Universitat Politectinca de Catalunya

Wing Beam Structure Analysis

Computational Engineering

Student name: Altadill Llasat, Miquel

Professor name: Cante Teran, Juan Carlos

Department: Aerospace Structures

Document: Deliverable 2

Delivery Date: 25/06/2020

MSc in Aeronautic Engineering - ESEIAAT
Universitat Politècnica de Catalunya (UPC)

Contents

List of Figures

List of Tables

1 Problem Statement

For this project it is asked to model the internal wing structure depicted in Figure 1, where it is clearly seen its composition. The wing shown in the upper right side is constituted by ribs and spars that are simplified into multiple elements. It is considered a typical wing structure of a commercial aircraft, composed of a front and rear spars and ribs. Other consideration to take into account for the analysis is the pylon attachment where the thrust and weight of the motor would be transmitted. Figure 1 also shows the reference frame and how the gravity force acts over the system.

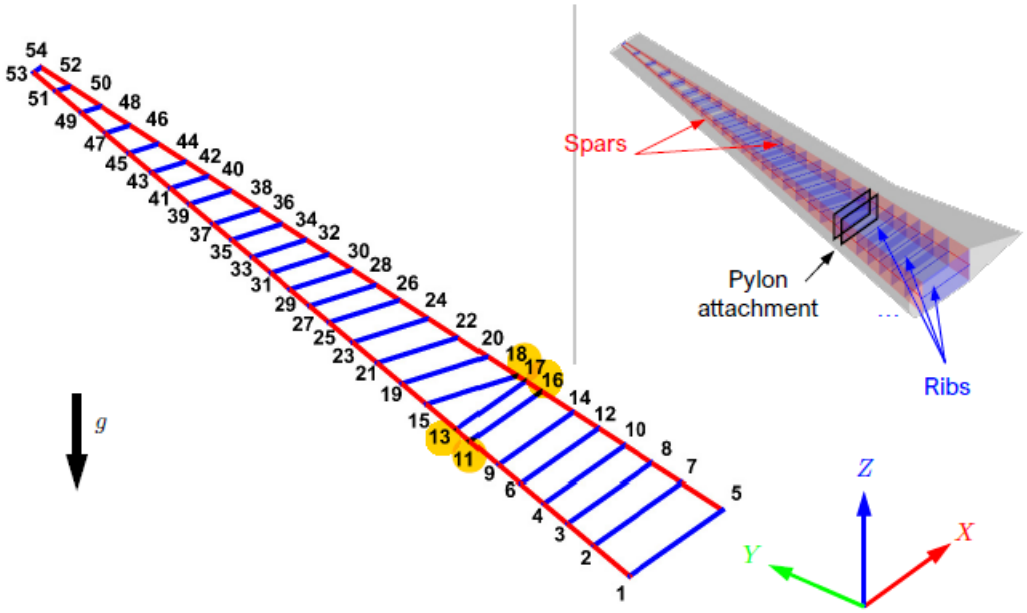

Figure 1: Wing structure representation

For modelling our structure it has been considered the beam element shown in Figure 2. It has been considered different material and beam properties for the ribs and the spars. Table 1 shows the material and beam properties for the ribs and the spars that compose the wing structure.

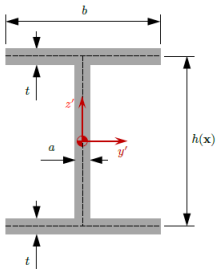

Figure 2: Beam element cross section area

Property	Spars	Ribs
Density ρ $[kg/m^3]$	2450	2050
Young's Modulus E $[GPa]$	70.6	62.5
ν	0.36	0.33
a $[mm]$	12	5
b $[mm]$	97	15
t $[mm]$	6	2.5

Table 1: Parameter that define the structure ribs and spars

For solving these structure it has been provided a data file with all the input data needed to model the wing structure such as the nodal coordinates, the connectivity matrix and the material connectives. In addition to the previous task, for these problem it is required from the geometrical parameters of the beam, for each beam element it is provided its geometrical parameters, its height and the rotation angles. Figure 3 shows the rotation angles $\{\alpha, \beta, \gamma\}$ which correspond with the rotation over the $\{X, Y, Z\}$ axis.

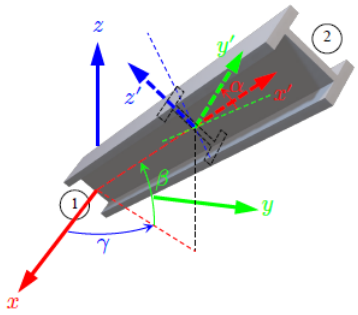

Figure 3: Beam element rotation angles

The following list shows how the nodal connectives are organised for defining each element in order to have a clear definition of its element numbering.

- Ribs elements: From **1** to **27**
- Front spar elements: From **28** to **53**
- Rear spar elements: From **54** to **79**

The elements are defined in the 'InputData.m' file shown in Attachment A.2 and there is a total of 79 elements. The material index for the spars and ribs are also defined:

- Spars: Material index **1**
- Ribs: Material index **2**

2 Loading Conditions

In this section it is defined the loading conditions needed to consider for performing the structural analysis. These conditions are selected in order to match approximately with a real load case. First of all it will be assumed that the aircraft is attached to the aircraft's fuselage at nodes **1** and **5**, at these points the displacements and rotations will be prescribed.

2.1 Wing Weight

Since the wing total weight is not only the sum of the weight of the ribs and spars it is necessary to implement some considerations on how the wing mass is computed. These consideration accounts for the extra weight of the wing fuselage and other internal components of the wing that would affect to the total weight. The total mass would be estimated using an effective density parameter. Then each element density would be computed as

$$\rho_{eff,e} = \rho_e + \hat{\rho} \tag{1}$$

where ρ_e correspond to the material density of the rib or spar element, and $\hat{\rho}$ is a pseudo-density that accounts for the mass of the rest of the wing elements. The pseudo-density can be estimated as

$$\hat{\rho} = \frac{M_w - M_s - Mr}{V_s + V_r} \tag{2}$$

with $M_w = 1550kg$ being the mass of the whole wing, M_s and M_r are respectively the total mass of the spars and the ribs, while V_s and V_r refer to their corresponding total volumes. Once computed the effective density for each element, the total mass of the structure could be computed as

$$M_{tot} = \sum_{e=1}^{N} \rho_{eff,e} \cdot V_e \tag{3}$$

where N is the total element number.

2.2 Engine Loads

In a real aircraft wing one of the most critical points of its design is the point where the motor is attached to the wing. The structure in charge of distributing the motor thrust and weight is called pylon. For this case of study it has been chosen a simplification that distributes the engine loads over two ribs of the structure. The thrust T_e and weight W_e of the engine is transmitted through the pylon as a distributed load into ribs **11-16** and **13-17**, corresponding to elements **7** and **8**. Figure 1 shows in yellow the position of the nodes that compose the pylon attachment.

Since there are two elements holding the motor, half the forces are then applied at each rib, acting as a distributed load. The equations that define the load distribution read as

$$t_e = \frac{T_e}{2 \cdot L_{rib}} \tag{4}$$

$$w_e = \frac{W_e}{2 \cdot L_{rib}} \tag{5}$$

Figure 4 shows a representation of how the thrust and the weight is distributed along the elements that hold the motor loads. For the analysis, the value of T_e must be such to compensate the total drag of the wing without taking into account drag effects produced by the plane body. Finally, the engine mass is considered to have a value of $M_e = 960kg$

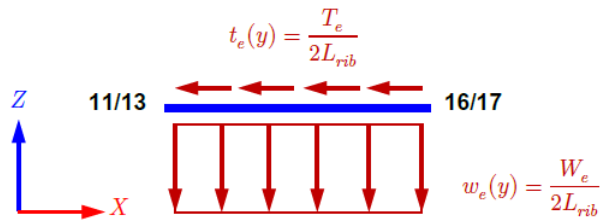

Figure 4: Pylon attachment force distribution

2.3 Aerodynamic Loads

The aerodynamic loads distribution has also been selected in a manner that it gives the problem realism. It is known that the drag and lift forces would have its greatest value near the wing root then it will reduce its value at the same time the profile chord is reduced. The minimum lift and drag values are found at the wing tip, where the profile chord tends to zero value. The lift force would be distributed along the front and rear spar and the drag would be only distributed along the rear spar, these simplification tries to keep physical meaning on how the problem is stated. Figure 5 shows a representation on how the aerodynamic loads are distributed. There it is seen that the lift is applied in the z-direction, while the drag load is applied in the x-direction.

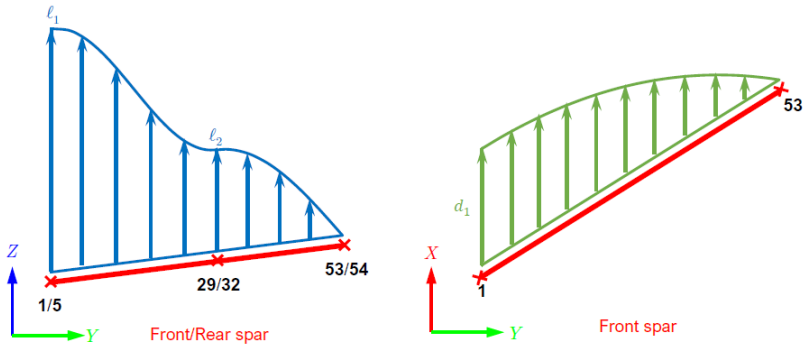

Figure 5: Aerodynamic loads distribution over the wing

The lift and drag distribution is defined by Equations 6 - 7, which are directly related to Figure 5 and also to the parameters shown in Table 2.

$$l(y) = \begin{cases} \frac{1}{2}\left(l_1 + l_2 + (l_1 - l_2)\left[\pi \cdot \frac{y-y_1}{y_2-y_1}\right]\right) & y_1 < y < y_2 \\ l_2 \cdot cos\left[\frac{\pi}{2} \cdot \frac{y-y_2}{y_3-y_2}\right] & y_2 < y < y_3 \end{cases} \tag{6}$$

$$d(y) = d_1\left[1 - \left(\frac{y - y_1}{y_3 - y_1}\right)^2\right] \tag{7}$$

Parameter		Front spar	Rear Spar
l_1	$[kN/m]$	15	4.5
l_2	$[kN/m]$	4.5	4.5
d_1	$[kN/m]$	1	-

Table 2: Parameter that define the structure ribs and spars

3 Problem Resolution

In this section it is explained the procedure followed for implementing the wing problem into a functional Matlab code that predict its behaviour under the previously defined load conditions.

3.1 Previous Calculations

First of all it is necessary to introduce the fixed nodes of the system, for the wing case these are the ones at the wing root shown in Figure 1 which correspond to nodal positions **1** and **5**. For the analysis the displacements and rotations over all the degrees of freedom of the fixed nodes has been set to zero.

- **Aerodynamic loads**

Since all the wing geometric parameters have already been provided in a data file the aerodynamic loads can be computed using the expressions shown in Section 2.3. It is important to consider that the Lift and Drag obtained from Equations 6 - 7 gives a result in $[N/m]$. Then if one wants to compute the overall lift and drag produced by the wing it is necessary to multiply the obtained values by its element length and then add each one of them.

$$L_{total} = \sum_{e=1}^{N} L_e \cdot l_e \tag{8}$$

$$D_{total} = \sum_{e=1}^{N} D_e \cdot l_e \tag{9}$$

Where,

$$l_e = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_1 - z_2)^2} \tag{10}$$

Subindex 1 and 2 accounts for the nodes that conforms the element. Adding the element lift and drag by its length for all the structure elements does not mean that all the elements have these forces acting on them, most of them would have zero value.

- **Volume and Mass**

Once computed the aerodynamic loads of the problem, it is required to calculate the overall structure mass and volume. Using the nodal positions, nodal connectivity and the material type matrices it is easily computed the element length. Then using the geometric parameters shown in Table 1 and beam height parameter provided in the '*dat*' matrix the area of the beam can be estimated as:

$$A_e = (h - t) \cdot a + 2 \cdot b \cdot t \tag{11}$$

With the density and area of each element the mass can be easily obtained by

$$M_e = A_e \cdot \rho_e \cdot l_e \tag{12}$$

Finally the total mass of the ribs, spars and the total structure is obtained.

- **Beam rotations and geometry**

For solving the problem related to the wing analysis it is needed to rotate from a global coordinate system to the beam local coordinate system. For that it is necessary to use the rotation angles $\{\alpha, \beta, \gamma\}$ shown in Figure 3. These angles are provided for each beam element in the input data file. Finally the rotation of the beam element nodal coordinates is accomplished by using the following expression:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{Bmatrix} cos\beta cos\gamma & cos\beta sin\gamma & sin\beta \\ -sin\alpha sin\beta cos\gamma - cos\alpha sin\gamma & -sin\alpha sin\beta sin\gamma + cos\alpha cos\gamma & sin\alpha cos\beta \\ -cos\alpha sin\beta cos\gamma + sin\alpha sin\gamma & -cos\alpha sin\beta sin\gamma - sin\alpha cos\gamma & cos\alpha cos\beta \end{Bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{13}$$

Once defined the rotation matrix used for solving the problem it is explained how to compute the inertia and torsional constant of each section. Figure 6 shows how the beam has been divided into three different sections for computing the aforementioned parameters.

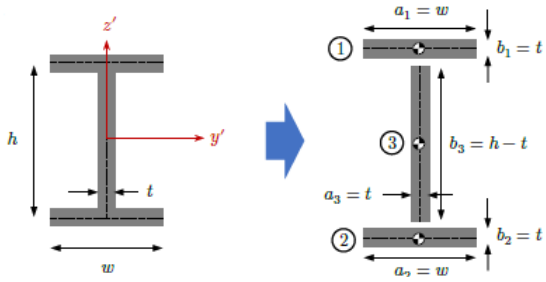

Figure 6: Beam subdivision for geometric parameters compute

Inertia and torsional constant for each element are computed using the following expressions:

$$I_y = \frac{1}{12} \cdot a \cdot (h - t)^3 + \frac{1}{6} \cdot b \cdot t^3 + b \cdot t \cdot \frac{h^2}{2} \tag{14}$$

$$I_z = \frac{1}{12} \cdot (h - t)^3 \cdot a^3 + \frac{1}{6} \cdot t \cdot b^3 + b \cdot t \cdot \frac{h^2}{2} \tag{15}$$

$$J = \frac{1}{3} \cdot \left(2b \cdot t^3 + h \cdot a^3\right) \tag{16}$$

Since these parameters need to be computed in different sections of the code it has been implemented a function called *BeamParameters* that compute the area, inertia and torsional constant of the section using the element index and its geometrical parameters as an input.

3.2 Solving Algorithm

Once it is presented the initial problem analysis it should be developed an algorithm capable predicting the behaviour of our structure. For that it has been followed a methodology shown in the project guides that it could be summarised in some steps.

1. **Data Input**

   The algorithm reads all the data needed for generating the problem structure and solution. It also contains all the parameters related to the beam element, such as its section, density and mechanical properties.

   (a) Nodal positions
   (b) Nodal connectivities for element definition
   (c) Material matrix for element material definition
   (d) Data table of element geometric information

2. **Pre-Process**

   These function is in charge of the previous calculations required for solving the overall structural system. The parameters and matrices defined in the pre-process function are:

   - Material matrix with containing all the information depending of each material type.
   - Boundary conditions - Fixes nodes
   - Connectivity matrix which contains the elements degrees of freedom in terms of the global system.
   - Aerodynamic loads for each element

   These part of the program would also be in charge of computing parameters such as the total structural mass, its volume, element length and the volume and mass for spars and ribs.

3. **Solver**

   The solver would be the one in charge of solving the structure behaviour. The solver follows the procedure shown below:

   (a) Global Stiffness Matrix
   (b) Force vector assembly
   (c) Global system of equations resolution
   (d) Tension Torsion and Moment extraction

4. **Post-Process**

   These function processes all the results and data generated during the algorithm for showing the desired results. Divided into a plotting and data displaying section.

4 Problem Results

Once the solving methodology has been moreover explained it is time to show the obtained results. Since the wing has been simulated using assumptions that try to bring realism into the problem the results are expected to keep a moreover realistic shape.

4.1 Nominal Parameters

During the analysis it has been obtained some structural nominal parameters such as the structural mass and the total aerodynamic loads value. These values help to have a clearer vision on how big is the structure modelled and which are the values of the forces acting over the overall structure. In order to give a better presentation of these results they are displayed into Table 3.

Parameter	Magnitude
Wing Mass m_w $[kg]$	2461.9019
Structure Mass m_t $[kg]$	911.9019
Ribs Mass m_r $[kg]$	278.1860
Spars Mass m_s $[kg]$	633.7159
Total Lift L_t $[N]$	175467.6262
Total Drag D_t $[N]$	11302.9777

Table 3: Wing Nominal Parameters

From these results it is clearly seen that the wing would be suffering from a lift force greater than its weight. These force is going to deform the structure pulling the wing upsides from its tip or at least this should be the expected results.

4.2 Reactions

One important result extracted from the solving algorithm are the reactions at the wing root. These point in any aircraft need to be carefully analysed since it is expected to transmit the wing loads to the fuselage. The structure that holds the wing to the fuselage and the one that supports the motor to the wing are the most critical ones and its design must consider that fact. Table 4 shows the reaction results obtained for the nodes that connect the wing to the fuselage, these are nodes **1** and **5**.

Parameter	Node 1	Node 2
Axial force N $[N]$	-3882.0858	3882.0858
Shear force - Y Q_y $[N]$	-7979.2278	7979.2278
Shear force - Z Q_z $[N]$	-102094.3416	-48750.1846
Torsion moment T $[N \cdot m]$	-456767.6305 9	-348515.6991
Bending moment M_y $[N \cdot m]$	244758.6180	126789.8739
Bending moment M_z $[N \cdot m]$	-2779.3424	-3056.0922

Table 4: Wing Nominal Parameters

The results show that the torsion moment for these nodes has a very large value. These confirms the idea that the structure that supports the wing to the fuselage has to be very rough. Due to the wing torsion these nodes would also suffer from a great shear force in its Z direction. It is left for further analysis how the wing root should be designed according to the obtained results.

4.3 Deformed Wing

The solver gives the solution of the structure reactions and the displacements and rotations at each node. Using these results one can represent how the wing is deformed. As it has been said, the lift force has a greater value than the overall wing weight so it is expected that the wing bends upwards.

The problem asks to represent the wing deformation using two different plotting functions called '*plotWing*' and '*plotBeams3D_def*'. The first one uses the nodal positions and deformations in global coordinates for representing the structure. The other uses the deformations in is local reference frame for the representation. Both of the representations are expected to have the same shape.

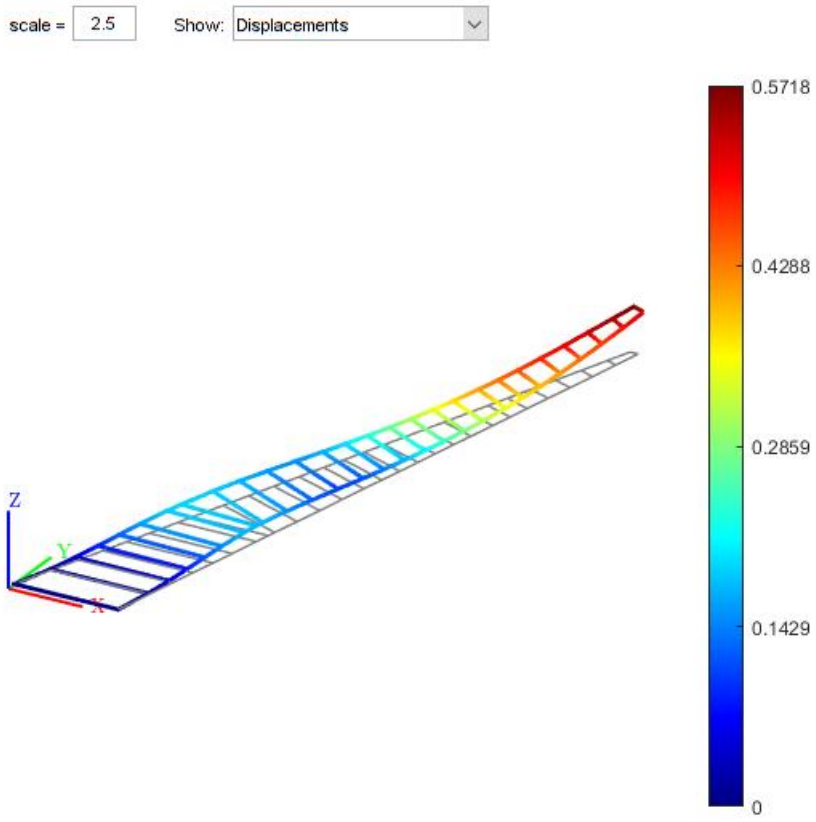

Figure 7: Deformed wing using '*plotWing*' plot

Figure 7 and 8 show that the wing has the same shape for the different plotting options as it was expected. Both of the representations have been done using a scaling factor of 2.5 over the displacements and rotation values. Both figures show the deformed structure coloured and the structure in its initial conditions in light grey. Figure 7 shows a colorbar with zero displacement value for blue tones and the maximum displacement in red tones. As it is expected, the zero displacement zone is near the wing root and the highest displacement values are at the wing tip. It is appreciated an increase in the displacements near the pylon attachment due to the thrust and motor weight force effects over the structure. After these zone the displacements reduce taking tones near to the dark blue for finally increasing until reaching its maximum value.

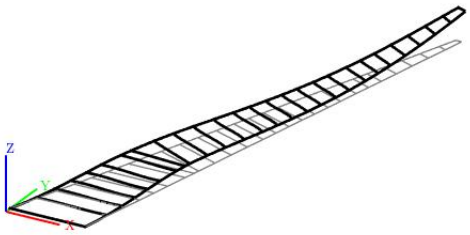

Figure 8: Deformed wing using '*plotBeams3D_def*' plot

4.4 Force and Moments Analysis

Once analysed how the structure is deformed it is time to study how the force and moments evolve over the wing structure when it is under the already specified loading conditions. In this section it is analysed the axial and shear forces and the torsional and bending moments of each element of the structure.

- **Axial Force**

These force is the one acting in the X direction of the beam. Its sign indicates weather it is a compressing or tractor force. Negative values indicate that the beam is under compressing load conditions. The following figure shows the axial force distribution.

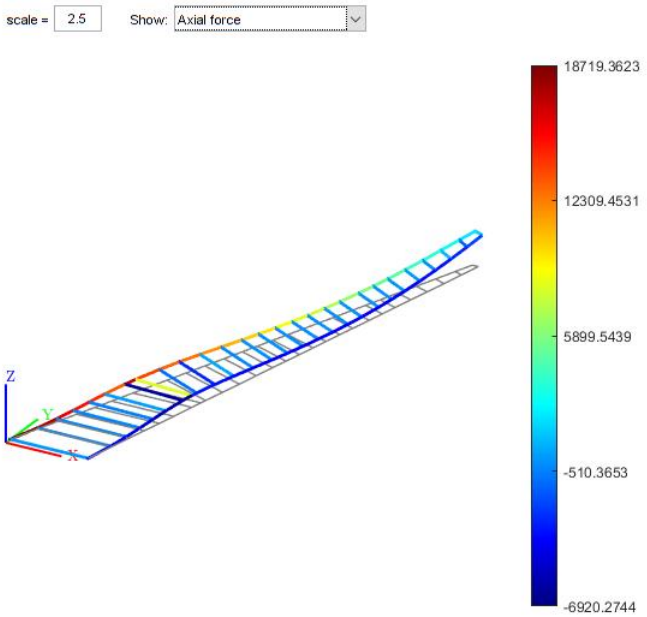

Figure 9: Axial Force distribution over the wing structure

Figure 9 shows in red tones the zones of maximum traction and in blue tones the maximum compression zones. The scale of the plot goes from maximum traction values of $18719.2623N$ to minimum compression values of $-6920.2744N$. The plot shows that the maximum axial force is near the zone where the motor is placed because of the influence of the motor thrust and weight. The motor thrust acts like a pulling force that acts against a drag force that has the same value but with the difference that the drag is divided into multiple elements along the rear spar. These arrangement of the forces produces that concentration of the axial force near the pylon attachment. The front spar of the wing is acting over traction forces, or positive axial force value, and the rear spar is acting over compression.

As a final comment about the axial force, it is seen that its value increases at the ribs attached tot he motor pylon, These increase is due to the fact that the trusts and weight is distributed over these ribs, making its axial force value increase considerable against the other ribs. All the ribs are acting over compressing forces except one.

- **Shear Force in the local y' direction**

The shear force in the local y' direction is another force that would be largely affected by the influence of the motor thrust. Figure 10 shows that its maximum values are located at the elements near the pylon attachment. Dark red tones indicate the maximum positive values of shear and the blue ones indicate its minimum values. The colorbar is comprehended between values of $5479,97$ and $-4939,1835$ N which are moreover the same value in absolute value terms.

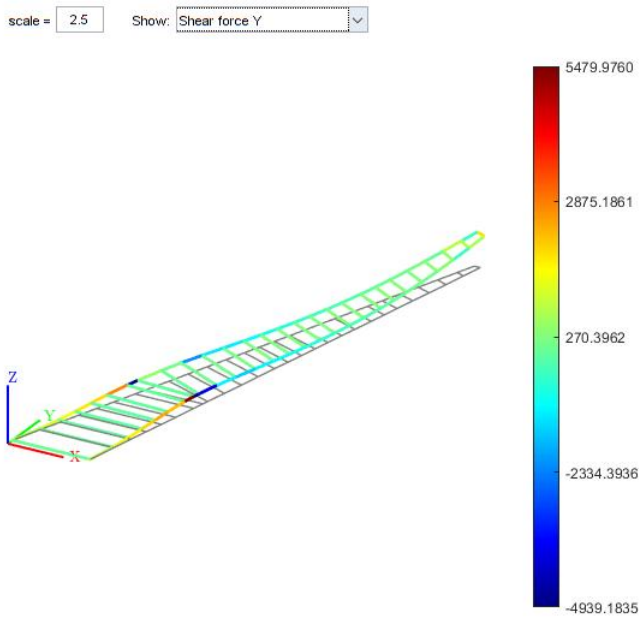

Figure 10: Shear Force (y-direction) distribution over the wing structure

In Figure 10 it has been already seen that the shear is concentrated around the motor zone of the wing. The spars have a greater vale than the ribs, that seem to present a a constant shear value except for the wing tip zone, where the shear force increases.

- **Shear Force in the local z' direction**

In the wing structure the shear force effect in the local z' direction is produced by the loads that affect our structure. In Figure 11 one can identify that the maximum shear is near the wing root zone. The maximum is there because how the bending moment affect to these zone of the structure. Apart from that it is observed that the zone near the pylon attachment also suffers from considerable shear force but they magnitude is moreover half of the maximum shear value.

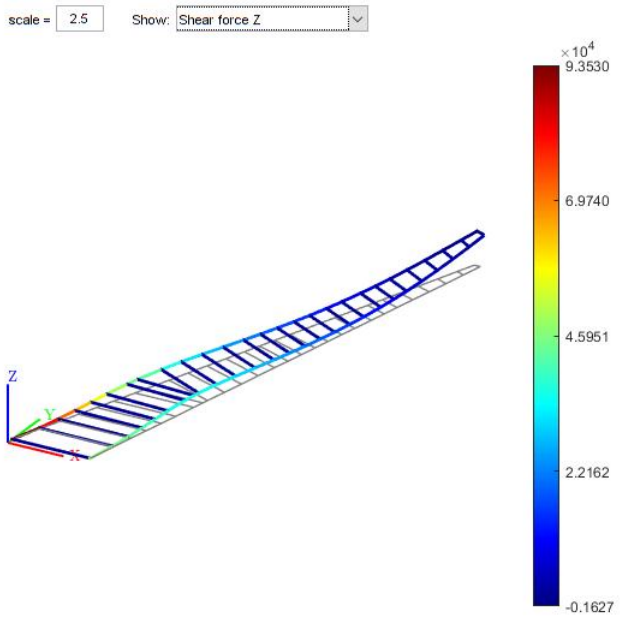

Figure 11: Shear Force (z-direction) distribution over the wing structure

The shear force scale shown in Figure 11 goes from negative vales approximately sixty times lower than the maximum positive values. In red it is coloured the maximum positive shear z and in blue tones it can be observed the negative values. All the ribs present negative shear force value. For the spars it can be observed that the shear force decreases gradually from the wing root until it reaches negative values at the wing tip. In between these transition it is expected to find a point of null shear force value, near the wing tip. Comparing with the shear in the y direction one can quickly identify that the magnitude of the force over the z direction is approximately twenty times greater for its maximum value. These means that it must be cautiously considered the behaviour study of the zones suffering from maximum shear stress force on the z direction.

- **Torsional moment**

The torsional moment representation shows how the wing behaves when it suffer from moments around the X axis, when the beam twist over itself. In the following figure it is shown the representation of the torsional moment over the whole wing structure. Since the plot represents the local direction, these means that these beams would suffer from a moment that would try to twist them over its central axis. Figure 12 show that the maximum torsional moment is near the wing tip zone. Also, it is easily identified that the spars are the elements that are mostly suffering these moment and the ribs have a moreover constant negative torsion value.

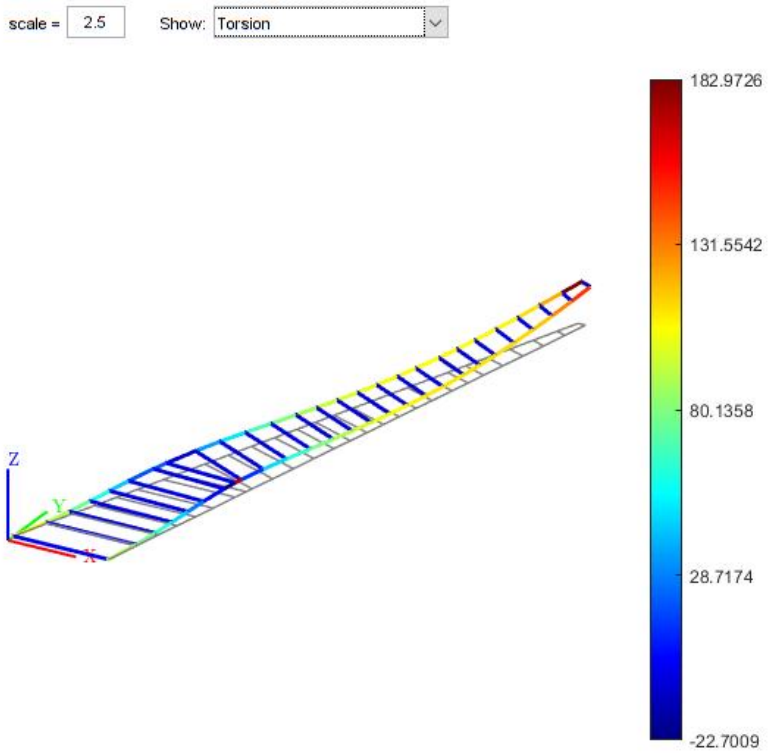

Figure 12: Torsional moment distribution over the wing structure

Figure 12 shows that the maximum torsion value in the scale is around 182.976 $N \cdot m$ and the maximum negative one is around -22.7009 $N \cdot m$ which is approximately 8 times lower. It observed that the motor weight produces a reducing effect in the torsion moment near the pylon attachment zone. After that zone the moment increases until it reaches its maximum value at the wing tip. It is also identified an element near the pylon attachment that presents considerable torsion moments. These element should be carefully considered because it also has great axial and shear-y forces that could affect to the structure performance. That means that these element would be a critical zone of the structure and its reinforcement would improve the structure safety. The maximum torsional moment could be directly related to the rotation of the elements over its local X direction since the maximum rotations are located at the wing tip.

- **Bending moment in the local y' direction**

As it was expected from the previous reaction analysis at the wing root, the bending moment has its maximum value near these zone. The forces are distributed in a way that produce maximum bending efforts at the spars near the wing root. Figure 13 shows these behaviour, in blue it can be identified the maximum negative values of bending, which are almost three hundred times greater than the maximum positive ones. The bending moment in the local y' direction has its maximum values at the wing root spars, then the value reduces along the spars until it reaches positive values near the wing tip. The motor has a reducing effect in the bending moment magnitude, making the transition to lower bend moments faster, In between these transition it is expected a zero bending moment zone.

The zero moment and force zone has also been identified at the same location in the previous results comments. This means that the elements suffering from minimum forces and reactions could be redesigned for reducing the overall structure weight. When reducing the weight one has to keep in mind that the structure should accomplish the resistance requirements. Finally it needs to be commented that for the ribs the bending moment is moreover constant.

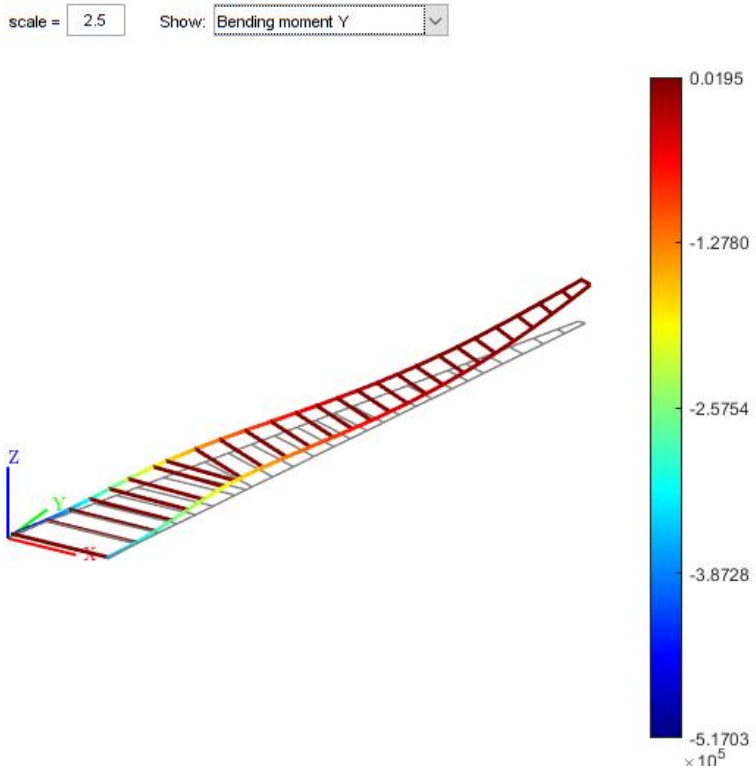

Figure 13: Bending moment (y-direction) distribution over the wing structure

- **Bending moment in the local z' direction**

For the bending moment in the local z' direction it would also be commented how it is distributed along the wing structure and also how the magnitude of its value changes comparing to the moment in the local y' direction.

Figure 14 shows how the drag and thrust forces affect to the wing structure, increasing the bending moment values at the wing root and the pylon attachment zone. In blue it is identified the maximum negative bending values and in red the maximum positive values. Since the drag acts in an opposite direction to the trust, the bend is also expected to have opposite values, Additionally, it is also known that the drag is distributed along the rear spars, and the thrust is concentrated in the pylon attachment ribs, that fact explain why the maximum bending moment is located near the pylon attachment zone.

The ribs of the structure suffer from a moreover constant bending moment in the local z' direction. For the spars, the bending moment gradually reduces from the wing root until it reach the maximum negative value near the motor zone. After that the spars reduces its bending value until they reach

positive values. The minimum positive values of bending are located at the wing tip and it could also be identified the zero bending zone near the motor, where the values goes from negative to positive.

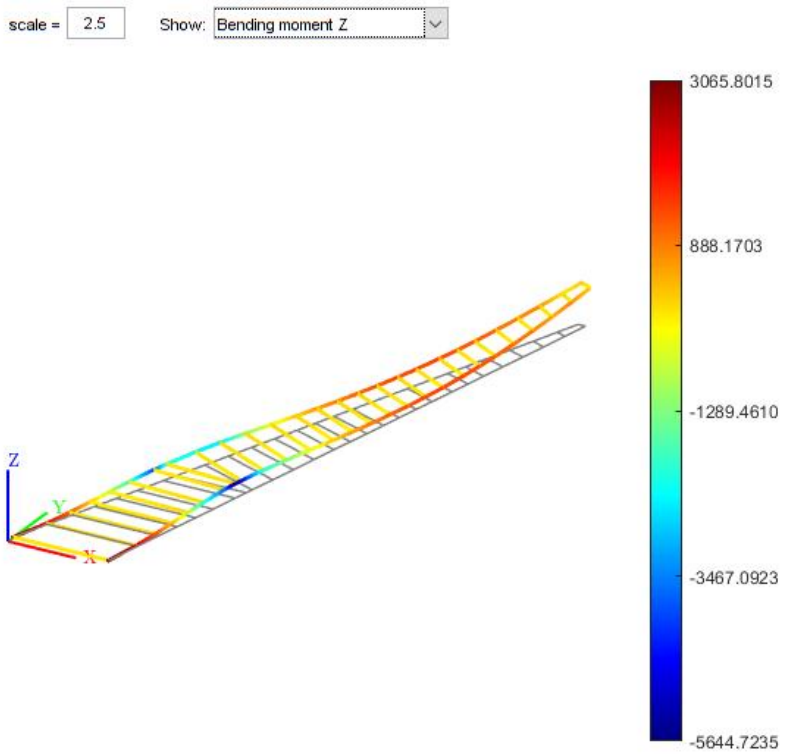

Figure 14: Bending moment (z-direction) distribution over the wing structure

4.5 Displacement and Rotations

In the Post-Processing function it has been implemented an algorithm that computes the beams suffering from maximum axial force, shear and bending moment over all its directions. The algorithm computes the maximum in absolute value since the negative values affect to the structure the same way it would affect a positive one. The only consideration is that negative values would act in an opposite direction from the positive ones.

From the result analysis shown in Section 4.4 it has been identified two critical points in the wing. These points are the wing root joint with the fuselage and the spar located behind the pylon attachment. It has also been noted that there is a zone of minimum forces and moment in between the motor and the wing tip.

Comparing Figure 10 and 11 scales it is easily identified that the maximum shear is the one acting in the local z-direction of the beam element. Figure 11 shows that the maximum shear is near the wing root front spar elements. Then comparing Figure 13 and 14 it is seen that the scale of the bending moment over the y-direction is greater. The explanation of the greater moment in y-direction is due to the fact that the lift force is greater than the drag, then it would produce greater bend moments in the y-direction. From Figure 13 it is seen that the maximum bending

over the y-direction has negative values and it is located at the front wing root spar. Finally, for the axial force Figure 9 shows that the maximum values are positive and that are also located at the front spar near the wing root.

The visual analysis matches with the results obtained with the developed algorithm. The algorithm that look for the maximum values in the saved tension and moment matrices also indicates that the maximum values are found in the first front spar element, which correspond with:

$$Element = \mathbf{28} \tag{17}$$

This element is composed by nodes **1** and **2**, just as shown in Figure 1. The maximum shear and bend moment directions also corresponds to the ones aforementioned in the visual analysis. The following Figure shows the deflection and rotation of the element **28** in the y and z direction.

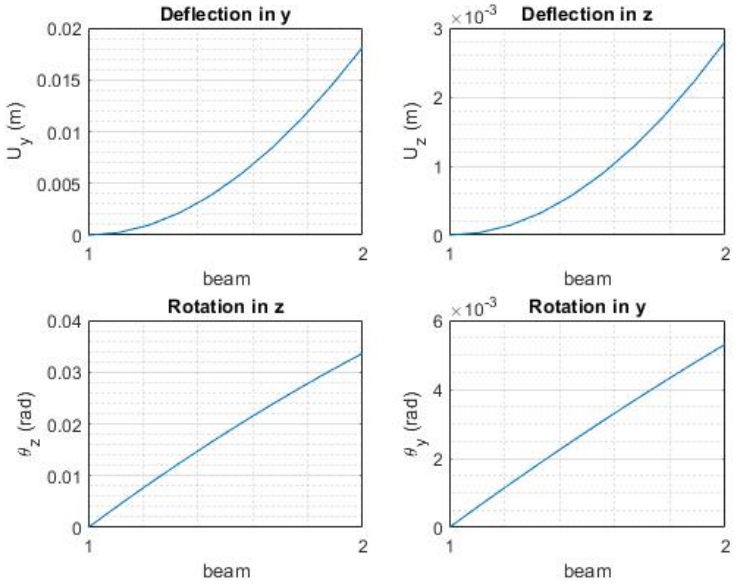

Figure 15: Deflection and rotation for most critical beam in y and z direction

As it is expected, Figure 15 shows that the displacement and rotations born from the prescribed zero value in nodal position **1**, which correspond to the restricted node attached to the fuselage. Then the plots show the evolution of def-lection and rotation for the x ans y direction from node **1** to node **2**. The maximum rotations are expected to be in the Z axis, just as shown in the bottom left plot, these rotations are due to the effect of the drag forces acting along the wing. Since the lift forces are moreover balanced along the front and rear spars, the rotation in y direction keep very low values. Finally, it is observed a very low deflection in the z direction comparing to the deflection suffered in the y direction.

5 Conclusions

From the previous result analysis it can be extracted the conclusion that the elements near the pylon attachment need to be rough enough to hold the loads that the motor transmit to this zone. It is left for further studies the analysis on how these beams should be designed and how would these affect to the structure performance.

One example of the possible analysis to perform with the developed algorithm would be to analyse how the structure performance increase when redesigning the critical elements. Its redesign would consist in improving the material resistance or making the beam more robust. It is also interesting to analyse other possible beam geometries and compare the results with the obtained ones.

As a final comment it should be said that this project helped a lot to visualise how the wing structure deforms under conditions that keep similarity with a real case where the plane is in flight. It has been really useful to analyse all the effects that the motors induce to the wing behaviour and it would be interesting to study how would the wing behave without having the motor load in the wing, maybe studying a light aircraft.

Furthermore the analysis of how each force and moment acts along the structure elements gave a better understanding on how they are divided and which values take comparing to the applied loads.

A Structural Analysis Code

In this Appendix we can see the code developed with Matlab for solving the structural analysis of a wing using the FEM. This results has been obtained using a self developed code[1]

A.1 Main Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%    FINITE ELEMENT METHOD     %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%     STRUCTURAL ANALYSIS      %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%  Miquel Altadill Llasat  %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Code developed for the structural analysis of a commercial
% aircraft wing.

clc
clear all
close all

%% ============================ INPUT DATA ================================ %%
%InputData
% - Inside each function (Cleaner Workspace)

%% ============================  PREPROCESS ============================  %%
[T2,Fnod, mat,Lift,Drag] = Preprocess();
[El_L,El_M,M_spar,M_rib, V_spar, V_rib,rho_eff,M_struct,V_tot] = ...
    StructureMass(mat);

%% ============================= SOLVER ================================  %%
[Kel,KG,R] = StiffnessMatrix(El_L,T2,mat);
[Fext,Drag_T,Lift_T] = ForceAssembly(El_L,rho_eff,Lift,Drag,R,mat,T2);
[U,F_int,u_int,N_x,Q_y,Q_z,T_x,M_y,M_z,rf,Rr] = solver(Fnod,KG,Kel,R,T2,Fext);


%% ============================ POSTPROCESS ============================  %%
PostProcess(M_struct,M_rib,M_spar,V_rib,V_spar,Lift_T,Drag_T,V_tot,U,u_int,...
    F_int,El_L,N_x,Q_y,Q_z,T_x,M_y,M_z,R,Rr);
```

A.2 Input Data

```

%% Phisical Properties

% Ribs parameters
E_r = 62.5e9;        % Young's Modulus [Pa]
v_r = 0.33;          % Poisson
G_r = E_r / (2*(1+v_r)); % G coef [Pa]
rho_r = 2050;        % Density [kg/m^3]
a_r = 5e-3;          % Beam geom [m]
b_r = 15e-3;         % Beam geom [m]
t_r = 2.5e-3;        % Beam geom [m]
```

[1]This code can be downloaded with a document that describes the case of study by clicking "here".

```
12
13
14 % Spars Parameters
15 E_s = 70.6e9;       % Young's Modulus [Pa]
16 v_s = 0.36;         % Poison
17 G_s = E_s / (2*(1+v_s)); % G coef [Pa]
18 rho_s = 2450;       % Density [kg/m^3]
19 a_s = 12e-3;        % Beam geom [m]
20 b_s = 97e-3;        % Beam geom [m]
21 t_s = 6e-3;         % Beam geom [m]
22
23 % Gravity and mass
24 M_w= 1550;          % Wing mass [kg]
25 M_eng = 960;       % Engine mass [kg]
26 g = 9.81;           % Gravity acc [m/s^2]
27
28
29
30
31
32
33 %% Wing Parameters
34 % Nodal coordinates (m)
35 % x coordinate:  x(:,1)
36 % y coordinate:  x(:,2)
37 % z coordinate:  x(:,3)
38 x = [
39   15.90000000,   1.85347500,  -1.26306070
40   16.37993300,   2.74947500,  -1.15390270
41   16.72863500,   3.40047500,  -1.07459260
42   17.04466200,   3.99047500,  -1.00271410
43   18.69000000,   1.85347500,  -1.26306070
44   17.34997700,   4.56047500,  -0.93327205
45   19.01494100,   2.74947500,  -1.15390270
46   19.25103100,   3.40047500,  -1.07459260
47   17.64404300,   5.10947500,  -0.86638842
48   19.46499900,   3.99047500,  -1.00271410
49   18.01470600,   5.80147500,  -0.78208338
50   19.67171300,   4.56047500,  -0.93327205
51   18.20003700,   6.14747500,  -0.73993086
52   19.87081200,   5.10947500,  -0.86638842
53   18.60033300,   6.89479850,  -0.64888586
54   20.12177100,   5.80147500,  -0.78208338
55   20.24725100,   6.14747500,  -0.73993086
56   20.31191500,   6.32578240,  -0.71820801
57   18.92707400,   7.50479850,  -0.57457073
58   20.54448900,   6.96708800,  -0.64007898
59   19.25435000,   8.11579850,  -0.50013376
60   20.77744500,   7.60944480,  -0.56182186
61   19.58162500,   8.72679850,  -0.42569679
62   21.01040000,   8.25180170,  -0.48356475
63   19.89765300,   9.31679850,  -0.35381822
64   21.23534900,   8.87208090,  -0.40799732
65   20.15904500,   9.80479850,  -0.29436611
66   21.42140800,   9.38512530,  -0.34549409
67   20.42097300,  10.29379900,  -0.23479217
68   21.60784900,   9.89922110,  -0.28286279
69   20.68236500,  10.78179900,  -0.17534006
70   21.79390800,  10.41226600,  -0.22035956
```

```
   20.93304500,  11.24979900,  -0.11832451
   21.97234200,  10.90428400,  -0.16041794
   21.20568500,  11.75879900,  -0.05631401
   22.16640800,  11.43940600,  -0.09522502
   21.45636500,  12.22679900,   0.00070154
   22.34484200,  11.93142400,  -0.03528340
   21.77239200,  12.81679900,   0.07258011
   22.56979100,  12.55170300,   0.04028402
   22.02360700,  13.28579900,   0.12971749
   22.74860600,  13.04477200,   0.10035372
   22.27439000,  13.75399200,   0.18675664
   22.92711400,  13.53699400,   0.16032015
   22.57906600,  14.32279900,   0.25605322
   23.14398200,  14.13499200,   0.23317308
   22.89562900,  14.91379900,   0.32805363
   23.36931200,  14.75632200,   0.30886859
   23.22236900,  15.52379900,   0.40236876
   23.60188600,  15.39762800,   0.38699762
   23.56329800,  16.16028900,   0.47991118
   23.84456000,  16.06678300,   0.46851956
   23.85800000,  16.71047500,   0.54693930
   24.07800000,  16.71047500,   0.54693930
]';

% Nodal connectivities
%  Ribs: rows 1 to 27
%  Front spar: rows 28 to 53
%  Rear spar: rows 54 to 79
Tnod = [
                1              5
                2              7
                3              8
                4             10
                6             12
                9             14
               11             16
               13             17
               15             18
               19             20
               21             22
               23             24
               25             26
               27             28
               29             30
               31             32
               33             34
               35             36
               37             38
               39             40
               41             42
               43             44
               45             46
               47             48
               49             50
               51             52
               53             54
                1              2
                2              3
                3              4
```

```
                4                6
                6                9
                9               11
               11               13
               13               15
               15               19
               19               21
               21               23
               23               25
               25               27
               27               29
               29               31
               31               33
               33               35
               35               37
               37               39
               39               41
               41               43
               43               45
               45               47
               47               49
               49               51
               51               53
                5                7
                7                8
                8               10
               10               12
               12               14
               14               16
               16               17
               17               18
               18               20
               20               22
               22               24
               24               26
               26               28
               28               30
               30               32
               32               34
               34               36
               36               38
               38               40
               40               42
               42               44
               44               46
               46               48
               48               50
               50               52
               52               54
];

% Material connectivities
%  Ribs: rows 1 to 27
%  Front spar: rows 28 to 53
%  Rear spar: rows 54 to 79
Tmat = [
                2
                2
                2
```

189	2
190	2
191	2
192	2
193	2
194	2
195	2
196	2
197	2
198	2
199	2
200	2
201	2
202	2
203	2
204	2
205	2
206	2
207	2
208	2
209	2
210	2
211	2
212	2
213	1
214	1
215	1
216	1
217	1
218	1
219	1
220	1
221	1
222	1
223	1
224	1
225	1
226	1
227	1
228	1
229	1
230	1
231	1
232	1
233	1
234	1
235	1
236	1
237	1
238	1
239	1
240	1
241	1
242	1
243	1
244	1
245	1
246	1
247	1

```
                1
                1
                1
                1
                1
                1
                1
                1
                1
                1
                1
                1
                1
                1
                1
];

% Geometrical parameters
%  I-section height: dat(1,:) (m)
%  alpha angle:      dat(2,:) (rad)
%  beta angle:       dat(3,:) (rad)
%  gamma angle:      dat(4,:) (rad)
dat = [
   1.03000,   0.00000,   0.00000,   0.00000
   0.97391,   0.00000,   0.00000,   0.00000
   0.93316,   0.00000,   0.00000,   0.00000
   0.89623,   0.00000,   0.00000,   0.00000
   0.86055,   0.00000,   0.00000,   0.00000
   0.82618,   0.00000,   0.00000,   0.00000
   0.78287,   0.00000,   0.00000,   0.00000
   0.76121,   0.00000,   0.00000,   0.00000
   0.73224,   0.00000,  -0.03841,  -0.32096
   0.69307,   0.00000,  -0.03841,  -0.32096
   0.65385,   0.00000,  -0.03841,  -0.32096
   0.61462,   0.00000,  -0.03841,  -0.32096
   0.57674,   0.00000,  -0.03841,  -0.32096
   0.54541,   0.00000,  -0.03841,  -0.32096
   0.51401,   0.00000,  -0.03841,  -0.32096
   0.48268,   0.00000,  -0.03841,  -0.32096
   0.45263,   0.00000,  -0.03841,  -0.32096
   0.41996,   0.00000,  -0.03841,  -0.32096
   0.38991,   0.00000,  -0.03841,  -0.32096
   0.35203,   0.00000,  -0.03841,  -0.32096
   0.32192,   0.00000,  -0.03841,  -0.32096
   0.29186,   0.00000,  -0.03841,  -0.32096
   0.25534,   0.00000,  -0.03841,  -0.32096
   0.21739,   0.00000,  -0.03841,  -0.32096
   0.17823,   0.00000,  -0.03841,  -0.32096
   0.13737,   0.00000,  -0.03841,  -0.32096
   0.10000,   0.00000,   0.00000,   0.00000
   1.00196,   0.00000,   0.10698,   1.07905
   0.95354,   0.00000,   0.10698,   1.07904
   0.91470,   0.00000,   0.10698,   1.07905
   0.87839,   0.00000,   0.10698,   1.07904
   0.84337,   0.00000,   0.10698,   1.07905
   0.80453,   0.00000,   0.10698,   1.07904
   0.77204,   0.00000,   0.10698,   1.07905
```

```
    0.73782,   0.00000,   0.10698,   1.07905
    0.69534,   0.00000,   0.10698,   1.07904
    0.65712,   0.00000,   0.10698,   1.07904
    0.61888,   0.00000,   0.10698,   1.07905
    0.58129,   0.00000,   0.10698,   1.07904
    0.54755,   0.00000,   0.10698,   1.07905
    0.51697,   0.00000,   0.10698,   1.07905
    0.48639,   0.00000,   0.10698,   1.07905
    0.45647,   0.00000,   0.10698,   1.07904
    0.42589,   0.00000,   0.10698,   1.07905
    0.39531,   0.00000,   0.10698,   1.07904
    0.36220,   0.00000,   0.10698,   1.07905
    0.32905,   0.00000,   0.10698,   1.07905
    0.29972,   0.00000,   0.10698,   1.07904
    0.26726,   0.00000,   0.10698,   1.07904
    0.23096,   0.00000,   0.10698,   1.07905
    0.19337,   0.00000,   0.10698,   1.07905
    0.15436,   0.00000,   0.10698,   1.07905
    0.11722,   0.00000,   0.10698,   1.07904
    1.00196,   0.00000,   0.11403,   1.22289
    0.95354,   0.00000,   0.11403,   1.22289
    0.91470,   0.00000,   0.11403,   1.22289
    0.87839,   0.00000,   0.11403,   1.22289
    0.84337,   0.00000,   0.11403,   1.22289
    0.80453,   0.00000,   0.11403,   1.22289
    0.77204,   0.00000,   0.11403,   1.22289
    0.75563,   0.00000,   0.11403,   1.22289
    0.72998,   0.00000,   0.11403,   1.22289
    0.68980,   0.00000,   0.11403,   1.22289
    0.64959,   0.00000,   0.11403,   1.22289
    0.61007,   0.00000,   0.11403,   1.22289
    0.57460,   0.00000,   0.11403,   1.22289
    0.54245,   0.00000,   0.11403,   1.22289
    0.51030,   0.00000,   0.11403,   1.22289
    0.47885,   0.00000,   0.11403,   1.22289
    0.44670,   0.00000,   0.11403,   1.22289
    0.41455,   0.00000,   0.11403,   1.22289
    0.37974,   0.00000,   0.11403,   1.22289
    0.34489,   0.00000,   0.11403,   1.22289
    0.31406,   0.00000,   0.11403,   1.22289
    0.27993,   0.00000,   0.11403,   1.22289
    0.24177,   0.00000,   0.11403,   1.22289
    0.20225,   0.00000,   0.11403,   1.22289
    0.16124,   0.00000,   0.11403,   1.22289
    0.12015,   0.00000,   0.11403,   1.22289
];
```

A.3 Pre-processing

A.3.1 Connectivity Matrix and Aerodynamic Loads

```
function [T2,Fnod, mat,Lift,Drag] = Preprocess()

%% Input Data

InputData;
```

```
6
7  % Material properties storage
8  mat = [%  E     Area     Density    G     Inertia     a        b      t
9            E_s,    0,     rho_s,    G_s,       0,     a_s,    b_s,    t_s;  % ...
                Spars material  = 1
10           E_r,    0,     rho_r,    G_r,       0,     a_r,    b_r,    t_r   % ...
                Ribs material   = 2
11           ]';
12 % Material:
13 %   - materiall 1 = mat(:, 1)
14 %   - materiall 2 (E_r) = mat( 1, 2 )
15
16
17
18 %% Fixed nodes
19
20 Fnod = [    % Node       DOF     Displacement
21                1,         1,         0;
22                1,         2,         0;
23                1,         3,         0;
24                1,         4,         0;
25                1,         5,         0;
26                1,         6,         0;
27                5,         1,         0;
28                5,         2,         0;
29                5,         3,         0;
30                5,         4,         0;
31                5,         5,         0;
32                5,         6,         0;
33 ]';
34
35 %% Problem dimensions
36
37 Ndim = size(x,1)*2;           % DoF for each node
38 Nnodes = size(x,2);           % Number of nodes
39 Nelements = size(Tnod,2);     % Number of elements
40 NnodesXelement = size(Tnod,1);% Number of nodes per element
41 Ndofs = Ndim*Nnodes;          % Total DoF of the system
42
43
44 %% Connectivity matrix - T2
45
46 T2=zeros(Nelements,Ndim*NnodesXelement);
47 T1=Tnod';
48 p=Ndim-1;
49 for i=1:Nelements
50     for j=1:NnodesXelement
51         for e=0:(Ndim-1)
52         t2(i,(j*Ndim-p+e))=Ndim*T1(i,j)-p+e;
53         end
54     end
55 end
56
57 T2=t2';
58
59
60 %% Aerodynamic load computation
61 % Ribs elements: 1 to 27
62 % Front spars:   28 to 53
```

```
% Rear spars:    54 to 79


Lift = zeros(Nelements,1);
Drag = zeros(Nelements,1);




for e = 1:Nelements

    % Nodal position y
    ye1 = x(2,Tnod(1,e));      % Node 1
    ye2 = x(2,Tnod(2,e));      % Node 2

    y_el = (ye1 + ye2)/2;
    % Front spar
    if e ≥ 28 && e ≤ 53
        y_1=x(2,1);
        y_2=x(2,29);
        y_3=x(2,53);
        % Drag Compute
        d_1 = 1e3;   % [N/m]
        Drag(e) = d_1*(1 - ((y_el-y_1)/(y_3 - y_1))^2);
        % Lift Compute
        l_1 = 15e3;  % [N/m]
        l_2 = 4.5e3; % [N/m]
        if y_el > y_1 && y_el < y_2
            % Formula 1
            Lift(e) = 1/2* (l_1 + l_2 + (l_1-l_2)* cos(pi*((y_el-y_1)/(y_2-y_1))));
        else
            % Formula 2
            Lift(e) = l_2*cos((pi/2)*((y_el-y_2)/(y_3-y_2)));
        end
    end

    % Rear spar
    if  e≥54 && e≤79
        y_1=x(2,5);
        y_2=x(2,32);
        y_3=x(2,54);
        % Lift Compute
        l_1 = 4.5e3;  % [N/m]
        l_2 = 4.5e3; % [N/m]
        if y_el > y_1 && y_el < y_2
            % Formula 1
            Lift(e) = 1/2* (l_1 + l_2 + (l_1-l_2)* cos(pi*((y_el-y_1)/(y_2-y_1))));
        else
            % Formula 2
            Lift(e) = l_2*cos((pi/2)*((y_el-y_2)/(y_3-y_2)));
        end
    end


end

end
```

A.3.2 Structure Mass

```
function [El_L,El_M,M_spar,M_rib, V_spar, V_rib,rho_eff,M_struct,V_tot] = ...
    StructureMass(mat)

InputData;


%% Problem dimensions

Ndim = size(x,1)*2;           % DoF for each node
Nnodes = size(x,2);           % Number of nodes
Nelements = size(Tnod,2);     % Number of elements
NnodesXelement = size(Tnod,1);% Number of nodes per element
Ndofs = Ndim*Nnodes;          % Total DoF of the system

%% Structure Computations

El_L = zeros (Nelements,1);     %Save Element Length
El_M = zeros (Nelements,1);     %Save Element Mass


M_spar = 0;
M_rib  = 0;
V_rib  = 0;
V_spar = 0;
V_tot  = 0;


for e = 1:Nelements
    x1=x(1,Tnod(1,e));
    y1=x(2,Tnod(1,e));
    z1=x(3,Tnod(1,e));
    x2=x(1,Tnod(2,e));
    y2=x(2,Tnod(2,e));
    z2=x(3,Tnod(2,e));

    El_L(e) = sqrt((x1-x2)^2 + (y1-y2)^2 + (z1-z2)^2);

    [Ae] = BeamParameters(mat,dat,Tmat, e);

    El_V  = Ae*El_L(e);
    rho_e = mat(3,Tmat(e));
    El_M(e) =El_V*rho_e;

    if Tmat(e)=
       M_spar = El_M(e) + M_spar;
       V_spar = El_V + V_spar;
    end
    if Tmat(e)== 2 % Rib
       M_rib = El_M(e) + M_rib;
       V_rib = El_V + V_rib;
    end

    V_tot = El_V + V_tot;

end
```

```
55
56 rho_hat = (M_w - M_spar-M_rib)/ (V_rib+V_spar);
57 rho_eff(1)  =  mat(3,1) + rho_hat; % Spar
58 rho_eff(2)  =  mat(3,2) + rho_hat; % Rib
59
60 M_struct = sum(El_M);
61 % Mtot  = V_rib*rho_eff_rib + V_spar*rho_eff_spar + M_w;
62 % Mtot2 = M_spar + M_rib + M_w;
63 %disp('Total Mass 1'); disp(Mtot);
64 %disp('Total Mass 2'); disp(Mtot2);
65
66
67
68 end
```

A.4 Solver

A.4.1 Stiffness Matrix

```
1  function [Kel,KG,R] = StiffnessMatrix(El_L,T2,mat)
2
3  InputData;
4
5
6  %% Problem Dimensions
7
8  Ndim = size(x,1)*2;           % DoF for each node
9  Nnodes = size(x,2);           % Number of nodes
10 Nelements = size(Tnod,2);     % Number of elements
11 NnodesXelement = size(Tnod,1);% Number of nodes per element
12 Ndofs = Ndim*Nnodes;          % Total DoF of the system
13
14
15 %% Element Stiffness matrices
16
17 Kel = zeros(NnodesXelement*Ndim, NnodesXelement*Ndim, Nelements);
18 R = zeros(NnodesXelement*Ndim, NnodesXelement*Ndim, Nelements);
19 for e = 1:Nelements
20
21     % ELEMENT ROTATION MATRIX
22
23     alp=dat(2,e);   % Alpha
24     bet=dat(3,e);   % Beta
25     gam=dat(4,e);   % Gamma
26
27     R(:,:,e)=[cos(bet)*cos(gam),                                    ...
           cos(bet)*sin(gam),                          sin(bet), 0,  0,  0,  0,  ...
           0,  0,  0,  0,  0;
28         -(sin(alp)*sin(bet)*cos(gam))-(cos(alp)*sin(gam)),   ...
              -(sin(alp)*sin(bet)*sin(gam))+(cos(alp)*cos(gam)),  ...
              sin(alp)*cos(bet), 0,  0,  0,  0,  0,  0,  0,  0;
29         -(cos(alp)*sin(bet)*cos(gam))+(sin(alp)*sin(gam)),   ...
              -(cos(alp)*sin(bet)*sin(gam))-(sin(alp)*cos(gam)),  ...
              cos(alp)*cos(bet),0,  0,  0,  0,  0,  0,  0,  0,  0;
30         0,  0,  0,cos(bet)*cos(gam),                                ...
              cos(bet)*sin(gam),                          sin(bet),  0,  0,  0,  ...
```

```
            0,  0,  0,;
31      0,  0,  0,-(sin(alp)*sin(bet)*cos(gam))-(cos(alp)*sin(gam)),   ...
            -(sin(alp)*sin(bet)*sin(gam))+(cos(alp)*cos(gam)),  ...
            sin(alp)*cos(bet),  0,  0,  0,  0,  0,  0,;
32      0,  0,  0,-(cos(alp)*sin(bet)*cos(gam))+(sin(alp)*sin(gam)),   ...
            -(cos(alp)*sin(bet)*sin(gam))-(sin(alp)*cos(gam)),  ...
            cos(alp)*cos(bet),  0,  0,  0,  0,  0,  0,;
33      0,  0,  0,  0,  0,  0,   cos(bet)*cos(gam),                                 ...
            cos(bet)*sin(gam),                          sin(bet),  0,  0,  0,;
34      0,  0,  0,  0,  0,  0,   ...
            -(sin(alp)*sin(bet)*cos(gam))-(cos(alp)*sin(gam)),   ...
            -(sin(alp)*sin(bet)*sin(gam))+(cos(alp)*cos(gam)),  ...
            sin(alp)*cos(bet),  0,  0,  0,;
35      0,  0,  0,  0,  0,  0,   ...
            -(cos(alp)*sin(bet)*cos(gam))+(sin(alp)*sin(gam)),   ...
            -(cos(alp)*sin(bet)*sin(gam))-(sin(alp)*cos(gam)),  ...
            cos(alp)*cos(bet),  0,  0,  0,;
36      0,  0,  0,  0,  0,  0,  0,  0,  0,   cos(bet)*cos(gam), ...
                                        cos(bet)*sin(gam), ...
                                    sin(bet);
37      0,  0,  0,  0,  0,  0,  0,  0,  0,   ...
            -(sin(alp)*sin(bet)*cos(gam))-(cos(alp)*sin(gam)),   ...
            -(sin(alp)*sin(bet)*sin(gam))+(cos(alp)*cos(gam)),  sin(alp)*cos(bet);
38      0,  0,  0,  0,  0,  0,  0,  0,  0,   ...
            -(cos(alp)*sin(bet)*cos(gam))+(sin(alp)*sin(gam)),   ...
            -(cos(alp)*sin(bet)*sin(gam))-(sin(alp)*cos(gam)),  cos(alp)*cos(bet);
39      ];
40
41  % LOCAL ELEMENT SIFFNESS MATRIX
42
43  [Ae, Iye, Ize, Je] = BeamParameters(mat,dat,Tmat, e);
44
45  le = El_L(e);
46  Ee = mat(1,Tmat(e));
47  Ge = mat(4,Tmat(e));
48
49  Ke =[((Ee*Ae)/le),    0,      0,      0,      0,      0,      (-(Ee*Ae)/le), ...
        0,      0,      0,      0,      0;
50          0, ((12*Ee*Ize)/(le^3)),    0,      0,      0,  ...
                ((6*Ee*Ize)/(le^2)),    0, ((-12*Ee*Ize)/(le^3)),    0,      0, ...
                    0,  ((6*Ee*Ize)/(le^2));
51          0, 0,   ((12*Ee*Iye)/(le^3)),    0,  ((-6*Ee*Iye)/(le^2)),    0, 0, ...
               0, ((-12*Ee*Iye)/(le^3)),    0,  ((-6*Ee*Iye)/(le^2)),    0;
52          0,  0,  0, ((Ge*Je)/le),   0,      0,      0,      0,      0,  ...
                ((-Ge*Je)/le),  0,      0;
53          0, 0,   ((-6*Ee*Iye)/(le^2)),    0,  ((4*Ee*Iye)/(le)),    0, 0, 0, ...
                ((6*Ee*Iye)/(le^2)),    0,  ((2*Ee*Iye)/(le)),    0;
54          0, ((6*Ee*Ize)/(le^2)),    0,      0,      0,  ((4*Ee*Ize)/(le)), ...
                   0, ((-6*Ee*Ize)/(le^2)),    0,      0,      0,  ...
                ((2*Ee*Ize)/(le));
55          ((-Ee*Ae)/le),    0,      0,      0,      0,      0,      ...
                ((Ee*Ae)/le),    0,      0,      0,      0,      0;
56          0, ((-12*Ee*Ize)/(le^3)),    0,      0,      0,  ...
                ((-6*Ee*Ize)/(le^2)),    0, ((12*Ee*Ize)/(le^3)),    0,      0, ...
                    0,  ((-6*Ee*Ize)/(le^2));
57          0, 0,   ((-12*Ee*Iye)/(le^3)),    0,  ((6*Ee*Iye)/(le^2)),    0, 0, ...
               0, ((12*Ee*Iye)/(le^3)),    0,  ((6*Ee*Iye)/(le^2)),    0;
58          0,  0,  0, ((-Ge*Je)/le),   0,      0,      0,      0,      0,  ...
                ((Ge*Je)/le),  0,      0;
```

```
            0, 0,   ((-6*Ee*Iye)/(le^2)),    0,  ((2*Ee*Iye)/(le)),    0, 0, 0, ...
                ((6*Ee*Iye)/(le^2)),    0,  ((4*Ee*Iye)/(le)),    0;
            0, ((6*Ee*Ize)/(le^2)),    0,      0,      0,  ((2*Ee*Ize)/(le)), ...
                   0, ((-6*Ee*Ize)/(le^2)),    0,      0,      0,  ...
                ((4*Ee*Ize)/(le));
            ];

    % GLOBAL ELEMENT STIFFNESS MATRIX
    Ke = R(:,:,e).' * Ke * R(:,:,e);

    % ELEMENT MATRIX ASSEMBLY
    for r=1:(NnodesXelement*Ndim)
         for s=1:(NnodesXelement*Ndim)
             Kel(r,s,e)=Ke(r,s);
         end
    end

end

%% GLOBAL STIFFNESS MATRIX
KG=zeros(Ndofs,Ndofs);
for e=1:Nelements
    for i=1:(NnodesXelement*Ndim)
        I = T2(i,e);
        for j=1:(NnodesXelement*Ndim)
            J = T2(j,e);
            KG(I,J)=KG(I,J)+ Kel(i,j,e);
        end
    end
end


end
```

A.4.2 Force Assembly

```
function [Fext,Drag_T,Lift_T] = ForceAssembly(El_L,rho_eff,Lift,Drag,R,mat,T2)

InputData;


%% Problem Dimensions

Ndim = size(x,1)*2;           % DoF for each node
Nnodes = size(x,2);           % Number of nodes
Nelements = size(Tnod,2);     % Number of elements
NnodesXelement = size(Tnod,1);% Number of nodes per element
Ndofs = Ndim*Nnodes;          % Total DoF of the system



%% FORCE VECTOR AND MATRIX ASSEMBLY
El_W = zeros(Nelements,1);  % Weight Vector
El_T = zeros(Nelements,1);  % Tension Vector
fe_local = zeros(Ndim*NnodesXelement,1);    % External force local
fe = zeros(Ndim*NnodesXelement,Nelements);  % External force global element
```

```
Fext=zeros(Ndofs,1);                          % System External force vector

Lift_T = 0;
Drag_T = 0;


for e = 1:Nelements
    Drag_T = Drag_T + Drag(e)*El_L(e);
    Lift_T = Lift_T + Lift(e)*El_L(e);
end

for e = 1:Nelements

    le = El_L(e);
    [Ae] = BeamParameters(mat,dat,Tmat, e);

    % Element distributed Weight
    El_W(e) = rho_eff(Tmat(e))*Ae*g;

    if e==7 || e==8
      El_W(e) = El_W(e) + ((M_eng*g)/(2*le));
      El_T(e) = Drag_T / (2*le);
    end

    qx = (Drag(e)- El_T(e));
    qz = (Lift(e)-El_W(e));
    qe = [qx; 0; qz; 0; 0; 0; 0; 0; 0; 0; 0; 0;];

    % COMPONENTS IN LOCAL COORDINATE SYSTEM
    qe_local = R(:,:,e)*qe;

%     EQUIVALENT ELEMENT FORCE VECTOR IN LOCAL COORDINATES
    for d = 1:Ndim*NnodesXelement
        % Axial distributed load x-direction
        if d == 1 || d==7
            ind = 1;
            fe_local(d) = (qe_local(ind)*le/2);
        end
        % Axial distributed load y-direction
        if d == 2 || d==6 || d==8 || d==12
            ind = 2;
            if d == 2 || d==8
                fe_local(d) = (qe_local(ind)*le/2)*1;
            end
            if d == 6
                fe_local(d) = (qe_local(ind)*le/2)* (le/6);
            end
            if d == 12
                fe_local(d) = (qe_local(ind)*le/2)* (-le/6);
            end
        end
        % Axial ditributed load z-direction
        if d == 3 || d==5 || d==9 || d==11
            ind = 3;
            if d == 3 || d==9
                fe_local(d) = (qe_local(ind)*le/2)*1;
            end
            if d == 5
                fe_local(d) = (qe_local(ind)*le/2)*(-le/6);
```

```
            end
            if d == 11
                fe_local(d) = (qe_local(ind)*le/2)*(le/6);
            end
        end
    end

    % EQUIVALENT ELEMENT FORCE VECTOR IN GLOBAL COORDINATES
    fe(:,e) = R(:,:,e).' * fe_local;

    % FORCE ASSEMBLY
    for i = 1:NnodesXelement*Ndim
        I = T2(i,e);
        Fext(I)=Fext(I)+fe(i,e);
    end
end


end
```

A.4.3 Solver Function

```
function [U,F_int,u_int,N_x,Q_y,Q_z,T_x,M_y,M_z,rf,Rr] = ...
    solver(Fnod,KG,Kel,R,T2,Fext)

InputData;

%% Problem Dimensions

Ndim = size(x,1)*2;            % DoF for each node
Nnodes = size(x,2);            % Number of nodes
Nelements = size(Tnod,2);      % Number of elements
NnodesXelement = size(Tnod,1);% Number of nodes per element
Ndofs = Ndim*Nnodes;           % Total DoF of the system



%% SYSTEM DEGREES OF FREEDOM

% RESTRICTED DoF's:

NFnod = size (Fnod,2);
vR = zeros(1,NFnod);     % Restricted DoF global
uR = zeros(NFnod,1);     % Restricted displacements

for i = 1:NFnod
   uR(i,1) = Fnod(3,i);
end
for i = 1:NFnod
   node = Fnod(1,i);
   dof  = Fnod(2,i);
   vR(1,i) = Ndim*node - Ndim + dof;
end


% FREE DoF's
```

```
34 vL = setdiff(1:Ndofs,vR);
35
36
37
38 %% REDUCED SYSTEM SOLUTION
39
40 KLL = KG(vL,vL);
41 KLR = KG(vL,vR);
42 KRL = KG(vR,vL);
43 KRR = KG(vR,vR);
44 fextL = Fext(vL,1);
45 fextR = Fext(vR,1);
46
47 % LINEAR SYSTEM SOLUTION
48 ul = KLL\(fextL - KLR*uR);
49 Rr = KRR*uR + KRL*ul - fextR;
50
51 % GLOBAL DISPLACEMENTS ASSEMBLY
52 U = zeros(Ndofs,1);
53 U(vL) = ul;
54 U(vR) = uR;
55
56 % GLOBAL REACTIONS ASSEMBLY
57 rf = zeros(Ndofs,1);
58 rf(vL) = 0;
59 rf(vR) = Rr;
60
61
62 %% INTERNAL FORCES AND MOMENTS
63
64 u_int = zeros(NnodesXelement*Ndim, Nelements);  % Internal displacements and rot
65 F_int = zeros(NnodesXelement*Ndim, Nelements);  % Internal forces in local
66 N_x = zeros(1,Nelements);
67 Q_y = zeros(1,Nelements);
68 Q_z = zeros(1,Nelements);
69 T_x = zeros(1,Nelements);
70 M_y = zeros(2,Nelements);
71 M_z = zeros(2,Nelements);
72
73 for  e = 1:Nelements
74     for i = 1:NnodesXelement*Ndim
75         I=T2(i,e);
76         % INTERNAL GLOBAL DISPLACEMENTS AND ROTATIONS
77         u_e(i,1) = U(I,1);
78     end
79     % INTERNAL LOCAL DISPLACEMENTS AND ROTATIONS
80     u_int(:,e) = R(:,:,e) * u_e;
81
82     % INTERNAL FORCE VECTOR IN GLOBAL
83     F_e = Kel(:,:,e) * u_e;
84     % INTERNAL FORCE VECTOR IN LOCAL
85     F_int(:,e) = R(:,:,e) * F_e;
86     % AXIAL FORCE
87     N_x(e) = -F_int(1,e);  % X direction
88     % SHEAR FORCE
89     Q_y(e) = - F_int(2,e);     % Y direction
90     Q_z(e) = - F_int(3,e);     % Z direction
91     %TORSION MOMENT
92     T_x(e) = - F_int(4,e);      % Arround X-axis
```

```
93      % BENDING MOMENT (Linear)
94      M_y(1,e)  = -F_int(5,e);
95      M_y(2,e)  = F_int(11,e);      % Arround Y-axis
96      M_z(1,e)  = -F_int(6,e);
97      M_z(2,e)  =  F_int(12,e);     % Arround Z-axis
98
99  end
100
101
102 end
```

A.5 Post-processing

```
1   function  ...
        PostProcess(M_struct,M_rib,M_spar,V_rib,V_spar,Lift_T,Drag_T,V_tot,U,u_int,F_int,El_L,N_x,Q_y,
2
3   InputData;
4
5   % Plot tittles with lattex typography
6   % set(groot,'defaulttextinterpreter','latex');
7   % set(groot,'defaultaxesticklabelinterpreter','latex');
8   % set(groot,'defaultlegendinterpreter','latex');
9
10
11  %% DATA DISPLAY - PLOTS
12
13  % PLOT 1 - PLOTWING
14  x = x.';
15  Tnod = Tnod.';
16  U = U.';
17  plotWing(x,Tnod,El_L,U,u_int,N_x,Q_y,Q_z,T_x,M_y,M_z);
18
19  % PLOT 2 - PLOTBEAMS3D DEFORMATION
20  nsub  = 10;     % Number of subdivisions
21  factor = 2.5;   % Amplification factor
22
23  plotBeams3D_def(x,Tnod,nsub,El_L,u_int,factor,R)
24
25  % PLOT BEAMS 3D - CRITICAL BEAMS
26  maxAxial_e = find(N_x(1,:) == max(N_x));
27
28  maxShear_y = max(abs(Q_y));
29  maxShear_z = max(abs(Q_z));
30
31  if maxShear_y > maxShear_z
32      maxShear_e = find(abs(Q_y(1,:)) == maxShear_y);
33      maxShear = maxShear_y;
34      dirs = 'Y';
35  else
36      maxShear_e = find(abs(Q_z(1,:)) == maxShear_z);
37      maxShear = maxShear_z;
38      dirs = 'Z';
39  end
40
41  maxBend_y = max(max(abs(M_y)));
42  maxBend_z = max(max(abs(M_z)));
```

```
43
44 if maxBend_y > maxBend_z
45     maxBend_e = find(abs(M_y(1,:)) == maxBend_y);
46     maxBend = maxBend_y;
47     dirb = 'Y';
48 else
49     maxBend_e = find(abs(M_z(1,:)) == maxBend_z);
50     maxBend = maxBend_z;
51     dirb = 'Z';
52 end
53
54
55 e_plot = maxAxial_e;
56 plotBeams3D(Tnod,nsub,El_L,u_int,e_plot)
57
58
59 %% DATA DISPLAY - CONSOLE
60 j=0;
61 for i =1:12
62     if i≤6
63         fprintf('Node:  1  Dof: %d  Reaction: %0.4f  \n',i,Rr(i));
64     else
65         j=j+1;
66         fprintf('Node:  5  Dof: %d  Reaction: %0.4f  \n',j,Rr(i));
67     end
68 end
69 disp('------------------------------------------------------------------------')
70 fprintf('Structure Mass %0.4f  Kg \n',M_struct);
71 fprintf('Structure Volume %0.4f  m^3 \n',V_tot);
72
73 fprintf('Ribs Mass %0.4f  Kg \n',M_rib);
74 fprintf('Ribs Volume %0.4f  m^3 \n',V_rib);
75
76 fprintf('Spars Mass %0.4f  Kg \n',M_spar);
77 fprintf('Spars Volume %0.4f  m^3 \n',V_spar);
78
79 fprintf('Wing Mass %0.4f  Kg \n',M_struct+M_w);
80 fprintf('Wing & Motor Mass %0.4f  Kg \n',M_struct+M_w+M_eng);
81 fprintf('Wing Lift %0.4f  N \n',Lift_T);
82 fprintf('Wing Drag %0.4f  N \n',Drag_T);
83 disp('------------------------------------------------------------------------')
84
85 fprintf('Max Axial force = %d    At element:  %d   \n',max(N_x),maxAxial_e);
86 fprintf('Max Shear force = %d    At element:  %d  Direction: %s ...
       \n',maxShear,maxShear_e,dirs);
87 fprintf('Max Bending moment = %d    At element:  %d  Direction: %s  ...
       \n',maxBend,maxBend_e,dirb);
88 disp('------------------------------------------------------------------------')
89
90
91 end
```

```
1 function plotWing(x,Tnod,le,u,uint,n,qy,qz,t,my,mz)
2 % Inputs:
3 %  x     nodal coordinates matrix [Nnodes x Ndim]
4 %  Tnod  nodal connectivities matrix [Nelements x NnodesXelement]
5 %  le    vector with element lengths [Nelements x 1]
```

```
6  %  u     global displacements/rotations vector [Ndofs x 1]
7  %  uint  matrix with element displacements and rotations in LOCAL axes [12 x ...
       Nelements]
8  %        uint(1,e) : ux for node 1 of element e in local reference frame
9  %        uint(2,e) : uy for node 1 of element e in local reference frame
10 %        uint(3,e) : uz for node 1 of element e in local reference frame
11 %        uint(4,e) : angle_x for node 1 of element e in local reference frame
12 %        uint(5,e) : angle_y for node 1 of element e in local reference frame
13 %        uint(6,e) : angle_z for node 1 of element e in local reference frame
14 %        uint(7,e) : ux for node 2 of element e in local reference frame
15 %        uint(8,e) : uy for node 2 of element e in local reference frame
16 %        uint(9,e) : uz for node 2 of element e in local reference frame
17 %        uint(10,e): angle_x for node 2 of element e in local reference frame
18 %        uint(11,e): angle_y for node 2 of element e in local reference frame
19 %        uint(12,e): angle_z for node 2 of element e in local reference frame
20 %  n     Axial force of element e in local reference frame [1 x Nelements]
21 %  qy    Shear force in y-direction of element e in local reference frame [1 x ...
       Nelements]
22 %  qz    Shear force in z-direction of element e in local reference frame [1 x ...
       Nelements]
23 %  t     Torsion moment of element e in local reference frame [1 x Nelements]
24 %  my    Bending moment in y-direction of element e in local reference frame [2 ...
       x Nelements]
25 %  mz    Bending moment in z-direction of element e in local reference frame [2 ...
       x Nelements]
26
27 % Get dimensions
28 Ndim = size(x,2);
29 Nnodes = size(x,1);
30 Nelements = size(Tnod,1);
31 NnodesXelement = size(Tnod,2);
32 NdofsXnode = 6;
33
34 % X,Y,Z data
35 X = reshape(x(Tnod',1),NnodesXelement,Nelements);
36 Y = reshape(x(Tnod',2),NnodesXelement,Nelements);
37 Z = reshape(x(Tnod',3),NnodesXelement,Nelements);
38
39 % Displacements
40 u = reshape(u,NdofsXnode,Nnodes);
41 Ux = reshape(u(1,Tnod'),NnodesXelement,Nelements);
42 Uy = reshape(u(2,Tnod'),NnodesXelement,Nelements);
43 Uz = reshape(u(3,Tnod'),NnodesXelement,Nelements);
44 D = sqrt(Ux.^2+Uy.^2+Uz.^2);
45
46 % Displacement X
47 ux = uint([1,7],:);
48
49 % Displacement Y
50 uy = uint([2,8],:);
51
52 % Displacement Z
53 uz = uint([3,9],:);
54
55 % Rotation X
56 theta_x = uint([4,10],:);
57
58 % Rotation Y
59 theta_y = uint([5,11],:);
```

```
60
61 % Rotation Z
62 theta_z = uint([6,12],:);
63
64 % Strain - Displacement X
65 Aux = repmat(diff(uint([1,7],:),1,1),2,1)./reshape(le,1,[]);
66
67 % Axial force
68 F = repmat(reshape(n,1,[]),2,1);
69
70 % Shear Y
71 Qy = repmat(reshape(qy,1,[]),2,1);
72
73 % Shear Z
74 Qz = repmat(reshape(qz,1,[]),2,1);
75
76 % Torsion
77 T = repmat(reshape(t,1,[]),2,1);
78
79 % Bending Moment Y
80 My = my;
81
82 % Bending Moment Z
83 Mz = mz;
84
85 % Initialize figure
86 figure('visible','off','color','w','Name','Wing','position',[50,50,565,540],'SizeChangedFcn',@res
87
88 % Create uicontrols
89 scale_lab = uicontrol('style','text','string','scale = ...
       ','position',[20,495,40,20],'backgroundcolor','w');
90 scale_edt = ...
       uicontrol('style','edit','string','1','position',[60,498,40,22],'callback',@scaleFactor);
91 show_lab = ...
       uicontrol('style','text','string','Show:','position',[120,495,40,20],'backgroundcolor','w');
92 show_pop = ...
       uicontrol('style','popupmenu','string',{'Displacements','Displacement ...
       X','Displacement Y','Displacement Z',...
93      'Rotation X','Rotation Y','Rotation Z','Axial strain','Axial force','Shear ...
           force Y','Shear force Z',...
94      'Torsion','Bending moment Y','Bending moment ...
           Z'},'position',[160,500,160,20],'callback',@showResults);
95 para_ax = ...
       axes('units','pixels','position',[20,20,400,450],'xcolor','none','ycolor','none','zcolor','nor
96
97 % Initial plot
98 sz = 2;
99 ori = [x(1,1),x(1,2),x(1,3)]-0.1;
100 hold on;
101 plot3(ori(1)+[0,sz],ori(2)+[0,0],ori(3)+[0,0],'r','linewidth',1.5); ...
       text(ori(1)+1.1*sz,ori(2),ori(3),'X','color','r');
102 plot3(ori(1)+[0,0],ori(2)+[0,sz],ori(3)+[0,0],'g','linewidth',1.5); ...
       text(ori(1),ori(2)+1.1*sz,ori(3),'Y','color','g');
103 plot3(ori(1)+[0,0],ori(2)+[0,0],ori(3)+[0,sz],'b','linewidth',1.5); ...
       text(ori(1),ori(2),ori(3)+1.1*sz,'Z','color','b');
104 patch(X,Y,Z,ones(size(X)),'edgecolor',[0.5,0.5,0.5],'linewidth',1);
105 p = patch(X+Ux,Y+Uy,Z+Uz,D,'edgecolor','interp','linewidth',2);
106 view(30,25);
107 axis equal;
```

```
colormap jet;
cbar = colorbar;
updateCase(D);
set(gcf,'visible','on');
set(gca,'xcolor','none','ycolor','none','zcolor','none');

function showResults(varargin)
    updateResults;
end

function resizeWindow(varargin)
    pos = get(gcf,'position');
    set(para_ax,'position',[20,20,pos(3)-165,pos(4)-90]);
    set(scale_lab,'position',[20,pos(4)-45,40,20]);
    set(scale_edt,'position',[60,pos(4)-42,40,22]);
    set(show_lab,'position',[120,pos(4)-45,40,20]);
    set(show_pop,'position',[160,pos(4)-40,160,20]);
end

function updateCase(var)
    set(p,'Cdata',var);
    caxis([min(var(:)),max(var(:))]);
    set(cbar,'Ticks',linspace(min(var(:)),max(var(:)),5));
    set(cbar,'visible','on');
end

function updateResults
    switch get(show_pop,'value')
        case 1
            updateCase(D);
        case 2
            updateCase(ux);
        case 3
            updateCase(uy);
        case 4
            updateCase(uz);
        case 5
            updateCase(theta_x);
        case 6
            updateCase(theta_y);
        case 7
            updateCase(theta_z);
        case 8
            updateCase(Aux);
        case 9
            updateCase(F);
        case 10
            updateCase(Qy);
        case 11
            updateCase(Qz);
        case 12
            updateCase(T);
        case 13
            updateCase(My);
        case 14
            updateCase(Mz);
    end
end

```

```
function scaleFactor(varargin)
    scale = str2double(get(scale_edt,'String'));
    set(p,'XData',X+scale*Ux,'YData',Y+scale*Uy,'ZData',Z+scale*Uz);
end

end
```

```
function plotBeams3D(Tnod,nsub,le,uint,e_plot)
% PLOTBEAMS3D — Plot displacements, rotations for 3D beam in local reference fram
% Inputs:
%   Tnod     Nodal connectivities matrix [Nelements x NnodesXelement]
%   nsub     Number of elements's subdivisions to evaluate displacements and ...
    rotations
%   le       vector with element lengths [Nelements x 1]
%   uint     matrix with element displacements and rotations in LOCAL axes [12 ...
    x Nelements]
%   e_plot   Element to plot


% Compute deflection and rotation polynomial coefficients
nel = size(Tnod,1);
puy = zeros(nel,4);
pty = zeros(nel,3);
puz = zeros(nel,4);
ptz = zeros(nel,3);

for e = 1:nel
    temp1 = [
        2,      le(e),   -2,    le(e);
        -3*le(e), -2*le(e)^2, 3*le(e), -le(e)^2;
        0,    le(e)^3,    0,     0;
        le(e)^3,       0,    0,     0];
    temp2 = temp1; temp2(:,[2,4]) = -temp2(:,[2,4]);
    matCoeff = zeros(8,12);
    matCoeff(1:4,[2,6,8,12]) = temp1;
    matCoeff(5:8,[3,5,9,11]) = temp2;
    coeff = 1/le(e)^3*matCoeff*uint(:,e);

    puy(e,:) = [coeff(1),coeff(2),coeff(3),coeff(4)];
    ptz(e,:) = [3*coeff(1),2*coeff(2),coeff(3)];
    puz(e,:) = [coeff(5),coeff(6),coeff(7),coeff(8)];
    pty(e,:) = [3*coeff(5),2*coeff(6),coeff(7)];
end

x_coord = linspace(0,le(e_plot),nsub);
y_coord = polyval(puy(e_plot,:),x_coord);
z_coord = polyval(puz(e_plot,:),x_coord);
theta_y = polyval(pty(e_plot,:),x_coord);
theta_z = polyval(ptz(e_plot,:),x_coord);

fig = figure();
set(fig,'Name',['Deformation of element ',num2str(e_plot)])

% Plot beam deflections
subplot(2,2,1)
plot(x_coord,y_coord);
set(gca,'XTickLabel',sprintf('%i\n',Tnod(e_plot,:)),'XTick',[0 le(e_plot)]);
```

```
xlabel('beam');
ylabel('U_y (m)');
title('Deflection in y');
grid
grid minor
xlim(gca,[0 le(e_plot)]);

subplot(2,2,2)
plot(x_coord,z_coord);
set(gca,'XTickLabel',sprintf('%i\n',Tnod(e_plot,:)),'XTick',[0 le(e_plot)]);
xlabel('beam');
ylabel('U_z (m)');
title('Deflection in z');
grid
grid minor
xlim(gca,[0 le(e_plot)]);

% Plot beam section rotations
subplot(2,2,4)
plot(x_coord,theta_y);
set(gca,'XTickLabel',sprintf('%i\n',Tnod(e_plot,:)),'XTick',[0 le(e_plot)]);
xlabel('beam');
ylabel('\theta_y (rad)');
title('Rotation in y');
grid
grid minor
xlim(gca,[0 le(e_plot)]);

subplot(2,2,3)
plot(x_coord,theta_z);
set(gca,'XTickLabel',sprintf('%i\n',Tnod(e_plot,:)),'XTick',[0 le(e_plot)]);
xlabel('beam');
ylabel('\theta_z (rad)');
title('Rotation in z');
grid
grid minor
xlim(gca,[0 le(e_plot)]);

end
```

```
function plotBeams3D_def(x,Tnod,nsub,le,uint,factor,Re)
% PLOTBEAMS3D_DEF — Plot deformed structure of 3D BEAMS
% Inputs:
%   x        Nodal coordinates matrix [Nnodes x Ndim]
%   Tnod     Nodal connectivities matrix [Nelements x NnodesXelement]
%   nsub     Number of elements's subdivisions to evaluate displacements and ...
    rotations
%   le       vector with element lengths [Nelements x 1]
%   uint     matrix with element displacements and rotations in LOCAL axes [12 ...
    x Nelements]
%   factor   Number to scale/amplify the displacements
%   Re       Rotation matrix [12 x 12 x Nelements]


fig1 = figure();
set(fig1,'Name','Deformed structure','Color','w');

```

```
% Initial plot
sz = 2;
ori = [x(1,1),x(1,2),x(1,3)]-0.1;
hold on;
plot3(ori(1)+[0,sz],ori(2)+[0,0],ori(3)+[0,0],'r','linewidth',1.5); ...
    text(ori(1)+1.1*sz,ori(2),ori(3),'X','color','r');
plot3(ori(1)+[0,0],ori(2)+[0,sz],ori(3)+[0,0],'g','linewidth',1.5); ...
    text(ori(1),ori(2)+1.1*sz,ori(3),'Y','color','g');
plot3(ori(1)+[0,0],ori(2)+[0,0],ori(3)+[0,sz],'b','linewidth',1.5); ...
    text(ori(1),ori(2),ori(3)+1.1*sz,'Z','color','b');
patch('Vertices',x,'Faces',Tnod,'edgecolor',[0.5,0.5,0.5],'linewidth',1);

% Compute deflection polynomial coefficients and plot the deformed beams
% using nsub subdivisions

nel = size(Tnod,1);
puy = zeros(nel,4);
puz = zeros(nel,4);

for e = 1:nel
    temp1 = [
        2,      le(e),   -2,    le(e);
        -3*le(e), -2*le(e)^2, 3*le(e), -le(e)^2;
        0,    le(e)^3,    0,     0;
        le(e)^3,       0,    0,     0];
    temp2 = temp1; temp2(:,[2,4]) = -temp2(:,[2,4]);

    matCoeff = zeros(8,12);
    matCoeff(1:4,[2,6,8,12]) = temp1;
    matCoeff(5:8,[3,5,9,11]) = temp2;

    coeff = 1/le(e)^3*matCoeff*uint(:,e);
    puy(e,:) = [coeff(1),coeff(2),coeff(3),coeff(4)];
    puz(e,:) = [coeff(5),coeff(6),coeff(7),coeff(8)];

    x_coord = linspace(0,le(e),nsub+1);
    uy_coord = polyval(puy(e,:),x_coord);
    uz_coord = polyval(puz(e,:),x_coord);
    ux_coord = linspace(uint(1,e),uint(7,e),nsub+1);

    temp_coord = x(Tnod(e,1),:)' +  Re(1:3,1:3,e)' * ...
        [x_coord;zeros(2,size(x_coord,2))]  + factor * Re(1:3,1:3,e)' * ...
        [ux_coord;uy_coord;uz_coord];
    patch('Vertices',temp_coord','Faces',[1:nsub;2:nsub+1]','FaceColor','none','LineWidth',2);
end

view(30,25);
axis equal;
set(gca,'xcolor','none','ycolor','none','zcolor','none');
```