DESCRIPTION OF THE DEVELOPER MODEL USED IN THE AE3212-II PROJECT

DEVELOPER MODEL

Authors:
S.J. VAN ELSLOO
Dr. ir. J.M.J.F. VAN CAMPEN
Dr. ir. W. VAN DER WAL

AE3212-II
Simulation, Verification & Validation
February 2021

Contents

Chapter 1: Introduction

This document aims to explain the methods used in the developer model that is provided to you as part of this project. Furthermore, it aims to explain how to use the program itself.

Chapter 2 - 6 aim to provide the theoretical background necessary to understand the developer model. Chapter 7 provides guidance on how to use the program.

You are reminded that naturally, your numerical model may not be an exact copy of the methods outlined in this document. You are referred to the assignment description for the precise requirements that are posed on your numerical model.

Chapter 2: Bending stiffness calculations

The first part of the program is to compute the bending stiffness properties, i.e. the location of the centroid and the second moments of area. All components are assumed to be thin-walled. The coordinate system used in subsequent sections is located at the leading edge of the aileron, with the z-axis aligned with the chord (and thus a symmetry-axis) and pointing towards the trailing edge, and y is the corresponding perpendicular axis (pointing upwards).

2.1 Centroid

The centroid is computed using elementary procedures. The stringer spacing Δ_{st} was computed by computing the perimeter of the aileron and dividing this by n_{st}. The first stringer was placed on the leading edge of the aileron. The placement of the remaining stringers then follows straightforwardly. The developer model only functions for cases with an odd number of stringers. Stringers are assumed to be point areas located on the skin itself, so no effort is made to estimate the precise location of the centroid of each stringer based on the shape of the stinger. Furthermore, as mentioned before all components are assumed to be thin-walled, and computation of the stringer areas is consistent with that assumption.

The centroid calculations are detailed in the following subsections.

2.1.1 Stringer placement

The stringer spacing Δ_{st} was computed by computing the perimeter of the aileron and dividing this by n_{st}. The first stringer was placed on the leading edge of the aileron. The number of stringers placed along the leading edge was computed via

$$n_{circ} = 2\left\lfloor \frac{h_a\pi/4}{\Delta_{st}} \right\rfloor + 1$$

where $h_a\pi/4$ represents the length of a quarter-circle with diameter h_a. Note that the above will always result an odd number.

2.1.2 Stringer centroid

Stringers are assumed to be point areas located on the skin itself, so no effort is made to estimate the precise location of the centroids of the stringers based on the shape of the stinger.

2.1.3 Stringer area

As mentioned before, all components are assumed to be thin-walled, thus the area of a stringer is simply computed via

$$A_{st} = h_{st}t_{st} + w_{st}t_{st}$$

2.1.4 Placement of stringers on semi-circle

The first stringer is placed on the leading edge. The angle between stringers is then computed using

$$\phi = \frac{\Delta_{st}}{h_a\pi} \cdot 2\pi$$

where $h_a\pi$ represents the perimeter of a full circle with diameter h_a. Thus, the z and y coordinates of the ith stringer after the stringer on the leading edge (on the positive y-axis) are given by (for $i\phi < \frac{\pi}{2}$):

$$\tilde{z}_{st_i} = \frac{h_a}{2}\left(1 - \cos\left(i\phi\right)\right) 1 \leq i < \frac{\pi}{2}/\phi$$
$$\tilde{y}_{st_i} = \frac{h_a}{2}\left(\sin\left(i\phi\right)\right) 1 \leq i < \frac{\pi}{2}/\phi$$

2.1.5 Placement of stringers on skin

The stringers on the skin are placed by first computing the inclination of the skin, equal to

$$\psi = \arctan\left(\frac{h_a/2}{C_a - h_a/2}\right) \tag{2.1}$$

The z-coordinate of the jth stringer (with $j = 0$ corresponding to the last stringer before the trailing edge, and $j > 0$ corresponding to the stringers before that) is equal to

$$z_j = C_a - \frac{\Delta_{st}}{2}\cos(\psi) - j\Delta_{st}\cos(\psi) \qquad \forall \quad j \ni z_j > h_a/2$$

Similarly, for y_j:

$$y_j = \frac{\Delta_{st}}{2}\sin(\psi) + j\Delta_{st}\sin(\psi) \qquad \forall \quad j \ni y_j < h_a/2$$

The coordinates of stringers placed on the lower skin are obtained by simply taking negative values for y_j.

Finally, the coordinates of all stringers are stored in a 2D `numpy`-array, with each row corresponding to a stringer, starting from the leading edge stringer and then ordering in clockwise direction.

2.1.6 Semi-circle centroid

The centroid of the semi-circle was taken to be located at (from standard tables)

$$\tilde{z}_{circ} = h_a\left(\frac{1}{2} - \frac{1}{\pi}\right)$$

The centroid of the the remaining elements (top and bottom skin, spar and semicircle) are known from literature, allowing for straightforward computation of the centroid of the aileron as a whole.

2.2 Second moment of area

The moment of inertia is also computed using elementary procedures. All components are assumed to be thin-walled. Stringers are assumed to be point areas, with no moment of inertia around their own axis; only Steiner terms are included (with their centroids again assumed to be located on the skin itself). The moment of inertia of the remaining elements (top and bottom skin, spar and semicircle) are again known from literature. Care was taken in selecting the right moment of inertia around the y-axis for the semicircle. The results are then combined straightforwardly using Steiner terms when appropriate.

The moment of inertia calculations are detailed in the following subsections.

2.2.1 Stringer second moment of area

Stringers are assumed to be point areas, with no moment of inertia around their own axis; only Steiner terms are included (with their centroids again assumed to be located on the skin itself).

2.2.2 Skin second moment of area

The moment of inertia of the two skins around their own centroid has been computed as

$$I_{zz} = \frac{l_{st} t_{st}\left(\frac{h_a}{2}\right)^2}{12}$$
$$I_{yy} = \frac{l_{st} t_{st}\left(C_a - \frac{h_a}{2}\right)^2}{12}$$

with

$$l_{sk} = \sqrt{\left(C_a - \frac{h_a}{2}\right)^2 + \left(\frac{h_a}{2}\right)^2} \tag{2.2}$$

denoting the length of the skin. Steiner terms were then appropriately taken into account (for both skins).

2.2.3 Spar second moment of area

The spar only has a moment of area about the horizontal axis, equal to

$$I_{zz} = \frac{h_a^3 t_{sp}}{12}$$

Then Steiner terms were appropriately taken into account.

2.2.4 Semi-circle second moment of area

The second moment of area about the horizontal axis is straightforwardly computed with

$$I_{zz} = \frac{\pi}{2} \left(\frac{h_a}{2} \right)^3 t_{sk}$$

The second moment of area about the vertical axis is a more delicate issue. First, the second moment of area about the vertical axis passing through the ends of the semi-circle (i.e., coincident with the spar) is computed, equal to

$$I_{\tilde{y}\tilde{y}} = \frac{\pi}{2} \left(\frac{h_a}{2} \right)^3 t_{sk}$$

Then, the Steiner term to move to the centroid of the semi-circle is *substracted* from this, i.e. the final second moment of area of the semi-circle about its own centroid is equal to

$$I_{yy} = I_{\tilde{y}\tilde{y}} - \left(\frac{h_a}{2} - \tilde{z}_{circ} \right)^2 \frac{h_a \pi}{2} t_{sk}$$

with $\tilde{z}_{circ}$ determined previously.

Chapter 3: Torsional stiffness calculations

The second part of the program is to compute the torsional stiffness properties, i.e. the location of the shear center and the torsional stiffness. All components are assumed to be thin-walled. Stringers are assumed to be point areas, but the skins and spar *are* assumed to carry bending stresses; the shear flow thus varies between booms, and the area of the booms (which are only placed at the location of the stringers) are set equal to the stringer area. The coordinate system used in subsequent sections is located at the leading edge of the aileron, with the z-axis aligned with the chord (and thus a symmetry-axis) and pointing towards the trailing edge, and y is the corresponding perpendicular axis (pointing upwards).

3.1 Shear center

The shear center is computed using the concept of shear flow distributions. Stringers are assumed to be point areas, but the skins *are* assumed to carry bending stresses; the shear flow thus varies between booms, and the area of the booms (which are only placed at the location of stringers) are set equal to the stringer area. The base shear flow is thus found analytically.

3.1.1 Base shear flow distribution

Due to symmetry, it is clearly only necessary to compute z_{sc}. As a result, we only have to apply a unit vertical shear force, and compute the corresponding base shear flow distribution. Figure 3.1 shows the aileron, along with some definitions:

- (1) - (6) denote the regions in which the shear flow shall be determined separately.
- The red dots represent the locations of the stringers.
- The blue arrows indicate the *assumed* positive direction for the shear flows.
- The pink arrows indicate the *assumed* positive direction of the redundant shear flows; $q_{0,1}$ corresponds to the left cell; $q_{0,2}$ corresponds to the right cell.
- The purple dot and arrow show how the position of the shear center is computed: the moment about the middle of the spar is computed (with clockwise positive); the vertical shear force is assumed to act a distance η to the right of this point (and thus the external moment is negative for positive η).
- The orange $r-\theta$ coordinate system indicates the coordinate system that will be used to evaluate the base shear flows in regions 1 and 2.
- s_3 and s_4 indicate the definition of s that will be used in regions (3) and (4).
- Double green lines indicate the locations of the cuts made to compute the base shear flow: the first cut is made in the middle of the stringer located on the leading edge (such that this half of the stringer area is part of region (1), and the other half is part of region (6), so that symmetry is preserved); the second cut is made in the middle of the spar.

The governing equation behind the base shear flow distribution that will be used is

$$q_b(s) = \frac{-S_y}{I_{zz}} \left(\int_0^s ty\,ds + \sum_i^{i \ni s_i < s} B_i y_i \right) + q_{b_0} \tag{3.1}$$

where B_i are the stringer areas (all equal to each other), y_i is the y-coordinate of the ith stringer, and q_{b_0} is the base shear flow resulting from the previous element, and where the sum accounts for all stringers that have been encountered up unto the current value of s.

Region (1) In region (1), it is easier to express the equation in terms of θ, with $y = h\sin\theta$ (with $h = h_a/2$) and $ds = hd\theta$; one obtains

$$\begin{aligned} q_{b_1}(\theta) &= -\frac{S_y t_{sk}}{I_{zz}} \left(\int_0^\theta h^2 \sin\theta d\theta + \sum_i^{i \ni \theta_i < \theta} B_i h \sin\theta_i \right) \\ &= -\frac{S_y t_{sk}}{I_{zz}} \left(h^2 [\cos\theta - 1] + \sum_i^{i \ni \theta_i < \theta} B_i h \sin\theta_i \right), \qquad 0 \le \theta \le \frac{\pi}{2} \end{aligned}$$

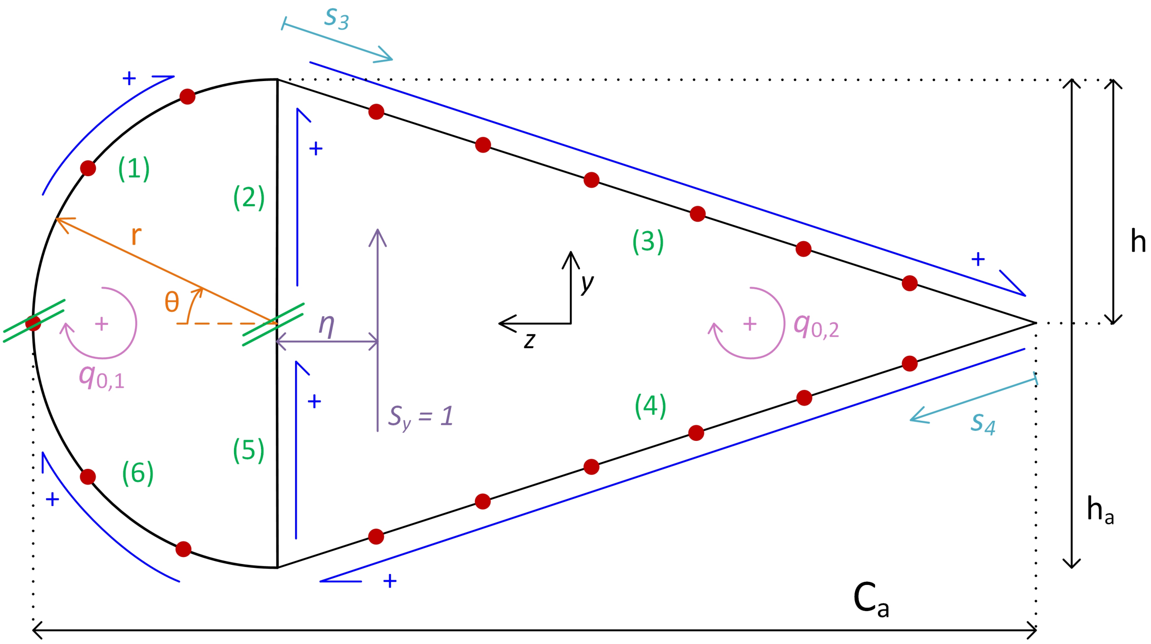

Figure 3.1: Definition of positive directions for shear flow calculations.

Evaluating the effect of the booms is simply a matter of programming.

Region (2) In region (2), there are no booms present, so we simply obtain (with $ds = dy$)

$$q_{b_2}(y) = -\frac{S_y t_{sp}}{I_{zz}} \int_0^y y\,dy = -\frac{S_y t_{sp}}{2I_{zz}} y^2 \qquad 0 \leq y \leq h$$

Region (3) In region (3), we measure s_3 from the intersection between the spar and the top skin. Consequently, we have

$$y = h + \frac{dy}{ds} s_3 = h - \frac{h}{l_{sk}} s_3$$

where l_{sk} has been computed in Equation (2.2). Thus, we obtain

$$\begin{aligned} q_{b_3}(s_3) &= -\frac{S_y t_{sk}}{I_{zz}} \left(\int_0^{s_3} \left[h - \frac{h}{l_{sk}} s_3 \right] ds_3 + \sum_i^{i \ni s_i < s_3} B_i y_i \right) + q_{b_1}\left(\frac{\pi}{2}\right) + q_{b_2}(h) \\ &= -\frac{S_y t_{sk}}{I_{zz}} \left(h s_3 - \frac{h}{2l_{sk}} s_3^2 + \sum_i^{i \ni s_i < s_3} B_i y_i \right) + q_{b_1}\left(\frac{\pi}{2}\right) + q_{b_2}(h), \qquad 0 \leq s_3 \leq l_{sk} \end{aligned}$$

Region (4) In region (4), we measure s_4 from the trailing edge. Consequently, we have

$$y = \frac{dy}{ds} s_3 = -\frac{h}{l_{sk}} s_3$$

such that

$$\begin{aligned} q_{b_4}(s_4) &= -\frac{S_y t_{sk}}{I_{zz}} \left(\int_0^{s_4} \left[-\frac{h}{l_{sk}} s_4 \right] ds_4 + \sum_i^{i \ni s_i < s_4} B_i y_i \right) + q_{b_3}(l_{sk}) \\ &= -\frac{S_y t_{sk}}{I_{zz}} \left[-\frac{h}{2l_{sk}} s_4^2 + \sum_i^{i \ni s_i < s_4} B_i y_i \right) + q_{b_3}(l_{sk}), \qquad 0 \leq s_4 \leq l_{sk} \end{aligned}$$

Region (5) In region (5), we simply integrate downwards from the cut in the spar; with $dy = ds$, we thus obtain

$$q_{b_5}(y) = -\frac{S_y t_{sp}}{I_{zz}} \int_0^y t y\, dy = -\frac{S_y t_{sp}}{2 I_{zz}} y^2, \qquad -h \leq y \leq 0$$

Region (6) In region (6), we apply the same transformation as before:

$$\begin{aligned} q_{b_6}(\theta) &= -\frac{S_y t_{sk}}{I_{zz}} \left(\int_{-\pi/2}^{\theta} h^2 \sin\theta\, d\theta + \sum_i^{i \ni \theta_i < \theta} B_i h \sin\theta_i \right) + q_{b_4}\left(l_{sk}\right) - q_{b_5}(-h) \\ &= -\frac{S_y t_{sk}}{I_{zz}} \left(h^2 \cos\theta + \sum_i^{i \ni \theta_i < \theta} B_i h \sin\theta_i \right) + q_{b_4}\left(l_{sk}\right) - q_{b_5}(-h), \qquad -\frac{\pi}{2} \leq \theta \leq 0 \end{aligned}$$

3.1.2 Redundant shear flow distribution

The redundant shear flow distribution may be found by setting the rate of twist equal to zero, resulting in

$$0 = \oint \frac{q}{t} ds$$

since the shear modulus is constant. Splitting this in a contribution to the readily computed base shear flows, and the to be determined redundant shear flow, this results in a matrix equation

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} q_{0,1} \\ q_{0,2} \end{bmatrix} = - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{3.2}$$

with

$$A_{11} = \int_0^{\pi/2} \frac{h d\theta}{t_{sk}} + \int_h^0 \frac{dy}{t_{sp}} + \int_0^{-h} \frac{dy}{t_{sp}} + \int_{-\pi/2}^{0} \frac{h d\theta}{t_{sk}} \tag{3.3}$$

$$A_{12} = -\int_h^0 \frac{dy}{t_{sp}} - \int_0^{-h} \frac{dy}{t_{sp}} \tag{3.4}$$

$$A_{21} = -\int_h^0 \frac{dy}{t_{sp}} - \int_0^{-h} \frac{dy}{t_{sp}} \tag{3.5}$$

$$A_{22} = \int_h^0 \frac{dy}{t_{sp}} + \int_0^{l_{sk}} \frac{ds}{t_{sk}} + \int_0^{l_{sk}} \frac{ds}{t_{sk}} + \int_{-h}^{0} \frac{dy}{t_{sp}} \tag{3.6}$$

and

$$\begin{aligned} b_1 &= \int_0^{\pi/2} \frac{q_{b_1}(\theta)\, h d\theta}{t_{sk}} - \int_h^0 \frac{q_{b_2}(y)\, dy}{t_{sp}} - \int_0^{-h} \frac{q_{b_2}(y)\, dy}{t_{sp}} + \int_{-\pi/2}^{0} \frac{q_{b_6}(\theta)\, h d\theta}{t_{sk}} \\ b_2 &= \int_0^{h} \frac{q_{b_2}(y)\, dy}{t_{sp}} + \int_0^{l_{sk}} \frac{q_{b_3}\left(s_3\right) ds_3}{t_{sk}} + \int_0^{l_{sk}} \frac{q_{b_4}\left(s_4\right) ds_4}{t_{sk}} + \int_{-h}^{0} \frac{q_{b_5}(y)\, dy}{t_{sp}} \end{aligned}$$

The integrals for A are straightforwardly found; the integrals for b_1 and b_2 were computed as follows:

$$\int_0^\theta q_{b_1}(\theta)\,hd\theta = \int_0^{\pi/2} -\frac{S_y t_{sk}}{I_{zz}}\left(h^2\left[\cos\theta - 1\right] + \sum_i^{i\ni\theta_i<\theta} B_i h\sin\theta_i\right)d\theta$$

$$= -\frac{S_y t_{sk}}{I_{zz}}\left(h^2\left[\sin\theta - \theta\right] + \sum_i^{i\ni\theta_i<\theta} B_i h\sin\theta_i\left[\theta\_\theta_i\right]\right)$$

$$\int_h^y q_{b_2}(y)\,dy = \int_h^y -\frac{S_y t_{sp}}{2I_{zz}} y^2 dy$$

$$= -\frac{S_y t_{sp}}{6I_{zz}}\left[h^3 - y^3\right]$$

$$\int_0^{s_3} q_{b_3}\left(s_3\right)ds_3 = \int_0^{s_3}\left(-\frac{S_y t_{sk}}{I_{zz}}\left[hs_3 - \frac{h}{2l_{sk}}s_3^2 + \sum_i^{i\ni s_i<s_3} B_i y_i\right] + q_{b_1}\left(\frac{\pi}{2}\right) + q_{b_2}(h)\right)ds_3$$

$$= -\frac{S_y t_{sk}}{I_{zz}}\left[\frac{h}{2}s_3^2 - \frac{h}{6l_{sk}}s_3^3 + \sum_i^{i\ni s_i<s_3} B_i y_i\left(s_i - s_3\right)\right] + q_{b_1}\left(\frac{\pi}{2}\right)s_3 + q_{b_2}(h)\,s_3$$

Other integrals may be found similarly but are omitted here for sake of brevity.

The redundant shear flow is then easily found from Equation (3.2), and may then may be superimposed on the base shear flow distribution, taking into consideration the positive directions as shown in Figure 3.1.

3.1.3 Calculation of shear center

Finally, the shear center may be computed by equating the moment generated by the internal shear flows about the middle point of the spar to the external moment generated (taking clockwise to be positive). The moment created by the internal shear flow is equal to

$$M_i = \int_0^{\pi/2} q_1(\theta)\,h^2 d\theta + \int_0^{l_{sk}} q_3\left(s_3\right) r ds_3 + \int_0^{l_{sk}} q_4\left(s_4\right) r ds_4 + \int_{-\pi/2}^{0} q_6(\theta)\,h^2 d\theta \tag{3.7}$$

with r the perpendicular distance between the skins and the middle of the spar, which can be computed by computing

$$r = l_{sk}\sin\psi$$

where ψ has been found in Equation (2.1). The external moment equals

$$M_e = \eta\cdot -S_y = -\eta S_y \tag{3.8}$$

Thus, η may be straightforwardly found by Equation (3.7) and (3.8) (particularly when setting $S_y = 1$). Lastly, the shear center is transformed to a coordinate system that is centered at the leading edge of the aileron, by simply computing $z_{sc} = h + \eta$.

3.2 Torsional stiffness

The torsional stiffness is also computed using the concept of shear flow distributions (Figure 3.1 is again used for the sense of positive directions for the shear flows).

3.2.1 Torque distribution

When under a pure torque, only the redundant shear flows are present, and will satisfy

$$T = 2A_1 q_{0,1} + 2A_2 q_{0,2} \tag{3.9}$$

with

$$A_1 = \frac{\pi h^2}{2} \tag{3.10}$$

$$A_2 = \left(C_a - h\right)h \tag{3.11}$$

3.2.2 Rate of twist

The product of the shear modulus and rate of twist of each cell is equal to

$$G\left(\frac{d\theta}{dx}\right)_1 = A_{11}q_{0,1} + A_{12}q_{0,2} \tag{3.12}$$

$$G\left(\frac{d\theta}{dx}\right)_2 = A_{21}q_{0,1} + A_{22}q_{0,2} \tag{3.13}$$

where A_{11}, A_{12}, A_{21} and A_{22} have been found in Equations (3.3)-(3.6).

3.2.3 Torsional constant

By solving Equations (3.9)-(4.3) (setting $(Gd\theta/dx)$ equal for both cells), we obtain values for $q_{0,1}$, $q_{0,2}$ and $(Gd\theta/dx)$. We can thus compute the torsional constant from

$$J = \frac{T}{G\frac{d\theta}{dx}}$$

Chapter 4: Aerodynamic load

For the deflection calculations, it is necessary to obtain a continuous function of the aerodynamic load, i.e. the gridded data provided is converted into a continuous function $\omega(x, z)$. This is done with a linear least squares regression.

4.1 Linear least squares regression

The aerodynamic load is approximated by a linear least squares regression of the form

$$\omega(x, z) \approx \sum_{i=0}^{k_x} \sum_{j=0}^{k_z} C_{ij} X(x)^i Z(z)^j, \qquad k_x \leq N_x, k_z \leq N_z \tag{4.1}$$

with N_x and N_z described in the assignment (the number of grid points in the x- and z-direction, respectively), and where $X(x)$ and $Z(z)$ represent the transformation functions

$$X(x) = -1 + 2\frac{x - \min\left(x_t\right)}{\max\left(x_t\right) - \min\left(x_t\right)} \tag{4.2}$$

$$Z(z) = -1 + 2\frac{z - \min\left(z_t\right)}{\max\left(z_t\right) - \min\left(z_t\right)} \tag{4.3}$$

where x_t and z_t are the training coordinates (the coordinates of the grid-points on which the data is regressed). This transformation ensures that the regression happens on a $[-1, 1] \times [-1, 1]$ grid, which improves numerical stability when using large values of k_x and k_z.

Based on the given training (x_t, z_t)-coordinates, a linear regression can be straightforwardly set up and solved for. With the coefficients known, the aerodynamic load at an arbitrary value of (x, z) can be immediately computed from Equation (4.1) (the transformations in Equation (**??**) and (**??**) are in the regression function itself, after all).

Chapter 5: Deflection calculations

The deflections of the aileron are computed by applying the principle of stationary total potential energy applied to a closed-section beam subject to Euler-Bernoulli beam theory and St. Venant Torsion, using the Rayleigh-Ritz method. This method is based on finding deflection functions that minimise the total potential energy. Since you are unlikely to have much experience with this method, the explanations that follow are more elaborate than before. The chapter is structured as follows. Section 5.1 introduces the Rayleigh-Ritz method to a simple beam, subject to only one lateral deflection, no twist, and homogeneous boundary conditions. Section 5.2 extends this to include nonhomogeneous boundary conditions. Section 5.3 extends this to a lateral and twist deflections, and Section 5.4 includes the second lateral deflection.

5.1 Introduction to Rayleigh-Ritz approach

The Rayleigh-Ritz approach is a variational approach where the solution is assumed to be a finite sum of basis functions, with coefficients chosen to minimise a certain quantity. In the context of beam deflections, the Rayleigh-Ritz approach can be used when considering the principle of stationary total energy, equal to

$$\Pi = U + V$$

where U is the total strain energy and V the work potential.

5.1.1 Coordinate system

The same coordinate system as before is used: the coordinate system is centered at the leading edge of the root of the aileron, with x pointing outboard, z pointing upstream (away from the aileron, and such that the z-axis is an axis of symmetry) and y complements the right-handed coordinate system.

5.1.2 Total strain energy

The total strain energy for a one-dimensional beam with deflections in only one lateral direction (call it y) is simply equal to

$$U = \int_0^{l_a} \frac{M^2}{2EI} dx$$

However, $M^2 = (EIv'')^2$, and thus (for a constant stiffness beam)

$$U = \frac{EI}{2} \int_0^{l_a} \left(\frac{d^2 v}{dx^2} \right)^2 dx$$

where $v(\cdot)$ denotes the deflection. In order to normalise the integrals, the transformation $\xi = 2x/l_a - 1$ is applied, i.e. $dx = l_a d\xi/2$, such that

$$U = \frac{4EI}{l_a^3} \int_{-1}^{1} \left(\frac{d^2 v}{d\xi^2} \right)^2 d\xi$$

5.1.3 Work potential

The work potential is equal to

$$V = -\int_0^{l_a} q_a v \, dx$$

where $q_a(\cdot)$ is the total distributed loading. Again, normalising this to the domain $\xi \in [-1, 1]$ results in

$$V = -\frac{l_a}{2} \int_{-1}^{1} q_a v \, d\xi$$

Point forces and couple moments are included in q_a by appropriate use of Dirac delta functions, that is:

- A point force P located at $\xi = \xi_0$ has a contribution to q_a equal to $P\delta\left(\xi_0\right)$.
- A couple moment M located at $\xi = \xi_0$ has a contribution to q_a equal to $M\delta^2\left(\xi_0\right)$.

5.1.4 Principle of stationary total potential energy

The total potential energy is thus equal to

$$\Pi = U + V = \frac{4EI}{l_a^3} \int_{-1}^{1} \left(\frac{d^2 v}{d\xi^2}\right)^2 d\xi - \frac{l_a}{2} \int_{-1}^{1} q_a v \, d\xi \tag{5.1}$$

The principle of stationary total potential energy states that the total potential energy of a system is at a minimum when the system is in equilibrium. Thus, the derivative of the total potential energy with respect to any quantity p shall be zero, i.e. $\partial\Pi/\partial p = 0$. One can therefore assume the solution to be a linear combination of a finite set of basis functions, i.e.

$$v(\xi) = \sum_{i=0}^{N-1} a_i \psi_i(\xi) = \mathbf{a}^T \boldsymbol{\psi}(\xi)$$

with N the dimension of the function space of the set of basis functions, and dropping the summation sign for sake of brevity. Let $\mathbf{a}$ denote the vector containing the N coefficients, and let $\boldsymbol{\psi}$ denote the vector containing the N basis functions, such that $v = \mathbf{a}^T\boldsymbol{\psi} = \boldsymbol{\psi}^T\mathbf{a}$. Substituting this into Equation (5.1) results in

$$\Pi = U + V = \frac{4EI}{l_a^3} \int_{-1}^{1} \left(\frac{d^2 v}{d\xi^2}\right)^2 d\xi - \frac{l_a}{2} \int_{-1}^{1} q_a v \, d\xi \tag{5.1}$$

$$= \frac{4EI}{l_a^3} \int_{-1}^{1} \left(\mathbf{a}^T \frac{d^2\boldsymbol{\psi}}{d\xi^2}\right)\left(\frac{d^2\boldsymbol{\psi}^T}{d\xi^2}\mathbf{a}\right) d\xi - \frac{l_a}{2} \int_{-1}^{1} q_a \mathbf{a}^T \boldsymbol{\psi} \, d\xi \tag{5.2}$$

Now, let K_1 be the matrix equal to

$$K_1 = \int_{-1}^{1} \frac{d^2\boldsymbol{\psi}}{d\xi^2} \frac{d^2\boldsymbol{\psi}^T}{d\xi^2} d\xi \tag{5.3}$$

i.e. with entries

$$K_{1_{ij}} = \int_{-1}^{1} \frac{d^2\psi_i}{d\xi^2} \frac{d^2\psi_j}{d\xi^2} d\xi \tag{5.4}$$

Note that K_1 is symmetric.

Differentiating with respect to $\mathbf{a}$ results in

$$\frac{\partial\Pi}{\partial\mathbf{a}} = \frac{4EI}{l_a^3}\left(K_1 + K_1^T\right)\mathbf{a} - \frac{l_a}{2} \int_{-1}^{1} q_a \boldsymbol{\psi} \, d\xi = 0 \tag{5.5}$$

Thus, let K_2 be equal to

$$K_2 = \int_{-1}^{1} q_a \boldsymbol{\psi} \, d\xi \tag{5.6}$$

i.e. with entries

$$K_{2_i} = \int_{-1}^{1} q_a \psi_i \, d\xi$$

then Equation (5.5) reduces to

$$\frac{8EI}{l_a^3} K_1 \mathbf{a} = \frac{l_a}{2} K_2 \tag{5.7}$$

5.1.5 Choice of basis functions

If the boundary conditions are homogeneous, and all basis functions individually satisfy the boundary conditions, then also any linear combination of basis functions satisfy the boundary conditions. Thus, no additional steps are necessary.

However, imposing the condition that all basis functions should satisfy the boundary conditions may not be the most computationally efficient. Consider Equation (5.4); if an orthogonal set of basis functions is chosen, then K_1 evidently becomes a diagonal matrix, significantly reducing computational cost. It is therefore desirable to find a way to include basis functions that do not meet the boundary conditions. Consider a case with N_{bc} boundary conditions of the form

$$\mathcal{L}\left\{v\left(\xi_i\right)\right\} = 0, \qquad 0 \leq i < N_{bc} \tag{5.8}$$

where $\mathcal{L}$ is a differential operator, e.g. $\mathcal{L} = 1$ in case of a Dirichlet boundary condition (a simple support), or $\mathcal{L} = 2/l_a \, d/d\xi$ in case of a Neumann boundary condition (when the beam is free to move laterally in y-direction, but restrained from rotation around the z-axis). More practically speaking, we consider the following two types of boundary conditions:

- A simple support, i.e. $v(x_1) = 0$. This would result in one row of Equation (5.8) being formed by

$$\sum_{i=0}^{N} a_i \psi_i \left(\xi_1\right) = 0$$

with $\xi_1 = 2x_1/l_a - 2$.
- A first-order condition, i.e. $v'(x_2) = 0$. This would result in one row of Equation (5.8) being formed by

$$\sum_{i=0}^{N} \frac{2}{l_a} a_i \frac{d\psi_i}{d\xi} \left(\xi_2\right) = 0$$

with $\xi_2 = 2x_2/l_a - 2$.

Evidently, in case of N basis functions (with N coefficients), this leads to $N - N_{bc}$ free variables. Therefore, let $\bar{\mathbf{a}}$ denote the first N_{bc} coefficients, and $\hat{\mathbf{a}}$ denote the remaining $N - N_{bc}$ coefficients. From Equation (5.8), it is then possible to express $\bar{\mathbf{a}}$ as function of $\hat{\mathbf{a}}$, as v is a linear combination of the basis functions; write Equation (5.8) in the form

$$\Upsilon_1 \bar{\mathbf{a}} + \Upsilon_2 \hat{\mathbf{a}} = \mathbf{0} \tag{5.9}$$

where Υ_1 is a $N_{bc} \times N_{bc}$ matrix and Υ_2 a $N_{bc} \times (N - N_{bc})$ matrix, with entries given by

$$\Upsilon_{1_{ij}} = \mathcal{L}\left\{\psi_j\left(\xi_i\right)\right\}$$
$$\Upsilon_{2_{ij}} = \mathcal{L}\left\{\psi_{j+N_{bc}}\left(\xi_i\right)\right\}$$

Then, solving Equation (5.7) results in

$$\bar{\mathbf{a}} = -\Upsilon_1^{-1}\Upsilon_2\hat{\mathbf{a}} = \Upsilon\hat{\mathbf{a}} \tag{5.10}$$

where we defined $\Upsilon = -\Upsilon_1^{-1}\Upsilon_2$. Essentially, this boils down to using a_i for $i \geq N_{bc}$ to minimise the total potential energy; the remaining a_i for $i < N_{bc}$ will be used to 'fix' the solution so that it satisfies the boundary conditions. However, Equation (5.7) no longer holds; when differentiating Equation (5.2), it was implicitly assumed that all coefficients were independent; however, evidently, now $d\bar{\mathbf{a}}/d\hat{\mathbf{a}} = \Upsilon \neq 0$, so the variation of $\bar{\mathbf{a}}$ with respect to $\hat{\mathbf{a}}$ ought to be taken into account. Therefore, let us rewrite Equation (5.2) as

$$\begin{aligned}\Pi = U + V = &= \frac{4EI}{l_a^3}\int_{-1}^{1}\left(\mathbf{a}^T\frac{d^2\boldsymbol{\psi}}{d\xi^2}\right)\left(\frac{d^2\boldsymbol{\psi}^T}{d\xi^2}\mathbf{a}\right)d\xi - \frac{l_a}{2}\int_{-1}^{1} q_a\mathbf{a}^T\boldsymbol{\psi}\,d\xi \\ &= \frac{4EI}{l_a^3}\int_{-1}^{1}\left[(\bar{\mathbf{a}},\hat{\mathbf{a}})^T\frac{d^2(\bar{\boldsymbol{\psi}},\hat{\boldsymbol{\psi}})}{d\xi^2}\right]\left[\frac{d^2(\bar{\boldsymbol{\psi}},\hat{\boldsymbol{\psi}})^T}{d\xi^2}(\bar{\mathbf{a}},\hat{\mathbf{a}})\right]d\xi - \frac{l_a}{2}\int_{-1}^{1} q_a(\bar{\mathbf{a}},\hat{\mathbf{a}})^T(\bar{\boldsymbol{\psi}},\hat{\boldsymbol{\psi}})\,d\xi\end{aligned} \tag{5.2}$$

where $(\mathbf{v}_1, \mathbf{v}_2)$ implies the concatenation of two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, and where $\bar{\boldsymbol{\psi}}$ is the vector containing the first N_{bc} basis functions, and $\hat{\boldsymbol{\psi}}$ the vector containing the remaining $N - N_{bc}$ coefficients. Working out the above results in

$$\begin{aligned}\Pi = U + V =& \frac{4EI}{l_a^3}\int_{-1}^{1}\left[\left(\bar{\mathbf{a}}^T\frac{d^2\bar{\boldsymbol{\psi}}}{d\xi^2}\right)\left(\frac{d^2\bar{\boldsymbol{\psi}}^T}{d\xi^2}\bar{\mathbf{a}}\right) + \left(\bar{\mathbf{a}}^T\frac{d^2\bar{\boldsymbol{\psi}}}{d\xi^2}\right)\left(\frac{d^2\hat{\boldsymbol{\psi}}^T}{d\xi^2}\hat{\mathbf{a}}\right)\right.\\ &\left. + \left(\hat{\mathbf{a}}^T\frac{d^2\hat{\boldsymbol{\psi}}}{d\xi^2}\right)\left(\frac{d^2\bar{\boldsymbol{\psi}}^T}{d\xi^2}\bar{\mathbf{a}}\right) + \left(\hat{\mathbf{a}}^T\frac{d^2\hat{\boldsymbol{\psi}}}{d\xi^2}\right)\left(\frac{d^2\hat{\boldsymbol{\psi}}^T}{d\xi^2}\hat{\mathbf{a}}\right)\right]d\xi\end{aligned} \tag{5.11}$$

$$- \frac{l_a}{2}\int_{-1}^{1} q_a\left(\bar{\mathbf{a}}^T\bar{\boldsymbol{\psi}} + \hat{\mathbf{a}}^T\hat{\boldsymbol{\psi}}\right)d\xi \tag{5.12}$$

Akin to how K_1 is defined in Equation (5.3) and K_2 in Equation (5.6), define $K_1^{(1,1)}$, $K_1^{(1,2)}$, $K_1^{(2,2)}$, $K_2^{(1)}$, $K_2^{(2)}$ to be equal to

$$K_1^{(1,1)} = \int_{-1}^{1}\frac{d^2\bar{\boldsymbol{\psi}}}{d\xi^2}\frac{d^2\bar{\boldsymbol{\psi}}^T}{d\xi^2}d\xi$$

$$K_1^{(1,2)} = \int_{-1}^{1}\frac{d^2\bar{\boldsymbol{\psi}}}{d\xi^2}\frac{d^2\hat{\boldsymbol{\psi}}^T}{d\xi^2}d\xi$$

$$K_1^{(2,1)} = \int_{-1}^{1}\frac{d^2\hat{\boldsymbol{\psi}}}{d\xi^2}\frac{d^2\bar{\boldsymbol{\psi}}^T}{d\xi^2}d\xi$$

$$K_1^{(2,2)} = \int_{-1}^{1}\frac{d^2\hat{\boldsymbol{\psi}}}{d\xi^2}\frac{d^2\hat{\boldsymbol{\psi}}^T}{d\xi^2}d\xi$$

$$K_2^{(1)} = \int_{-1}^{1} q_a\bar{\boldsymbol{\psi}}\,d\xi$$

$$K_2^{(2)} = \int_{-1}^{1} q_a\hat{\boldsymbol{\psi}}\,d\xi$$

Note that

$$K_1 = \begin{bmatrix} K_1^{(1,1)} & K_1^{(1,2)} \\ K_1^{(2,1)} & K_2^{(2,2)} \end{bmatrix}$$
$$K_2 = \begin{bmatrix} K_2^{(1)} \\ K_2^{(2)} \end{bmatrix}$$

and note that $K_1^{(1,1)}$ and $K_2^{(2,2)}$ are both symmetric, and that $K_1^{(1,2)} = (K_1^{(2,1)})^T$. This way, Equation (5.12) reduces to

$$\Pi = U + V = \frac{4EI}{l_a^3}\left[\bar{\mathbf{a}}^T K_1^{(1,1)}\bar{\mathbf{a}} + \bar{\mathbf{a}}^T K_1^{(1,2)}\hat{\mathbf{a}} + \hat{\mathbf{a}}^T K_1^{(2,1)}\bar{\mathbf{a}} + \hat{\mathbf{a}}^T K_1^{(2,2)}\hat{\mathbf{a}}\right] - \frac{l_a}{2}\left[K_2^{(1)}\bar{\mathbf{a}} + K_2^{(2)}\hat{\mathbf{a}}\right] \tag{5.13}$$

Substituting $\bar{\mathbf{a}} = \Upsilon\hat{\mathbf{a}}$ results in

$$\begin{aligned}\Pi = U + V &= \frac{4EI}{l_a^3}\left[\hat{\mathbf{a}}^T\Upsilon^T K_1^{(1,1)}\Upsilon\hat{\mathbf{a}} + \hat{\mathbf{a}}^T\Upsilon^T K_1^{(1,2)}\hat{\mathbf{a}} + \hat{\mathbf{a}}^T K_1^{(2,1)}\Upsilon\hat{\mathbf{a}} + \hat{\mathbf{a}}^T K_1^{(2,2)}\hat{\mathbf{a}}\right] - \frac{l_a}{2}\left[K_2^{(1)}\Upsilon\hat{\mathbf{a}} + K_2^{(2)}\hat{\mathbf{a}}\right] \\ &= \frac{4EI}{l_a^3}\left[\hat{\mathbf{a}}^T\left(\Upsilon^T K_1^{(1,1)}\Upsilon + \Upsilon^T K_1^{(1,2)} + K_1^{(2,1)}\Upsilon + K_1^{(2,2)}\right)\hat{\mathbf{a}}\right] - \frac{l_a}{2}\left[K_2^{(1)}\Upsilon + K_2^{(2)}\right]\hat{\mathbf{a}}\end{aligned}$$

Differentiating now with respect to $\hat{\mathbf{a}}$ results in

$$\begin{aligned}\frac{d\Pi}{d\hat{\mathbf{a}}} =& \frac{4EI}{l_a^3}\left[\left(\Upsilon^T K_1^{(1,1)}\Upsilon + \Upsilon^T K_1^{(1,2)} + K_1^{(2,1)}\Upsilon + K_1^{(2,2)}\right)^T + \left(\Upsilon^T K_1^{(1,1)}\Upsilon + \Upsilon^T K_1^{(1,2)} + K_1^{(2,1)}\Upsilon + K_1^{(2,2)}\right)\right]\hat{\mathbf{a}} \\ &- \frac{l_a}{2}\left[K_2^{(1)}\Upsilon + K_2^{(2)}\right]\end{aligned}$$

Setting the derivative equal to 0, this results in a $N_{bc} \times N_{bc}$ system of equations:

$$\frac{8EI}{l_a^3}\left(\Upsilon^T K_1^{(1,1)}\Upsilon + \Upsilon^T K_1^{(1,2)} + K_1^{(2,1)}\Upsilon + K_1^{(2,2)}\right)\hat{\mathbf{a}} = \frac{l_a}{2}\left[K_2^{(1)}\Upsilon + K_2^{(2)}\right]$$

$\bar{\mathbf{a}}$ is then trivially computed from

$$\bar{\mathbf{a}} = \Upsilon\hat{\mathbf{a}}$$

5.2 Introducing non-homogeneous boundary conditions

The boundary conditions introduced in Equation (5.8) are homogeneous. However, consider now the more general case,

$$\mathcal{L}\left\{v\left(\xi_i\right)\right\} = f\left(\xi_i\right), \qquad 0 \le i < N_{bc} \tag{5.14}$$

In that case, Equation (5.9) may be written as

$$\Upsilon_1\bar{\mathbf{a}} + \Upsilon_2\hat{\mathbf{a}} = \mathbf{f}$$

with $\mathbf{f}$ the vector containing the boundary conditions $f(\xi_i)$. We thus obtain

$$\bar{\mathbf{a}} = -\Upsilon_1^{-1}\Upsilon_2\hat{\mathbf{a}} + \Upsilon_1^{-1}\mathbf{f} = \Upsilon\hat{\mathbf{a}} + F$$

with $\Upsilon = -\Upsilon_1^{-1}\Upsilon_2$ and $F = \Upsilon_1^{-1}\mathbf{f}$. Substituting this into Equation (5.13) results in

$$\begin{aligned}\Pi = U + V =& \frac{4EI}{l_a^3}\left[(\Upsilon\hat{\mathbf{a}} + F)^T K_1^{(1,1)}(\Upsilon\hat{\mathbf{a}} + F) + (\Upsilon\hat{\mathbf{a}} + F)^T K_1^{(1,2)}\hat{\mathbf{a}} + \hat{\mathbf{a}}^T K_1^{(2,1)}(\Upsilon\hat{\mathbf{a}} + F) + \hat{\mathbf{a}}^T K_1^{(2,2)}\hat{\mathbf{a}}\right] \\ &- \frac{l_a}{2}\left[K_2^{(1)}(\Upsilon\hat{\mathbf{a}} + F) + K_2^{(2)}\hat{\mathbf{a}}\right] \\ =& \frac{4EI}{l_a^3}\left\{\left[\hat{\mathbf{a}}^T\Upsilon^T K_1^{(1,1)}\Upsilon\hat{\mathbf{a}} + \hat{\mathbf{a}}^T\Upsilon^T K_1^{(1,2)}\hat{\mathbf{a}} + \hat{\mathbf{a}}^T K_1^{(2,1)}\Upsilon\hat{\mathbf{a}} + \hat{\mathbf{a}}^T K_1^{(2,2)}\hat{\mathbf{a}}\right]\right. \\ &+ \left.\left[F^T K_1^{(1,1)}\Upsilon\hat{\mathbf{a}} + \hat{\mathbf{a}}^T\Upsilon^T K_1^{(1,1)}F + F^T K_1^{(1,2)}\hat{\mathbf{a}} + \hat{\mathbf{a}}^T K_1^{(2,1)}F\right] + \left[F^T K_1^{(1,1)}F\right]\right\} \\ &- \frac{l_a}{2}\left\{\left(K_2^{(1)}\Upsilon + K_2^{(2)}\right)\hat{\mathbf{a}} + K_2^{(1)}F\right\}\end{aligned}$$

This can be rewritten to

$$\begin{aligned}\Pi = U + V = &\frac{4EI}{l_a^3}\left\{\left[\hat{\mathbf{a}}^T\left(\Upsilon^T K_1^{(1,1)}\Upsilon + \Upsilon^T K_1^{(1,2)} + K_1^{(2,1)}\Upsilon + K_1^{(2,2)}\right)\hat{\mathbf{a}}\right]\right. \\ &\left. + \hat{\mathbf{a}}^T\left(\Upsilon^T K_1^{(1,1)}F + K_1^{(2,1)}F\right) + \left(F^T K_1^{(1,1)}\Upsilon + F^T K_1^{(1,2)}\right)\hat{\mathbf{a}} + F^T K_1^{(1,1)}F\right\} \\ &- \frac{l_a}{2}\left\{\left(K_2^{(1)}\Upsilon + K_2^{(2)}\right)\hat{\mathbf{a}} + K_2^{(1)}F\right\}\end{aligned}$$

Differentiating with respect to $\hat{\mathbf{a}}$ results in

$$\begin{aligned}\frac{d\Pi}{d\hat{\mathbf{a}}} = &\frac{4EI}{l_a^3}\left\{\left[\left(\Upsilon^T K_1^{(1,1)}\Upsilon + \Upsilon^T K_1^{(1,2)} + K_1^{(2,1)}\Upsilon + K_1^{(2,2)}\right) + \left(\Upsilon^T K_1^{(1,1)}\Upsilon + \Upsilon^T K_1^{(1,2)} + K_1^{(2,1)}\Upsilon + K_1^{(2,2)}\right)^T\right]\hat{\mathbf{a}}\right. \\ &\left. + \left(\Upsilon^T K_1^{(1,1)}F + K_1^{(2,1)}F\right) + \left(F^T K_1^{(1,1)}\Upsilon + F^T K_1^{(1,2)}\right)^T\right\} - \frac{l_a}{2}\left[K_2^{(1)}\Upsilon + K_2^{(2)}\right]\end{aligned}$$

Combining equivalent terms results in

$$\frac{d\Pi}{d\hat{\mathbf{a}}} = \frac{8EI}{l_a^3}\left(\Upsilon^T K_1^{(1,1)}\Upsilon + \Upsilon^T K_1^{(1,2)} + K_1^{(2,1)}\Upsilon + K_1^{(2,2)}\right)\hat{\mathbf{a}} = -\frac{8EI}{l_a^3}\left(\Upsilon^T K_1^{(1,1)}F + K_1^{(2,1)}F\right) + \frac{l_a}{2}\left[K_2^{(1)}\Upsilon + K_2^{(2)}\right]$$

$\bar{\mathbf{a}}$ is then trivially computed from

$$\bar{\mathbf{a}} = \Upsilon\hat{\mathbf{a}} + F$$

5.2.1 Choice of basis functions

The set of basis functions that is chosen is a set of monomials. Although this does not take advantage of any orthogonality relations, integrating the dot products is still very straightforward. The use of actual orthogonal series is generally difficult for this kind of problem, since it is required that the *second derivative* of the series is orthogonal. This is generally difficult to accomplish, usually results in numerical difficulties elsewhere.

5.3 Inclusion of torque distribution

The problem will now be extended to also include deflections due to torque; the beam will still only be able to deflect in one lateral direction (call it the y-direction), but will be able to twist about the x-axis.

5.3.1 Total strain energy

The total strain energy is now equal to

$$U = \int_0^{l_a} \frac{M^2}{2EI}dx + \int_0^{l_a} \frac{T^2}{2GJ}dx$$

where $M^2 = (EIv'')^2$ and $T^2 = (GJ\phi')^2$ (for a constant stiffness beam), where $v(x)$ denotes the deflection of the flexural axis of the beam, and $\phi(x)$ denotes the twist around the flexural axis. Substituting these, and normalising the integrals as before, results in

$$U = \frac{4EI}{l_a^3}\int_{-1}^{1}\left(\frac{d^2v}{d\xi^2}\right)^2 d\xi + \frac{GJ}{l_a}\int_{-1}^{1}\left(\frac{d\phi}{d\xi}\right)^2 d\xi$$

5.3.2 Work potential

The work potential is now equal to

$$V = -\int_0^{l_a} q_a v\,dx - \int_0^{l_a} \tau_x\phi\,dx$$

where $\tau(\cdot)$ denotes the total distributed torque. Again, normalising this to the domain $\xi \in [-1, 1]$ results in

$$V = -\frac{l_a}{2}\int_{-1}^{1} q_a v \, d\xi - \frac{l_a}{2}\int_{-1}^{1} \tau_x \phi \, d\xi$$

$\tau_x(x)$ is the summation of the contribution of each force to the distributed torque, with the following types of forces having the following contribution:

- A distributed torque, $\tau(x)$ [Nm/m], has a contribution equal to $\tau(x)$.
- A direct torque, T [Nm], located at $x = x_0$, has a contribution equal to $T\delta(x_0)$.
- A doubly distributed load (i.e. a distributed load that is both a function of x and z), $\omega(x, z)$ [N/m^2], has a contribution equal to $\int_0^{C_a} \omega(x, z)\left(z - z_{s.c}\right) dz$, where $z_{s.c.}$ is the coordinate of the shear center.
- A distributed load, $q(x) \cdot d(x)$ [Nm/m], where $q(x)$ represents the magnitude of the distributed load, and $d(x)$ is the spanwise variation of the z-coordinate of the point application of the distributed load. Note that a distributed load with constant $d(x) = \bar{z}$ is essentially a distributed torque equal to $\tau(x) = -\bar{z}q(x)$ (note the minus sign, due to the way the coordinate system is defined).
- A point load, P [N], located at $x = x_0$ and $z = z_0$, has a contribution equal to $-Pz_0\delta(x_0)$.
- A couple moment: does not have an influence on the torque distribution.

5.3.3 Total potential energy

The total potential energy is thus equal to

$$\Pi = U + V = \frac{4EI}{l_a^3}\int_{-1}^{1}\left(\frac{d^2v}{d\xi^2}\right)^2 d\xi + \frac{GJ}{l_a}\int_{-1}^{1}\left(\frac{d\phi}{d\xi}\right)^2 d\xi - \int_0^{l_a} q_a v \, dx - \int_0^{l_a} \tau\phi \, dx \tag{5.15}$$

Application of the principle of stationary total potential energy is similar to before. First, we assume the solutions to be given by

$$v(\xi) = \sum_{i=0}^{N-1} a_i\psi_i(\xi) = \mathbf{a}^T\boldsymbol{\psi}(\xi)$$
$$\phi(\xi) = \sum_{i=0}^{N-1} c_i\psi_i(\xi) = \mathbf{c}^T\boldsymbol{\psi}(\xi)$$

where, for sake of simplicity, the same set of basis functions is used to describe the lateral and torsional deflections; the number of coefficients is also assumed equal for both deflections (in both cases N coefficients are used). Substituting into Equation (5.15) results in

$$\begin{aligned}\Pi = U + V &= \int_0^{l_a}\frac{M^2}{2EI}dx + \int_0^{l_a}\frac{T^2}{2GJ}dx - \int_0^{l_a} q_a v \, dx - \int_0^{l_a}\tau\phi \, dx \\ &= \frac{4EI}{l_a^3}\int_{-1}^{1}\left(\mathbf{a}^T\frac{d^2\boldsymbol{\psi}}{d\xi^2}\right)\left(\frac{d^2\boldsymbol{\psi}^T}{d\xi^2}\mathbf{a}\right)d\xi + \frac{GJ}{l_a}\int_{-1}^{1}\left(\mathbf{c}^T\frac{d\boldsymbol{\psi}}{d\xi}\right)\left(\frac{d\boldsymbol{\psi}^T}{d\xi}\mathbf{c}\right)d\xi \\ &\quad - \frac{l_a}{2}\int_{-1}^{1} q_a\mathbf{a}^T\boldsymbol{\psi}\, d\xi - \frac{l_a}{2}\int_{-1}^{1}\tau_x\mathbf{c}^T\boldsymbol{\psi}\, d\xi\end{aligned} \tag{5.16}$$

5.3.4 Boundary conditions

As before, we now consider the boundary conditions. Formally speaking, consider a case with N_{bc} boundary conditions of the form

$$\mathcal{L}\left\{v\left(\xi_i\right)\right\} = f_i, \qquad 0 \le i < N_{bc} \tag{5.17}$$

where $\mathcal{L}$ is a differential operator. Three types of boundary conditions will be considered:

- A simple support, e.g. $v(x_1, z_1) = f_1$. Note that this boundary condition also requires a z-argument. This is because $v(x)$ normally describes the deflection of the flexural axis. However, in case a simple support is not mounted on the flexural axis, but located at $z_1 \neq z_{sc}$ (with z_{sc} the coordinate of the shear center), then part of the lateral deflection may be caused by a non-zero twist at this spanwise station (this is for example the case for the hinges of the aileron, which are not likely to coincide with the flexural axis of the aileron). This means that the corresponding boundary condition is of the form

$$\sum_{i=0}^{N-1} a_i \psi_i \left(\xi_1\right) - \left(z_1 - z_{sc}\right) \sum_{i=0}^{N-1} c_i \psi_i \left(\xi_1\right) = f_1$$

  with $\xi_1 = 2x_1/l_a - 2$.
- A first-order support on the slope of the flexural axis, e.g. $v'(x_2) = f_2$. In this case, we impose the boundary condition strictly on the flexural axis, so there is no need for a z-argument. This boundary condition results in an equation of the form

$$\sum_{i=0}^{N-1} \frac{2}{l_a} a_i \frac{d\psi_i}{d\xi} \left(\xi_2\right) = f_2$$

  with $\xi_2 = 2x_2/l_a - 2$.
- A twist constraint, e.g. $\phi(x_3) = f_3$. Again, there is no dependence on the lateral coordinate. This boundary condition results in an equation of the form

$$\sum_{i=0}^{N-1} c_i \psi_i \left(\xi_3\right) = f_3$$

  with $\xi_3 = 2x_3/l_a - 2$.

In case of N basis functions (with N coefficients), this leads to $2N - N_{bc}$ free variables (as there are two functions that need to be approximated). Therefore, let $\bar{\mathbf{a}}$ denote the first N_a coefficients a_i; let $\hat{\mathbf{a}}$ denote the remaining $N - N_a$ a_i; let $\bar{\mathbf{c}}$ denote the first N_c coefficients c_i; let $\hat{\mathbf{c}}$ denote the remaining $N - N_c$ c_i. That is, let

$$\begin{aligned}
\bar{\mathbf{a}} &= \begin{bmatrix} a_0 & \cdots & a_{N_a - 1} \end{bmatrix} \\
\hat{\mathbf{a}} &= \begin{bmatrix} a_{N_a} & \cdots & a_N \end{bmatrix} \\
\bar{\mathbf{c}} &= \begin{bmatrix} c_0 & \cdots & c_{N_c - 1} \end{bmatrix} \\
\hat{\mathbf{c}} &= \begin{bmatrix} c_{N_x} & \cdots & c_N \end{bmatrix}
\end{aligned}$$

Choosing N_a and N_c is arbitrary to a certain degree; the only immediate constraint seems to be $N_a + N_c = N_{bc}$(and $N_a \geq 0$ and $N_c \geq 0$). However, it will soon be established that choosing N_a and N_c may require more attention, and not any combination of N_a and N_c is valid.

Similar to before, write Equation (5.17) as

$$\Upsilon_1 \bar{\mathbf{a}} + \Upsilon_2 \hat{\mathbf{a}} + \Upsilon_3 \bar{\mathbf{c}} + \Upsilon_4 \hat{\mathbf{c}} = \mathbf{0} \tag{5.18}$$

where Υ_1 is a $N_{bc} \times N_a$ matrix, Υ_2 a $N_{bc} \times (N - N_a)$ matrix, Υ_3 is a $N_{bc} \times N_c$ matrix, Υ_4 a $N_{bc} \times (N - N_c)$ matrix, with entries given by

$$\begin{aligned}
\Upsilon_{1_{ij}} &= \mathcal{L}\left\{\psi_j\left(\xi_i\right)\right\} \\
\Upsilon_{2_{ij}} &= \mathcal{L}\left\{\psi_{j+N_a)}\left(\xi_i\right)\right\} \\
\Upsilon_{3_{ij}} &= \mathcal{L}\left\{\psi_j\left(\xi_i\right)\right\} \\
\Upsilon_{4_{ij}} &= \mathcal{L}\left\{\psi_{j+N_c}\left(\xi_i\right)\right\}
\end{aligned}$$

It is now convenient to define $\bar{\boldsymbol{\alpha}} = (\bar{\mathbf{a}}, \bar{\mathbf{c}})$ and $\hat{\boldsymbol{\alpha}} = (\hat{\mathbf{a}}, \hat{\mathbf{c}})$, with $(\mathbf{v}_1, \mathbf{v}_2)$ implying the concatenation of two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Previously it was our goal to write everything in terms of $\hat{\mathbf{a}}$; it will now be our goal to write everything in terms of $\hat{\boldsymbol{\alpha}}$. We will first find expressions for $\bar{\mathbf{a}}$ and $\bar{\mathbf{c}}$. Let $\Upsilon_{13} = \begin{bmatrix}\Upsilon_1 & \Upsilon_3\end{bmatrix}$ and $\Upsilon_{24} = \begin{bmatrix}\Upsilon_2 & \Upsilon_4\end{bmatrix}$, then Equation (5.18) may be written as

$$\Upsilon_{13}\bar{\boldsymbol{\alpha}} = -\Upsilon_{24}\hat{\boldsymbol{\alpha}} + \mathbf{f}$$

leading to

$$\bar{\boldsymbol{\alpha}} = -\Upsilon_{13}^{-1}\Upsilon_{24}\hat{\boldsymbol{\alpha}} + \Upsilon_{13}^{-1}\mathbf{f} = \Upsilon\hat{\boldsymbol{\alpha}} + F$$

with $\Upsilon = -\Upsilon_{13}^{-1}\Upsilon_{24}$ and $F = \Upsilon_{13}^{-1}\mathbf{f}$. It can thus now be concluded that we require N_a and N_c to be chosen such that Υ_{13} is non-singular! Unfortunately, different sets of boundary conditions will require different N_a and N_c (even if N_{bc} is the same); for example, twist boundary conditions will require higher N_c, but clamped conditions will require a higher number of N_a. Nonetheless, there will always be at least one combination of N_a and N_c possible, given that the boundary conditions are linearly independent, and do not physically over- or underdetermine the system.

In any case, we can now write

$$\bar{\mathbf{a}} = \Upsilon_a\hat{\boldsymbol{\alpha}} + F_a \tag{5.19}$$

$$\bar{\mathbf{c}} = \Upsilon_c\hat{\boldsymbol{\alpha}} + F_c \tag{5.20}$$

with Υ_a and F_a corresponding to the first N_a rows of Υ and F, and Υ_c and F_c corresponding to the last N_c rows of Υ and F. It is now desirable to obtain $\hat{\mathbf{a}}$ and $\hat{\mathbf{c}}$ in terms of $\hat{\boldsymbol{\alpha}}$. This is a simple mapping from $\mathbb{R}^{N-N_a}$ or $\mathbb{R}^{N-N_c}$ to $\mathbb{R}^{2N}$, respectively:

$$\hat{\mathbf{a}} = H_a\hat{\boldsymbol{\alpha}} \tag{5.21}$$

$$\hat{\mathbf{c}} = H_c\hat{\boldsymbol{\alpha}} \tag{5.22}$$

with

$$H_a = \begin{bmatrix}I_{N_a} & 0_{N_a,N_c}\end{bmatrix}$$
$$H_c = \begin{bmatrix}0_{N_c,N_a} & I_{N_c}\end{bmatrix}$$

where I_n is the identity matrix of size n, and $0_{n_1,n_2}$ is the zero-matrix of size $n_1 \times n_2$.

5.3.5 Stiffness matrices

Finally, it is beneficial to introduce the stiffness matrices

$$K_{1,i} = \begin{bmatrix} K_{1,i}^{(1,1)} & K_{1,i}^{(1,2)} \\ K_{1,i}^{(2,1)} & K_{1,i}^{(2,2} \end{bmatrix} = \begin{cases} \int_{-1}^{1} \frac{d^2\boldsymbol{\psi}}{d\xi^2}\frac{d^2\boldsymbol{\psi}^T}{d\xi^2}\,d\xi, & i = a \\ \int_{-1}^{1} \frac{d\boldsymbol{\psi}}{d\xi}\frac{d\boldsymbol{\psi}^T}{d\xi}\,d\xi, & i = c \end{cases} \tag{5.23}$$

$$K_{2,i} = \begin{bmatrix} K_{2,i}^{(1)} \\ K_{2,i}^{(2)} \end{bmatrix} = \begin{bmatrix} \begin{cases} \int_{-1}^{1} q_i\psi\,d\xi, & i = a \\ \int_{-1}^{1} \tau\psi\,d\xi, & i = c \end{cases} \end{bmatrix} \tag{5.24}$$

where $K_{1,i}^{(1,1)}$ is of size $(N - N_i) \times (N - N_i)$, and $K_{2,i}^{(1)}$ is of size $N - N_i$.

5.3.6 Resulting governing equation

Then, plugging in Equation (5.19)-(5.24) into Equation (5.16), rewriting as before and differentiating eventually results in

$$\begin{aligned}\frac{d\Pi}{d\hat{\boldsymbol{\alpha}}} =& \frac{8EI}{l_a^3}\left(\Upsilon_a^T K_{1,a}^{(1,1)}\Upsilon_a + \Upsilon_a^T K_{1,a}^{(1,2)} H_a + H_a^T K_{1,a}^{(2,1)}\Upsilon_a + H_a^T K_{1,a}^{(2,2)} H_a\right)\hat{\boldsymbol{\alpha}} \\ &+ \frac{2GJ}{l_a}\left(\Upsilon_c^T K_{1,c}^{(1,1)}\Upsilon_c + \Upsilon_c^T K_{1,c}^{(1,2)} H_c + H_c^T K_{1,c}^{(2,1)}\Upsilon_c + H_c^T K_{1,c}^{(2,2)} H_c\right)\hat{\boldsymbol{\alpha}} \\ =& -\frac{8EI}{l_a^3}\left(\Upsilon_a^T K_{1,a}^{(1,1)} F_a + H_a^T K_1^{(2,1)} F_a\right) + \frac{l_a}{2}\left[K_{2,a}^{(1)}\Upsilon_a + K_{2,a}^{(2)} H_a\right] \\ &- \frac{2GJ}{l_a}\left(\Upsilon_c^T K_{1,c}^{(1,1)} F_c + H_c^T K_1^{(2,1)} F_c\right) + \frac{l_c}{2}\left[K_{2,c}^{(1)}\Upsilon_c + K_{2,c}^{(2)} H_c\right]\end{aligned}$$

which can be solved for $\hat{\boldsymbol{\alpha}}$. The remaining coefficients can be found by

$$\bar{\boldsymbol{\alpha}} = \Upsilon\hat{\boldsymbol{\alpha}} + F$$

5.4 Inclusion of second shear lateral deflection

The problem will now at last be expanded to also include deflections due to deflections in z-direction.

5.4.1 Total strain energy

The total strain energy is now equal to

$$U = \int_0^{l_a} \frac{M_z^2}{2EI_{zz}} dx + \int_0^{l_a} \frac{M_y^2}{2EI_{yy}} dx + \int_0^{l_a} \frac{T^2}{2GJ} dx$$

where $M_z^2 = (EI_{zz}v'')^2$, $M_y^2 = (EI_{yy}w'')^2$, and $T^2 = (GJ\phi')^2$ (for a constant stiffness beam), where $v(x)$ denotes the deflection of the flexural axis of the beam in y-direction, $w(x)$ denotes the deflection of the flexural axis of the beam in z-direction, and $\phi(x)$ denotes the twist around the flexural axis. Substituting these, and normalising the integrals as before, results in

$$U = \frac{4EI_{zz}}{l_a^3}\int_{-1}^{1}\left(\frac{d^2v}{d\xi^2}\right)^2 d\xi + \frac{4EI_{yy}}{l_a^3}\int_{-1}^{1}\left(\frac{d^2w}{d\xi^2}\right)^2 d\xi + \frac{GJ}{l_a}\int_{-1}^{1}\left(\frac{d\phi}{d\xi}\right)^2 d\xi$$

5.4.2 Work potential

The work potential is now equal to

$$V = -\int_0^{l_a} q_a v\, dx - \int_0^{l_a} q_b w\, dx - \int_0^{l_a} \tau_x \phi\, dx$$

where $q_a(x)$ denotes the distributed load in y-direction, $q_b(x)$ denotes the distributed load in z-direction, and $\tau_x(x)$ denotes the total distributed torque. Again, normalising this to the domain $\xi \in [-1, 1]$ results in

$$V = -\frac{l_a}{2}\int_{-1}^{1} q_a v\, d\xi - \frac{l_a}{2}\int_{-1}^{1} q_b w\, d\xi - \frac{l_a}{2}\int_{-1}^{1} \tau_x \phi\, d\xi$$

We can distinguish the following forces, each having certain contributions to q_a, q_b or τ_x:

1. A distributed torque, $\tau_i(x)$ [Nm/m]. This load has the following contributions:

$$\begin{aligned}q_{a_i}(x) &= 0 \\ q_{b_i}(x) &= 0 \\ \tau_{x_i}(x) &= \tau_i(x)\end{aligned}$$

2. A direct torque, T_i [Nm], located at $x = x_i$. This load has the following contributions:

$$q_{a_i}(x) = 0$$
$$q_{b_i}(x) = 0$$
$$\tau_{x_i}(x) = T_i \delta\left(x_i\right)$$

3. A doubly distributed load acting perpendicular to the xz-plane, $\omega(x, z)$ [N/m^2]. This load has the following contributions:

$$q_{a_i}(x) = \int_0^{C_a} \omega(x, z)\, dz$$
$$q_{b_i}(x) = 0$$
$$\tau_{x_i}(x) = -\int_0^{C_a} \omega(x, z)\left(z - z_{sc}\right)\, dz$$

4. A double distributed load acting perpendicular to the xy-plane, $\omega(x, y)$ [N/m^2]. This load has the following contributions:

$$q_{a_i}(x) = 0$$
$$q_{b_i}(x) = \int_0^{h_a} \omega(x, y)\, dy$$
$$\tau_{x_i}(x) = \int_0^{h_a} \omega(x, y)\left(y - y_{sc}\right)\, dy$$

5. A distributed load, $q(x)$ [N/m], with its point of application in the cross-section being described by $y_i(x)$ and $z_i(x)$, and its orientation within the yz-plane described by $\theta_i(x)$ (measured from the positive y-axis, with positive direction given by the right-hand rule). This load has the following contributions:

$$q_{a_i}(x) = \cos\left(\theta_i(x)\right) q(x)$$
$$q_{b_i}(x) = \sin\left(\theta_i(x)\right) q(x)$$
$$\tau_{x_i}(x) = -\cos\left(\theta_i(x)\right) q(x)\left(z_i(x) - z_{sc}\right) + \sin\left(\theta_i(x)\right) q(x)\left(y_i(x) - q_{sc}\right)$$

6. A point load, P_i [N], located at $x = x_i$, $y = y_i$ and $z = z_i$, and acting at an angle θ_i (measured from the positive y-axis, with positive direction given by the right-hand rule). This load has the following contributions:

$$q_{a_i}(x) = P_i \cos\left(\theta_i\right) \delta\left(x_i\right)$$
$$q_{b_i}(x) = P_i \sin\left(\theta_i\right) \delta\left(x_i\right)$$
$$\tau_{x_i}(x) = -P_i \cos\left(\theta_i\right)\left(z_i - z_{sc}\right) \delta\left(x_i\right) + P_i \sin\left(\theta_i\right)\left(y_i - y_{sc}\right) \delta\left(x_i\right)$$

7. A couple moment, M_i [Nm], located at $x = x_i$, and acting at an angle θ_i (measured from the positive y-axis, with positive direction given by the right-hand rule). This load has the following contributions:

$$q_{a_i}(x) = M_i \sin\left(\theta_i\right) \delta^2\left(x_i\right)$$
$$q_{b_i}(x) = -M_i \cos\left(\theta_i\right) \delta^2\left(x_i\right)$$
$$\tau_{x_i}(x) = 0$$

5.4.3 Total potential energy

The total potential energy is thus equal to

$$\Pi = U + V = \frac{4EI_{zz}}{l_a^3}\int_{-1}^{1}\left(\frac{d^2 v}{d\xi^2}\right)^2 d\xi + \frac{4EI_{yy}}{l_a^3}\int_{-1}^{1}\left(\frac{d^2 w}{d\xi^2}\right)^2 d\xi + \frac{GJ}{l_a}\int_{-1}^{1}\left(\frac{d\phi}{d\xi}\right)^2 d\xi$$
$$- \int_0^{l_a} q_a v \, dx - \int_0^{l_a} q_b w \, dx - \int_0^{l_a} \tau_x \phi \, dx \tag{5.25}$$

Application of the principle of stationary total potential energy is similar to before. First, we assume the solutions to be given by

$$v(\xi) = \sum_{i=0}^{N-1} a_i \psi_i(\xi) = \mathbf{a}^T \boldsymbol{\psi}(\xi)$$
$$w(\xi) = \sum_{i=0}^{N-1} b_i \psi_i(\xi) = \mathbf{b}^T \boldsymbol{\psi}(\xi)$$
$$\phi(\xi) = \sum_{i=0}^{N-1} c_i \psi_i(\xi) = \mathbf{c}^T \boldsymbol{\psi}(\xi)$$

where, for sake of simplicity, the same set of basis functions is used to describe the lateral and torsional deflections; the number of coefficients is also assumed equal for all deflections (in all three cases N coefficients are used). Substituting into Equation (5.25) results in

$$\Pi = U + V = \frac{4EI_{zz}}{l_a^3}\int_{-1}^{1}\left(\mathbf{a}^T \frac{d^2\boldsymbol{\psi}}{d\xi^2}\right)\left(\frac{d^2\boldsymbol{\psi}^T}{d\xi^2}\mathbf{a}\right) d\xi + \frac{4EI_{yy}}{l_a^3}\int_{-1}^{1}\left(\mathbf{b}^T \frac{d^2\boldsymbol{\psi}}{d\xi^2}\right)\left(\frac{d^2\boldsymbol{\psi}^T}{d\xi^2}\mathbf{b}\right) d\xi$$
$$+ \frac{GJ}{l_a}\int_{-1}^{1}\left(\mathbf{c}^T \frac{d\boldsymbol{\psi}}{d\xi}\right)\left(\frac{d\boldsymbol{\psi}^T}{d\xi}\mathbf{c}\right) d\xi$$
$$- \frac{l_a}{2}\int_{-1}^{1} q_a \mathbf{a}^T \boldsymbol{\psi} \, d\xi - \frac{l_a}{2}\int_{-1}^{1} q_b \mathbf{b}^T \boldsymbol{\psi} \, d\xi - \frac{l_a}{2}\int_{-1}^{1} \tau_x \mathbf{c}^T \boldsymbol{\psi} \, d\xi \tag{5.26}$$

5.4.4 Boundary conditions

As before, we now consider the boundary conditions. Formally speaking, consider a case with N_{bc} boundary conditions of the form

$$\mathcal{L}\left\{v\left(\xi_i\right)\right\} = f_i, \qquad 0 \leq i < N_{bc} \tag{5.27}$$

where $\mathcal{L}$ is a differential operator. Five types of boundary conditions will be considered:

- A simple support in the y-direction, e.g. $v\left(x_1, y_1, z_1\right) = f_1$. Note that this boundary condition requires both a y- and z-argument, as the simple support is applied at an arbitrary position $(y, z) = (y_1, z_1)$ in the cross-section. Again, this is because $v(x)$ normally describes the deflection of the flexural axis. However, in case a simple support is not mounted on the flexural axis, but located at $z_1 \neq z_{sc}$ and $y_1 \neq y_{sc}$ (with (y_{sc}, z_{sc}) denoting the coordinates of the shear center), then part of the lateral deflection may be caused by a non-zero twist at this spanwise station (this is for example the case for the hinges of the aileron, which are not likely to coincide with the flexural axis of the aileron). This means that the corresponding boundary condition is of the form

$$\sum_{i=0}^{N-1} a_i \psi\left(\xi_1\right) - \left(z_1 - z_{sc}\right)\sum_{i=0}^{N-1} c_i \psi_i\left(\xi_1\right) = f_1$$

with $\xi_1 = 2x_1/l_a - 2$.

- A simple support in the z-direction, e.g. $w\left(x_2, y_2, z_2\right) = f_2$. This is similar to the one above, except that it restricts the w-deflection instead of the v-deflection. The corresponding boundary condition is of the form

$$\sum_{i=0}^{N-1} b_i \psi\left(\xi_2\right) + \left(y_1 - y_{sc}\right) \sum_{i=0}^{N-1} c_i \psi_i\left(\xi_2\right) = f_2$$

  with $\xi_2 = 2x_2/l_a - 2$.
- A simple support in arbitrary direction, with its orientation within the yz-plane described by θ_i (measured from the positive y-axis, with positive direction given by the right-hand rule), e.g. $\cos\left(\theta_3\right) v\left(x_3, y_3, z_3\right) + \sin\left(\theta_3\right) w\left(x_3, y_3, z_3\right) = f_3$. This is a generalisation of the previous two boundary conditions, and restricts the deflection in an arbitrary direction (rather than simply the y- or z-direction). The corresponding boundary condition is of the form

$$\begin{aligned}&\cos\left(\theta_3\right) \sum_{i=0}^{N-1} a_i \psi\left(\xi_3\right) + \sin\left(\theta_3\right) \sum_{i=0}^{N-1} b_i \psi\left(\xi_3\right)\\ &- \cos\left(\theta_3\right)\left(z_3 - z_{sc}\right) \sum_{i=0}^{N-1} c_i \psi_i\left(\xi_3\right) + \sin\left(\theta_3\right)\left(y_3 - y_{sc}\right) \sum_{i=0}^{N-1} c_i \psi_i\left(\xi_2\right) = f_3\end{aligned}$$

  with $\xi_3 = 2x_3/l_a - 2$.
- A first-order support on the slope of the flexural axis around the z-axis, e.g. $v'\left(x_4\right) = f_4$. In this case, we impose the boundary condition strictly on the flexural axis, so there is no need for a z-argument. This boundary condition results in an equation of the form

$$\sum_{i=0}^{N-1} \frac{2}{l_a} a_i \frac{d\psi_i}{d\xi}\left(\xi_4\right) = f_4$$

  with $\xi_4 = 2x_4/l_a - 2$.
- A first-order support on the slope of the flexural axis around the y-axis, e.g. $-w'\left(x_5\right) = f_5$. Note the minus sign: applying the right-hand rule around the y-axis results in a negative slope of w being defined as positive. This boundary condition results in an equation of the form

$$-\sum_{i=0}^{N-1} \frac{2}{l_a} b_i \frac{d\psi_i}{d\xi}\left(\xi_5\right) = f_5$$

  with $\xi_5 = 2x_5/l_a - 2$.
- A first-order support on the slope of the flexural axis around an arbitrary axis, with its orientation within the yz-plane described by θ_i (measured from the positive y-axis, with positive direction given by the right-hand rule), e.g. $-\cos\left(\theta_6\right) w'\left(x_6\right) + \sin\left(\theta_6\right) v'\left(x_6\right) = f_6$. This boundary condition results in an equation of the form

$$-\cos\left(\theta_6\right) \sum_{i=0}^{N-1} \frac{2}{l_a} b_i \frac{d\psi_i}{d\xi}\left(\xi_6\right) + \sin\left(\theta_6\right) \sum_{i=0}^{N-1} \frac{2}{l_a} a_i \frac{d\psi_i}{d\xi}\left(\xi_6\right) = f_6$$

  with $\xi_6 = 2\xi_6/l_a - 2$.
- A twist constraint, e.g. $\phi(x_7) = f_7$. Again, there is no dependence on the lateral coordinate. This boundary condition results in an equation of the form

$$\sum_{i=0}^{N-1} c_i \psi_i\left(\xi_7\right) = f_7$$

  with $\xi_7 = 2x_7/l_a - 2$.

In case of N basis functions (with N coefficients), this leads to $3N - N_{bc}$ free variables (as there are three functions that need to be approximated). Therefore, let $\bar{\mathbf{a}}$ denote the first N_a coefficients a_i, let $\hat{\mathbf{a}}$ denote the remaining $N - N_a$ a_i; let $\bar{\mathbf{b}}$ denote the first N_b coefficients b_i, let $\hat{\mathbf{b}}$ denote the remaining $N - N_b$ b_i; let $\bar{\mathbf{c}}$ denote the first N_c coefficients c_i, let $\hat{\mathbf{c}}$ denote the remaining $N - N_c$ c_i. That is, let

$$
\begin{aligned}
\bar{\mathbf{a}} &= \begin{bmatrix} a_0 & \cdots & a_{N_a-1} \end{bmatrix} \\
\hat{\mathbf{a}} &= \begin{bmatrix} a_{N_a} & \cdots & a_N \end{bmatrix} \\
\bar{\mathbf{b}} &= \begin{bmatrix} b_0 & \cdots & b_{N_b-1} \end{bmatrix} \\
\hat{\mathbf{b}} &= \begin{bmatrix} b_{N_b} & \cdots & b_N \end{bmatrix} \\
\bar{\mathbf{c}} &= \begin{bmatrix} c_0 & \cdots & c_{N_c-1} \end{bmatrix} \\
\hat{\mathbf{c}} &= \begin{bmatrix} c_{N_x} & \cdots & c_N \end{bmatrix}
\end{aligned}
$$

Choosing N_a, N_b and N_c is arbitrary to a certain degree; the only immediate constraint seems to be $N_a + N_b + N_c = N_{bc}$(and $N_a \geq 0$, $N_b \geq 0$, and $N_c \geq 0$). However, it will soon be established that choosing N_a, N_b and N_c may require more attention, and not any combination of N_a, N_b and N_c is valid.

Similar to before, write Equation (5.27) as

$$\Upsilon_1\bar{\mathbf{a}} + \Upsilon_2\hat{\mathbf{a}} + \Upsilon_3\bar{\mathbf{b}} + \Upsilon_4\hat{\mathbf{b}} + \Upsilon_5\bar{\mathbf{c}} + \Upsilon_6\hat{\mathbf{c}} = \mathbf{0} \tag{5.28}$$

where Υ_1 is a $N_{bc} \times N_a$ matrix, Υ_2 a $N_{bc} \times (N - N_a)$ matrix, Υ_3 is a $N_{bc} \times N_b$ matrix, Υ_4 is a $N_{bc} \times (N - N_b)$ matrix, Υ_5 is a $N_{bc} \times N_c$ matrix, Υ_6 is a $N_{bc} \times (N - N_c)$ matrix, with entries given by

$$
\begin{aligned}
\Upsilon_{1_{ij}} &= \mathcal{L}\left\{\psi_j\left(\xi_i\right)\right\} \\
\Upsilon_{2_{ij}} &= \mathcal{L}\left\{\psi_{j+N_a}\left(\xi_i\right)\right\} \\
\Upsilon_{3_{ij}} &= \mathcal{L}\left\{\psi_j\left(\xi_i\right)\right\} \\
\Upsilon_{4_{ij}} &= \mathcal{L}\left\{\psi_{j+N_b}\left(\xi_i\right)\right\} \\
\Upsilon_{5_{ij}} &= \mathcal{L}\left\{\psi_j\left(\xi_i\right)\right\} \\
\Upsilon_{6_{ij}} &= \mathcal{L}\left\{\psi_{j+N_c}\left(\xi_i\right)\right\}
\end{aligned}
$$

It is now convenient to define $\bar{\boldsymbol{\alpha}} = (\bar{\mathbf{a}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})$ and $\hat{\boldsymbol{\alpha}} = (\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{c}})$, with $(\mathbf{v}_1, \mathbf{v}_2)$ implying the concatenation of two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Previously it was our goal to write everything in terms of $\hat{\mathbf{a}}$; it will now be our goal to write everything in terms of $\hat{\boldsymbol{\alpha}}$. We will first find expressions for $\bar{\mathbf{a}}$, $\bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$. Let $\Upsilon_{135} = \begin{bmatrix}\Upsilon_1 & \Upsilon_3 & \Upsilon_5\end{bmatrix}$ and $\Upsilon_{246} = \begin{bmatrix}\Upsilon_2 & \Upsilon_4 & \Upsilon_6\end{bmatrix}$, then Equation (5.18) may be written as

$$\Upsilon_{135}\bar{\boldsymbol{\alpha}} = -\Upsilon_{246}\hat{\boldsymbol{\alpha}} + \mathbf{f}$$

leading to

$$\bar{\boldsymbol{\alpha}} = -\Upsilon_{135}^{-1}\Upsilon_{246}\hat{\boldsymbol{\alpha}} + \Upsilon_{135}^{-1}\mathbf{f} = \Upsilon\hat{\boldsymbol{\alpha}} + F$$

with $\Upsilon = -\Upsilon_{135}^{-1}\Upsilon_{246}$ and $F = \Upsilon_{135}^{-1}\mathbf{f}$. It can thus now be concluded that we require N_a, N_b and N_c to be chosen such that Υ_{135} is non-singular! Unfortunately, different sets of boundary conditions will require different N_a, N_b and N_c (even if N_{bc} is the same); for example, twist boundary conditions will require higher N_c, but clamped conditions will require a higher number of N_a or N_b. Nonetheless, there will always be at least one combination of N_a, N_b and N_c possible, given that the boundary conditions are linearly independent, and do not physically over- or underdetermine the system.

In any case, we can now write

$$\bar{\mathbf{a}} = \Upsilon_a\hat{\boldsymbol{\alpha}} + F_a \tag{5.29}$$

$$\bar{\mathbf{a}} = \Upsilon_b\hat{\boldsymbol{\alpha}} + F_b \tag{5.30}$$

$$\bar{\mathbf{c}} = \Upsilon_c\hat{\boldsymbol{\alpha}} + F_c \tag{5.31}$$

with Υ_a and F_a corresponding to the first N_a rows of Υ and F; Υ_c and F_c corresponding to the last N_c rows of Υ and F; and Υ_b and F_b corresponding to the remaining N_b rows of Υ and F. It is now desirable to obtain $\hat{\mathbf{a}}$, $\hat{\mathbf{b}}$

and $\hat{\mathbf{c}}$ in terms of $\hat{\boldsymbol{\alpha}}$. This is a simple mapping from $\mathbb{R}^{N_a}$, $\mathbb{R}^{N_b}$ or $\mathbb{R}^{N_c}$ to $\mathbb{R}^{3N-N_{bc}}$, respectively:

$$\hat{\mathbf{a}} = H_a\hat{\boldsymbol{\alpha}} \tag{5.32}$$

$$\hat{\mathbf{b}} = H_b\hat{\boldsymbol{\alpha}} \tag{5.33}$$

$$\hat{\mathbf{c}} = H_c\hat{\boldsymbol{\alpha}} \tag{5.34}$$

with

$$\begin{aligned} H_a &= \begin{bmatrix} I_{N_a} & 0_{N_a,N_b} & 0_{N_a,N_c} \end{bmatrix} \\ H_a &= \begin{bmatrix} 0_{N_b,N_a} & I_{N_b} & 0_{N_b,N_c} \end{bmatrix} \\ H_c &= \begin{bmatrix} 0_{N_c,N_a} & 0_{N_c,N_b} & I_{N_c} \end{bmatrix} \end{aligned}$$

where I_n is the identity matrix of size n, and $0_{n_1,n_2}$ is the zero-matrix of size $n_1 \times n_2$.

Finally, it is beneficial to introduce the stiffness matrices

$$K_{1,i} = \begin{bmatrix} K_{1,i}^{(1,1)} & K_{1,i}^{(1,2)} \\ K_{1,i}^{(2,1)} & K_{1,i}^{(2,2} \end{bmatrix} = \begin{cases} \int_{-1}^{1} \frac{d^2\boldsymbol{\psi}}{d\xi^2}\frac{d^2\boldsymbol{\psi}^T}{d\xi^2} d\xi, & i = a \cup i = b \\ \int_{-1}^{1} \frac{d\boldsymbol{\psi}}{d\xi}\frac{d\boldsymbol{\psi}^T}{d\xi} d\xi, & i = c \end{cases} \tag{5.35}$$

$$K_{2,i} = \begin{bmatrix} K_{2,i}^{(1)} \\ K_{2,i}^{(2)} \end{bmatrix} = \begin{cases} \int_{-1}^{1} q_i\boldsymbol{\psi}\, d\xi, & i = a \cup i = b \\ \int_{-1}^{1} \tau_x\boldsymbol{\psi}\, d\xi, & i = c \end{cases} \tag{5.36}$$

where $K_{1,i}^{(1,1)}$ is of size $(N - N_i) \times (N - N_i)$, and $K_{2,i}^{(1)}$ is of size $N - N_i$.

5.4.5 Resulting governing equation

Then, plugging in Equation (5.29)-(5.36) into Equation (5.26), rewriting as before and differentiating eventually results in

$$\begin{aligned} \frac{d\Pi}{d\hat{\boldsymbol{\alpha}}} =& \frac{8EI_{zz}}{l_a^3}\left(\Upsilon_a^T K_{1,a}^{(1,1)}\Upsilon_a + \Upsilon_a^T K_{1,a}^{(1,2)}H_a + H_a^T K_{1,a}^{(2,1)}\Upsilon_a + H_a^T K_{1,a}^{(2,2)}H_a\right)\hat{\boldsymbol{\alpha}} \\ &+ \frac{8EI_{yy}}{l_a^3}\left(\Upsilon_b^T K_{1,b}^{(1,1)}\Upsilon_b + \Upsilon_b^T K_{1,b}^{(1,2)}H_b + H_b^T K_{1,b}^{(2,1)}\Upsilon_b + H_b^T K_{1,b}^{(2,2)}H_b\right)\hat{\boldsymbol{\alpha}} \\ &+ \frac{2GJ}{l_a}\left(\Upsilon_c^T K_{1,c}^{(1,1)}\Upsilon_c + \Upsilon_c^T K_{1,c}^{(1,2)}H_c + H_c^T K_{1,c}^{(2,1)}\Upsilon_c + H_c^T K_{1,c}^{(2,2)}H_c\right)\hat{\boldsymbol{\alpha}} \\ =& -\frac{8EI_{zz}}{l_a^3}\left(\Upsilon_a^T K_{1,a}^{(1,1)}F_a + H_a^T K_1^{(2,1)}F_a\right) + \frac{l_a}{2}\left[K_{2,a}^{(1)}\Upsilon_a + K_{2,a}^{(2)}H_a\right] \\ &- \frac{8EI_{yy}}{l_a^3}\left(\Upsilon_b^T K_{1,b}^{(1,1)}F_b + H_b^T K_1^{(2,1)}F_b\right) + \frac{l_a}{2}\left[K_{2,b}^{(1)}\Upsilon_b + K_{2,b}^{(2)}H_b\right] \\ &- \frac{2GJ}{l_a}\left(\Upsilon_c^T K_{1,c}^{(1,1)}F_c + H_c^T K_1^{(2,1)}F_c\right) + \frac{l_c}{2}\left[K_{2,c}^{(1)}\Upsilon_c + K_{2,c}^{(2)}H_c\right] \end{aligned} \tag{5.37}$$

which can be solved for $\hat{\boldsymbol{\alpha}}$. The remaining coefficients can be found by

$$\bar{\boldsymbol{\alpha}} = \Upsilon\hat{\boldsymbol{\alpha}} + F$$

5.5 Sign conventions

The chosen coordinate system is the same as the aileron coordinate system: x points outboards, z points upstream (away from the aileron), and y points upwards (i.e., a right-handed coordinate system), with its origin located at the leading edge of the inboard end. The right-hand rule is used to determine the positive direction for twist and torque; v and w and all forces are positive in the positive direction of y and z, respectively. For an applied moment, the positive direction is governed by the right-hand rule.

Torque, distributed torque, moment, shear and distributed force relations are found as follows:

$$
\begin{aligned}
T(x) &= GJ\frac{d\phi}{dx} \\
\tau(x) &= GJ\frac{d^2\phi}{dx^2} \\
v'(x) &= \frac{dv}{dx} \\
M_z(x) &= -EI_{yy}\frac{d^2v}{dx^2} \\
S_y(x) &= -EI_{yy}\frac{d^3v}{dx^2} \\
M_y(x) &= -EI_{zz}\frac{d^2w}{dx^2} \\
S_z(x) &= -EI_{zz}\frac{d^3w}{dx^2}
\end{aligned}
$$

Chapter 6: Stress calculations

With the loading on the aileron obtained, the stress distribution may be obtained. This involves the following steps:

1. Computing the shear stress distribution:
   (a) Computing the shear flow distribution due to the vertical shear force, S_y.
   (b) Computing the shear flow distribution due to the horizontal shear force, S_z.
   (c) Computing the shear flow distribution due to the torque, T.
   (d) Summing the corresponding shear flow distributions and dividing the shear flow by the thickness.
2. Computing the direct stress distribution:
   (a) Computing the direct stress distribution due to the bending moment M_z.
   (b) Computing the direct stress distribution due to the bending moment M_y.
   (c) Summing the direct stress distributions.
3. Computing the Von Mises stresses.

6.1 Shear flow distribution

The shear flow distribution for a unit vertical shear force and unit torque has conveniently already been computed in Section 3.1 and 3.2, respectively. The shear flow distribution for a unit horizontal shear force may be obtained similarly to how the shear flow distribution for a unit vertical shear force was obtained. Use is made of the same sketch of Figure 3.1, and the governing equation behind the base shear flow distribution that will be used is

$$q_b(s) = \frac{-1}{I_{yy}}\left(\int_0^s tz\,ds + \sum_i^{i \ni s_i < s} B_i z_i\right) + q_{b_0} \tag{6.1}$$

where q_{b_0} represents the base shear flow of any previous walls, B_i the stringer area, and where the limit of the summation means that all booms should be included for which it holds that $s > s_i$). The following particularities hold for each region ($\bar{z}$ is the z-coordinate of the centroid, expressed in a coordinate frame centered at the leading edge of the aileron):

- For region (1) and (6), the substitution $ds = h\,d\theta$ and $z = -(1-\cos\theta)h - \bar{z}$ is made, using the θ defined in Figure 3.1. Integration runs between $\theta = 0$ and $\theta \leq \pi/2$ for region (1), and between $\theta = -\pi/2$ and $\theta \leq 0$ for region (6). Furthermore, for region (6), $q_{b_0} = q_{b_4}\left(s_4 = l_{sk}\right) - q_{b_5}(y = -h)$.
- For region (2) and (5), the substitution $ds = dy$ is made, and $z = -h - \bar{z}$. Integration runs between $y = 0$ and $y \leq h$ for region (2), and between 0 and $y \leq -h$ for region (5).
- For region (3), the substitution $z = (-h - \bar{z}) - (C_a - h)/l_{sk}s$ is made. Integration runs between $s = 0$ and $s \leq l_{sk}$, where l_{sk} is the length of the skin. Furthermore, $q_{b_0} = q_{b_1}(\theta = \pi/2) + q_{b_2}(y = h)$.
- For region (4), the substitution $z = (-C_a - \bar{z}) + (C_a - h)/l_{sk}s$ is made. Integration runs between $s = 0$ and $s \leq l_{sk}$. Furthermore, $q_{b_0} = q_{b_3}\left(s = l_{sk}\right)$.

The integral in Equation (6.1) is then easily computed in all cases. It is noted that the redundant shear flow will be equal due to 0 (since the stringer at the leading edge was symmetrically split over regions (1) and (6)), thus bypassing a significant chunk of the calculations.

The resulting shear flow distribution can be found by simply taking a linear combination of the unit shear flow distributions, weighted by the applied loading.

6.2 Direct stress distribution

The direct stress distribution is again found by taking a linear combination of the unit direct stress distributions, one due to the moment M_y, and due to M_z. For M_y, the contribution at an arbitrary location on the aileron is simply computed by

$$\sigma_{xx}(z) = M_y \frac{z - \bar{z}}{I_{yy}} \tag{6.2}$$

For M_z, the contribution is equal to

$$\sigma_{xx}(y) = M_z \frac{y}{I_{zz}} \tag{6.3}$$

Thus, by taking a linear combination of these two stress distributions, the total stress distribution is easily found.

6.3 Von Mises stress distribution

The Von Mises stress distribution is found by first computing the shear stress distribution τ_{yz} by dividing the shear flow by the local thickness, and then computing

$$\sigma_{vm} = \sqrt{\frac{\left(\sigma_{xx} - \sigma_{yy}\right)^2 + \left(\sigma_{yy} - \sigma_{zz}\right)^2 + \left(\sigma_{zz} - \sigma_{xx}\right)^2}{2} + 3\left(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{xz}^2\right)}$$

Chapter 7: Program description

The program consists of a main file called `main.py`, and three `.pyd` files, which contain compiled libraries of the source code. The source code has been compiled with `Cython`; since `Cython` produces platform specific machine code, it depends on your platform which version you should use:

- If you have a Windows system with Python 3.6, 32 bits: use the `*.cp36-win32.pyd` files.
- If you have a Windows system with Python 3.6, 64 bits: use the `*.cp36-win_amd64.pyd` files.
- If you have a Windows system with Python 3.7, 32 bits: use the `*.cp37-win32.pyd` files.
- If you have a Windows system with Python 3.7, 64 bits: use the `*.cp37-win_amd64.pyd` files.
- If you have a Windows system with Python 3.8, 32 bits: use the `*.cp38-win32.pyd` files.
- If you have a Windows system with Python 3.8, 64 bits: use the `*.cp38-win_amd64.pyd` files.

No support is provided for Mac OS or Linux systems. Please be aware that it matters whether the Python version is 64 bits or 32 bits, not whether your Windows system is 64 bits or 32 bits.

`main.py` and the `*.pyd` files should all be placed in the same folder. Do *not* change the name of the `*.pyd` files (although `main.py` may be renamed).

7.1 Program

Since the modules have been compiled to machine code, you do not have access to source code of the modules. Nonetheless, you are given access to nearly all important (intermediary) results by use of the `main.py` file. The program is divided into five parts.

Unless otherwise noted, the coordinate system used in the calculations is centered in the leading edge of the aileron, rotated such that the z-axis is a symmetry axis.

Note that the program is relatively flexible and allows for a moderately wide variety of beam configurations, and e.g. the parts below describe how you can modify the loading of the aileron yourself. This should aid you in doing verification of the developer model, independent of your own numerical model.

7.1.1 Part I

Part I contains the parameters as they appear in the assignment; the required units are written in a comment next to each line. You may modify the cross-section of the wing box as desired, although subject to the following constraints:

- `Ca` should always remain larger than `ha/2`.
- `la` should always be positive.
- `x1`, `x2`, `x3` should all be between `0` and `la`.
- `xa` should be such that `x2+xa/2` and `x2-xa/2` are both between `0` and `la`.
- `ha` should always be positive.
- `tsk` can be set equal to 0, but the shear center, torsional stiffness and stress distribution calculations will no longer work (the bending stiffness calculations will still work), since the aileron is suddenly an open section.
- `tsp` may be set equal to 0; the program will automatically consider the aileron to be a single cell.
- `tst` may be set equal to 0; this will remove all stringers from the program (contrary to setting `nst` equal to 0, which causes an error in the program).
- `hst` may be set equal to 0.
- `wst` may be set equal to 0.
- `nst` must be an odd number, and must be chosen such that there are at least three stringers on the semi-circular arc of the aileron.
- `d1`, `d3`, `theta` and `P` may be set to any value (also negative values are allowed).

The initial parameters are set according to the A320 aircraft, so obviously you should change them if that is not your aircraft. Note that the file-name of the aerodynamic loading needs to be provided in the first line of part I.

7.1.2 Part II

Part II computes the bending properties of the aileron. First, a `Crosssection` object (from the `Stiffness`-module) is defined. This simply takes the previously defined cross-sectional parameters of the ailerons and defines a `Crosssection` object.

Executing `crosssection.compute_bendingproperties()` performs all calculations necessary to compute the moment of inertia of the aileron. `crosssection.plot_crosssection()` will provide an overview of the cross-section; red crosses indicate locations of the stringers; the blue cross indicates the location of the centroid.

A number of important results can be readily accessed. `.stcoord` returns a 2D-array of the coordinates of the stringers; the first column contains the z-coordinates, the second column the y-coordinates. Stringers are ordered in clockwise direction, starting from the stringer at the leading edge. `.totarea` returns the total area of the cross-section (in m^2). `.yc` and `.zc` contain the y- and z-coordinate of the centroid respectively (both in m). `.Iyy` and `.Izz` contain I_{yy} and I_{zz} respectively (both in m^4). It should be noted that these can be manually overwritten, to make verification of subsequent parts of the program more convenient. Nonetheless, it is self-evident that altering `.zc`, `.Iyy` or `.Izz` will automatically render the stress calculations inaccurate.

7.1.3 Part III

Part III computes the torsional properties of the aileron. First, `crosssection.compute_shearcenter()` computes the shear center; `crosssection.compute_torsionalstiffness()` computes the torsional constant J. These results can be accessed in `.ysc`, `.zsc` and `.J`; again, these values may be overwritten if desired.

7.1.4 Part IV

Part IV is the most involved part of `main.py`, and takes care of the computation of the deflection profiles and the loading diagrams of the aileron. The first step is to define three additional parameters: `N`, which is the number of basis functions that will be used for each of the deflections, `E`, which is the elastic modulus, and `G`, which is the shear modulus. `Energy.Beam(la,crosssection,N,E,G)` will then create an aileron with length `la`, cross-sectional properties taken from `crosssection`, `N` basis functions for each deflection, and material properties `E` and `G`.

The initial `N` is set to 15. It is up to you to judge whether this `N` is sufficiently high to allow for a meaningful comparison with your numerical model, and if not, how much `N` should be increased.

Boundary conditions It is then necessary to introduce the boundary conditions on the aileron. There are three functions that can be used to create a boundary condition (each corresponding to a different type of boundary condition):

- `.addbcss(x,y,z,theta,f)` introduces a simply-supported boundary condition. `x` denotes the spanwise coordinate of the point of application. However, as explained in Section 5.4.4, it is also necessary to prescribe the point of application within the cross-section; this is done by the arguments `y` and `z`. The direction of the simply-supported boundary condition is given by `theta`: `theta = 0` results in imposing a boundary condition in (positive) y-direction; `theta = m.pi/2` results in a boundary condition in (positive) z-direction. Finally, `f` denotes the value of the nonhomogeneous boundary condition (e.g. setting it equal to `f = 0.01` implies that this boundary is deflected 0.01 m); setting `f = 0` results in a homogeneous boundary condition.
- `.addbcfo(x,theta,f)` introduces a boundary condition on the slope of the flexural axis. `x` again denotes the spanwise coordinate of this boundary condition. As this boundary condition can only be applied on the flexural axis, there are no `y` and `z` arguments. `theta` controls the direction of the boundary condition: `theta = 0` corresponds to setting a condition on the slope of the deflection in y; `theta = m.pi/2` corresponds to a condition on the slope of the deflection in z. Finally, `f` denotes the value of the nonhomogeneous boundary condition.
- `.addbct(x,f)` introduces a twist boundary condition. `x` denotes the spanwise coordinate of this boundary condition, and `f` sets the value of the boundary condition.

All boundary conditions need to be given in the physical coordinate system (not the transformed coordinate system). In order to clarify the above, the list of boundary conditions that is included in the original version of `main.py` are the boundary conditions that are actually applied on the aileron. This should give an understanding of how these functions work.

Note that boundary conditions that over- or underdetermine the system will result in the program not functioning. The program does not check whether the system is over- or underdetermined, so if you input a set of boundary conditions and the program returns an error, you have to verify yourself that the boundary conditions form a proper set of boundary conditions.

External loads The external loads also need to be explicitly declared. There are seven types of boundary conditions that can be imposed, which are analogous to the list of types of external forces listed in Section 5.4.2:

1. `.addfdistt(x1,x2,f)` introduces a distributed torque. This distributed torque corresponds to a function `f(x)` and runs between `x1` and `x2` (with `x2>x1`, and with positive direction given by right-hand rule around x-axis)..
2. `.addfdirectt(x,T)` introduces a direct torque with magnitude `T` at spanwise coordinate `x` (with positive direction given by right-hand rule around x-axis).
3. `.addfddxz(x1,x2,z1,z2,f)` introduces a distributed torque `f(x,z)` that acts perpendicular to the xz-plane. Spanwise, its starting and ending position are denoted by `x1` and `x2`, and `z1` and `z2` denote the cross-sectional starting and ending position. Positive values of `f(x,z)` mean that the force points in the positive y-direction.
4. `.addfddxy(x1,x2,y1,y2,f)` introduces a distributed torque `f(x,y)` that acts perpendicular to the xz-plane. Spanwise, its starting and ending position are denoted by `x1` and `x2`, and `y1` and `y2` denote the cross-sectional starting and ending position. Positive values of `f(x,y)` mean that the force points in the positive y-direction.
5. `.addfdl(x1,x2,yf,zf,thetaf,f)` introduces a distributed load `f(x)`, with its local y-coordinate given by `yf(x)`, its local z-coordinate given by `zf(x)`, and its local inclination with respect to the y-axis given by `thetaf(x)` (that is, `thataf = 0` means that the distributed load acts in positive y-direction; `thetaf = m.pi/2` means that the distributed load acts in positive z-direction). Spanwise, its starting and ending position are denoted by `x1` and `x2`.
6. `.addfpl(x,y,z,theta,P)` introduces a point load. This point load has magnitude `P` and is applied at spanwise coordinate `x`, with its location within the crosssection given by the arguments `y` and `z`. Its inclination with respect to the y-axis is given by `theta`, analogous to how `thetaf(x)` was defined for `.addfdl`.
7. `.addfcm(x,theta,M)` introduces a couple moment. This couple moment has magnitude `M` and is applied at spanwise coordinate `x`, with its inclination with respect to the y-axis given by `theta`. In other words, `\theta = 0` corresponds to pure couple moment about the y-axis; `theta = m.pi/2` corresponds to a pure couple moment about the z-axis.

As many loads may be introduced as desired. Note that `scipy.integrate` is used for any numerical integration routines.

Computations The deflections corresponding to the defined boundary conditions and external loads are then computed by calling `aileron.compute_deflections()`.

Note that increasing the number of basis functions, N, and the order of the linear regression of the aerodynamic load, k_x and k_z may increase the computation time significantly.

Auxiliary functions Some simplistic plotting procedures are given in `.plotv()`, `.plotw()` and `.plotphi()`, which plot the y-deflection, z-deflection (of the axis passing through the shear center) and twist, respectively, as well as some of their derivatives and corresponding loads, resulting from

$$\begin{aligned}
T(x) &= GJ\frac{d\phi}{dx} \\
\tau(x) &= GJ\frac{d^2\phi}{dx^2} \\
v'(x) &= \frac{dv}{dx} \\
M_z(x) &= -EI_{yy}\frac{d^2v}{dx^2} \\
S_y(x) &= -EI_{yy}\frac{d^3v}{dx^2} \\
M_y(x) &= -EI_{zz}\frac{d^2w}{dx^2} \\
S_z(x) &= -EI_{zz}\frac{d^3w}{dx^2}
\end{aligned}$$

A number of auxiliary equations enable you to produce the numerical results necessary to construct the plots yourself. These are:

- `.eval(x)` computes the deflection at `x`. It produces three outputs: the first output is the $v(x)$; the second is $w(x)$, and the third is $\phi(x)$.
- `.fdeval(x)` computes the first derivatives of the deflections; it also produces three outputs, one for each deflection.
- `.sdeval(x)` computes the second derivatives of the deflections; it also produces three outputs, one for each deflection.
- `.tdeval(x)` computes the third derivatives of the deflections; it also produces three outputs, one for each deflection.
- `.Sy(x)` computes $S_y(x)$.
- `.Sz(x)` computes $S_z(x)$.
- `.My(x)` computes $M_y(x)$.
- `.Mz(x)` computes $M_z(x)$.
- `.T(x)` computes $T(x)$.
- `.tau(x)` computes $\tau(x)$.

All of these functions accept `numpy`-arrays as argument.

Important results Many intermediate results are directly available to you. These are:

- `.Na`: returns N_a.
- `.Nb`: returns N_b.
- `.Nc`: returns N_c.
- `.nbc`: returns N_{bc}.
- `.nbcv`: returns $N - N_a$.
- `.nbcw`: returns $N - N_b$.
- `.nbct`: returns $N - N_c$.
- `.Ha`: returns H_a.
- `.Hb`: returns H_b.
- `.Hc`: returns H_c.
- `.Ua`: returns Υ_a.
- `.Ub`: returns Υ_b.
- `.Uc`: returns Υ_c.
- `.K1`: returns the $K_{1,a}$ / $K_{1,b}$ matrix (these matrices are identical).
- `.C1`: returns the $K_{1,c}$ matrix.
- `.K2a`: returns the $K_{2,a}$ matrix.
- `.K2b`: returns the $K_{2,b}$ matrix.
- `.C2`: returns the $K_{2,c}$ matrix.
- `.F`: returns the F vector.

- `.LHS`: returns the left-hand-side of Equation (5.37).
- `.RHS`: returns the right-hand-side of Equation (5.37).
- `.sol.coef`: returns $\boldsymbol{\alpha}$ vector. The first N coefficients corresponds to **a**; the second N coefficients to **b** and the last N coefficients to **c**. The coefficients are ordered based on their index.

7.1.5 Part V

Part V is the final part of `main.py` and is concerned with computing the stress distributions.

First, `Stress.Stresstate(crosssection)` creates a `stressstate` object, which will contain the shear flows and stresses of the aileron.

Calling `Stressobject.compute_unitstressdistributions()` computes unit shear flow / stress distributions. That is, it will compute the shear flow distribution due to a unit shear force in horizontal direction, due to a unit shear force in vertical direction and due to a unit torque, and the stress distribution due to a unit bending moment about y and due to a bending moment about z. This means that in subsequent calculations, the program can simply take a linear combination of these distributions rather than having to recompute the stress distribution at each instance.

Then, the values of the desired S_y, S_z, M_y, M_z and T are set. Note that only scalars are accepted.

`Stressobject.compute_stressdistributions(Sy,Sz,My,Mz,T)` then computes the corresponding shear flow, direct stress and Von Mises stress distributions:

- `.plot_shearflowdistributions()` plots the shear flow distribution. Positive directions for the shear flows follow from Figure 3.1.
- `.plot_directstressdistributions()` plots the direct stress distribution.
- `.plot_vonmisstressdistributions()` plots the Von Mises stress distribution.

Finally, the shear flow, direct stress and Von Mises stress distributions may be obtained manually as well. `q*f`, `sigma*f`, `vm*f` and `coord*` are all functions that compute the shear flow distribution, direct stress distribution, Von Mises stress distribution and coordinate list in the corresponding region. The following coordinate systems are used for each region:

- Region 1 and 6: a polar coordinate system, as denoted in Figure 3.1, with angle θ with respect to the negative z-axis, measured in clockwise direction. For region 1, this runs between $0 \leq \theta \leq \pi/2$; for region 6, this runs between $-\pi/2 \leq \theta \leq 0$.
- Region 2 and 5: the only relevant coordinate is y, which needs to run between $0 \leq y \leq ha/2$ for region 2 and $-ha/2 \leq y \leq 0$ for region 5.
- Region 3 and 4: the s_3 and s_4 coordinates are used, as denoted in Figure 3.1, which both need to run between [0, $\sqrt{\left(h_a/2\right)^2 + \left(C_a - h_a/2\right)^2}$].