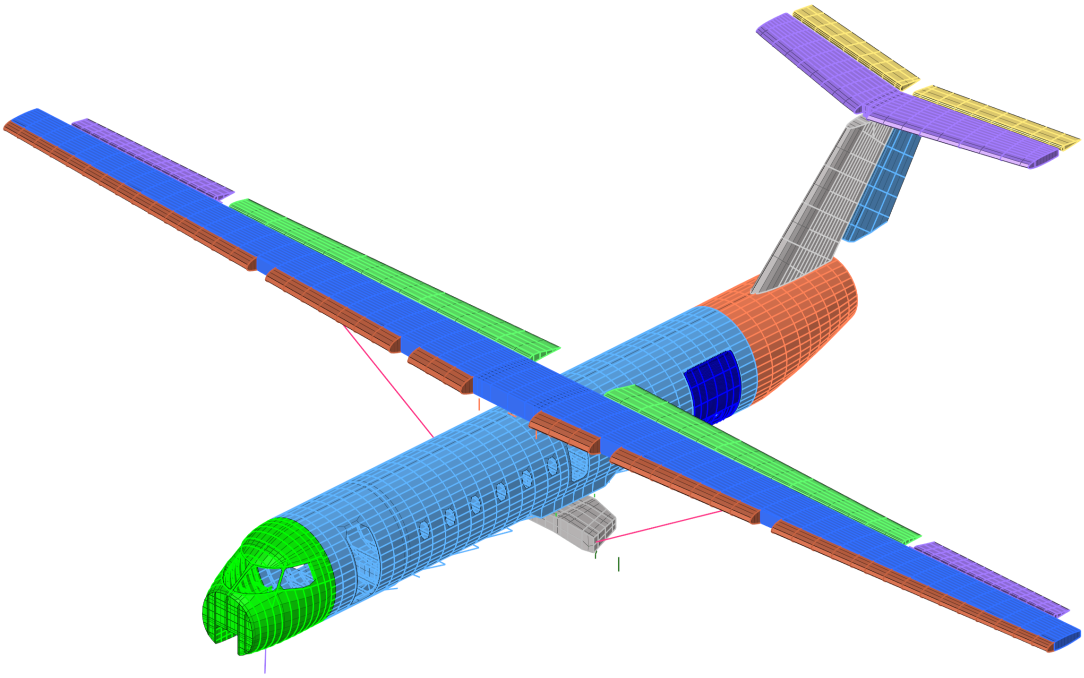


A Post-Processing Automation Tool for Finite Element Analysis on Airplane Fuselage

Master's Thesis in Master Programme Applied Mechanics

Jacob Wennersten

DEPARTMENT OF INDUSTRIAL AND MATERIAL SCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

A Post-Processing Automation Tool for Finite Element Analysis on Airplane Fuselage

JACOB WENNERSTEN

Department of Industrial and Material Science
Division of Material and Computational Mechanics
Chalmers University of Technology
Gothenburg, Sweden 2024

A Post-Processing Automation Tool for Finite Element Analysis on Airplane Fuselage
JACOB WENNERSTEN

© JACOB WENNERSTEN, 2024.

Supervisor: Pouya Shahbabai, Heart Aerospace
Supervisor: Kim Louisa Auth, Chalmers, Department of Industrial and Material Science
Examiner: Fredrik Larsson, Chalmers, Department of Industrial and Material Science

Master's Thesis 2024
Department of Industrial and Material Science
Division of Material and Computational Mechanics
Chalmers University of Technology
SE-412 96 Gothenburg
Chalmers Telephone +46 31 772 1000

Cover: GFE of the Heart Aerospace ES-30 aircraft.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

A Post-Processing Automation Tool for Finite Element Analysis on Airplane Fuselage
JACOB WENNERSTEN
Department of Industrial and Material Science
Chalmers University of Technology

Abstract

The work in this project revolves around investigating how automation can be used to improve the post-processing efficiency and quality when analyzing and sizing an airplane structure. The project is carried out at Heart Aerospace, based in Gothenburg, Sweden who is currently working on the design of an electric 30-seater regional aircraft for short travel called the ES-30, whose main engines are powered by batteries.

A program called ”H-ADAPT” (Heart Aerospace Design Analysis & Post-processing Tool), which serves as a tool with a graphical user interface, was created and used to perform the necessary analytical strength and stability analysis for the stringers and skins of the airplane fuselage. The program imports the large data files generated in the finite element analysis and summarize the different load cases, checks margin of safety and suggests optimized geometries for the elements. The user can edit the geometrical and material parameters of the elements, save and load the progress, as well as export the results to an Excel table for documentation. Validation of the analytical solutions is performed through hand-calculations, as well as comparison against a finite element analysis, for certain elements.

Unfortunately, the global finite element model lacked some geometrical information, and the user must manually give this input to the program for every element. This disrupts the automation and is time-consuming, which was one of the main purposes of the program to optimize. However, in the future if the geometrical information has been added, the program provides useful analysis and an easy-to-use graphical user interface to help with the sizing of the elements. It was also created in such a way that more complex analysis can be added, as well as more parts of the airplane.

Keywords: FEM, Post-processing automation, Stress analysis, Aerospace, PyNastran, PyQt, Electric aviation, Heart Aerospace, ES-30.

Acknowledgements

A big thank you to my supervisor Pouya Shahbabai at Heart Aerospace for giving me the opportunity to learn and practice what it is like to be a stress engineer in the world of aerospace engineering, challenging me to reach my full potential. I also appreciate the assistance and help from my colleges Eamonn Magee and Rushabh Kathote whom always took the time to answer my questions and provide explanations.

I also want to thank my examiner Fredrik Larsson and supervisor Kim Louisa Auth at Chalmers University of Technology for their courses and lectures throughout my educational journey, preparing me for this thesis, as well as all the support I have been given during the making of it. Thank you for all the detailed feedback and reviews, helping me make this thesis the best it could be.

Jacob Wennersten, Gothenburg, May 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BDF	Bulk Data File
FEA	Finite Element Analysis
FEM	Finite Element Method
GFEM	Global Finite Element Model
GUI	Graphical User Interface
H-ADAPT	Heart Aerospace Design Analysis & Post-processing Tool
JSON	JavaScript Object Notation
MS	Margin of Safety
OP2	Output2

Nomenclature

Below is the nomenclature of variables and parameters that have been used throughout this thesis.

Parameters

σ	Stress
σ_{yield}	Yield stress
$\sigma_{\text{yield, tension}}$	Yield stress in tension
$\sigma_{\text{yield, comp}}$	Yield stress in compression
σ_{ultimate}	Ultimate stress
$\sigma_{\text{ultimate, shear}}$	Ultimate stress in shear
σ_{vM}	Von Mises stress
F	Force
P_{cr}	Critical buckling force stringers
$N_{x,\text{cr}}$	Critical buckling force skins, x-direction
$N_{y,\text{cr}}$	Critical buckling force skins, y-direction
$N_{xy,\text{cr}}$	Critical buckling force skins, xy-direction
R	Stress ratio
R_x	Stress ratio in x-direction
R_y	Stress ratio in y-direction
R_{xy}	Stress ratio in xy-direction
A	Area
A_{T}	Area of T-section
A_{C}	Area of C-section
A_{Z}	Area of Z-section
h	Height of stringer section
b	Width of stringer section

t_{f}	Flange thickness of stringer section
t_{w}	Web thickness of stringer section
L	Length of stringer section
a	Length of skin in x-direction
b	Length of skin in y-direction
t	Thickness of skin in z-direction
ρ	Density
λ	Slenderness ratio
λ_{cr}	Critical slenderness ratio
E	Elastic modulus
ν	Poisson ratio
K	Column effective length factor
K_x	Compression buckling coefficient in x-direction
K_y	Compression buckling coefficient in y-direction
K_{xy}	Compression buckling coefficient in xy-direction
r	Radius of gyration
D	Flexural rigidity of skin

Contents

List of Figures

List of Tables

1

Introduction

Since 1990, the CO2 emissions from the aviation industry have increased by four times. And by 2050, it is projected to increase by another three times with current technologies. This would mean going from a total of 2% of the world's emissions in 2023 being from the aviation industry, to over 25% in 2050 [2]. A large contributing factor for this increase is the emerging markets of air travel in countries such as Brazil, Russia, India, and China, as more people will start to fly [11]. To meet EU:s goal of having net-zero emissions from aviation in 2050, new technologies must be developed, and one could turn to batteries [12].

1.1 Background

Up until recently, batteries have had a too low energy to weight ratio for powering airplanes. It also weighs just as much when empty or full, while other conventional jet fuels are expelled during combustion. But with the increased energy density, batteries could power airplanes for shorter flights. Some advantages would be a low carbon footprint, low maintenance costs because of the lack of a combustion engine and cheap electricity price compared to the price of conventional jet fuels [8].

Heart Aerospace, based in Gothenburg, Sweden is currently working on the design of an electric 30-seater regional aircraft for short travel called the ES-30, whose main engines are powered by batteries. With a range of 200 km this could fill an important role in reducing emissions in the aviation industry. A render of the ES-30 is seen below in figure 1.1.

Figure 1.1: Illustration of the electric 30-seater regional aircraft for short travel called the ES-30, developed by Heart Aerospace.

The design of the aircraft is in its conceptual phase, and obtaining the forces from different load combinations experienced in flight is of great importance for the structural design.

1.2 Structural analysis

1.2.1 The global finite element method

A Global Finite Element Model (GFEM) is a mathematical model representing the aircraft structure made up of idealized frames, panels, stringers, etc., that are represented by a coarse mesh with the use of shell, bar, and rigid body elements. The response is then solved for in the finite element program Nastran. This model is analyzed for various load combinations and flight cases. The internal loads extracted from the model are used as input for analyzing a structure's specific behavior and strength utilizing automated tools. Currently, this is a manual task, and is as such repetitive, time-consuming and error-prone. The GFEM model can be seen below in figure 1.2.

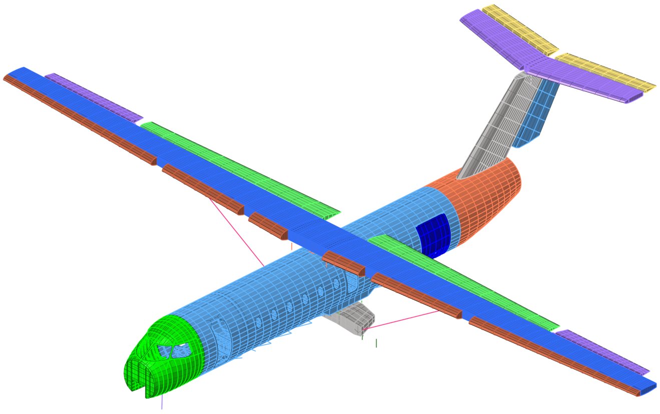


Figure 1.2: GFEM of the electric 30-seater regional aircraft for short travel called the ES-30, developed by Heart Aerospace.

1.2.2 Airplane fuselage

The GFEM has a semi-monocroque structure consisting of stringers, frames and skins. A monocroque structure is an airframe where loads are supported by the object's skin, with "semi" meaning that it still has some type of reinforcement. The stringer is represented by rod elements and the skin is represented by plate elements. An illustration of these members can be seen below i figure 1.3.

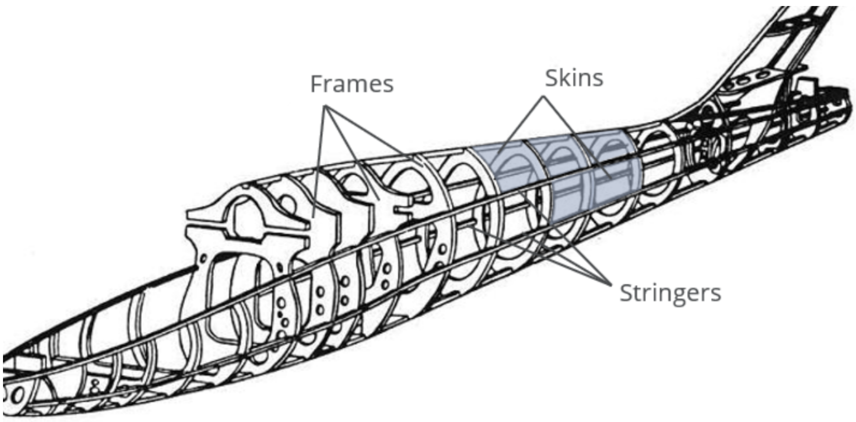


Figure 1.3: Sketch of a semi-monocroque airframe [14].

The skin covers the outer surface of the aircraft and is in the semi-monocroque airframe a load carrying structure. Stringer increases the stiffness of the skin under torsion and bending loads, as well as supporting the skin to prevent buckling under compression or shear loads. The frames form circular sections which run perpendicular to the longitudinal axis of the aircraft. They form the shape of the aircraft and also column support for the stringers to prevent buckling instabilities [6]. Note that the analysis of the frames will not be considered in this thesis.

1.2.3 GFEM and elements

In the GFEM, the airplane fuselage is idealized by element types such as rod and shell elements. A rod is a circular element with and area and length. It can also have a geometrical section assigned to it (see 2.1.3) and this will be needed for the strength and stability analysis. The rod can only take loads in the uni-axial direction as well as torsion, which makes it a simple one-dimensional element. A shell is a rectangular element with a width, height and thickness. It can take in-plane forces and bending moments in the x-, y- and xy-direction, as well as torsion. However, only the in-plane forces will be used in the analysis for simplicity.

Both element types can in tension go above the margin of safety (MS) when exceeding the yield or ultimate limit, depending on the type of analysis. But when in compression and shear, one must check for buckling, which is a sudden deformation that happens on thin members when exceeding a critical buckling load.

1.3 Purpose, scope and limitations

This thesis involves investigating how automation can be used to improve the post-processing efficiency (time to process the data) and quality (removing human error in repetitive tasks). The focus will be to develop a tool for extracting loads from the solved GFEM, process the data and check its strength and stability.

The tool should be able to summarize the loads from several load-cases, perform strength and stability analysis, and then show the user the worst load-case for every element. The user should then be able to alter the geometry and material of an element to see how it affects the MS in real time for optimization purposes. This process is called sizing, and it is the main purpose of the tool. It will also be written in such a way that it can scale well to be of use for larger and more complex analyses. The tool should be easy to use, and thus a Graphical user interface (GUI) will be created.

The main objective of this master thesis is thus to automate the post-processing analysis and provide the user with a strength and stability analysis summarized for all the load-cases via this tool and its GUI. Thought this main objective, the thesis aims to answer the following question formulations:

- Can a tool be created that extracts information and automates the repetitive post-processing analysis of the GFEM?
- Can it be written in such a generic way that it easily can be scaled and used on many different load cases, elements and geometries?
- Can a GUI be created to increase the ease of use and efficiency?

Outside the scope of these question formulations, some limitations are established. This is because of time and complexity. But since the aim is to make the tool generic and scalable, they could be added to the tool in the future. The limitations are the following:

1. The tool will be tested on the fuselage of the aircraft because of its simplicity. Even though it will be a generic code that could be scaled, the scaling to more complex parts of the aircraft will not be investigated as of now.
2. The tool will optimize the analyses for how the GFEM is structured as of now. If more information is later provided from this (such as section geometries or new materials) the code may have to be altered to handle this new information.
3. The tool can handle two different element types, which are rods and quadrilateral plates, that are used for stringer and skin analysis in the fuselage. Other types of elements will not be considered (such as bar, triangle, beam etc.). It will however be written in such a way that new types of elements could be added if desired.
4. After the elements have been analyses and edited, the user will be able to get an Excel table with all the parameters. How these parameters will be re-read into the GFEM will not be investigated.

2

Theory

Here, equations for strength and stability analysis for stringers and skins are presented. For boundary conditions, all elements are considered simply supported on all sides. No boundary is free in the GFEM, so this is a conservative estimate compared to a clamped boundary condition.

2.1 Stringers

2.1.1 Engineering stress equation

When the element is in tension, the engineering stress equation is used below.

$$\sigma = F/A \tag{2.1}$$

where A is the cross-section area and F is the force. Engineering meaning that the A is the original area.

2.1.2 The margin of safety

The Margin of Safety for limit case can be defined as,

$$\text{MS} = \frac{\sigma_{\text{yield}}}{\sigma} - 1 \tag{2.2}$$

when in tension, where σ_{yield} is the yield allowable stress and σ is the applied stress. For compression, it can be defined as

$$\text{MS} = \frac{P_{\text{cr}}}{F} - 1 \tag{2.3}$$

where P_{cr} is the critical buckling load and F is the applied load. For ultimate cases, the margin of safety can be defined as $\text{MS} = \frac{\sigma_{\text{u}}}{1.5 \times \sigma} - 1$ in tension and $\text{MS} = \frac{N_{\text{cr}}}{1.5 \times F} - 1$ where then σ_{ultimate} is the ultimate allowable stress.

2.1.3 Geometrical properties (second moment of inertia)

The second moment of inertia is an important geometrical property that is used when buckling is considered [5]. It can be defined in the x- or y- direction as I_x and I_y. The dimensions of the sections that will be the geometries of interest can be found below in figure 2.1. One can observe that for the T-section, the flange thickness t_f and web thickness t_w can be larger than the height h or width b. For both the Z-section and C-section, t_w can't be larger than b but t_f can't be larger than $h/2$.

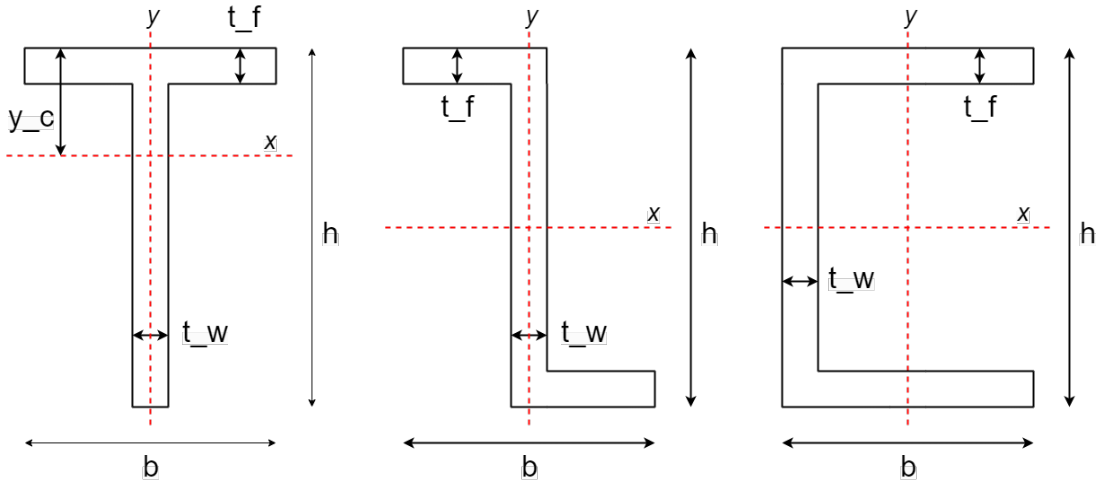


Figure 2.1: Geometries and parameters for a T-, Z- and C-section.

For the T-section, one can define

$$h - y_c = \frac{1}{A_T}\left(\frac{t_w h^2}{2} + \frac{(b - t_w)t_f^2}{2}\right) \tag{2.4}$$

$$A_T = (b - t_w)t_f + ht_w \tag{2.5}$$

These together can form the inertia in the x- and y-direction below.

$$I_x = \frac{t_w h^3}{3} + \frac{(b - t_w)t_f^3}{3} - A_T(h - y_c)^2 \tag{2.6}$$

$$I_y = \frac{(h - t_f)t_w^3}{12} + \frac{t_f b^3}{12} \tag{2.7}$$

For the Z-section, one can define $b_f = b - t_w$ and from that again get the inertia in the x- and y-direction below.

$$I_x = \frac{b_f t_f (h - t_f)^2}{2} + \frac{b_f t_f^3}{6} + \frac{t_w h^3}{12} \tag{2.8}$$

$$I_y = \frac{b_f t_f (b_f + t_w)^2}{2} + \frac{t_f b_f^3}{6} + \frac{h t_w^3}{12} \tag{2.9}$$

And finally for the C-section, define

$$A_C = 2bt_f + (h - 2t_f)t_w \tag{2.10}$$

$$d = (\frac{1}{A_C})\left(\frac{(h - 2t_f)t_w^2}{2} + t_f b^2\right) \tag{2.11}$$

$$I_{y0} = \frac{(h - 2t_f)t_w^3}{3} + 2\frac{t_f b^3}{3} \tag{2.12}$$

to get the inertia below.

$$I_x = \frac{bh^3}{12} - \frac{(b - t_w)(h - 2t_f)^3}{12} \tag{2.13}$$

$$I_y = I_{y0} - A_C d^2 \tag{2.14}$$

2.1.4 Mass

The mass for any column can be defined as

$$m = \rho L A \tag{2.15}$$

where ρ is the density, L is the length and A is the area of the cross-section.

2.1.5 Euler-Johnson column equation

The Euler-Johnson column equation provides a transition period between the yield allowable stress and the critical buckling stress when the element is in compression and has a low slenderness ratio [6]. A plot of this is shown in figure 2.2.

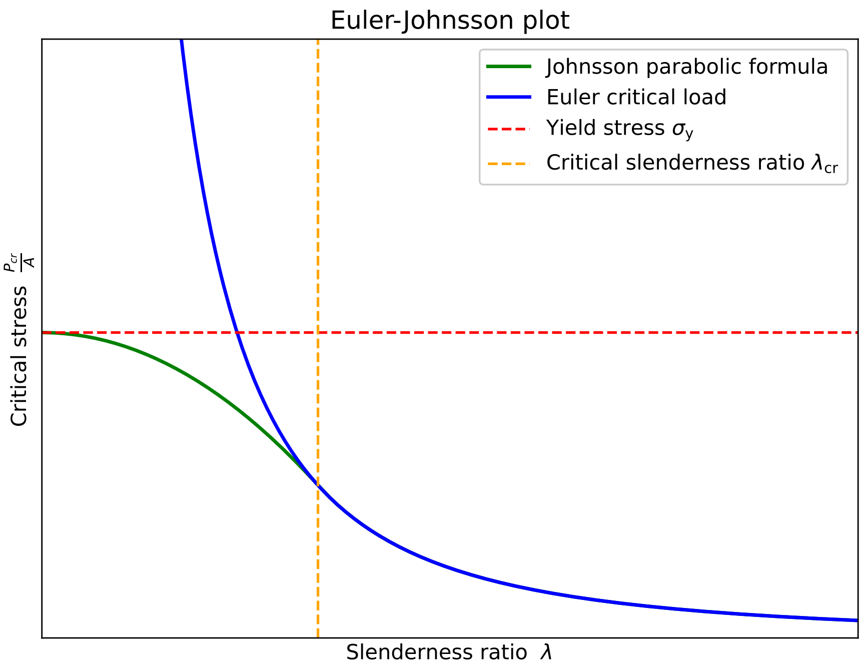


Figure 2.2: Critical bucking stress versus the slenderness ratio for a column with varying length.

The critical force is then defined as

$$P_{\text{cr}} = \sigma_{\text{yield}} A \left(1 - \frac{\sigma_{\text{y}}}{4\pi^2 E}\lambda^2\right), \text{ for } 0 < \lambda < \lambda_{\text{cr}} \tag{2.16}$$

$$P_{\text{cr}} = \frac{\pi^2 EI}{(KL)^2}, \text{ for } \lambda > \lambda_{\text{cr}} \tag{2.17}$$

where I is the lowest second moment of inertia and $K = 1$ for the column effective length factor with simply supported plates. $\lambda = \frac{LK}{r}$ with radius of gyration $r = \sqrt{\frac{I}{A}}$, and $\lambda_{\text{cr}} = \sqrt{\frac{2\pi^2 E}{\sigma_y}}$. The critical force is considered positive in compression, therefore the negative experienced compressive force has to switch sign.

2.2 Skins

2.2.1 Stress ratio

The stress ratio for buckling is defined as $R = \frac{F}{N_{\text{cr}}}$ where F is the applied load in force per unit length and N_{cr} is the critical buckling or failing load in force per unit length. Since compressive force is negative, the force again has to switch its sign before being used in this equation. The allowable stress ratio is defined as $R_a = \frac{N_a}{N_{\text{cr}}}$ where N_a is the allowable buckling or failing load in combined loading. These are later used to compute the MS in section 2.2.6. This is for the limit case for compression. For the ultimate case, the force is multiplied by a factor of 1.5 and all the yield allowable stresses in the plate equations are swapped for the ultimate allowable stress. For tension, the same MS as for tension of the bar element is used. The geometry of a skin is shown below in figure 2.3.

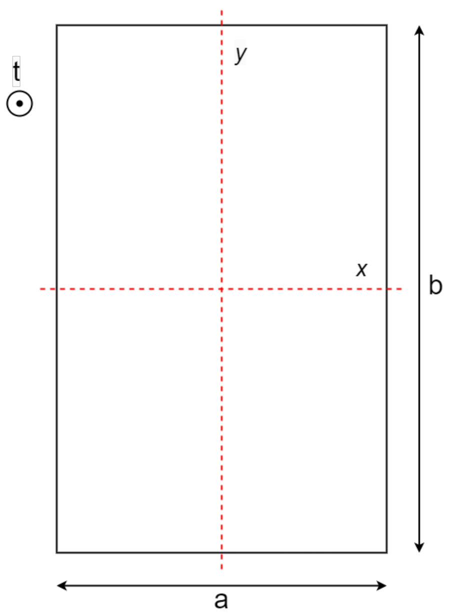


Figure 2.3: Geometry and parameters of a skin.

2.2.2 Von Mises stress

A skin can have a combined loading in the x-, y- and xy-direction. For these stress scenarios, use von Mises stress to check against the allowable [13].

$$\sigma_{\mathrm{vM}} = \sqrt{\sigma_x^2 + \sigma_y^2 + 3\sigma_{xy}^2 - \sigma_x\sigma_y} \tag{2.18}$$

where σ_x is the stress in the x-direction, σ_y is the stress in the y-direction and σ_{xy} is the stress in the xy-direction.

2.2.3 Mass

The mass for any skin can be defined as

$$m = \rho abt \tag{2.19}$$

where ρ is the density, a and b are the sides and t is the thickness.

2.2.4 Uni-axial compression

Now the force is in Newton per unit length. The critical load in uni-axial compression is defined below as

$$N_{x,\mathrm{cr}} = \frac{\pi^2 K_x D}{b^2} \tag{2.20}$$

or

$$N_{x,\mathrm{cr}} = \sigma_{\mathrm{yield}} t \tag{2.21}$$

whichever is smaller. Where K_x is the compression buckling coefficient found in "Airframe Stress Analysis and Sizing" by Chun-Yung Niu [6], and b is the length of the loaded edge of the plate and t is the thickness. And the flexural rigidity of the plate is defined as

$$D = \frac{Et^2}{12(1-\nu^2)} \tag{2.22}$$

If computing the force in the y-direction, the length a will be the loaded edge of the plate and used instead of b in equation 2.20.

2.2.5 Shear load

The critical load for shear is defined below as

$$N_{xy,\mathrm{cr}} = \frac{\pi^2 K_s D}{b^2} \tag{2.23}$$

or

$$N_{xy,\mathrm{cr}} = \sigma_{\mathrm{ultimate,shear}} t \tag{2.24}$$

whichever is smaller. Where K_s is the shear buckling coefficient also found in "Airframe Stress Analysis and Sizing" by Chun-Yung Niu [6], b is the shortest dimension of the plate, t is the thickness and D is the same flexural rigidity as above in equation 2.22.

2.2.6 Combined loading

From the uni-axial compression and shear, one can get the defined stress ratios and compute the margin of safety for combined loading [4].

$$\mathrm{MS} = \frac{2}{R_x + (R_x^2 + 4R_{xy}^2)^{0.5}} - 1 \tag{2.25}$$

$$\mathrm{MS} = \frac{2}{R_y + (R_y^2 + 4R_{xy}^2)^{0.5}} - 1 \tag{2.26}$$

where R_x and R_y are the stress ratios in the x- and y-direction and R_{xy} is the stress ratio for the shear force in the xy-direction.

2.3 Algorithm for minimization of thickness

To find the minimum thickness of a stringer or skin, Brent's algorithm is used [3]. This will find the minimum of $(\mathrm{MS} - 0.05)^2$ by varying the thickness of the element, which gives a MS = 0.05. A plot of this can be found in figure 2.4 below.

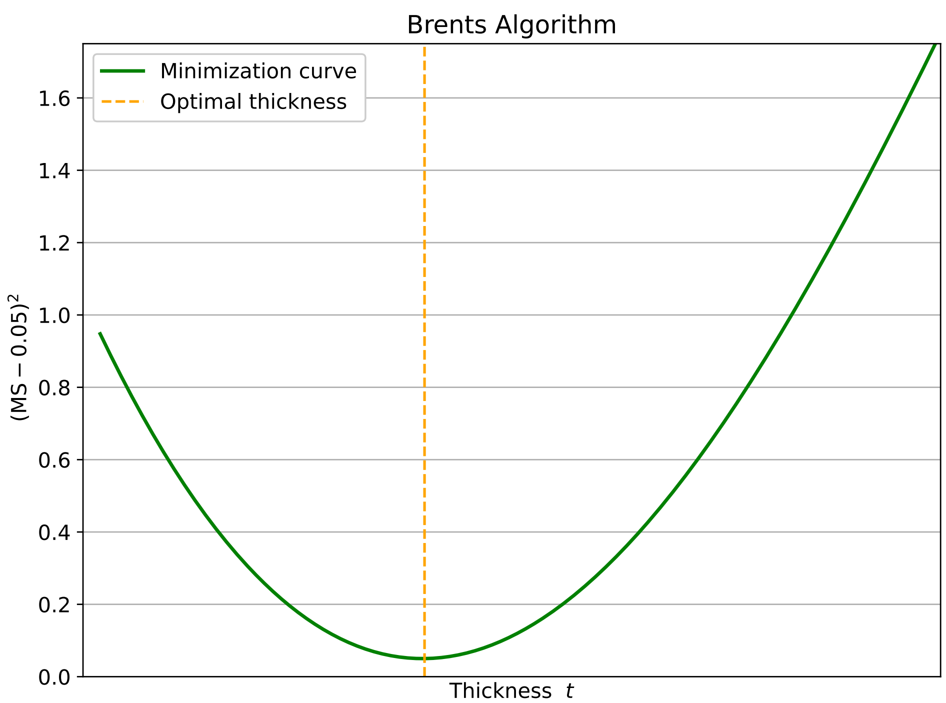


Figure 2.4: $(\mathrm{MS} - 0.05)^2$ vs increasing thickness for a stringer or skin.

If the section can't be thick enough, which can happen on the stringers because of the geometrical constraints, a negative MS will be found where the thickness will be equal to the width or height of the section. The optimization point will then be to the left of the yellow line, which means negative MS. For stringers, it will optimize the thickness of the flange t_f and the web t_w as the same value, since this is a single variable optimization algorithm. This might result in a geometry that would be able to be further optimized for the stringers. This will however not be a problem for the skins since it only has one single thickness and will thus be as optimized as can be. It will however be up to the engineer to decide if this thickness is feasible in reality.

However, when the section and geometries has been edited, the model has to be solved again in Nastran for a new response where the thickness will have to be optimized again with the algorithm. This is iteration could continue several times before the real optimum thickness is found for the elements.

3
Methods

In this chapter, the methods used to achieve the objectives of the thesis are discussed, such as programs, libraries and concepts.

3.1 GFEM and data files

This section gives a brief overview of the software and its file formats used in the finite element analysis (FEA) process, with HyperMesh for pre-processing, Nastran for solving and generated input and output files in BDF and OP2 formats.

HyperMesh is a pre-processing software used in FEA and to create the GFEM [1]. It allows one to define the geometry of a model by creating nodes and elements, which then can be assigned material properties. One also defines the constraints (fixed or limited movement of specific locations) and loads as boundary conditions. The output from HyperMesh is typically a file in BDF format, which stands for ”Bulk Data File” and is a text file. This file contains a complete description of the finite element model, ready for analysis by a solver program such as Nastran.

Nastran takes the BDF file generated by HyperMesh as input and utilizes the information in the file to perform complex numerical calculations based on the chosen analysis type (e.g., stress analysis, modal analysis) [9]. These calculations involve solving a large system of equations that govern the behavior of the model under applied loads and constraints.

Once Nastran completes the calculations, it generates an output file, typically in OP2 format, which stands for Output2 and in a binary format. This file contains the results of the analysis, such as:

- Deformations
- Internal loads
- Stresses
- Strains
- External loads and reaction forces

These files are then used for the post-processing analysis by extraction of the element information and loads in Python.

3.2 Extraction of element information and loads

To perform the post-processing analysis, the element information and loads must be extracted from the BDF and OP2 files to a more usable format. This is completed in Python with the package PyNastran.

PyNastran is a free, open-source Python library used for reading and manipulating Nastran files such as BDF and OP2 and is useful for automating a post-processing analysis [7]. One then goes from having text files full of information, to a user selected format in Python. A useful format to store this information in is a Pandas DataFrame. Pandas is also a free, open-source Python library, but designed for data visualization and manipulation [10]. One can build data structures called DataFrames which is a spreadsheet that holds tabular data for various data types, such as numerical and textual data. These are used to contain all the information from the BDF and OP2 files once extracted with PyNastran. Analysis on this data can now be performed and summarized.

Apart from input and output files from Nastran, information about the elements will be read from an Excel table like the one shown below in figure 3.1.

ElementID	PropID
261	3
263	3
266	3
267	3
268	3
269	3
317	3
321	3
325	3
326	3
327	3
328	3
376	3
378	3
381	3
382	3
383	3
384	3
1100	3
1101	3
1102	3
1103	3
1104	3

MatID	Material	Emod	nu	rho	yield_t	ultimate_t	yield_c	ultimate_shear
1	2024-T3	72394	0.33	2.76	324	441	268	268
2	2024-T351	72394	0.33	2.76	386	468	324	289
3	7075-T62	71016	0.33	2.79	468	524	468	330
4	7150-T775	71016	0.33	2.82	510	551	510	317
5	7475-T735	71016	0.33	2.79	406	482	399	282
6	2024-T8	71016	0.33	2.76	324	441	268	268
7	2024-T9	71016	0.33	2.76	324	441	268	268
8	2024-T10	71016	0.33	2.76	324	441	268	268
9	2024-T11	71016	0.33	2.76	324	441	268	268
17	2024-T12	71016	0.33	2.76	324	441	268	268
9E+07	2024-T13	71016	0.33	2.76	324	441	268	268
9E+07	2024-T14	71016	0.33	2.76	324	441	268	268

PropID	Section	h	b	t_w	t_f
1	Plate	nan	nan	nan	nan
2	Z	10	10	3	3
3	C	10	10	3	3
4	T	10	10	3	3

Figure 3.1: Excel input file with parameters needed for the analysis in the created tool.

The user here provides a list of elements defined in the BDF file and assign a property ID "PropID" to them. This is to give them a T-, Z- or C-section (if stringer) as well as dimensions. This will eventually be read from the BDF file itself, but not as of now. The elements will also have an assigned material ID "MatID" from the BDF file and depending on the identification number, the tool will assign one of the defined materials from the Excel to that element.

3.3 Analysis tool, visualization and usebility

Once the DataFrames with all the necessary information are obtained, analysis can be performed. This will be a strength and stability analysis on all the selected elements. Checking against the yield allowable stress when in tension and for buckling when in compression. The user can also edit the elements within the tool, such as the assigned geometry and material. The tool will be accessible and visualized through a GUI powered by Python though the package PyQt.

PyQt is a GUI framework used to create desktop applications, coded in Python. The application is created by adding features to one or several windows such as buttons, menus, text boxes and DataFrames. Thus, providing a user-friendly interaction to complex computations and analysis.

3.4 Graphical User Interface, concept illustration

A concept illustration of the GUI can be seen below in figure 3.2.

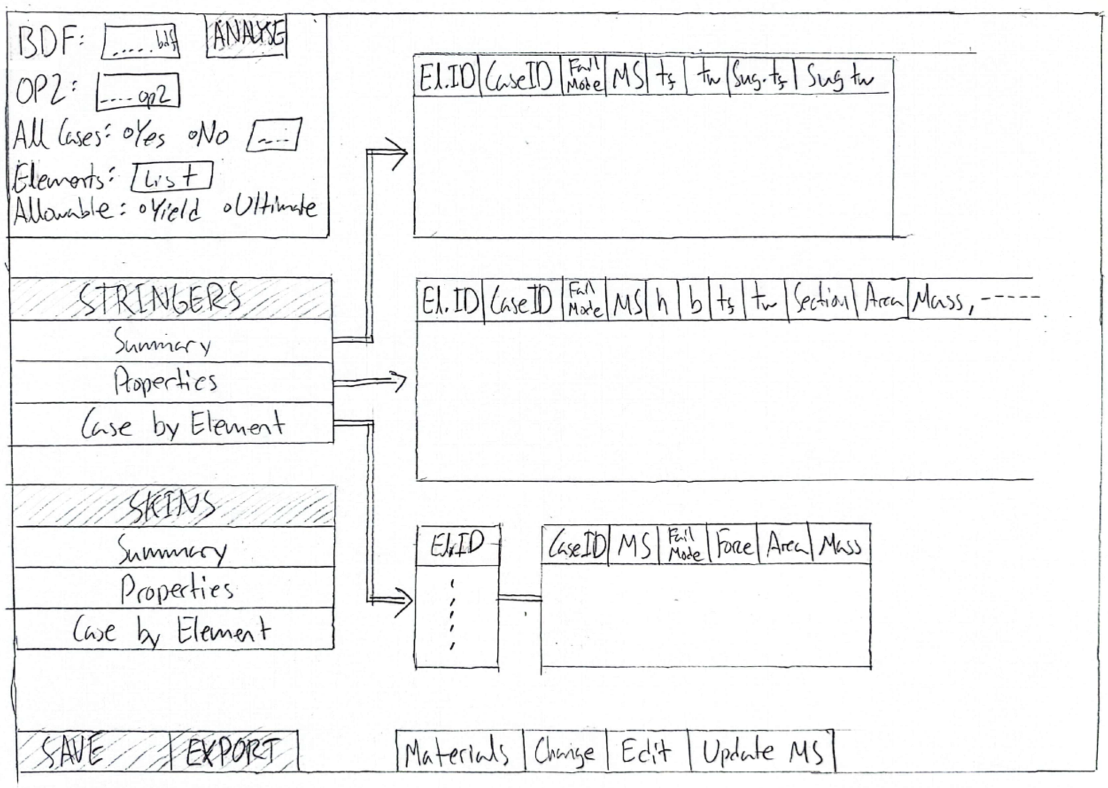


Figure 3.2: Concept illustration of GUI.

To start the analysis, the user imports the BDF and OP2 files in the top right corner. It is also chosen if all the cases should be considered or not. If not, a list of the cases must be provided in the Excel file. A similar list of element IDs (that have a defined property ID in the Excel file) is pasted. At last, one must choose if the allowable will be a yield case or ultimate case. Then the "ANALYSE" button can be pressed to start the analysis. When completed, three different DataFrames will appear per structural component (stringer or skin).

- When "Case by Element" is pressed, a table of all the elements is shown and when an element is selected by the user, a table of all the cases and the different forces, MS and failure modes experienced during that case for that element are displayed, as well as area and mass.

- The ”Properties” tab shows the case with the worst MS for every element, and the rest of the available information that could be of interest to the user such as nodes, length, material and suggested thickness.
- ”Summary” shows the same but fewer columns as ”Properties”, to give the user an overview of the most critical information.

The ”Properties” tab has editable columns called h, b, t_w, t_f, Section and Material. When these are edited and the ”Update MS” button is pressed, the whole analysis is updated with these new parameters. Other buttons are the ”Materials button, which will show a table of all the defined materials from the Excel file. A ”Change” button, so the user can select several rows in a column and edit them all to the same value if desired to save time. An ”Edit” button which will, when an element is selected, open a new window showing a figure of the geometry for that section as well as inertia and center of gravity (CG). Here the user can also edit h, b, t_w, t_f, Section and Material to see the changes to the inertia and CG.

The columns called ”Sug. t_w” and ”Sug. t_f” are a suggested thickness. This thickness provides an MS of 0.05 for that failure mode. The user then must change the columns t_f or t_w to that suggested thickness to apply the change. A new failure mode or case might then be the worst one, which the user should notice and iterate if needed. The algorithm behind this is explained in section 2.3.

The user can always press the ”SAVE” button in the top right corner to save all the progress. The program then creates a JSON file, which stands for ”JavaScript Object Notation” and is an open standard file format. The user can later load the JSON file into the program to continue where it left off. When the analysis is all done, one can press the ”EXPORT” button to get an Excel table looking like the ”Properties” DataFrame for referencing.

3.5 Flow of code

After the user imports the required files and lists and press "ANALYSE", the flow of code displayed in figure 3.3 happens to create the necessary DataFrames.

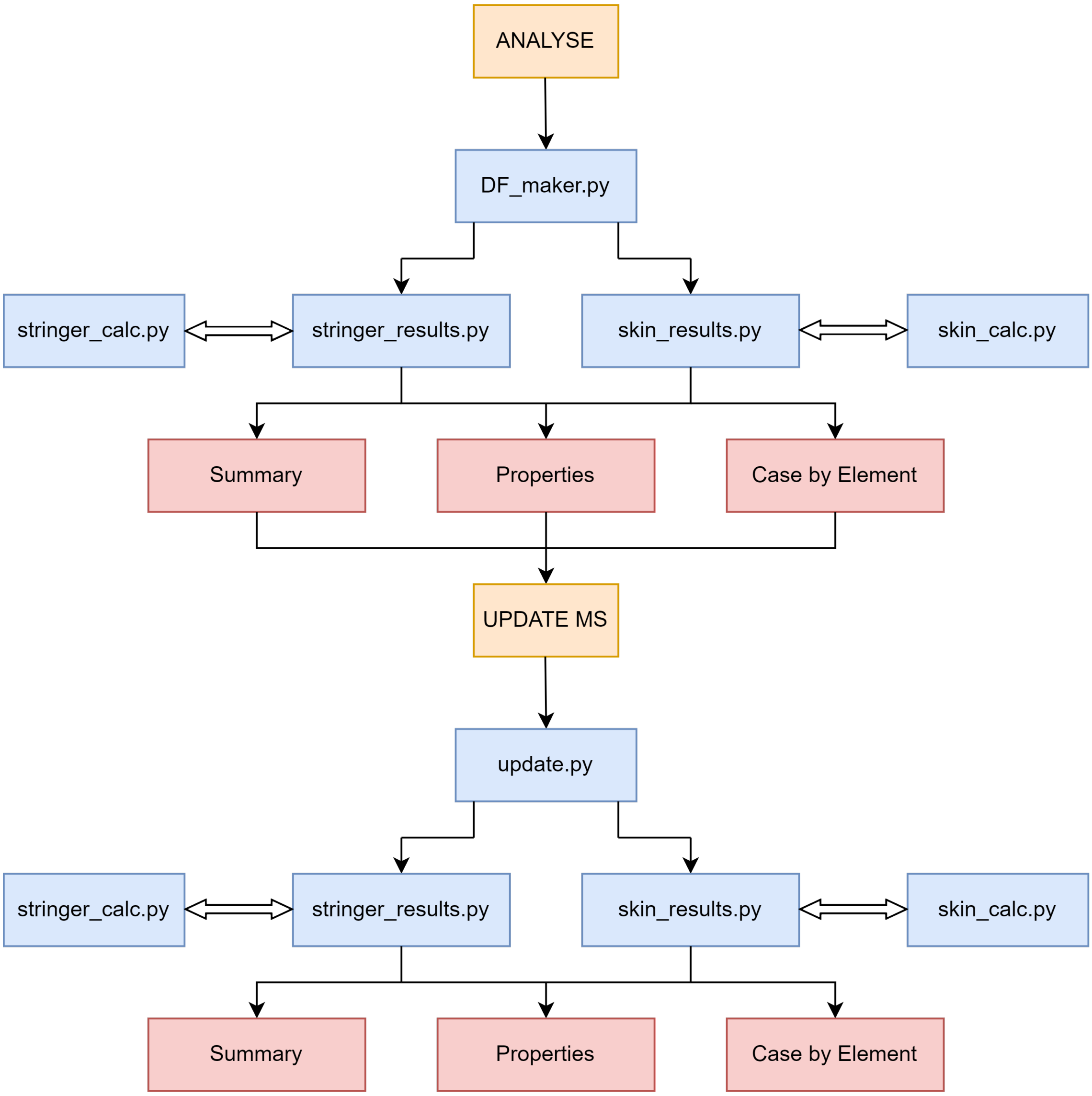


Figure 3.3: Flow of code where orange are buttons in the GUI, blue are the called Python files and red are the displayed DataFrames accessible by the GUI tabs.

- **df_maker.py** extracts all the data from the BDF, OP2 and Excel file though PyNastran, so Python only has to work in Pandas and its DataFrames. It also creates a dictionary where the entries are every element and if an element is chosen, it displays the different cases and forces associated with it. This is what is shown in the "Case by Element" tab.

- **stinger_results.py** and **skin_results.py** chooses the worst case for every element, where the worst case is the case with the lowest MS for that particular element. This is calculated though **stinger_calc.py** and **skin_calc.py**. The result is displayed in the "Properties" tab in the GUI, as well as a short version in the "Summary" tab. The file also runs an algorithm that find the optimum thickness for every element to reach an MS of 0.05 for the current failure mode. The flange and web thickness t_f and t_w are then optimized as one variable, and they can't get thicker than the width or high of the section. This will sometimes result in a lower MS than 0.05 since the section can't get thick enough without altering the high h and width b.

- **stinger_calc.py** and **skin_calc.py** contains all the equations for stress, inertia, mass, MS etc. Essentially all the equations defined in section 2 are present here. A demonstration of what this file does is done in the chapter 5.1 for validation.

- **update.py** runs when the 'UPDATE MS' button is pressed. It then takes the updated geometrical and material parameters and runs a slightly altered version of **df_maker.py**, without re-reading the BDF, OP2 and Excel file. Afterward, it is again sent to the **stinger_results.py** and **skin_results.py** files to optimize the thickness and sort out the worst cases for every element.

3.6 Validation and assertions

To ensure that the program works as expected, a validation needs to be performed, as well as continues assertions within the code.

3.6.1 Validation

When the strength and stability analysis has been done on the elements, it must be checked that these calculations work as intended and give accurate results. Therefore, the validation will consist of hand calculations for one stringer and one skin element, to check if the value of the MS and other outputs are the same as displayed in the GUI. In addition to this, a detailed FEM analysis will also be done on these elements to compare the analytical results with the numerical.

3.6.2 Assertions

To ensure that the code runs smoothly, and bugs don't go unnoticed, many assertion checks have been integrated into the code. This helps with faster development and addition of features without introducing unknown errors. Another type of assertion is to check that the user is handling the program correctly, and if not, notify them with an error. Some of these errors could for example be:

- Not all elements in the pasted element list have an assigned property ID in the Excel file.
- Elements have material IDs in the BDF which are not defined in the Excel file.

- The editable parameters h, b, t_w, t_f, Section and Material have an invalid value such as wrong data type, invalid geometry, section that isn't an L-, Z- or C-section or a non-defined material.

4

The post-processing tool

The result of the thesis is the program itself, which will be shown in detail here. The program has been given the name of "H-ADAPT" (Heart Aerospace Design Analysis & Post-processing Tool). When starting the program H-ADAPT, the window shown in figure 4.1 below appears.

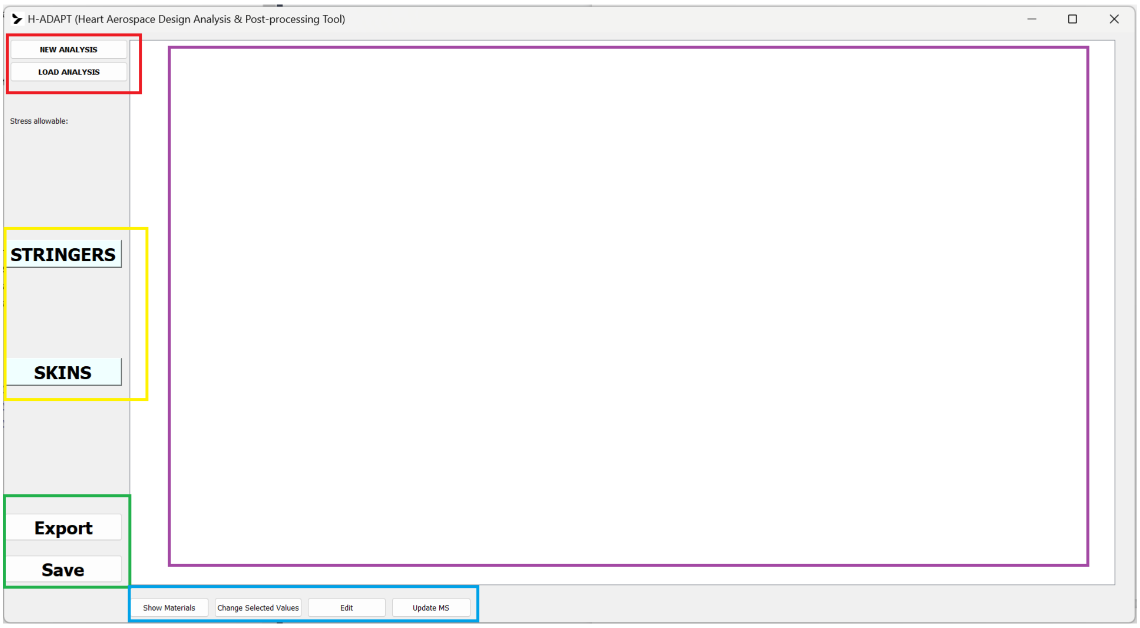


Figure 4.1: Start screen of the H-ADAPT program.

Different GUI sections such as analysis section, DataFrame window, stringers and skins section, update and functionalities section, save and export section has for visual purposes been color coded and will be explained below.

4.1 New analysis section

The analysis section is where the user has to go to start an analysis. One has the option to start a new analysis or load a previous one which may or may not have been altered, more on this later. This is done by pressing the "NEW ANALYSIS" or "LOAD ANALYSIS" button found in the red color coded box, with zoom shown below in figure 4.2.

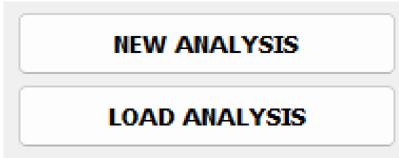


Figure 4.2: Zoom in of the analysis section in the GUI.

When the "NEW ANALYSIS" button is pressed, a new window shown below in figure 4.3 to the left appears. Here the user needs to take several steps.

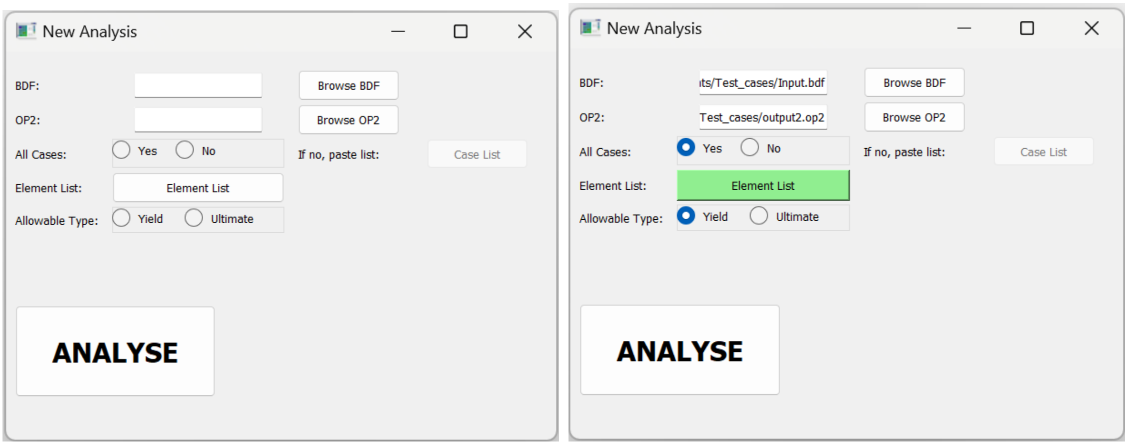


Figure 4.3: An empty "New Analysis" windows to the left and the same window filled out to the right.

- Import the BDF-file and OP2-file via a click button to navigate the folders on the computer and select the files.
- Choose if all the cases defined in the model should be considered in the analysis or not. If not, the user has to press the "Case List" button, where an empty list will appear. Here one must paste a list of numbers containing the case-IDs that are of interest for the analysis.
- Press the "Element List" button where an empty list shown below in figure 4.4 will appear where the user has to paste the stringer and skin elements that are of interest for the analysis. This is similar to the window that appears when the "Case List" button is pressed.
- Choose if the analysis should use the yield or ultimate allowable. If ultimate is chosen, the all the forces in the analysis will be scaled by 1.5 before the MS is calculated. The forces displayed in the DataFrame will however be shown unaltered, since the scaling only comes in the MS calculation. As shown in section 2.
- Press the "ANALYSE" button to run the analysis.

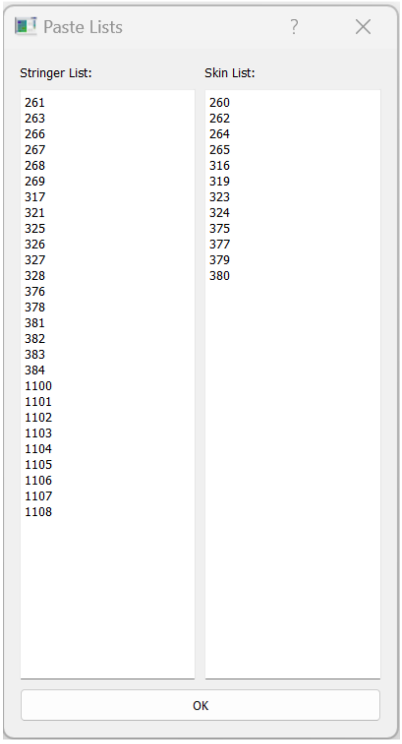


Figure 4.4: The "Paste Lists" window that appears when the user press the "Element List" button, filled with example element IDs.

When the "ANALYSE" button has been pressed, the "New Analysis" window closes and the DataFrame window fills with the results seen below in figure 4.5.

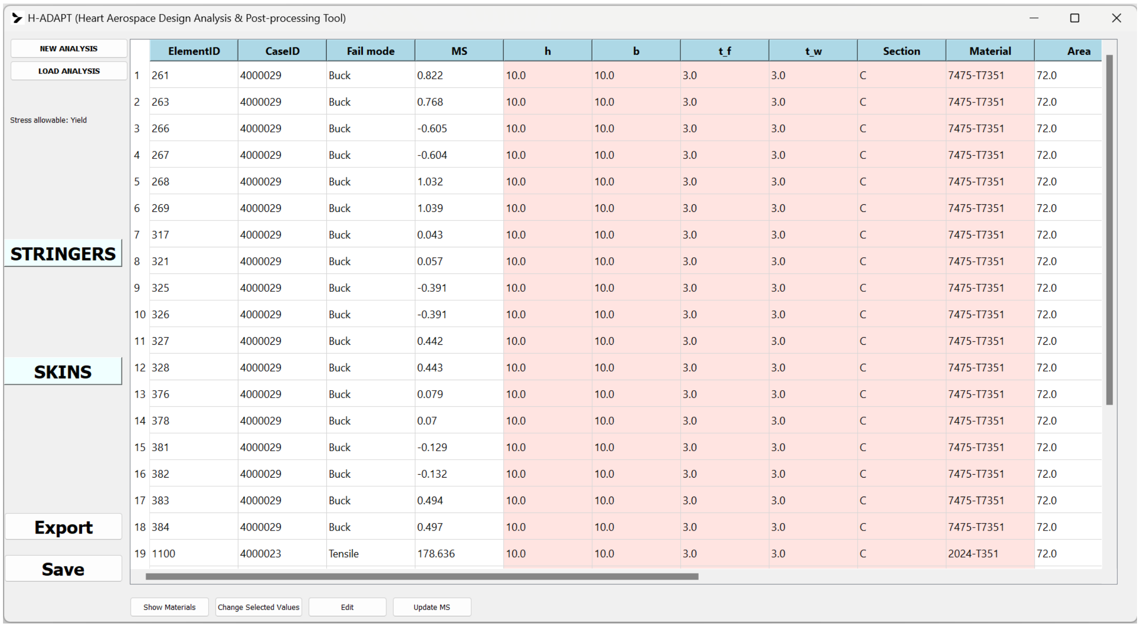
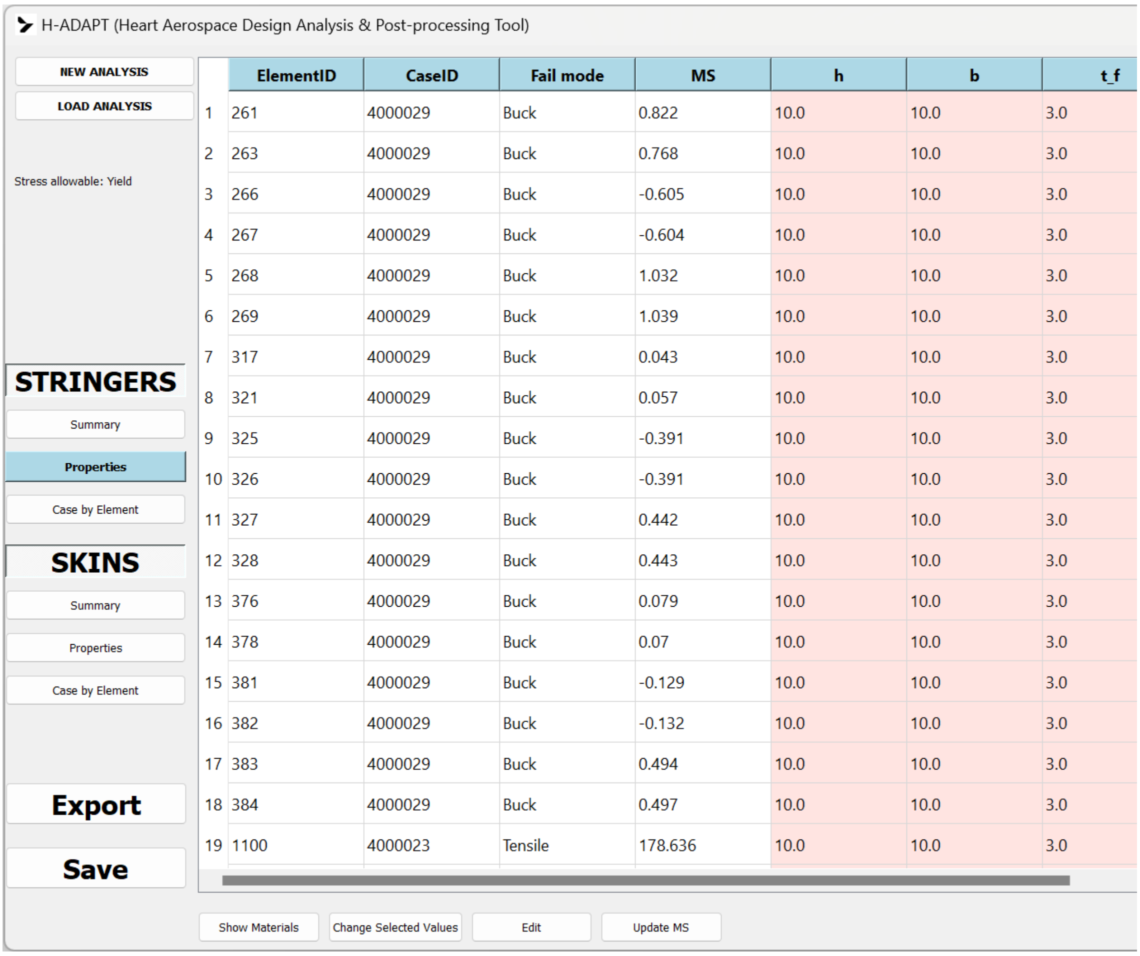


Figure 4.5: GUI with loaded analysis, showing the DataFrame window filled with results.

4.2 Stringer section

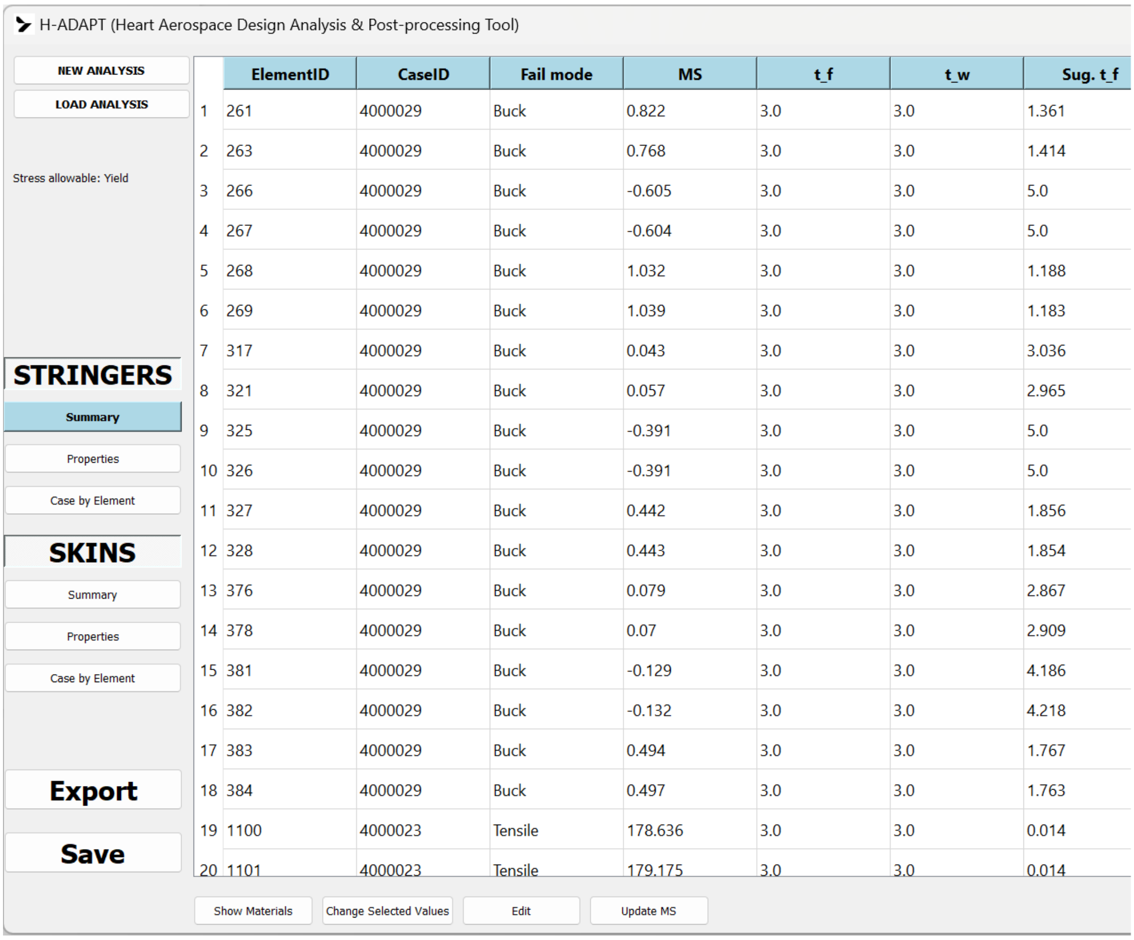
After an analysis has been loaded, one can press the drop-down menus to the left in the stringers and skins section to reveal what type of tabs can be shown. This part of this chapter will focus on the stringer sections. The first DataFrame window shown below in figure 4.6 is from the "Properties" tab.



	ElementID	CaseID	Fail mode	MS	h	b	t_f
1	261	4000029	Buck	0.822	10.0	10.0	3.0
2	263	4000029	Buck	0.768	10.0	10.0	3.0
3	266	4000029	Buck	-0.605	10.0	10.0	3.0
4	267	4000029	Buck	-0.604	10.0	10.0	3.0
5	268	4000029	Buck	1.032	10.0	10.0	3.0
6	269	4000029	Buck	1.039	10.0	10.0	3.0
7	317	4000029	Buck	0.043	10.0	10.0	3.0
8	321	4000029	Buck	0.057	10.0	10.0	3.0
9	325	4000029	Buck	-0.391	10.0	10.0	3.0
10	326	4000029	Buck	-0.391	10.0	10.0	3.0
11	327	4000029	Buck	0.442	10.0	10.0	3.0
12	328	4000029	Buck	0.443	10.0	10.0	3.0
13	376	4000029	Buck	0.079	10.0	10.0	3.0
14	378	4000029	Buck	0.07	10.0	10.0	3.0
15	381	4000029	Buck	-0.129	10.0	10.0	3.0
16	382	4000029	Buck	-0.132	10.0	10.0	3.0
17	383	4000029	Buck	0.494	10.0	10.0	3.0
18	384	4000029	Buck	0.497	10.0	10.0	3.0
19	1100	4000023	Tensile	178.636	10.0	10.0	3.0

Figure 4.6: "Properties" tab and its DataFrame window cropped.

This tab shows all the available information about each stringer element. The user can for every ElementID see which CaseID is the worst, its failure mode and MS. All the geometrical parameters, material name and parameters, nodes and suggested thickness. The red columns which would be h, b, t_f, t_b, Section and Material are the only editable variables. When these are altered and the analysis is updated, one will get a different MS and maybe a different failure mode. Altering any other part of the DataFrame is not possible. Next, the "Summary" tab is shown in figure 4.7 below.



	ElementID	CaseID	Fail mode	MS	t_f	t_w	Sug. t_f
1	261	4000029	Buck	0.822	3.0	3.0	1.361
2	263	4000029	Buck	0.768	3.0	3.0	1.414
3	266	4000029	Buck	-0.605	3.0	3.0	5.0
4	267	4000029	Buck	-0.604	3.0	3.0	5.0
5	268	4000029	Buck	1.032	3.0	3.0	1.188
6	269	4000029	Buck	1.039	3.0	3.0	1.183
7	317	4000029	Buck	0.043	3.0	3.0	3.036
8	321	4000029	Buck	0.057	3.0	3.0	2.965
9	325	4000029	Buck	-0.391	3.0	3.0	5.0
10	326	4000029	Buck	-0.391	3.0	3.0	5.0
11	327	4000029	Buck	0.442	3.0	3.0	1.856
12	328	4000029	Buck	0.443	3.0	3.0	1.854
13	376	4000029	Buck	0.079	3.0	3.0	2.867
14	378	4000029	Buck	0.07	3.0	3.0	2.909
15	381	4000029	Buck	-0.129	3.0	3.0	4.186
16	382	4000029	Buck	-0.132	3.0	3.0	4.218
17	383	4000029	Buck	0.494	3.0	3.0	1.767
18	384	4000029	Buck	0.497	3.0	3.0	1.763
19	1100	4000023	Tensile	178.636	3.0	3.0	0.014
20	1101	4000023	Tensile	179.175	3.0	3.0	0.014

Figure 4.7: "Summary" tab and its DataFrame window cropped.

This tab shows a summary of what is shown in the "Properties" tab. Everything shown in the "Summary" tab is also shown in the "Properties" tab. This gives the user a faster look of the most important information. Since no column is red, nothing here can be edited. The user must go to the "Properties" tab to be able to edit. Next, the "Case by Element" tab is shown in figure 4.8 below.

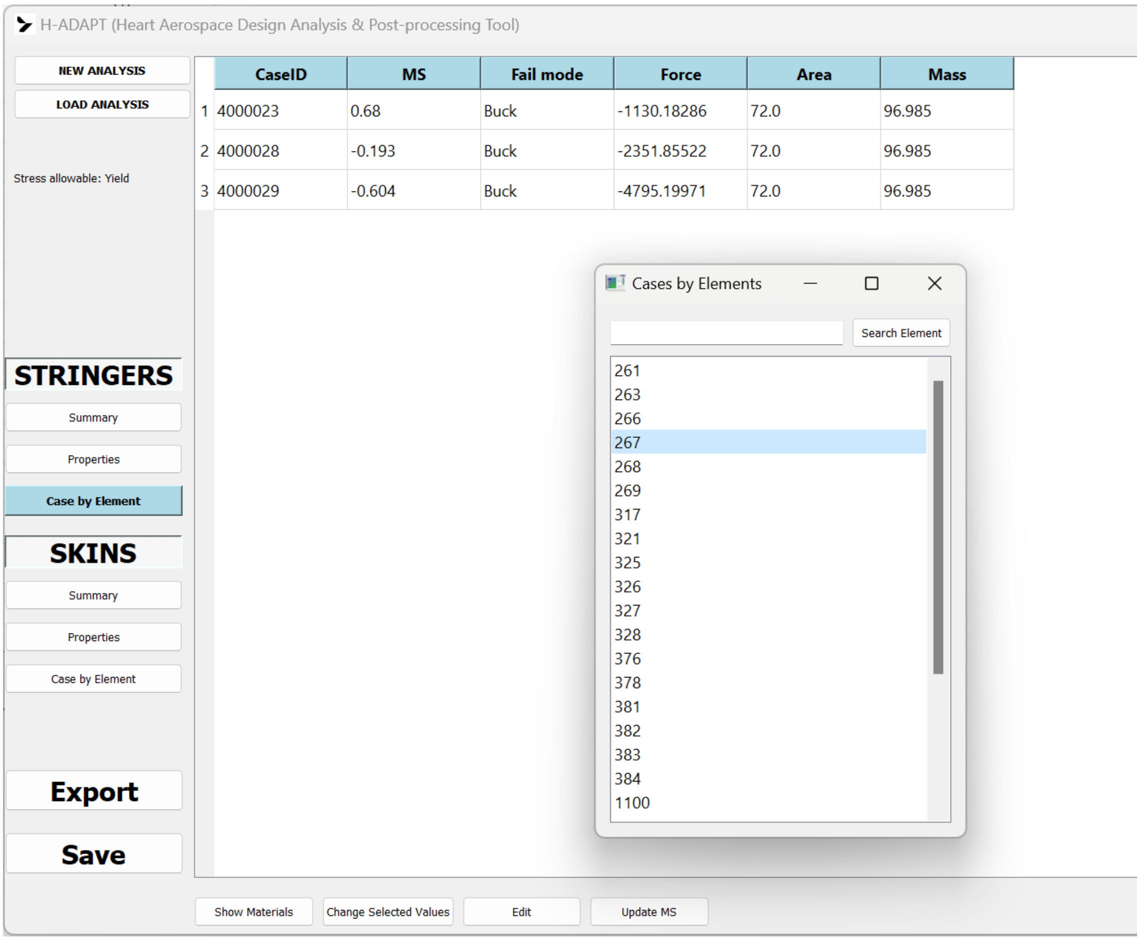


Figure 4.8: "Case by element" tab with its DataFrame and element selection window.

When this tab is pressed, a window called "Case by Element" appears. Here, the user can choose or search for an element ad show all the different load cases and forces associated with it. Here one can see for example how the forces and failure modes changes between the different cases and thus the MS.

The different tabs are the same for the skins, with a difference of which geometrical parameters and forces are shown. The skin has for example a length in a and b direction and only one thickness t, and no section type. It also has forces in x-, y- and xy-direction.

4.3 Update and functionalities section

Below the DataFrame window we find the update and functionalities section shown below in figure 4.9.

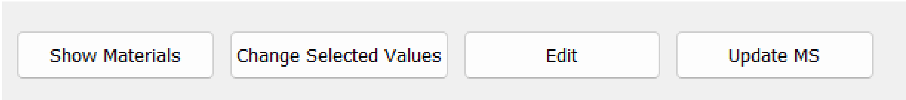


Figure 4.9: The update and functionalities section buttons.

- ”Show Materials” makes a window pop up that show all the defined materials and its properties such as σ_y, E, ν and more shown in figure 3.1.
- ”Change Selected Values” lets the user select several rows in the red editable area and change them all to the same value. This could be for example if one has several elements which should have the same thickness or the same type of geometrical section.
- ”Edit” takes a selected element and opens an external window where the user can change the editable variables in the red columns and see how it affects the inertia, mass and center of gravity (CG). This window is shown below in figure 4.10. But to update the MS, one also must press the ”Update MS” button. The ”Update Values” button in this window only updates the variables but does not solve the strength and stability analysis.
- ”Update MS” reads all the changed variables in the red columns and re-runs the analysis to compute for all the elements the new worst case-ID, failure mode, MS, mass etc.

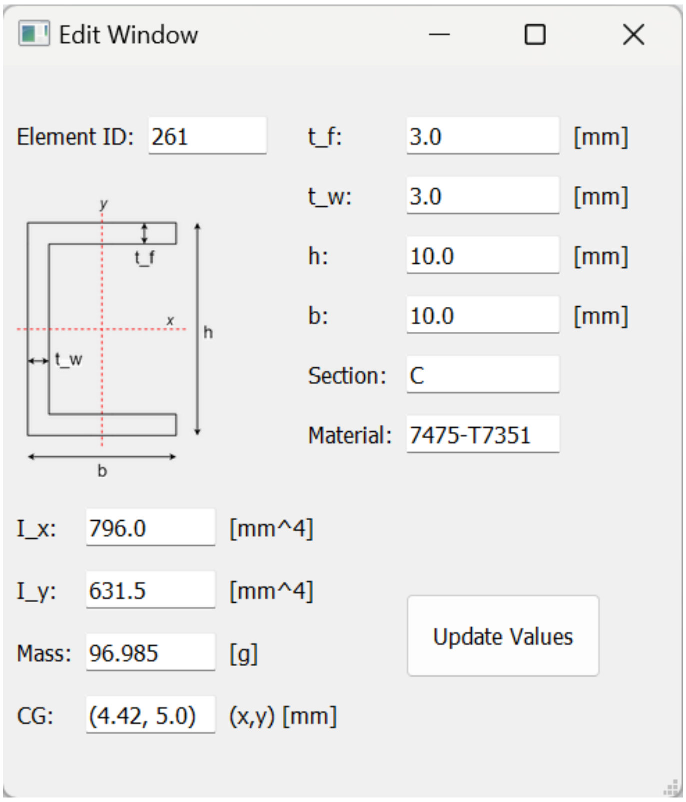


Figure 4.10: ”Edit Window” which opens when the user press the ”Edit” button where one can see the inertia, mass and CG changes for an element.

4.4 Save and export section

The user can continue where it left off by saving the progress with the ”Save” button in the save and export section seen below in figure 4.11. All the DataFrames and other information are then saved in a JSON type file which can be loaded next time the program has started with the ”LOAD ANALYSIS” button in the Analysis section.

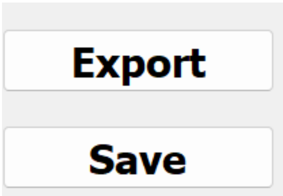
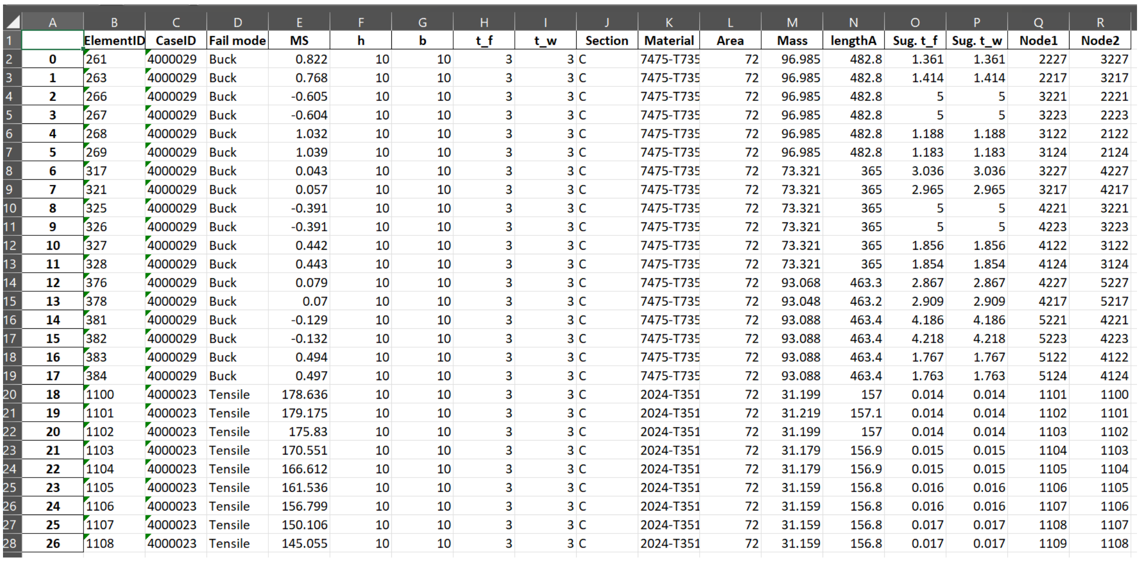


Figure 4.11: Zoom on the save and export section in the GUI.

The user can also export the results to an Excel file with the ”Export” button when the analysis is done. One then gets the choice of exporting stringers or skin and then select a name and location on the computer. An example of an exported Excel file for stringers is shown below in figure 4.12.



	ElementID	CaseID	Fail mode	MS	h	b	t_f	t_w	Section	Material	Area	Mass	lengthA	Sug. t_f	Sug. t_w	Node1	Node2
0	261	4000029	Buck	0.822	10	10	3	3	C	7475-T735	72	96.985	482.8	1.361	1.361	2227	3227
1	263	4000029	Buck	0.768	10	10	3	3	C	7475-T735	72	96.985	482.8	1.414	1.414	2217	3217
2	266	4000029	Buck	-0.605	10	10	3	3	C	7475-T735	72	96.985	482.8	5	5	3221	2221
3	267	4000029	Buck	-0.604	10	10	3	3	C	7475-T735	72	96.985	482.8	5	5	3223	2223
4	268	4000029	Buck	1.032	10	10	3	3	C	7475-T735	72	96.985	482.8	1.188	1.188	3122	2122
5	269	4000029	Buck	1.039	10	10	3	3	C	7475-T735	72	96.985	482.8	1.183	1.183	3124	2124
6	317	4000029	Buck	0.043	10	10	3	3	C	7475-T735	72	73.321	365	3.036	3.036	3227	4227
7	321	4000029	Buck	0.057	10	10	3	3	C	7475-T735	72	73.321	365	2.965	2.965	3217	4217
8	325	4000029	Buck	-0.391	10	10	3	3	C	7475-T735	72	73.321	365	5	5	4221	3221
9	326	4000029	Buck	-0.391	10	10	3	3	C	7475-T735	72	73.321	365	5	5	4223	3223
10	327	4000029	Buck	0.442	10	10	3	3	C	7475-T735	72	73.321	365	1.856	1.856	4122	3122
11	328	4000029	Buck	0.443	10	10	3	3	C	7475-T735	72	73.321	365	1.854	1.854	4124	3124
12	376	4000029	Buck	0.079	10	10	3	3	C	7475-T735	72	93.068	463.3	2.867	2.867	4227	5227
13	378	4000029	Buck	0.07	10	10	3	3	C	7475-T735	72	93.048	463.2	2.909	2.909	4217	5217
14	381	4000029	Buck	-0.129	10	10	3	3	C	7475-T735	72	93.088	463.4	4.186	4.186	5221	4221
15	382	4000029	Buck	-0.132	10	10	3	3	C	7475-T735	72	93.088	463.4	4.218	4.218	5223	4223
16	383	4000029	Buck	0.494	10	10	3	3	C	7475-T735	72	93.088	463.4	1.767	1.767	5122	4122
17	384	4000029	Buck	0.497	10	10	3	3	C	7475-T735	72	93.088	463.4	1.763	1.763	5124	4124
18	1100	4000023	Tensile	178.636	10	10	3	3	C	2024-T351	72	31.199	157	0.014	0.014	1101	1100
19	1101	4000023	Tensile	179.175	10	10	3	3	C	2024-T351	72	31.219	157.1	0.014	0.014	1102	1101
20	1102	4000023	Tensile	175.83	10	10	3	3	C	2024-T351	72	31.199	157	0.014	0.014	1103	1102
21	1103	4000023	Tensile	170.551	10	10	3	3	C	2024-T351	72	31.179	156.9	0.015	0.015	1104	1103
22	1104	4000023	Tensile	166.612	10	10	3	3	C	2024-T351	72	31.179	156.9	0.015	0.015	1105	1104
23	1105	4000023	Tensile	161.536	10	10	3	3	C	2024-T351	72	31.159	156.8	0.016	0.016	1106	1105
24	1106	4000023	Tensile	156.799	10	10	3	3	C	2024-T351	72	31.159	156.8	0.016	0.016	1107	1106
25	1107	4000023	Tensile	150.106	10	10	3	3	C	2024-T351	72	31.159	156.8	0.017	0.017	1108	1107
26	1108	4000023	Tensile	145.055	10	10	3	3	C	2024-T351	72	31.159	156.8	0.017	0.017	1109	1108

Figure 4.12: Exported Excel file for stringers., containing all the information available under the ”Properties” tab in the GUI.

The information exported is the same shown in the DataFrame under the ”Properties” tab.

4.5 Assertions

To ensure that the user is warned when an input or operation fails, several assertions have been adapted into the code. Examples of these warnings were discussed in section 3.6 and are shown in figure 4.13 below.

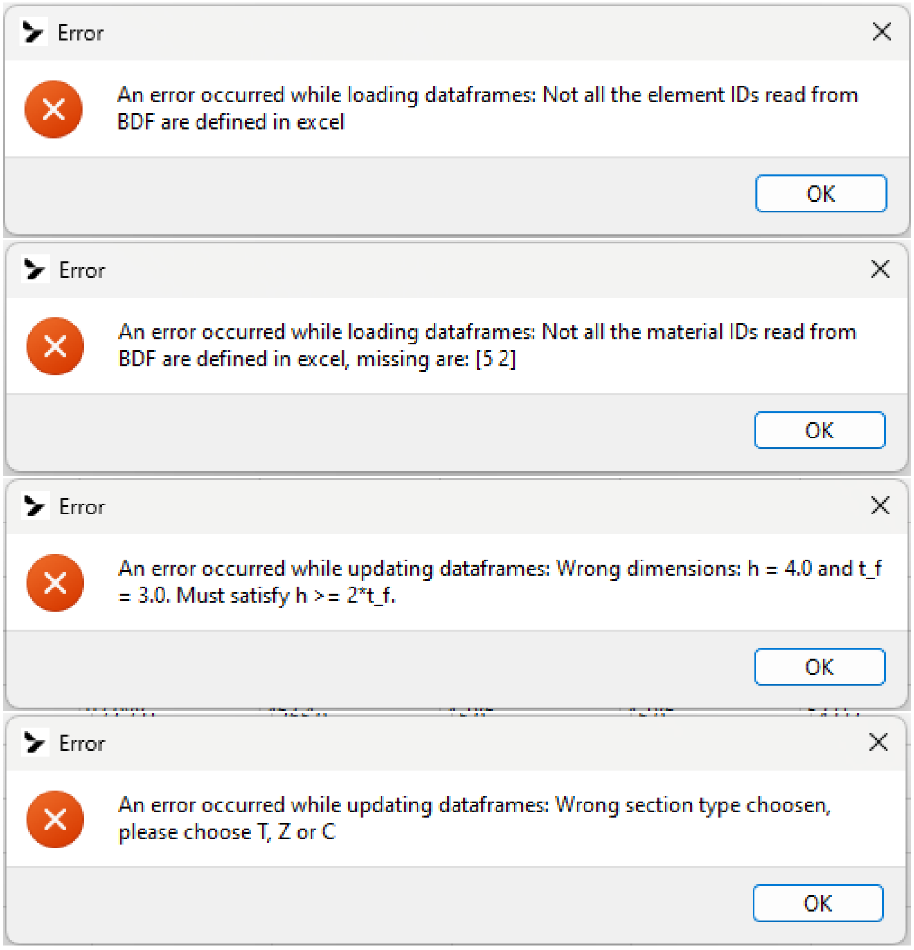


Figure 4.13: Examples of four different warning messages that can be displayed to the user when performing a non-accepted operation.

The first ”Error” occurs when the user tries to create a new session and in the ”Paste Lists” window writes an element number that is not defined in the Excel file. This means that there is no geometry associated with this element. The second ”Error” occurs when there are materials defined in the BDF file with associated material IDs that are not defined in the Excel file. The program then can’t find the material properties to give this element. The third ”Error” occurs when the geometrical parameter h is too small compared to t_f. This constraint was discussed in section 2.1.3. The fourth and final ”Error” occurs when the user tries to input a section to an element which is either a T-, Z- or C-section. Since these are the only defined sections for now, anything else will throw an error.

5

Validation

To validate the results, hand calculations have been done for both a stringer and skin element, as well as a FEA in HyperMesh. This is to ensure that the program is working properly and that the results from the analytical solutions show trustworthy results compared to the numerical results.

5.1 Hand calculations

For the hand calculations, the results are shown all the way from input to output, exactly the way the program does its computations. A certain Nastran run and flight case ID is chosen for both a stringer and skin element, and the results are shown below.

5.1.1 Stringer

For element number 50606, all the input information is found below in table 5.1.

h	25 mm
b	20 mm
t_f	1.5 mm
t_w	1.5 mm
L	840.9 mm
Section	C
Material	Al 7475-T7351
E	71 016 Pa
ρ	0.002 76 g/mm^3
$\sigma_{\text{yield,tension}}$	406 Pa
$\sigma_{\text{yield,comp}}$	399 Pa
F	−7065.760 N

Table 5.1: Input values for element 998.

Negative force means compressive force, so check for buckling. First use equations 2.10, 2.11 and 2.12 to get what is needed for the inertia of a C-section.

$$A_C = 2bt_f + (h - 2t_f)t_w = 2 \cdot 20 \cdot 1.5 + (25 - 2 \cdot 1.5) \cdot 1.5 = 93.00\,\text{mm}^2 \qquad (5.1)$$

$$d = \left(\frac{1}{A_C}\right)\left(\frac{(h-2t_f)t_w^2}{2} + t_f b^2\right) = \left(\frac{1}{93}\right)\left(\frac{(25-2\cdot 1.5)\cdot 1.5^2}{2} + 1.5\cdot 20^2\right) \\ = 6.72\,\text{mm} \tag{5.2}$$

$$I_{y0} = \frac{(h-2t_f)t_w^3}{3} + 2\frac{t_f b^3}{3} = \frac{(25-2\cdot 1.5)\cdot 1.5^3}{3} + 2\cdot\frac{1.5\cdot 20^3}{3} = 8024.75\,\text{mm}^4 \tag{5.3}$$

And the inertia in the x-direction and y-direction from equations 2.13 and 2.14 can then be computed below.

$$I_x = \frac{bh^3}{12} - \frac{(b-t_w)(h-2t_f)^3}{12} = \frac{20\cdot 25^3}{12} - \frac{(20-1.5)(25-2\cdot 1.5)^3}{12} \\ = 9626.00\,\text{mm}^4 \tag{5.4}$$

$$I_y = I_{y0} - A_C d^2 = 8024.75 - 93.00\cdot 6.72^2 = 3827.89\,\text{mm}^4 \tag{5.5}$$

I_y is the lowest second moment of inertia, and will thus be used as our inertia I for the buckling analysis. The mass defined from equation 2.19 is then computed.

$$m = \rho A L = 0.00276\cdot 93.00\cdot 840.90 = 215.84\,\text{g} \tag{5.6}$$

By selecting the element in the tool and then pressing the "Edit" button, one can find the outputs from the tool below in figure 5.1.

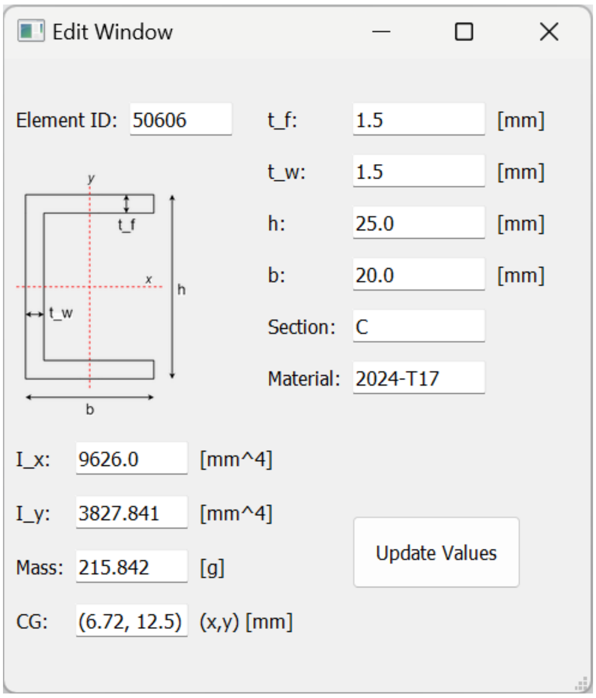


Figure 5.1: Printscreen from the GUI when editing element 50606. Right side are inputs and left side are the outputs.

Which shows the same results as the hand calculations for the inertia and mass. Next, for the MS to be computed, one continues with the slenderness.

$$r = \sqrt{\frac{I}{A}} = \sqrt{\frac{3827.89}{93.00}} = 6.42\,\text{mm} \tag{5.7}$$

$$\lambda = \frac{LK}{r} = \frac{840.90 \cdot 1}{6.42} = 130.98 \tag{5.8}$$

$$\lambda_{\text{cr}} = \sqrt{\frac{2\pi^2 E}{\sigma_y}} = \sqrt{\frac{2 \cdot \pi^2 \cdot 71016}{406}} = 58.75 \tag{5.9}$$

Since $\lambda > \lambda_{\text{cr}}$, equation 2.17 will be used.

$$P_{\text{cr}} = \frac{\pi^2 EI}{(KL)^2} = \frac{\pi^2 \cdot 71016 \cdot 3827.89}{(1 \cdot 840.90)^2} = 3794.25\,\text{N} \tag{5.10}$$

The MS can now be checked with equation 2.3, where the force F flips it sign.

$$\text{MS} = \frac{P_{\text{cr}}}{F} - 1 = \frac{3794.25}{7065.76} - 1 = -0.463 \tag{5.11}$$

Which is the same as the MS calculated within the tool shown below in figure 5.2.

	ElementID	CaseID	Fail mode	MS	h	b	t_f	t_w	Section	Material
1	3915	1011011	Buck	-0.546	25.0	20.0	1.5	1.5	C	2024-T17
2	50606	1011011	Buck	-0.463	25.0	20.0	1.5	1.5	C	2024-T17
3	998	1011002	Tensile	-0.157	25.0	20.0	1.0	1.0	C	2024-T17
4	21161730	1011007	Buck Trans. region	1.782	20.0	20.0	1.5	1.5	C	2024-T16

Figure 5.2: Printscreen from the GUI when under the stringers "Properties" tab, showing the MS for some elements, including element 50606.

The tool also suggests a new thickness of $t_w = t_f = 3.377\,\text{mm}$ which is found through Brent's algorithm shown in section 2.3. This thickness should result in a new MS = 0.05, with the assumption that both thicknesses t_f, t_w are the same and that the failure mode won't change. Below, this thickness is inserted and the moment of inertia is re-calculated.

$$A_C = 2bt_f + (h - 2t_f)t_w = 2 \cdot 20 \cdot 3.377 + (25 - 2 \cdot 3.377) \cdot 3.377 = 196.70\,\text{mm}^2 \tag{5.12}$$

$$\begin{aligned} d &= \left(\frac{1}{A_C}\right)\left(\frac{(h-2t_f)t_w^2}{2} + t_f b^2\right) = \left(\frac{1}{196.70}\right)\left(\frac{(25 - 2 \cdot 3.377) \cdot 3.377^2}{2} + 3.377 \cdot 20^2\right) \\ &= 7.40\,\text{mm} \end{aligned} \tag{5.13}$$

$$\begin{aligned} I_{y0} &= \frac{(h-2t_f)t_w^3}{3} + 2\frac{t_f b^3}{3} = \frac{(25 - 2 \cdot 3.377) \cdot 3.377^3}{3} + 2 \cdot \frac{3.377 \cdot 20^3}{3} \\ &= 18\,244.90\,\text{mm}^4 \end{aligned} \tag{5.14}$$

$$I_x = \frac{bh^3}{12} - \frac{(b-t_w)(h-2t_f)^3}{12} = \frac{20 \cdot 25^3}{12} - \frac{(20-3.377)(25-2 \cdot 3.377)^3}{12} \\ = 17\,627.11\,\mathrm{mm}^4 \tag{5.15}$$

$$I_y = I_{y0} - A_C d^2 = 18244.90 - 196.70 \cdot 7.40^2 = 7473.61\,\mathrm{mm}^4 \tag{5.16}$$

For the new inertia $I = I_y$, a new slenderness ratio must be computed below.

$$r = \sqrt{\frac{I}{A}} = \sqrt{\frac{7473.61}{196.70}} = 6.16\,\mathrm{mm} \tag{5.17}$$

$$\lambda = \frac{LK}{r} = \frac{840.90 \cdot 1}{6.16} = 136.50 \tag{5.18}$$

where $\lambda > \lambda_{\mathrm{cr}}$ which means that equation 2.17 will again be used.

$$P_{\mathrm{cr}} = \frac{\pi^2 EI}{(KL)^2} = \frac{\pi^2 \cdot 71016 \cdot 7473.61}{(1 \cdot 840.9)^2} = 7407.94\,\mathrm{N} \tag{5.19}$$

The MS can now be checked with equation 2.3.

$$\mathrm{MS} = \frac{P_{\mathrm{cr}}}{F} - 1 = \frac{7407.94}{7065.76} - 1 = 0.050 \tag{5.20}$$

Which confirms that the suggested thickness results in a MS = 0.050.

5.1.2 Skin

For element number 319, all the input information is found below in table 5.2.

a	83.4 mm
b	365.0 mm
t	1.0 mm
Material	Al 7475-T7351
E	71 016 Pa
ν	0.33
ρ	0.002 79 $\mathrm{g/mm^3}$
$\sigma_{\mathrm{yield,tension}}$	386 Pa
$\sigma_{\mathrm{yield,comp}}$	324 Pa
$\sigma_{\mathrm{ultimate,shear}}$	289 Pa
F_x	−3.783 N/mm
F_y	−8.541 N/mm
F_{xy}	1.061 N/mm

Table 5.2: Input values for element 319.

For this combined loading, one should check both von Mises and buckling to see which one has the lowest MS. Start with equation 2.22 to get the flexural rigidity.

$$D = \frac{Et^2}{12(1-\nu^2)} = \frac{71016 \cdot 1.0^2}{12 \cdot (1-0.33^2)} = 6641.23\,\text{Nmm} \tag{5.21}$$

Continue with the critical load for Uni-axial force.

$$N_{x,\text{cr}} = \frac{\pi^2 K_x D}{b^2} = \frac{3.14^2 \cdot 21.21 \cdot 6641.23}{365.0^2} = 10.435\,\text{N/mm} \tag{5.22}$$

$$N_{y,\text{cr}} = \frac{\pi^2 K_y D}{a^2} = \frac{3.14^2 \cdot 4.00 \cdot 6641.23}{83.4^2} = 37.694\,\text{N/mm} \tag{5.23}$$

Also check against the allowable.

$$\sigma_{\text{yield,comp}} t = 324 \cdot 1.0 = 324\,\text{N/mm} \tag{5.24}$$

Where $N_{x,\text{cr}}$, $N_{y,\text{cr}} < \sigma_{\text{yield,comp}} t$ which means that the allowable will not be used. Next, compute the stress ratios, where F_x and F_y flips their signs.

$$R_x = \frac{F_x}{N_{x,\text{cr}}} = \frac{3.7831}{10.4352} = 0.362 \tag{5.25}$$

$$R_y = \frac{F_y}{N_{y,\text{cr}}} = \frac{8.5417}{37.6943} = 0.226 \tag{5.26}$$

Repeat the last steps, but for shear.

$$N_{xy,\text{cr}} = \frac{\pi^2 K_s D}{a^2} = \frac{3.14^2 \cdot 5.60 \cdot 6641.23}{83.4^2} = 52.718\,\text{N/mm} \tag{5.27}$$

$$\sigma_{\text{ultimate,shear}} t = 289 \cdot 1.0 = 289\,\text{N/mm} \tag{5.28}$$

Where, $N_{xy,\text{cr}} < \sigma_{\text{ultimate,shear}} t$ which means that the allowable will not be used.

$$R_{xy} = \frac{F_{xy}}{N_{xy,\text{cr}}} = \frac{1.061}{52.718} = 0.020 \tag{5.29}$$

Now compute the MS for the combined loading, combining the shear with the force in the x- and y-direction separately. As well, checking the von Mises stress against our allowable.

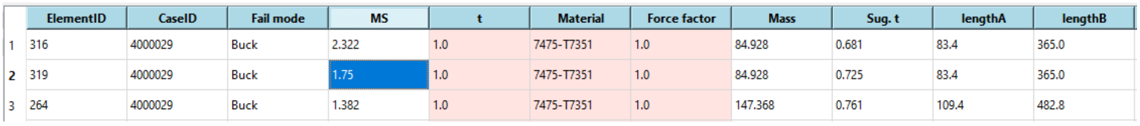
$$\begin{aligned} \text{MS} = \frac{2}{R_x + (R_x^2 + 4R_{xy}^2)^{0.5}} - 1 &= \frac{2}{0.362 + (0.362^2 + 4 \cdot 0.020^2)^{0.5}} - 1 \\ &= 1.750 \end{aligned} \tag{5.30}$$

$$\begin{aligned} \text{MS} = \frac{2}{R_y + (R_y^2 + 4R_{xy}^2)^{0.5}} - 1 &= \frac{2}{0.226 + (0.226^2 + 4 \cdot 0.020^2)^{0.5}} - 1 \\ &= 3.381 \end{aligned} \tag{5.31}$$

$$
\begin{aligned}
\sigma_{vM} &= \sqrt{\sigma_x^2 + \sigma_y^2 + 3\sigma_{xy}^2 - \sigma_x\sigma_y} = \sqrt{(F_x/t)^2 + (F_y/t)^2 + 3(F_{xy}/t)^2 - F_xF_y/t^2} \\
&= \sqrt{(3.78/1.0)^2 + (8.54/1.0)^2 + 3\cdot(1.06/1.0)^2 - 3.78\cdot 8.54/1.0^2} \\
&= 7.623\,\mathrm{Pa}
\end{aligned} \tag{5.32}
$$

$$
\mathrm{MS} = \frac{\sigma_{\text{yield,tension}}}{\sigma_{vM}} - 1 = \frac{386}{7.623} - 1 = 49.661 \tag{5.33}
$$

The worst case was with the force in the x-direction combined with the shear for a MS = 1.750. This is the same value display in the GUI which can be seen in figure 5.3 below.



	ElementID	CaseID	Fail mode	MS	t	Material	Force factor	Mass	Sug. t	lengthA	lengthB
1	316	4000029	Buck	2.322	1.0	7475-T7351	1.0	84.928	0.681	83.4	365.0
2	319	4000029	Buck	1.75	1.0	7475-T7351	1.0	84.928	0.725	83.4	365.0
3	264	4000029	Buck	1.382	1.0	7475-T7351	1.0	147.368	0.761	109.4	482.8

Figure 5.3: Printscreen from the GUI when under the skins "Properties" tab, showing the MS for some elements, including element 319.

The tool also suggests a new thickness of $t = 0.725$ mm which is again found through Brent's algorithm shown in section 2.3. Below, the flexural rigidity is re-calculated with this new thickness.

$$
D = \frac{Et^2}{12(1-\nu^2)} = \frac{71016 \cdot 0.725^2}{12\cdot(1-0.33^2)} = 3490.80\,\mathrm{Nmm} \tag{5.34}
$$

And the critical forces.

$$
N_{x,\text{cr}} = \frac{\pi^2 K_x D}{b^2} = \frac{3.14^2 \cdot 21.21 \cdot 3490.80}{365.0^2} = 5.479\,\mathrm{N/mm} \tag{5.35}
$$

$$
N_{y,\text{cr}} = \frac{\pi^2 K_y D}{a^2} = \frac{3.14^2 \cdot 4.00 \cdot 3490.80}{83.4^2} = 19.793\,\mathrm{N/mm} \tag{5.36}
$$

$$
N_{xy,\text{cr}} = \frac{\pi^2 K_s D}{a^2} = \frac{3.14^2 \cdot 5.60 \cdot 3490.80}{83.4^2} = 27.710\,\mathrm{N/mm} \tag{5.37}
$$

$$
\sigma_{\text{yield,comp}}t = 324 \cdot 0.725 = 234.9\,\mathrm{N/mm} \tag{5.38}
$$

$$
\sigma_{\text{ultimate,shear}}t = 289 \cdot 0.725 = 209.525\,\mathrm{N/mm} \tag{5.39}
$$

where $N_{x,\text{cr}}, N_{y,\text{cr}} < \sigma_{\text{yield,comp}}t$ and $N_{xy,\text{cr}} < \sigma_{\text{ultimate,shear}}t$. Next, the stress ratios.

$$
R_x = \frac{F_x}{N_{x,\text{cr}}} = \frac{3.783}{5.479} = 0.690 \tag{5.40}
$$

$$
R_y = \frac{F_y}{N_{y,\text{cr}}} = \frac{8.541}{19.793} = 0.431 \tag{5.41}
$$

$$R_{xy} = \frac{F_{xy}}{N_{xy,\text{cr}}} = \frac{1.061}{27.710} = 0.038 \tag{5.42}$$

And finally, the MS.

$$\begin{aligned} \text{MS} = \frac{2}{R_x + (R_x^2 + 4R_{xy}^2)^{0.5}} - 1 &= \frac{2}{0.690 + (0.690^2 + 4 \cdot 0.038^2)^{0.5}} - 1 \\ &= 0.048 \end{aligned} \tag{5.43}$$

$$\begin{aligned} \text{MS} = \frac{2}{R_y + (R_y^2 + 4R_{xy}^2)^{0.5}} - 1 &= \frac{2}{0.431 + (0.431^2 + 4 \cdot 0.038^2)^{0.5}} - 1 \\ &= 1.299 \end{aligned} \tag{5.44}$$

$$\begin{aligned} \sigma_{vM} &= \sqrt{\sigma_x^2 + \sigma_y^2 + 3\sigma_{xy}^2 - \sigma_x\sigma_y} = \sqrt{(F_x/t)^2 + (F_y/t)^2 + 3(F_{xy}/t)^2 - F_xF_y/t^2} \\ &= \sqrt{(3.78/0.725)^2 + (8.54/0.725)^2 + 3 \cdot (1.06/0.725)^2 - 3.78 \cdot 8.54/0.725^2} \\ &= 10.532\,\text{Pa} \end{aligned} \tag{5.45}$$

$$\text{MS} = \frac{\sigma_{\text{yield,tension}}}{\sigma_{vM}} - 1 = \frac{386}{10.532} - 1 = 36.650 \tag{5.46}$$

The worst case was again the force in the x-direction combined with the shear for a MS = 0.048 which is close to the sought MS = 0.050

5.2 FEA validation

Here, the same elements and loads as used in the hand calculations are created in HyperMesh and simulated in Nastran. Though this, one can get a numerical result to verify that the analytical method gives good results. The output from the simulation are the real eigenvalues, which corresponds to $\frac{P_{\text{cr}}}{F}$. This means that MS = eigenvalue − 1, similar to equation 2.3.

5.2.1 Stringer

The same stringer as section 5.1 with its original thickness is created in HyperMesh and can be seen below in figure 5.4. The ends are simply supported and are only free to move in the in-plane direction. A fictitious boundary condition has also been added in the center to prevent rigid body motion by constraining movement in the in-plane direction.

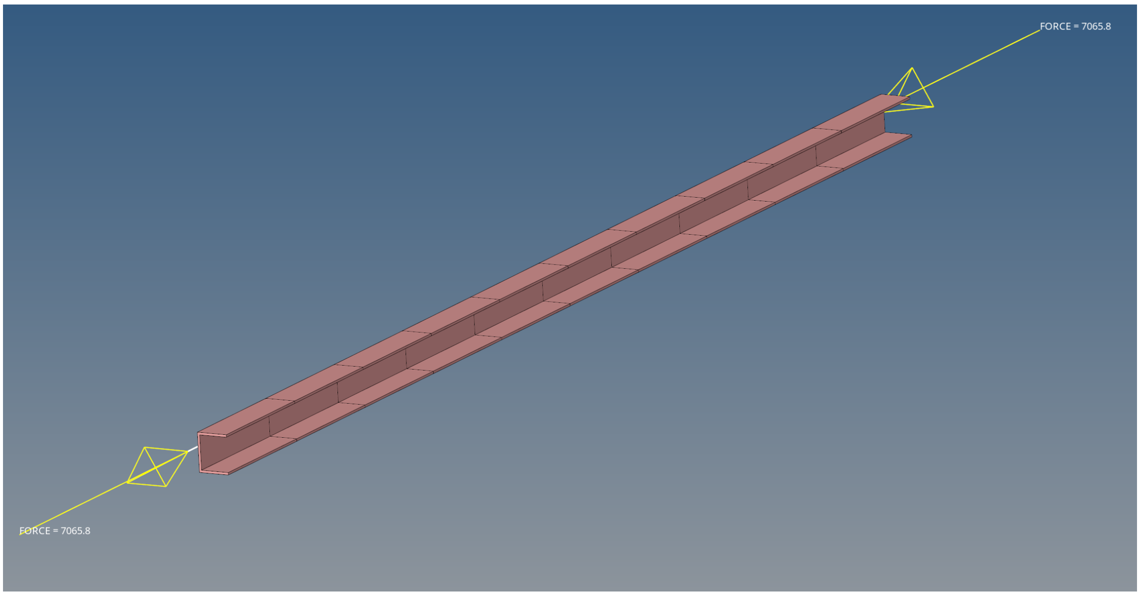


Figure 5.4: Geometry of stringer, with yellow arrows being the forces it is subjected to.

It is also simply supported on both ends, with the yellow-colored load vectors visible. A 2D image of the C-section can be seen below in figure 5.5.

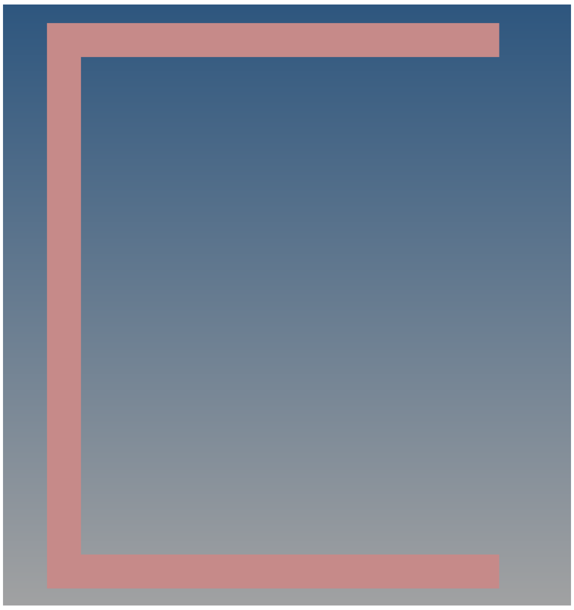


Figure 5.5: C-section of stringer, thickness $t_w = t_f = 1.5\,\text{mm}$

The analysis is sent to Nastran for solving and the result gives two corresponding eigenvalues seen in table 5.3 below.

Mode 1	0.536
Mode 2	0.775

Table 5.3: The real eigenvalues of the simulated C-section stringer with $t_w = t_f = 1.5\,\text{mm}$.

The first buckling mode is always the lowest, and will thus correspond to the lowest MS, which is the one of interest. It can be computed though the relation

$$\text{MS} = \text{eigenvalue} - 1 = 0.536 - 1 = -0.536 \tag{5.47}$$

which is close to the analytical result of MS $= -0.463$ in section 5.1.1. Next, the thickness is changed to $t_w = t_f = 3.377\,\text{mm}$. The new cross-section and eigenvalues can be seen below in figure 5.6 and table 5.4.

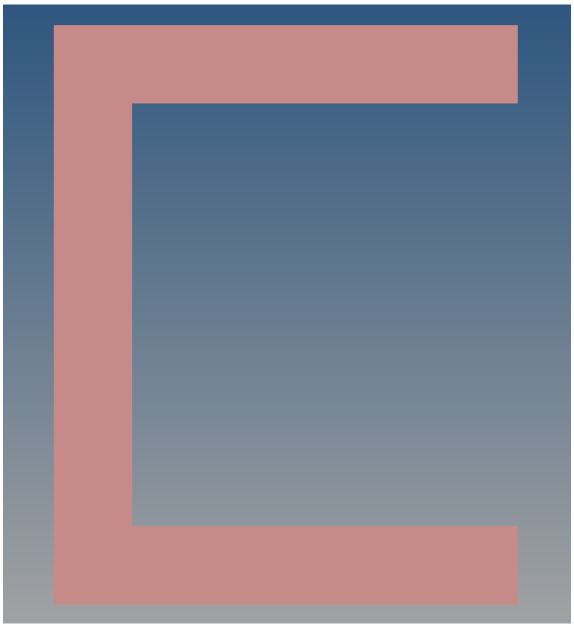


Figure 5.6: C-section of stringer, new thickness $t_w = t_f = 3.377\,\text{mm}$

Mode 1	1.048
Mode 2	2.464

Table 5.4: The real eigenvalues of the simulated C-section stringer with thickness $t_w = t_f = 3.377\,\text{mm}$

The first buckling mode gives

$$\text{MS} = \text{eigenvalue} - 1 = 1.048 - 1 = 0.048 \tag{5.48}$$

which again is close to the desired MS $= 0.050$, confirming the analytical solution for the stringer.

5.2.2 Skin

The same procedure is done but for the skin in section 5.1.2. All boundaries are simply supported and locked in the z-direction. Fictitious boundary conditions has also been added on two opposite corner nodes, where one is constrained in the x-direction and the other in both x- and y-direction. This is again to prevent rigid body motion. The created geometry and loads can be seen in figure 5.7 below.

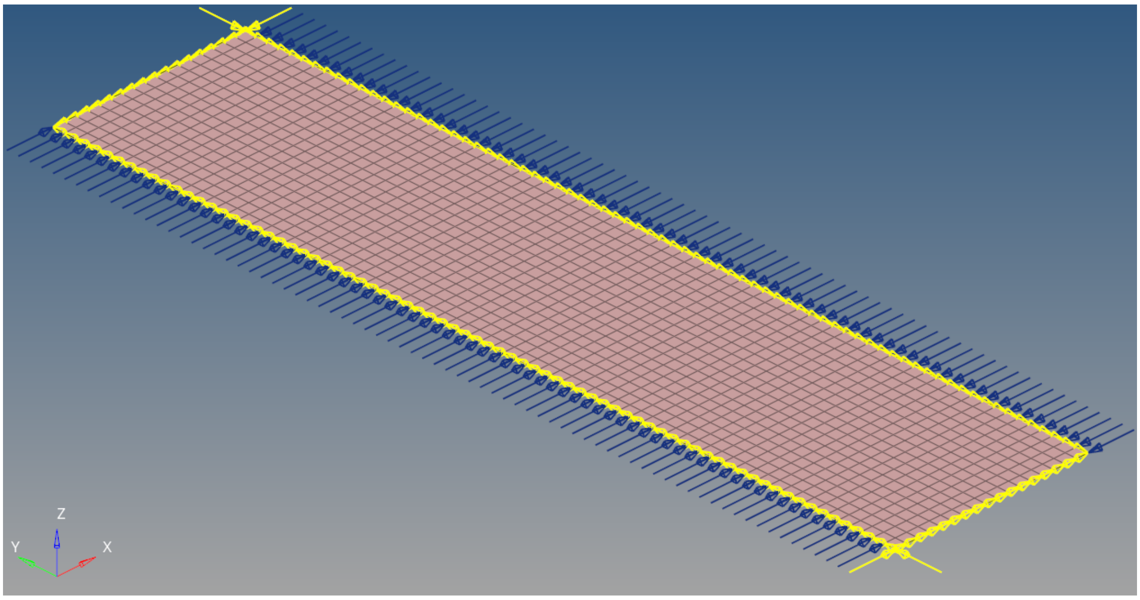


Figure 5.7: Skin with forces in x- and xy-direction. Thickness of $t = 1.0\,\text{mm}$
.

Here the skin is simply supported on all sides. The blue arrows being the force in the x-direction with $F_x = -3.783\,08\,\text{N/mm}$ and the yellow arrows being the shear force in the xy-direction with $F_{xy} = 1.061\,53\,\text{N/mm}$. The eigenvalues obtained when solved can be seen in table 5.5 below.

Mode 1	2.758
Mode 2	3.616
Mode 3	5.276
Mode 4	7.962
Mode 5	10.035

Table 5.5: The real eigenvalues of the simulated skin with thickness $t = 1.0\,\text{mm}$, $F_x = -3.783\,\text{N/mm}$ and $F_{xy} = 1.061\,\text{N/mm}$.

Here, the first eigenvalue gives

$$\text{MS} = \text{eigenvalue} - 1 = 2.758 - 1 = 1.758 \tag{5.49}$$

which is the same as the analytical value of MS = 1.750. Next the same skin is simulated but with the force being in the y-direction instead of x-direction with

$F_y = -8.541\,\text{N/mm}$. This can be seen below in figure 5.8 where F_y is the red arrows.

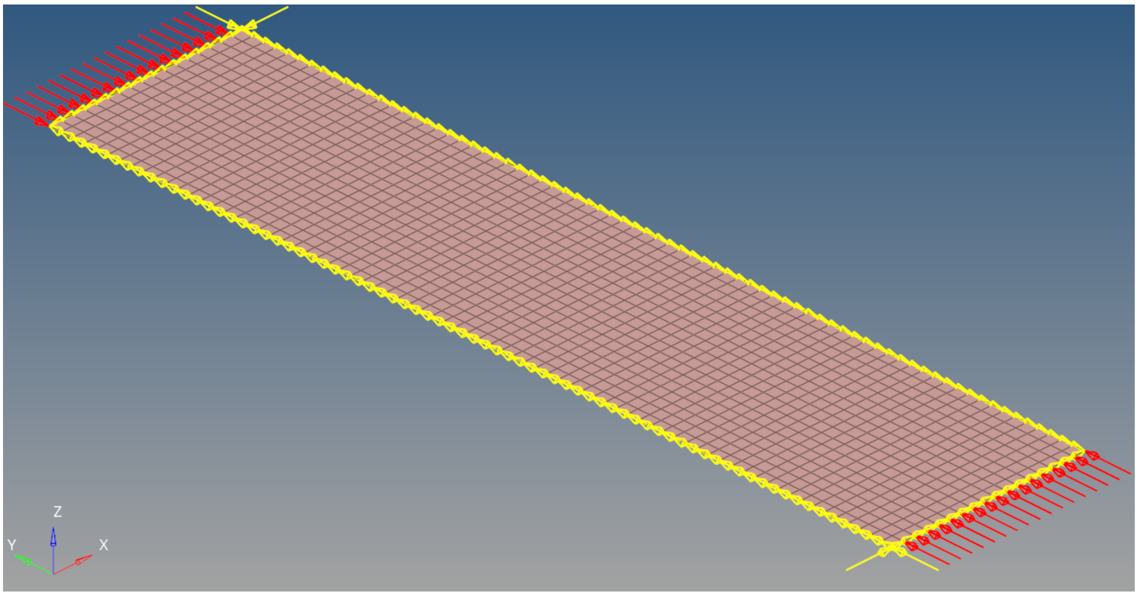


Figure 5.8: Skin with forces in y- and xy-direction. Thickness of $t = 1.0\,\text{mm}$.

The corresponding eigenvalues are seen below in table 5.6.

Mode 1	4.391
Mode 2	4.433
Mode 3	4.806
Mode 4	5.008
Mode 5	5.396

Table 5.6: The real eigenvalues of the simulated skin with thickness $t = 1.0\,\text{mm}$, $F_y = -8.541\,\text{N/mm}$ and $F_{xy} = 1.061\,\text{N/mm}$.

This gives

$$\text{MS} = \text{eigenvalue} - 1 = 4.391 - 1 = 3.391 \tag{5.50}$$

which is close to the analytical MS = 3.380. For these two solutions, the lowest MS was MS = 1.758. The same answer and case as in the analytical solution.

Lastly, the thickness is changed to $t = 0.725\,\text{mm}$ and the results for the two simulations can be seen below in table 5.7 and 5.8.

Mode 1	1.051
Mode 2	1.379
Mode 3	2.013
Mode 4	3.038
Mode 5	3.948

Table 5.7: The real eigenvalues of the simulated skin with thickness $t = 0.725\,\text{mm}$, $F_x = -3.783\,\text{N/mm}$ and $F_{xy} = 1.061\,\text{N/mm}$.

Mode 1	1.676
Mode 2	1.692
Mode 3	1.834
Mode 4	1.911
Mode 5	2.060

Table 5.8: The real eigenvalues of the simulated skin with thickness $t = 0.725\,\text{mm}$, $F_y = -8.541\,\text{N/mm}$ and $F_{xy} = 1.061\,\text{N/mm}$.

The margin of safety for the lowest eigenvalue in the two tables are

$$\text{MS} = \text{eigenvalue} - 1 = 1.051 - 1 = 0.051 \tag{5.51}$$

$$\text{MS} = \text{eigenvalue} - 1 = 1.676 - 1 = 0.676 \tag{5.52}$$

where the lowest MS = 0.051 which is close to the desired MS = 0.050. But, the second simulation with force in F_y have a MS = 0.676 and is deviating from the analytical MS = 1.299. This could be due to an error in the simulation when applying load and boundary conditions, or the ratio between a and b. Since the skin is quite long when comparing the side lengths, both K_x and K_y are found in the extreme values of their graphs since they are a function of the ratio b/a. There have been limitations of what values b/a can be for these combine loading condition equations for skins in some literature, but no limit was found in the literature for this method.

6
Assessment

The assessment of the results is considered with regard to the purpose and question formulations in section 1.3.

6.1 Automation

When the program is up-and-running, it does what it set out to do. It extracts and summarizes the results from the GFEM which somewhat automates the post-processing analysis. Showing critical information such as MS, failure mode, geometrical and material information.

But for the tool to fulfil its purpose of removing all repetitive error-prone processes as well as saving time, this will not be done until the sections and its geometries are defined in the GFEM and the extensive use of the Excel file disappears. And until this is done, the problem of automatically getting the changes made to the elements back into the GFEM cannot be solved, but that is outside the scope this thesis.

6.2 Scaling

The program has been written in such a way that it should easily allow for addition of more analysis and other changes. It is hard to say if this goal has been reached, but it has always been in mind when the code was written. A new developer could hopefully, with the commented code, see how the current analysis has been done and, through the same DataFrame, perform more analysis.

A change that should be done to the code in the future is to be able to read and extract the section and its geometries from the GFEM when it has been added. It is predicted that this can be done with only a small alteration to the code.

6.3 Ease of use

The program is easy to use and gives the user several different options. It shows short summaries or a more detailed DataFrame, as well as giving the option to see all the cases for every element to compare them. Instead of starting the analysis from scratch when closed, the save and load buttons gives the user the option to continue where it left off. The results are also easily exported to an Excel table for documentation. Assertions are integrated into the GUI to warn the user when something has gone wrong and what, which also makes it easy to edit the elements.

The ease of use to go from the GFEM to the tool is again not optimal because of the Excel file, where the user must define geometries and manually assign them to every element that will be part of the analysis. The same goes with getting the exported changes done to the materials and geometries back into the GFEM.

6.4 Validation

The hand calculations confirmed the same results as displayed in the GUI. This confirms that the used formulas are coded correctly and that the minimization find the right minimum.

The FEA confirmed the analytical results by giving the same MS for almost all the tested cases. It confirmed the combined loading cases with the lowest MS for the skins, but one simulation differed significantly from the analytical results. This was the skin with the optimized thickness, and force in the y- and xy-direction. The FEA and analytical solution for this skin did however show the same result with the original thickness of $t = 1.0\,\mathrm{mm}$ applied.

The reason for this uncertainty is unclear. It could be a numerical error, lack of mesh density, wrong settings or analytically unsupported geometry (b/a too large). But for the skin with the force in the x- and xy-direction, the same exact results were obtained for both $t = 1.0\,\mathrm{mm}$ and $t = 0.725\,\mathrm{mm}$ when comparing the analytical and numerical solution. The same procedure was done when changing the thickness of the plate with the force in the y- and xy-direction, which makes the difference in the results hard to understand.

7
Concluding remarks and future work

In this last chapter, improvements that can be made to the tool in the future will be discussed and concluding remarks will be made.

7.1 Concluding remarks

As of right now, ”H-ADAPT” is not as efficient as it could be, with the biggest drawbacks being that the geometries must be read from the Excel file. This slows the post-processing automation considerably. When the geometries have been added into the pre-processor HyperMesh and are being able to be read through the BDF file, the tool can start to reach its full potential with automating the sizing of the structure, until it must be re-read into the GFEM in the pre-processor. However, when the user has added the geometries and assign them to the elements in the Excel file, the tool does what it was set out to do. Fast analysis of the elements where the user easily can see the worst cases and what forces are associated with it, along with optimization algorithms for sizing and an Excel output table.

The tool has been created with the mindset that it should be scalable and generic, which means that it could be the start of a broader analysis program that can be used for sizing of larger areas of the airplane structures.

7.2 Future work

The tool is not completed, and developments can be done in the future to automate more of the process and broaden the analysis.

7.2.1 Dependence on Excel table

The whole tool is now dependent on the Excel table seen in figure 3.1 as an input, which contains certain material and geometrical information. As long as a material ID is assigned to every element in the GFEM, having the material information in an external Excel table is a good solution. One can then easily change the material for an element at the source in HyperMesh without having to alter anything in the code or Excel table. And if a material is added, it is easy to add it to the rest of the materials.

The geometrical properties of the stringers on the other hand have room for improvement. Right now, the user must define the different geometries in the Excel table and also assign the geometrical ID to all elements of interest before the analysis. This is a slow and cumbersome process which could be streamlined to increase the automation of the post-processing analysis. For this to happen, the geometrical sections have to be assigned to the elements directly in HyperMesh which it is not at the moment. This is the goal for the future and until then, it must be read in the Excel table. This is however not a problem for the skins since they don't have a geometrical section, only a thickness, which is currently defined in the HyperMesh model.

7.2.2 Expansion of analysis

With the goal of only analyzing the stringer and skins and their effect on each other, a thorough analytical analysis has been done for preliminary sizing of the elements. But to do a complete analysis of the fuselage, one would have to broaden the tool. First step would be to add analysis of the frame. To analyze these, one would need the element forces from adjacent elements which is currently present in the OP2 file, but unable to be read by PyNastran. A function would have to be written to read this information, either from the binary OP2 file or the text based F06 file, which is out of the scope of this thesis. Other examples of future improvement for analysis could be to add information about the rivet and fastener placements to add inter rivet buckling.

Right now, the tool only does analysis on the elements of the fuselage. It could be expanded to handle more areas of the airplane, such as empennage (tail section) and wings, which would be a whole new thesis in itself because of the more complex structures, theory and equations this would bring. But it could build on this program because of the way the tool can be scaled for addition of functionalities.

For the skin analysis, the chosen analytical solution for buckling allows for x- or y-force together with the xy-force. Preferably, one would like all three forces to be present at the same time to get a better answer.

7.2.3 Automation

When the user is done with the analysis and sized its elements, it needs to be re-read into the pre-processor at some time to be solved again for the new load paths and cases. For the process to be fully automated, in the sense without manual repetitive copy pastes, this expansion of the tool would also be beneficial.

Bibliography

[1] Altair. *Altair HyperMesh.* URL: https://altair.com/hypermesh. Accessed: 07.05.2024.

[2] Candelaria Bergero et al. "Pathways to net-zero emissions from aviation". In: *Nature Sustainability* 6.4 (Apr. 2023), pp. 404–414. ISSN: 2398-9629. DOI: 10.1038/s41893-022-01046-9. URL: https://doi.org/10.1038/s41893-022-01046-9.

[3] *Brent's method.* https://mathsfromnothing.au/brents-method/?i=1. Accessed: 2024-04-11.

[4] E. F. Bruhn. *Analysis and design of flight vehicle structures.* Ohio: Tri-State Offset Company, 1973.

[5] *Calculation Tools and Engineering Resources.* https://calcresource.com/. Accessed: 2024-03-8.

[6] Michael Chun-Yung Niu. *Airframe Stress Analysis and Sizing, Second Edition.* Hong Kong: Hong Kong Conmilit Press LTD, 1999.

[7] Steven Doyle. *pyNastran's documentation.* URL: https://pynastran-git.readthedocs.io/en/latest/. Accessed: 2024-04-26.

[8] Bjorn Fehrm. *The true cost of Electric Aircraft.* URL: https://leehamnews.com/2021/07/01/the-true-cost-of-electric-aircraft/. Accessed: 2024-05-07.

[9] Hexagon. *MSC Nastran Multidisciplinary structural analysis.* URL: https://hexagon.com/products/product-groups/computer-aided-engineering-software/msc-nastran. Accessed: 07.05.2024.

[10] *Pandas documentation.* URL: https://pandas.pydata.org/docs/. Accessed: 2024-04-26.

[11] Hannah Ritchie, Pablo Rosado, and Max Roser. "CO2 and Greenhouse Gas Emissions". In: *Our World in Data* (2023). https://ourworldindata.org/co2-and-greenhouse-gas-emissions.

[12] Johan Kos Thijs Boonekamp Elisabeth van der Sman Bram Peerlings. *Destination 2050 – A route to net zero European Aviation.* URL: https://www.destination2050.eu/. Accessed: 25.01.2024.

[13] S. Timoshenko. *Strength of materials , Part I Elementary Theory and Problems.* D. Van Nostrand Company, Inc., 250 Fourth Avenue, New York 3: D. Van Nostrand Company, I, 1940.

[14] Andrew Wood. *Aircraft Fuselage Structural Design and Layout.* URL: https://aerotoolbox.com/fuselage-structure/. Accessed: 30.04.2024.

A

Python example code

```
import pandas as pd
import numpy as np

def Inertia_T(h, b, t_f, t_w):
    """
    Second moment of inertia for a T-section

    Parameters
    ----------
    h : Float
        Hight.
    b : Float
        Width.
    t_f : Float
        Flange thickness.
    t_w : Float
        Webb thickness.

    Returns
    -------
    I : Float
        Lowest second moment of inertia.
    Ix : Float
        Second moment of inertia, x-direction.
    Iy : Float
        Second moment of inertia, y-direction.
    A : Float
        Cross section area.

    """
    A = (b - t_w) * t_f + h *t_w
    Ix1 = t_w * h**3 / 3 + (b - t_w) * t_f**3 / 3
    center = (1/A) * ((t_w * h**2)/2 + ((b - t_w) * t_f**2)/2)

    Ix = Ix1 - A * center**2
    Iy = (h - t_f) * t_w**3 / 12 + t_f * b**3 / 12
```

```
    if Iy < Ix:
        I = Iy
    else:
        I = Ix
    assert h >= t_f, f'Wrong dimensions: h = {h} and t_f = {t_f}. Must satisfy h >=
    assert b >= t_w, f'Wrong dimensions: b = {b}, t_w = {t_w}. Must satisfy b >= t_
    return I, Ix, Iy, A

def Inertia_Z(h, b, t_f, t_w):
    """
    Second moment of inertia for a Z-section

    Parameters
    ----------
    h : Float
        Hight.
    b : Float
        Width.
    t_f : Float
        Flange thickness.
    t_w : Float
        Webb thickness.

    Returns
    -------
    I : Float
        Lowest second moment of inertia.
    Ix : Float
        Second moment of inertia, x-direction.
    Iy : Float
        Second moment of inertia, y-direction.
    A : Float
        Cross section area.

    """

    b_f = b - t_w
    A = h * t_w + 2 * b_f * t_f

    Ix = (b_f * t_f *(h - t_f)**2)/2 + b_f * t_f**3 / 6 + t_w * h**3 / 12
    Iy = (b_f * t_f *(b_f + t_w)**2) / 2 + (t_f * b_f**3) / 6 + (h * t_w**3) / 12

    if Iy < Ix:
        I = Iy
    else:
        I = Ix
```

```
    assert h >= 2*t_f, f'Wrong dimensions: h = {h} and t_f = {t_f}. Must satisfy h
    assert b >= t_w, f'Wrong dimensions: b = {b}, t_w = {t_w}. Must satisfy b >= t_
    return I, Ix, Iy, A

def Inertia_C(h, b, t_f, t_w):
    """
    Second moment of inertia for a C-section

    Parameters
    ----------
    h : Float
        Hight.
    b : Float
        Width.
    t_f : Float
        Flange thickness.
    t_w : Float
        Webb thickness.

    Returns
    -------
    I : Float
        Lowest second moment of inertia.
    Ix : Float
        Second moment of inertia, x-direction.
    Iy : Float
        Second moment of inertia, y-direction.
    A : Float
        Cross section area.

    """

    A = 2 * b * t_f + (h - 2 * t_f) * t_w

    Ix = (b * h**3) / 12 - ((b - t_w) * (h - 2 * t_f)**3) / 12

    d = (1/A) * (((h - 2*t_f) * t_w**2) / 2 + t_f * b**2)
    Iy0 = ((h - 2 * t_f) * t_w**3) / 3 + 2 * (t_f * b**3) / 3
    Iy = Iy0 - A * d**2

    if Iy < Ix:
        I = Iy
    else:
        I = Ix
    assert h >= 2*t_f, f'Wrong dimensions: h = {h} and t_f = {t_f}. Must satisfy h
    assert b >= t_w, f'Wrong dimensions: b = {b}, t_w = {t_w}. Must satisfy b >= t_
```

```python
    return I, Ix, Iy, A

def slenderness_func(I, A, L):
    """
    Function to compute the slenderness ratio

    Parameters
    ----------
    I : Float
        Lowest second moment of inertia.
    A : Float
        Area of the cross-section.
    L : Float
        Length of the element.

    Returns
    -------
    Lambda : Float
        The slenderness ratio.

    """

    r_gyration = np.sqrt(I / A)
    Lambda = L / r_gyration
    return Lambda

def mass_func(rho, A, L):
    """
    The mass of the elemtn

    Parameters
    ----------
    rho : float
        Density of the material.
    A : Float
        Area of the cross-section.
    L : Float
        Lenght of the element.

    Returns
    -------
    m : Float
        Mass of the elemtn in grams.

    """
```

```
    m = rho * (A * L) / 1000
    return m

def euler_buckling(E, L, I):
    """
    Regular Euler buckling

    Parameters
    ----------
    E : Float
        Elastic modulus.
    L : Float
        Lenght.
    I : Float
        Lowest second moment of inertia.

    Returns
    -------
    P_cr : Float
        Critical buckling load.

    """

    P_cr = np.pi**2 * E * I / (L**2)
    return P_cr

def euler_transit(E, A, Lambda, sigma_yield_comp):
    """
    Eulerbuckling in the transit region when element is not slender.

    Parameters
    ----------
    E : Float
        Elastic modulus.
    A : Float
        Area of cross-section.
    Lambda : Float
        Slenderness ratio.
    sigma_yield_comp : Float
        Compressive yield strength of material.

    Returns
    -------
    P_cr : Float
        Critical buckling load.
```

```
    """

    P_cr = sigma_yield_comp * (1 - sigma_yield_comp / (4 * np.pi**2 * E) * Lambda**
    return P_cr

def tensile(F, A):
    """
    Tensile stress

    Parameters
    ----------
    F : Float
        Tensile Force.
    A : Float
        Area of cross-section.

    Returns
    -------
    stress : Float
        Experienced tensile stress on cross-section.

    """

    stress = F / A
    return stress

def material_func(mat_nr, file_path):
    """
    Read the material file to get material parameters.

    Parameters
    ----------
    mat_nr : Int
        The identification number for the material.
    file_path : Str
        The file path to the excel material file.

    Returns
    -------
    E : Float
        Elastic modulus.
    rho : Float
        Density.
    sigma_yield : Float
        Tensile yield stress.
    sigma_yield_comp : Float
```

```
        Compressive yield stress.

    """

    bdf_mat = pd.read_excel(file_path, usecols='G:P')
    bdf_mat.dropna(how='all', inplace=True)
    bdf_mat.rename(columns={bdf_mat.columns[0]: 'MatID'}, inplace=True)
    assert (bdf_mat['MatID'] == mat_nr).any(), 'Choosen MatID does not exist in exc

    bdf_mat.set_index('MatID', inplace=True)
    E = bdf_mat['Emod'][mat_nr]
    rho =bdf_mat['rho'][mat_nr]
    sigma_yield =bdf_mat['yield_t'][mat_nr]
    sigma_yield_comp =bdf_mat['yield_c'][mat_nr]
    return E, rho, sigma_yield, sigma_yield_comp

def MS_buck_func(P_cr, F):
    """
    Margin of safety to buckling

    Parameters
    ----------
    P_cr : Float
        Critical buckling force.
    F : Float
        Force.

    Returns
    -------
    MS_buck : Float
        Marigin of safety to buckling.

    """

    MS_buck = P_cr / (F*(-1)) - 1 #Mult F by -1 since compression is positive
    return MS_buck

def MS_tensile_func(sigma_yield, stress):
    """
    Margin of safety in tensile stress

    Parameters
    ----------
    sigma_yield : Float
        Yield stress in tension.
    stress : Float
```

```
        Experienced stress.

    Returns
    -------
    MS_tensile : Float
        Margin of safety in tensile stress.

    """

    MS_tensile = sigma_yield / stress - 1
    return MS_tensile


def function_df(h, b, t_f, t_w, L, F, Section_type, E, rho, sigma_yield, sigma_yiel
    """
    Calling all the functions and giving the required outputs for the GUI.

    Parameters
    ----------
    h : Float
        Hight.
    b : Float
        Width.
    t_f : Float
        Flange thickness.
    t_w : Float
        Webb thickness.
    L : Float
        Lenght.
    F : Float
        Force.
    Section_type : Str
        Type of section.
    E : Float
        Elastic modulus.
    rho : Float
        Density.
    sigma_yield : Float
        Yield stress in tension.
    sigma_yield_comp : Float
        Yield stress in compression.

    Returns
    -------
    MS : Float
        Marigin of Safety.
```

```
    m : Float
        Mass.
    A : Float
        Area of cross section.
    Mode : Str
        Failure more.

    """

    #Calling of functions
    if Section_type == 'T':
        I, Ix, Iy, A = Inertia_T(h, b, t_f, t_w)
    elif Section_type == 'Z':
        I, Ix, Iy, A = Inertia_Z(h, b, t_f, t_w)
    elif Section_type == 'C':
        I, Ix, Iy, A = Inertia_C(h, b, t_f, t_w)
    else:
        assert not Section_type not in ['T', 'Z', 'C'], \
            'Wrong section type choosen, please choose T, Z or C'

    m = mass_func(rho, A, L)

    #Compression
    if F < 0:
        Lambda = slenderness_func(I, A, L)
        Lambda_cr = np.sqrt(2 * np.pi**2 * E / sigma_yield_comp)
        #if sigma_yield_comp*A / (-F) < 1:
            #print('Cladding factor needed, ratio:',sigma_yield_comp*A / (-F))

        if Lambda >= Lambda_cr:
            P_cr = euler_buckling(E, L, I)
            MS = MS_buck_func(P_cr, F)
            Mode = 'Buck'

        elif Lambda < Lambda_cr:
            P_cr = euler_transit(E, A, Lambda, sigma_yield_comp)
            MS = MS_buck_func(P_cr, F)
            Mode = 'Buck Trans. region'
    #Tension
    else:
        stress = tensile(F, A)
        MS = MS_tensile_func(sigma_yield, stress)
        P_cr = 0
        Mode = 'Tensile'
    return MS, m, A, Mode
```

```python
def edit_function(h, b, t_f, t_w, Section_type, rho, lengthA):
    """
    Function used for the "edit" button in the GUI.

    Parameters
    ----------
    h : Float
        Hight.
    b : Float
        Width.
    t_f : Float
        Flange thickness.
    t_w : Float
        Webb thickness.
    Section_type : Str
        Type of sectionh.
    rho : Float
        Density.
    lengthA : Float
        Length of element.

    Returns
    -------
    Ix: Float
        Second moment of inertia in x-direction.
    Iy: Float
        Second moment of inertia in x-direction.
    mass: Float
        Mass.
    Cg : Float
        Center of gravity position.

    """

    #Calling of functions
    if Section_type == 'T':
        I, Ix, Iy, A = Inertia_T(h, b, t_f, t_w)
        x_center = round(b/2,2)
        y_center = round((1 / (2 * A)) * (t_w * h**2 + (b - t_w) * t_f**2),2)
        Cg = f"({x_center}, {y_center})"

    elif Section_type == 'Z':
        I, Ix, Iy, A = Inertia_Z(h, b, t_f, t_w)
        x_center = 0.0
        y_center = round(h/2,2)
        Cg = f"({x_center}, {y_center})"
```

```
elif Section_type == 'C':
    I, Ix, Iy, A = Inertia_C(h, b, t_f, t_w)
    x_center = round((1/A) * ((h - 2 * t_f) * t_w**2 / 2 + t_f * b**2 ),2)
    y_center = round(h/2, 2)
    Cg = f"({x_center}, {y_center})"
else:
    assert not Section_type not in ['T', 'Z', 'C'], \
        'Wrong section type choosen, please choose T, Z or C'

mass = mass_func(rho, A, lengthA)
return round(Ix,3), round(Iy,3), round(mass,3), Cg
```

DEPARTMENT OF INDUSTRIAL AND MATERIAL SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY