidth',1,'Color','b')
xlim([0 10]);
hold on;
% tip chord

% leading edge
plot([0,r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75],[0,4.88],...
    'LineWidth',1,'Color','b')
% trailing edge
plot([r_chord, r_chord*0.75 + (4.88*sind(40)) + t_chord*0.25],[0,4.88],...
    'LineWidth',1,'Color','b')
% tip chord
h_tc = [r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75, r_chord*0.75 + ...
    (4.88*sind(40)) + t_chord*0.25];
y_tc = [4.88,4.88];
plot(h_tc,y_tc,'LineWidth',1,'Color','b')

% wing box
plot([0.15*r_chord, r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.15*t_chord],[0, 4.88],'LineWidth',1,'Color','r')
plot([0.65*r_chord, r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.65*t_chord],[0,4.88],'LineWidth',1,'Color','r')
plot([0.15*r_chord, 0.65*r_chord],[0,0],'LineWidth',1,'Color','r')
plot([r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + 0.15*t_chord, ...
    r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + 0.65*t_chord],...
    [4.88,4.88],'LineWidth',1,'Color','r')

% flexural axis
mid_box_tip = ((r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.15*t_chord) + (r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.65*t_chord))/2;
plot([0.4*r_chord,mid_box_tip],[0,4.88],'LineWidth',1,'Color','r', ...
    'LineStyle','--')
m_qc = 4.88/(mid_box_tip-0.4*r_chord);

rib_location = 2;

m_box_left = 4.88/(r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.15*t_chord - 0.15*r_chord);
box_left = @(x) m_box_left*(x - 0.15*r_chord);
m_box_right = 4.88/(r_chord*0.75 + (4.88*sind(40)) - t_chord*0.75 + ...
    0.65*t_chord - 0.65*r_chord);
box_right = @(x) m_box_upper*(x - 0.65*r_chord);

m_ribs = -1/m_qc;
```

```
x_ribs = r_chord*0.75 + (rib_location.*cosd(40));
y_ribs = rib_location.*sind(40);
c_ribs = y_ribs - m_ribs.*x_ribs;

ribs_eq = @(x) c_ribs + m_ribs*x;

x_inter_upper = (-0.15*m_box_left*r_chord - c_ribs)/(m_ribs-m_box_left);
x_inter_lower = (-0.65*m_box_right*r_chord - c_ribs)/(m_ribs-m_box_right);
x_ribrange = linspace(x_inter_lower,x_inter_upper,2);
plot(x_ribrange,ribs_eq(x_ribrange),'Color','magenta');

grid on;
xlim([0 7]);

x_start = 0.473322 + (2.0516-0.473322)/7;
x_inter1 = ((m_qc*x_start) - (0.15*r_chord*m_box_left))/(m_qc-m_box_left);
x_strrange = linspace(x_start,x_inter1,2);
str_eq = @(x) m_qc*(x - x_start);
plot(x_strrange,str_eq(x_strrange),'Color','green');

x_inter2 = [];

for i = 2:5
    x_start(i) = 0.473322 + (2.0516-0.473322)*i/7;
    x_inter2(i-1) = (4.88/m_qc) + x_start(i);
    x_strrange = linspace(x_start(i),x_inter2(i-1),2);
    str_eq = @(x) m_qc*(x - x_start(i));
    plot(x_strrange,str_eq(x_strrange),'Color','green');
end

x_start = 0.473322 + (2.0516-0.473322)*6/7;
x_inter3 = ((m_qc*x_start) - (0.65*r_chord*m_box_right))...
    /(m_qc-m_box_right);
x_strrange = linspace(x_start,x_inter3,2);
str_eq = @(x) m_qc*(x - x_start);
plot(x_strrange,str_eq(x_strrange),'Color','green');

%% Stringers x span plot
close all
clc

span = [0, 4.0029,5.1224];
stringers = [6, 5, 4];

figure()
plot(span, stringers,'Marker','o')
xlabel('Spanwise direction','FontWeight','bold')
ylabel('Number of stringers','FontWeight','bold')

xlim([0 5.1224])
ylim([3 7])

%%
```

```
function [c, ceq] = combinedconstraints(x)

% setup variables
% CHECK THESE AFTER DOING MATERIAL SELECTION
M_yy = 6646942.7;
direct_yield_stress = 4.31*10^8;
comp_yield_stress = 4.42e+8;
shear_yield_stress = 3.1e+08;
aluminium_ym = 7.2e+10;
D = 2.81;
K = 3.62;
L = 0.5;
h_box = 0.454;
LD = 35;
M_xx = M_yy/LD;
w_length = 4.88;

I_xx = 1/6 * ((x(2)-2*x(1))*x(1)) * (x(3)*x(1)*2) * x(4)*(x(4)+1)*...
    (x(4)*2 + 1)*(5.16/x(4)) + x(5)*w_length^3/4;

%Stress due to bending constraint inequality:
c(1) = direct_yield_stress - (M_yy*(h_box/2))/((x(2)-2*x(1))*x(1)) + ...
    (x(3)*x(1)*2) * x(4)*(h_box/2)^2 * 2 - M_xx*5.16/(1/6 * ...
    ((x(2)-2*x(1))*x(1)) * (x(3)*x(1)*2) * x(4)*(x(4)+1)*(x(4)*2 + 1)...
    *(5.16/x(4)) + x(5)*w_length^3/4);

%Buckling constraint inequality:
c(2) = comp_yield_stress - (pi^2*aluminium_ym * ((2*((x(2)-x(1))/2)^2)*...
    (x(3)*x(1)) + x(1)/12*(x(2)-2*x(1))^3 )) / ...
    ((x(3)*x(1)*2+((x(2)-2*x(1))*x(1)))*L^2);
c(3) = shear_yield_stress - K*alumiSnium_ym*(x(5)/(((5.16/x(4))))^2);

ceq = [];

end
```

A.4 Detailed Design

A.4.1 Engineering Drawings

A.4.2 Mesh Convergence

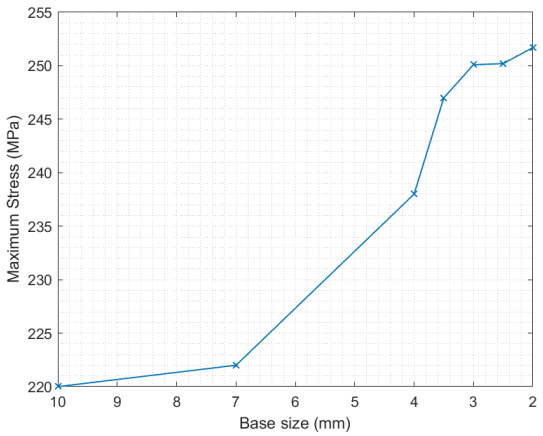


Figure A.8: Limit load mesh convergence

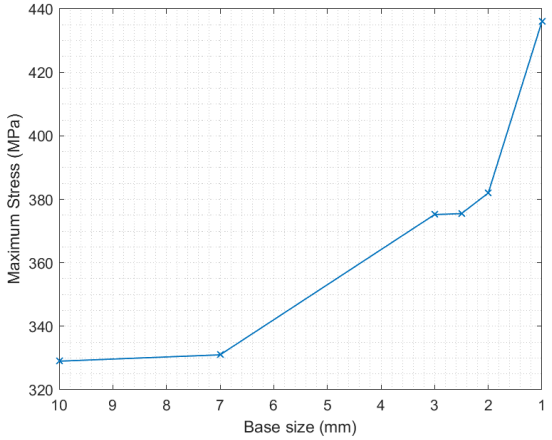


Figure A.9: Ultimate load mesh convergence

A.4.3 Buckling Plots

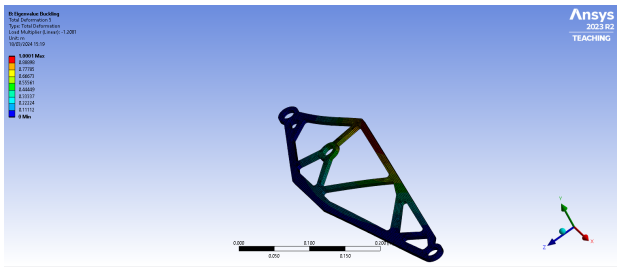


Figure A.10: 5th buckling mode

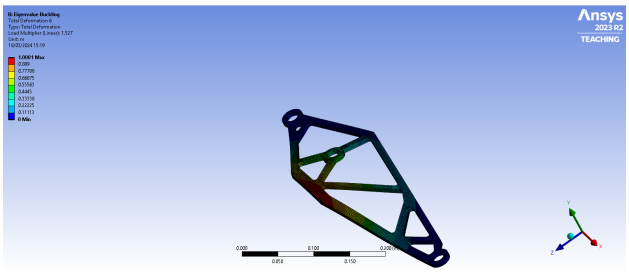


Figure A.11: 6th buckling mode

A.5 Yielding Plots

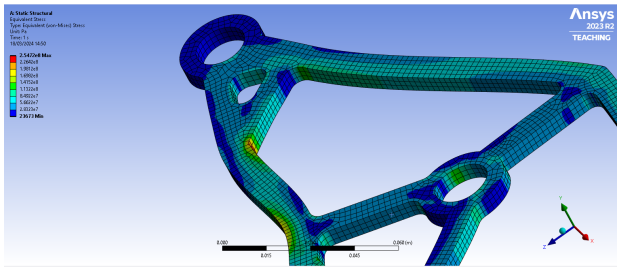


Figure A.12: Yielding location at limit load

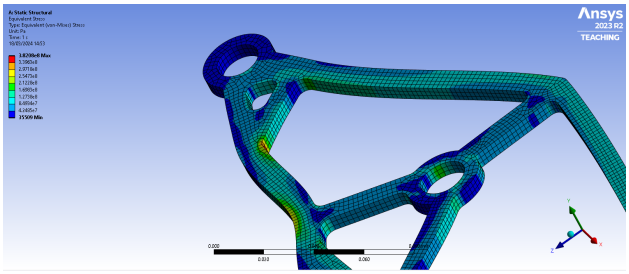


Figure A.13: Yielding location at ultimate load