Aeroelastic Study of the Flutter Conditions of an Aircraft Wing

Document:

Appendices

Author/Authoress:

Guillermo Adroher Bolla

Director/Directress - Codirector/Codirectress:

Dr. David Roca Cazorla - Dr. Juan Carlos Cante Teran

Degree:

Bachelor's Degree in Aerospace Technologies Engineering

Examination Session:

Spring, 2022

BACHELOR FINAL THESIS

Contents

List of Figures

List of Tables

Listings

Appendix A

Structural Theory

This appendix exposes the mathematical formulation of the structural models for beam and plate elements, together with the computational implementation in MATLAB language.

A.1 Structural Basics

In this section, two theories are exposed, the Timoshenko for beams and the Reissner-Mindlin for flat plates. Both are described from the hypotheses to the FEM application, going through the mathematical formulation.

A.1.1 Timoshenko beam theory

The Timoshenko beam theory comes from the Euler-Bernoulli model, erasing one hypothesis from the latter that make the results more accurate.

Hypotheses of Timoshenko

To begin with the Euler-Bernoulli basis is exposed. The beam local axes are depicted in Figure A.1.

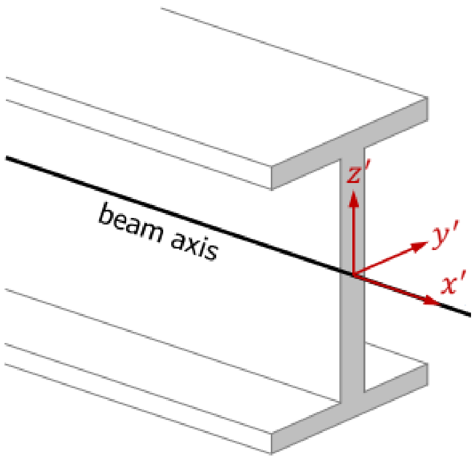


Figure A.1 Beam local axes considering an arbitrary I cross-section. (Source: [1])

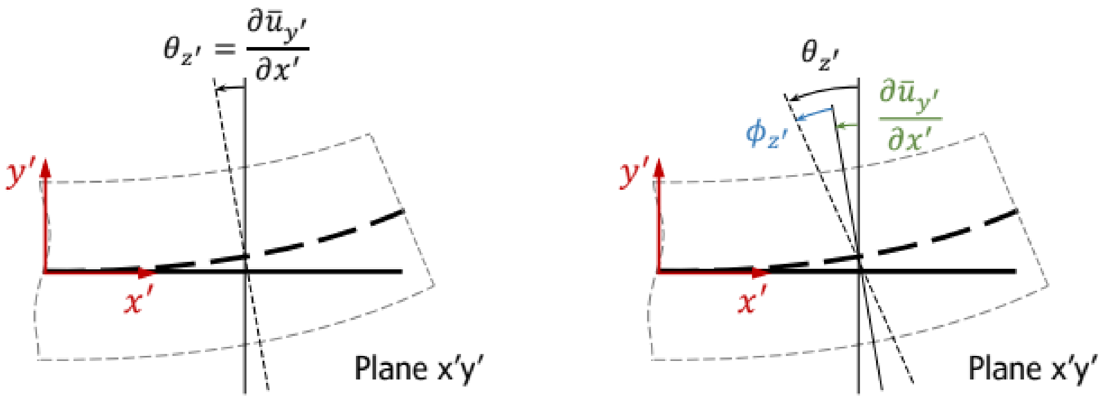

$\theta_{z'} = \frac{\partial \bar{u}_{y'}}{\partial x'}$
y'
x'
Plane x'y'
$\theta_{z'}$
$\phi_{z'}$
$\frac{\partial \bar{u}_{y'}}{\partial x'}$
y'
x'
Plane x'y'

Figure A.2 Plane $x'y'$ deflection comparison between the Euler-Bernoulli and the Timoshenko theories. (Source: [1])

- For shear loads applied on y' and bending moments about z' [1]:
  - Deflection $u_{y'}$ of the points on a cross- section are small and equal to the deflection of the beam axis.
  - Lateral displacement $u_{z'}$ is zero.
  - Cross-sections to the beam axis remain plane.
  - Cross-sections to the beam axis remain orthogonal to the beam axis after deformation.
- Equivalently, for shear loads applied on z' and bending moments about y' [1]:
  - Deflection $u_{z'}$ of the points on a cross- section are small and equal to the deflection of the beam axis.
  - Lateral displacement $u_{y'}$ is zero.

At this point, the only difference between the Timoshenko model and the Euler-Bernoulli is the removal of the last hypothesis when applying shear loads on y' and bending moments about z'. Timoshenko erases the restriction for the cross-sections to be orthogonal to the beam axis after deformation, obtaining a more realistic model.

Now, twist is the only type of deformation which is not considered yet. The Saint-Venant theory is accurate when considering twist, its hypotheses being the following.

- For beams with circular cross-section subjected to a twist moment about x' [1]:
  - The motion of each cross-section is a rigid body rotation about x'.
  - All points in the cross-section remain plane (there are no displacements in the x' direction).

When the cross-section of the beam is not circular, the last hypothesis is not true and displacements in the x' direction appear. This type of displacements are called *warping* (see Figure A.3).

When considering the warping phenomenon in practice, a twist constant is created so to compute the equivalent of the total deformation energy of the beam.

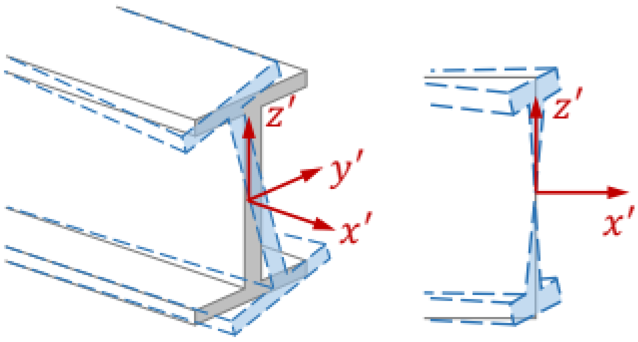


Figure A.3 Rigid-body rotation deformation (left) and warping phenomenon (right). (Source: [1])

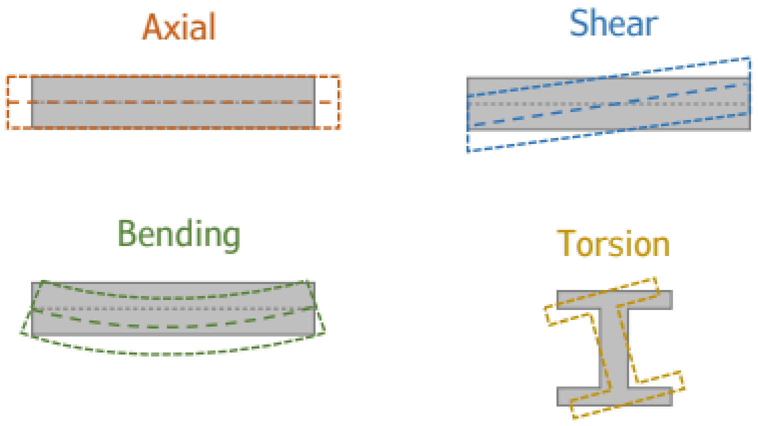


Figure A.4 Beam decomposition of the different types of deformation. (Source: [1])

Local displacements and strain

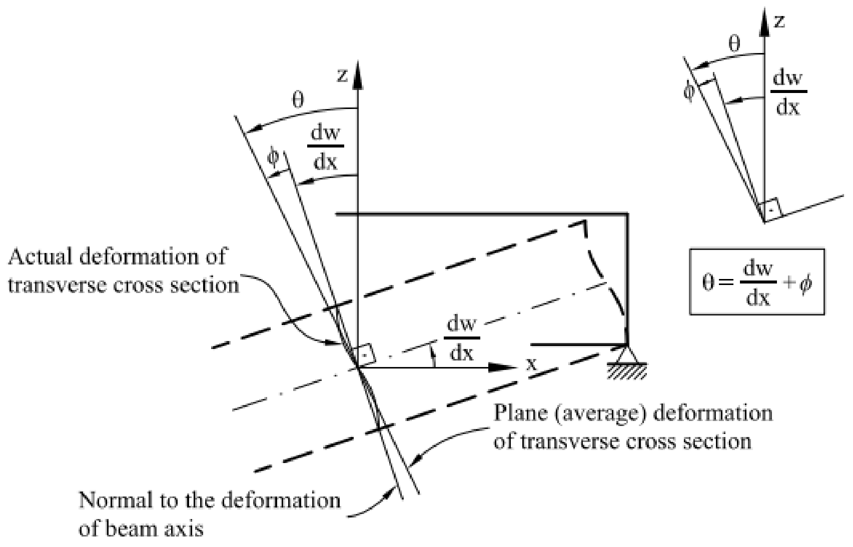
The local displacements of the beam considering twist (rotation about x' axis), bending moments (rotation about y' and z' axes), shear loads (applied on y' and z' axes) and axial load (applied on x' axis) are shown below [2].

$$\begin{aligned} u_{x'} &= \bar{u}_{x'} + z'\theta_{y'} - y'\theta_{z'} \\ u_{y'} &= \bar{u}_{y'} - z'\theta_{x'} \\ u_{z'} &= \bar{u}'_{z} + y'\theta_{x'} \end{aligned} \tag{A.1}$$

Where the different terms that appear mean the following[1]:

- $\bar{u}_{x'} \longrightarrow$ deformation in the longitudinal x' axis due to an axial load.
- $\bar{u}_{y'}$ and $\bar{u}_{z'} \longrightarrow$ deformation in the y' and z' axis due to shear loads on these axes.
- $z'\theta_{y'}$ and $y'\theta_{z'} \longrightarrow$ deformation due to bending moments around y' and z'.
- $z'\theta_{x'}$ and $y'\theta_{x'} \longrightarrow$ deformation due to the twist moments around the longitudinal axis x'.

[1]The rotations are assumed to be small in magnitude and because of that the deformations are computed under the assumption of small angles.



z
θ
dw/dx
ϕ
Actual deformation of transverse cross section
dw/dx
x
Plane (average) deformation of transverse cross section
Normal to the deformation of beam axis
θ = dw/dx + ϕ

Figure A.5 Rotation of the transverse cross-section as the Timoshenko theory states. (Source: [2])

Oñate's book nomenclature is not the same as in this project. The axes it uses are global while the axes in here are local (') and the vertical displacement in here is $u_{z'}$ while in the book it is w

Having the displacement field, the strain tensor[2] is easy to compute.

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{x'} & \frac{1}{2}\gamma_{x'y'} & \frac{1}{2}\gamma_{x'z'} \\ \frac{1}{2}\gamma_{x'y'} & \varepsilon_{y'} & \frac{1}{2}\gamma_{y'z'} \\ \frac{1}{2}\gamma_{x'z'} & \frac{1}{2}\gamma_{y'z'} & \varepsilon_{z'} \end{bmatrix} \tag{A.2}$$

Each term in the tensor can be computed as in [1],

$$\begin{aligned} \varepsilon_{x'} &= \frac{\partial u_{x'}}{\partial x'} = \frac{\partial \bar{u}_{x'}}{\partial x'} + z'\frac{\partial \theta_{y'}}{\partial x'} - y'\frac{\partial \theta_{z'}}{\partial x'} \\ \varepsilon_{y'} &= \frac{\partial u_{y'}}{\partial y'} = 0 \\ \varepsilon_{z'} &= \frac{\partial u_{z'}}{\partial z'} = 0 \\ \gamma_{x'y'} &= \frac{\partial u_{y'}}{\partial x'} + \frac{\partial u_{x'}}{\partial y'} = \frac{\partial \bar{u}_{y'}}{\partial x'} - \theta_{z'} - z'\frac{\partial \theta_{x'}}{\partial x'} = -\phi_{z'} - z'\frac{\partial \theta_{x'}}{\partial x'} \\ \gamma_{x'z'} &= \frac{\partial u_{z'}}{\partial x'} + \frac{\partial u_{x'}}{\partial z'} = \frac{\partial \bar{u}_{z'}}{\partial x'} + \theta_{y'} + y'\frac{\partial \theta_{x'}}{\partial x'} = \phi_{y'} + y'\frac{\partial \theta_{x'}}{\partial x'} \\ \gamma_{y'z'} &= \frac{\partial u_{z'}}{\partial y'} + \frac{\partial u_{y'}}{\partial z'} = 0 \end{aligned} \tag{A.3}$$

It is worth noticing that the terms $\phi_{y'}$ and $\phi_{z'}$ refer to an additional due to the rotation of the transverse cross-section. The total rotation is deduced from Figure A.5,

$$\theta = \frac{du_{z'}}{dx} + \phi \tag{A.4}$$

[2]The diagonal elements give the normal or longitudinal strains in each direction while the non-diagonal terms refer to the angular or shear strains.

thus it means the total rotation θ does not coincide with the slope of the linear distribution of the vertical displacement $u_{z'}$ along the x' axis.

The displacements and the strains fields (Equations A.1 and A.3) can be written in vector form, it will help later when talking about the FEM implementation.

$$\{\boldsymbol{u}'\} = \begin{Bmatrix} u_{x'} \\ u_{y'} \\ u_{z'} \end{Bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & z' & -y' \\ 0 & 1 & 0 & -z' & 0 & 0 \\ 0 & 0 & 1 & y' & 0 & 0 \end{bmatrix}}_{[\mathbf{S}(y',z')]} \underbrace{\begin{Bmatrix} \bar{u}_{x'} \\ \bar{u}_{y'} \\ \bar{u}_{z'} \\ \theta_{x'} \\ \theta_{y'} \\ \theta_{z'} \end{Bmatrix}}_{\{\bar{\boldsymbol{u}}'(x')\}} \tag{A.5}$$

$$\{\boldsymbol{\varepsilon}'\} = \begin{Bmatrix} \varepsilon_{x'} \\ \gamma_{x'y'} \\ \gamma_{x'z'} \end{Bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & z' & -y' \\ 0 & 1 & 0 & -z' & 0 & 0 \\ 0 & 0 & 1 & y' & 0 & 0 \end{bmatrix}}_{[\mathbf{S}(y',z')]} \underbrace{\begin{Bmatrix} \bar{u}_{x',x'} \\ \bar{u}_{y',x'} - \theta_{z'} \\ \bar{u}_{z',x'} + \theta_{y'} \\ \theta_{x',x'} \\ \theta_{y',x'} \\ \theta_{z',x'} \end{Bmatrix}}_{\{\bar{\boldsymbol{\varepsilon}}'(x')\}} \tag{A.6}$$

Note that the notation $\frac{\partial a_i}{\partial x_j} = a_{i,j}$ refers to the partial derivative of parameter a_i with respect to x_j.

Moreover, the local stresses vector form is also needed in further steps. In the following equation E is the Young's modulus, ν is the Poisson's ratio and G is the Shear modulus[3].

$$\{\boldsymbol{\sigma}'\} = \begin{Bmatrix} \sigma_{x'x'} \\ \tau_{x'y'} \\ \tau_{x'z'} \end{Bmatrix} = \underbrace{\begin{bmatrix} E & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{bmatrix}}_{[\mathbf{C}']} \begin{Bmatrix} \varepsilon_{x'x'} \\ \gamma_{x'y'} \\ \gamma_{x'z'} \end{Bmatrix} \tag{A.7}$$

Equilibrium equation and principle of virtual work (PVW)

The strong form of the equilibrium equation comes from the linear momentum equation and states

$$\dot{\boldsymbol{p}}' = \nabla_{x'} \cdot \boldsymbol{\sigma}' + \rho \boldsymbol{b}' + \boldsymbol{q}' \tag{A.8}$$

where $\dot{\boldsymbol{p}}' = \rho \boldsymbol{dotu'}$ is the linear momentum, ρ is the density in kg/m^3, $\boldsymbol{b'}$ is the external body forces vector in N/kg and $\boldsymbol{q'}$ is the volume forces vector in N/m^3.

The general boundary conditions are either prescribed displacements or external traction forces:

- Displacements: $\boldsymbol{u}'|_{x'\in\Gamma_u} = \boldsymbol{u}'_{\mathrm{p}}$
- Traction forces: $\boldsymbol{\sigma}' \cdot \widehat{\boldsymbol{n}}|_{x'\in\Gamma_\sigma} = \boldsymbol{t'}$

[3]The relation between them is $G = \frac{E}{2\,(1+\nu)}$ as isotropic materials are the only considered in this work.

Then, the principle of virtual work (PTW) gives the equation [1]

$$\int_{\Omega} \{\delta\boldsymbol{\varepsilon}'\}^{\mathrm{T}} \{\boldsymbol{\sigma}'\}\, d\Omega + \int_{\Omega} \{\delta\boldsymbol{u}'\}^{\mathrm{T}} \rho \{\ddot{\mathbf{u}}'\}\, d\Omega = \int_{\Omega} \{\delta\boldsymbol{u}'\}^{\mathrm{T}} \left(\rho \{\boldsymbol{b}'\} + \{\boldsymbol{q}'\}\right) d\Omega + \int_{\Gamma_\sigma} \{\delta\boldsymbol{u}'\}^{\mathrm{T}} \{\boldsymbol{t}'\}\, d\Gamma \tag{A.9}$$

For all $\delta\boldsymbol{u}'|_{x'\in\Gamma_u} = \mathbf{0}$, and such that $\{\delta\boldsymbol{\varepsilon}'\}^{\mathrm{T}} = \{\delta u_{x',x'}, \delta u_{x',y'} + \delta u_{y',x'}, \delta u_{x',z'} + \delta u_{z',x'}\}$

and assuming $\{\delta\boldsymbol{u}'\} = [\mathbf{S}(y', z')]\{\delta\bar{\mathbf{u}}'(x')\}$ and $\{\delta\boldsymbol{\varepsilon}'\} = [\mathbf{S}(y', z')]\{\delta\bar{\boldsymbol{\varepsilon}}'(x')\}$

The LHS of the PVW equation gives the stiffness and inertial terms while the RHS gives the external body and volume forces and the external surface forces applied over different cross-sections along the beam.

As what is sought in this work is to find the stiffness matrix and mass matrices so to perform a modal analysis later, the focus is exclusively on the LHS. There are two integrals to compute[4]:

- Stiffness term: It is the first term of the PVW equation and depends on material and cross-section parameters.

$$\int_{\Omega} \{\delta\boldsymbol{\varepsilon}'\}^{\mathrm{T}} \{\boldsymbol{\sigma}'\}\, d\Omega = \int_{\ell} \{\delta\bar{\boldsymbol{\varepsilon}}'\}^{\mathrm{T}} \underbrace{\left(\int_A [\mathbf{S}]^{\mathrm{T}} [\mathbf{C}'] [\mathbf{S}] dA\right)}_{[\overline{\mathbf{C}}']} \{\bar{\boldsymbol{\varepsilon}}'\}\, d\ell \tag{A.10}$$

$$\left[\overline{\mathbf{C}}'\right] = \int_A [\mathbf{S}]^{\mathrm{T}} [\mathbf{C}'] [\mathbf{S}] dA = \begin{bmatrix} EA & 0 & 0 & 0 & 0 & 0 \\ 0 & k_y GA & 0 & 0 & 0 & 0 \\ 0 & 0 & k_z GA & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{\mathrm{t}} GJ & 0 & 0 \\ 0 & 0 & 0 & 0 & EI_{y'} & 0 \\ 0 & 0 & 0 & 0 & 0 & EI_{z'} \end{bmatrix} \tag{A.11}$$

Where $I_{y'} = \int_A z'^2 dA$ and $I_{z'} = \int_A y'^2 dA$ are the area inertias of the section, $J = \int_A \left(y'^2 + z'^2\right) dA = I_{y'} + I_{z'}$ and k_y, k_z and k_t are the shear and twist correction constants.

- Inertial term: It is the second term of the PVW equation and depends also on material and cross-section parameters.

$$\int_{\Omega} \{\delta\boldsymbol{u}'\}^{\mathrm{T}} \rho \{\ddot{\mathbf{u}}'\}\, d\Omega = \int_{\ell} \{\delta\bar{\mathbf{u}}'\}^{\mathrm{T}} \underbrace{\left(\int_A [\mathbf{S}]^{\mathrm{T}} \rho [\mathbf{S}] dA\right)}_{[\bar{\boldsymbol{\rho}}']} \left\{\ddot{\bar{\mathbf{u}}}'\right\} d\ell \tag{A.12}$$

$$[\bar{\boldsymbol{\rho}}'] = \int_A [\mathbf{S}]^{\mathrm{T}} \rho [\mathbf{S}] dA = \rho \begin{bmatrix} A & 0 & 0 & 0 & 0 & 0 \\ 0 & A & 0 & 0 & 0 & 0 \\ 0 & 0 & A & 0 & 0 & 0 \\ 0 & 0 & 0 & J & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{y'} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{z'} \end{bmatrix} \tag{A.13}$$

[4]The cross-section of the beam is symmetric so the coupling terms of both matrices are 0, giving as a result two diagonal matrices.

The matrices are diagonal because of the symmetry of the cross-section about its two access. The shear and mass centers coincide with the intersection of the symmetry axes and that is where the origin of the local reference system is placed.

At this point, the LHS of the PVW equation yields:

$$\int_\Omega \{\delta\boldsymbol{\varepsilon}'\}^{\mathrm{T}}\{\boldsymbol{\sigma}'\}\,d\Omega + \int_\Omega \{\delta\boldsymbol{u}'\}^{\mathrm{T}}\rho\{\ddot{\boldsymbol{u}}'\}\,d\Omega = \int_\ell \{\delta\overline{\boldsymbol{\varepsilon}}'\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'\right]\{\overline{\boldsymbol{\varepsilon}}'\}\,d\ell + \int_\ell \{\delta\overline{\boldsymbol{u}}'\}^{\mathrm{T}}[\overline{\boldsymbol{\rho}}']\left\{\ddot{\overline{\boldsymbol{u}}}'\right\}d\ell \tag{A.14}$$

Decomposition of stiffness term

The stiffness term can be decomposed into different components. Each component refers to one of the types of deformation which affect the beam. This way the term becomes

$$\begin{aligned}\int_\ell \{\delta\overline{\boldsymbol{\varepsilon}}'\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'\right]\{\overline{\boldsymbol{\varepsilon}}'\}\,d\ell = \int_\ell \{\delta\overline{\varepsilon}'_{\mathrm{a}}\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{a}}\right]\{\overline{\varepsilon}'_{\mathrm{a}}\}\,d\ell + \int_\ell \{\delta\overline{\varepsilon}'_{\mathrm{b}}\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{b}}\right]\{\overline{\varepsilon}'_{\mathrm{b}}\}\,d\ell + \\ \int_\ell \{\delta\overline{\varepsilon}'_{\mathrm{s}}\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{s}}\right]\{\overline{\varepsilon}'_{\mathrm{s}}\}\,d\ell + \int_\ell \{\delta\overline{\varepsilon}'_{\mathrm{t}}\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{t}}\right]\{\overline{\varepsilon}'_{\mathrm{t}}\}\,d\ell\end{aligned} \tag{A.15}$$

where

$$\begin{aligned}
\{\overline{\boldsymbol{\varepsilon}}'_{\mathrm{a}}\} &= \{\bar{u}_{x',x'}\} & \left[\overline{\mathbf{C}}'_{\mathrm{a}}\right] &= [EA] & &\longrightarrow \text{Axial}\\
\{\overline{\boldsymbol{\varepsilon}}'_{\mathrm{b}}\} &= \left\{\begin{array}{c}\theta_{y',x'}\\ \theta_{z',x'}\end{array}\right\} & \left[\overline{\mathbf{C}}'_{\mathrm{b}}\right] &= \begin{bmatrix} EI_{y'} & 0\\ 0 & EI_{y'}\end{bmatrix} & &\longrightarrow \text{Bending}\\
\{\overline{\boldsymbol{\varepsilon}}'_{\mathrm{s}}\} &= \left\{\begin{array}{c}\bar{u}_{y',x'}-\theta_{z'}\\ \bar{u}_{z',x'}+\theta_{y'}\end{array}\right\} & \left[\overline{\mathbf{C}}'_{\mathrm{s}}\right] &= \begin{bmatrix} k_yGA & 0\\ 0 & k_zGA\end{bmatrix} & &\longrightarrow \text{Shear}\\
\{\overline{\boldsymbol{\varepsilon}}'_{\mathrm{t}}\} &= \{\theta_{x',x'}\} & \left[\overline{\mathbf{C}}'_{\mathrm{t}}\right] &= [k_{\mathrm{t}}GJ] & &\longrightarrow \text{twist}
\end{aligned} \tag{A.16}$$

This decomposition of the stiffness term is a way to simplify the computation of the global matrix. As it is defined in the computational methodology, not all the components are calculated using the same number of Gauss points[5].

Discretized displacements and strains

After the decomposition of the stiffness term, one can define the displacements and strain vectors in a discretized form. This will help when writing the computational code.

$$\left\{\bar{u}'^{(e)}\right\} = \begin{bmatrix}\cdots & \mathbf{N}^{(e,i)} & \cdots\end{bmatrix}\left\{\begin{array}{c}\vdots\\ \hat{u}^{(e,i)}\\ \vdots\end{array}\right\} \tag{A.17}$$

[5]Shear locking needs to be avoided so to obtain a realistic result (see subsection A.1.3).

$$\begin{gathered}\left\{\bar{\varepsilon}_{\mathrm{a}}^{\prime(e)}\right\}=\left[\begin{array}{ccc}\cdots & \mathbf{B}_{\mathrm{a}}^{\prime(e,i)} & \cdots\end{array}\right]\left\{\begin{array}{c}\vdots \\ \hat{u}^{(e,i)} \\ \vdots\end{array}\right\} \qquad \left\{\bar{\varepsilon}_{\mathrm{s}}^{\prime(e)}\right\}=\left[\begin{array}{ccc}\cdots & \mathbf{B}_{\mathrm{s}}^{\prime(e,i)} & \cdots\end{array}\right]\left\{\begin{array}{c}\vdots \\ \hat{u}^{(e,i)} \\ \vdots\end{array}\right\} \\ \left\{\bar{\varepsilon}_{\mathrm{b}}^{\prime(e)}\right\}=\left[\begin{array}{ccc}\cdots & \mathbf{B}_{\mathrm{b}}^{\prime(e,i)} & \cdots\end{array}\right]\left\{\begin{array}{c}\vdots \\ \hat{u}^{(e,i)} \\ \vdots\end{array}\right\} \qquad \left\{\bar{\varepsilon}_{\mathrm{t}}^{\prime(e)}\right\}=\left[\begin{array}{ccc}\cdots & \mathbf{B}_{\mathrm{t}}^{\prime(e,i)} & \cdots\end{array}\right]\left\{\begin{array}{c}\vdots \\ \hat{u}^{(e,i)} \\ \vdots\end{array}\right\}\end{gathered} \tag{A.18}$$

The second term in all the last discretization equations of element (e) is

$$\left\{\begin{array}{ccc}\cdots & \hat{u}^{(e,i)} & \cdots\end{array}\right\}^{\mathrm{T}}=\left\{\begin{array}{cccccc}\hat{u}_{x'}^{(e,i)} & \hat{u}_{y'}^{(e,i)} & \hat{u}_{z'}^{(e,i)} & \hat{\theta}_{x'}^{(e,i)} & \hat{\theta}_{y'}^{(e,i)} & \hat{\theta}_{z'}^{(e,i)}\end{array}\right\}^{\mathrm{T}} \tag{A.19}$$

and the first terms are

$$\begin{aligned}\left[\begin{array}{ccc}\cdots & \mathbf{N}^{(e,i)} & \cdots\end{array}\right]&=\sum_i\left[\begin{array}{cccccc}N^{(e,i)} & 0 & 0 & 0 & 0 & 0 \\ 0 & N^{(e,i)} & 0 & 0 & 0 & 0 \\ 0 & 0 & N^{(e,i)} & 0 & 0 & 0 \\ 0 & 0 & 0 & N^{(e,i)} & 0 & 0 \\ 0 & 0 & 0 & 0 & N^{(e,i)} & 0 \\ 0 & 0 & 0 & 0 & 0 & N^{(e,i)}\end{array}\right] \\ \left[\begin{array}{ccc}\cdots & \mathbf{B}_{\mathrm{a}}^{\prime(e,i)} & \cdots\end{array}\right]&=\sum_i\left[\begin{array}{cccccc}N_{,x'}^{(e,i)} & 0 & 0 & 0 & 0 & 0\end{array}\right] \\ \left[\begin{array}{ccc}\cdots & \mathbf{B}_{\mathrm{b}}^{\prime(e,i)} & \cdots\end{array}\right]&=\sum_i\left[\begin{array}{cccccc}0 & 0 & 0 & 0 & N_{,x'}^{(e,i)} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{,x'}^{(e,i)}\end{array}\right] \\ \left[\begin{array}{ccc}\cdots & \mathbf{B}_{\mathrm{s}}^{\prime(e,i)} & \cdots\end{array}\right]&=\sum_i\left[\begin{array}{cccccc}0 & N_{,x'}^{(e,i)} & 0 & 0 & 0 & -N^{(e,i)} \\ 0 & 0 & N_{,x'}^{(e,i)} & 0 & N^{(e,i)} & 0\end{array}\right] \\ \left[\begin{array}{ccc}\cdots & \mathbf{B}_{\mathrm{t}}^{\prime(e,i)} & \cdots\end{array}\right]&=\sum_i\left[\begin{array}{cccccc}0 & 0 & 0 & N_{,x'}^{(e,i)} & 0 & 0\end{array}\right]\end{aligned} \tag{A.20}$$

where the positions of the shape functions and their derivatives depend on the DOFs which are affected by each contribution. For example, the axial strain affects the first DOF (displacement in the x' axis) or twist affects the fourth DOF (rotation in the x' axis).

Discretized FEM global matrix system

Finally, the discretized FEM global matrices (stiffness and mass) can be computed. All the terms below are built using the same scheme: transposed rotation matrix, integral with the transposed matrix of the shape functions or their derivatives multiplied by the characteristic matrix[6] and the matrix of the shape functions or their derivatives without transposing and, at the end, the rotation matrix without transposing.

$$\left[\mathbf{M}^{(\mathbf{e})}\right]=\left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}}\int_{\ell^{(e)}}\left[\mathbf{N}^{(\mathbf{e})}\right]^{\mathbf{T}}\left[\overline{\boldsymbol{\rho}}^{\prime(e)}\right]\left[\mathbf{N}^{(\mathbf{e})}\right]d\ell\left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right] \tag{A.21}$$

[6]The characteristic matrix refers to the density matrix in the mass case and to the type of deformation in the stiffness case.

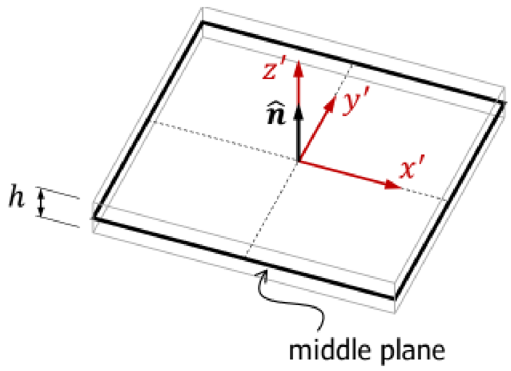


Figure A.6 Thin-plate middle plane and its local axes. (Source: [3])

$$\begin{aligned}
\left[\mathbf{K}_{\mathrm{a}}^{(e)}\right] &= \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{\ell^{(e)}} \left[\mathbf{B}_{\mathrm{a}}^{\prime(e)}\right]^{\mathbf{T}} \left[\overline{\mathbf{C}}_{\mathrm{a}}^{\prime(e)}\right] \left[\mathbf{B}_{\mathrm{a}}^{\prime(e)}\right] d\ell \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right] \\
\left[\mathbf{K}_{\mathrm{b}}^{(e)}\right] &= \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{\ell^{(e)}} \left[\mathbf{B}_{\mathrm{b}}^{\prime(e)}\right]^{\mathbf{T}} \left[\overline{\mathbf{C}}_{\mathrm{b}}^{\prime(e)}\right] \left[\mathbf{B}_{\mathrm{b}}^{\prime(e)}\right] d\ell \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right] \\
\left[\mathbf{K}_{\mathrm{s}}^{(e)}\right] &= \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{\ell^{(e)}} \left[\mathbf{B}_{\mathrm{s}}^{\prime(e)}\right]^{\mathbf{T}} \left[\overline{\mathbf{C}}_{\mathrm{s}}^{\prime(e)}\right] \left[\mathbf{B}_{\mathrm{s}}^{\prime(e)}\right] d\ell \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right] \\
\left[\mathbf{K}_{\mathrm{t}}^{(e)}\right] &= \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{\ell^{(e)}} \left[\mathbf{B}_{\mathrm{t}}^{\prime(e)}\right]^{\mathbf{T}} \left[\overline{\mathbf{C}}_{\mathrm{t}}^{\prime(e)}\right] \left[\mathbf{B}_{\mathrm{t}}^{\prime(e)}\right] d\ell \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]
\end{aligned} \tag{A.22}$$

The stiffness matrix is, then, the sum of all the components (axial, bending, shear and twist). Once both matrices are computed, one can run a modal analysis, for example.

A.1.2 Reissner-Mindlin plate theory

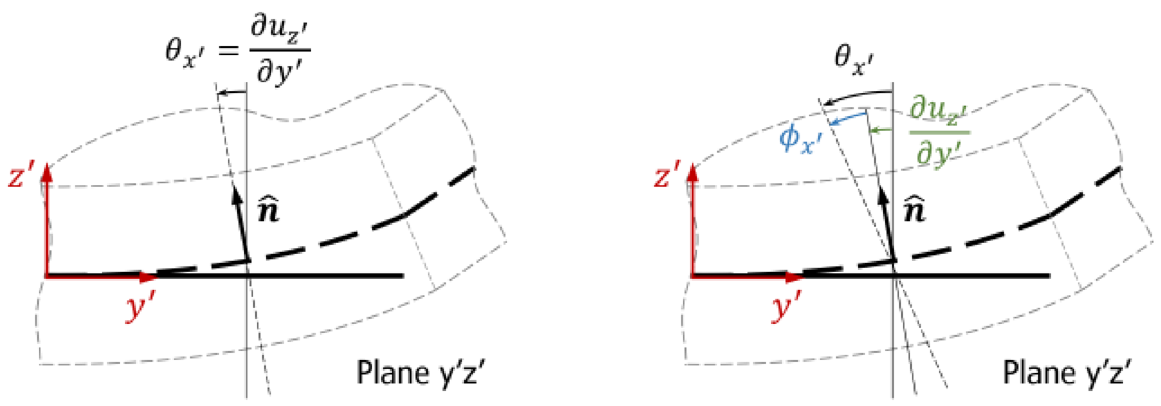
Although the Reissner-Mindlin theory comes from the Kirchhoff thin plate theory, the results are more precise than the latter. There is a restriction of the model which is not considered.

Hypotheses of Reissner-Mindlin

Firstly, the Kirchhoff basis for thin plates is stated (based on [3]). The displacements are computed in the middle plane (see Figure A.6), due to the thin thickness assumption.

- Points along the normal to the middle plane have the same vertical displacement (the thickness remains constant during deformation).
- Normal stress component is negligible, $\sigma_z \approx 0$ (plane stress assumption).
- A straight line normal to the undeformed middle plane remains straight after deformation.
- Points on the middle plane only move vertically ($\bar{u}_{x'} = \bar{u}_{y'} = 0$).
- A straight line normal to the undeformed middle plane remains normal to the deformed middle plane.

At this point, the only difference between the Reissner-Mindlin model and the Kirchhoff is the removal of the last hypothesis. The Ressner-Mindlin formulation erases the restriction for a straight line normal to the undeformed middle plane to remain normal to the deformed middle plane, thus a more realistic approximation



$\theta_{x'} = \frac{\partial u_{z'}}{\partial y'}$
z'
y'
$\hat{n}$
Plane y'z'
$\theta_{x'}$
$\phi_{x'}$
$\frac{\partial u_{z'}}{\partial y'}$
z'
y'
$\hat{n}$
Plane y'z'

Figure A.7 Plane $y'z'$ mid-plane deflection comparison between the Kirchhoff and the Reissner-Mindlin theories. (Source: [3])

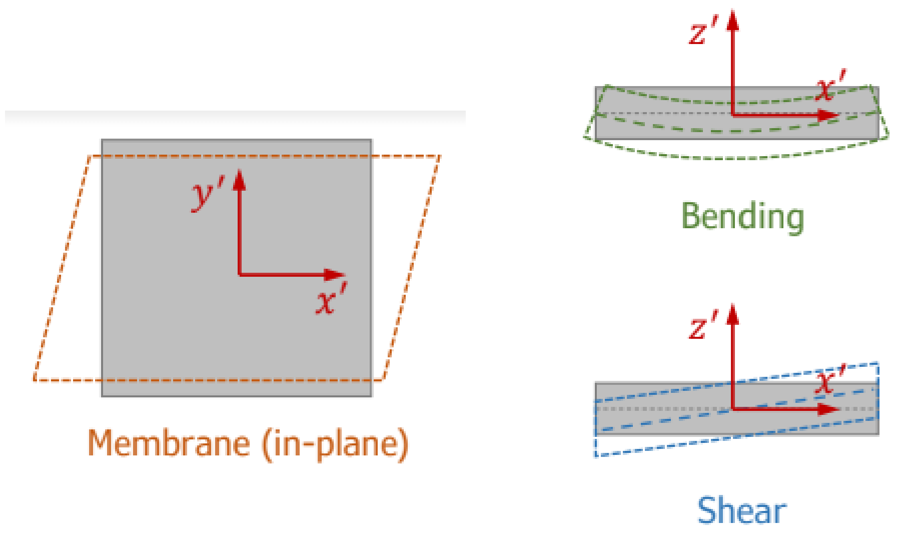


Figure A.8 Flat shell decomposition of the different types of deformation. (Source: [3])

is considered.

Local displacements and strain

The local displacements of the flat shell considering bending moments (rotation about x' and y' axes), shear loads (applied on z') and membrane deformation (applied on x' and y' axes) are shown below [2].

$$\begin{aligned} u_{x'} &= \bar{u}_{x'} + z'\theta_{y'} \\ u_{y'} &= \bar{u}_{y'} - z'\theta_{x'} \\ u_{z'} &= \bar{u}_z' \end{aligned} \tag{A.23}$$

Each term in the tensor can be computed as in [3],

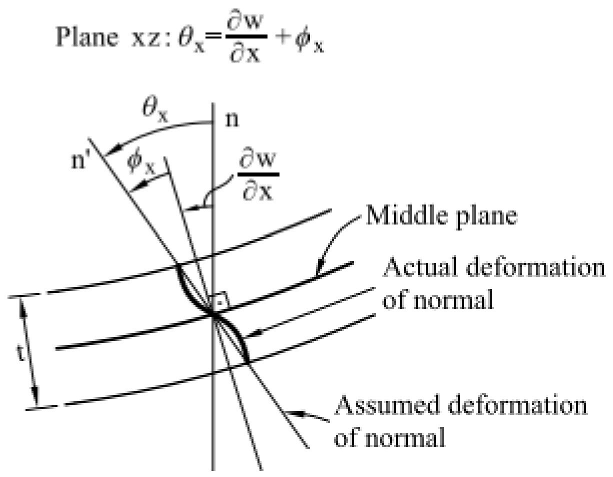

Plane xz: $\theta_x = \frac{\partial w}{\partial x} + \phi_x$
θ_x
n
n'
ϕ_x
$\frac{\partial w}{\partial x}$
Middle plane
Actual deformation of normal
t
Assumed deformation of normal

Figure A.9 Rotation of the transverse cross-section as the Reissner-Mindlin theory states. (Source: [2])

Oñate's book nomenclature is not the same as in this project. The axes it uses are global while the axes in here are local (') and the vertical displacement in here is $u_{z'}$ while in the book it is w

$$\begin{aligned}
\varepsilon_{x'x'} &= \frac{\partial u_{x'}}{\partial x'} = \frac{\partial \bar{u}_{x'}}{\partial x'} + z'\frac{\partial \theta_{y'}}{\partial x'} \\
\varepsilon_{y'y'} &= \frac{\partial u_{y'}}{\partial y'} = \frac{\partial \bar{u}_{y'}}{\partial y'} - z'\frac{\partial \theta_{x'}}{\partial y'} \\
\varepsilon_{z'z'} &= \frac{\partial u_{z'}}{\partial z'} = 0 \\
\gamma_{x'y'} &= \frac{\partial u_{y'}}{\partial x'} + \frac{\partial u_{x'}}{\partial y'} = \frac{\partial \bar{u}_{y'}}{\partial x'} + \frac{\partial \bar{u}_{x'}}{\partial y'} + z'\left(\frac{\partial \theta_{y'}}{\partial y'} - \frac{\partial \theta_{x'}}{\partial x'}\right) \\
\gamma_{x'z'} &= \frac{\partial u_{z'}}{\partial x'} + \frac{\partial u_{x'}}{\partial z'} = \frac{\partial \bar{u}_{z'}}{\partial x'} + \theta_{y'} = -\phi_{y'} \\
\gamma_{y'z'} &= \frac{\partial u_{z'}}{\partial y'} + \frac{\partial u_{y'}}{\partial z'} = \frac{\partial \bar{u}_{z'}}{\partial y'} - \theta_{x'} = -\phi_{x'}
\end{aligned} \tag{A.24}$$

Analog to the Timoshenko beam case, the total rotation of the flat plate adds and additional term ϕ due to the rotation of the transverse cross-section. In this case, as the plate is 2D, the equation is the same in the $x'z'$ and the $y'z'$ planes and the derivative is partial.

$$\begin{aligned}
\theta_x &= \frac{\partial u_{z'}}{\partial x'} + \phi_{x'} \\
\theta_y &= \frac{\partial u_{z'}}{\partial y'} + \phi_{y'}
\end{aligned} \tag{A.25}$$

Following the same procedure as in the beam, the displacements and strains fields (Equations A.23 and A.24) are organized in vector form.

$$\{\boldsymbol{u'}\} = \left\{ \begin{array}{c} u_{x'} \\ u_{y'} \\ u_{z'} \end{array} \right\} = \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & z' & 0 \\ 0 & 1 & 0 & -z' & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}^{[\boldsymbol{D}(z')]} \overbrace{\left\{ \begin{array}{c} \bar{u}_{x'} \\ \bar{u}_{y'} \\ \bar{u}_{z'} \\ \theta_{x'} \\ \theta_{y'} \\ \theta_{z'} \end{array} \right\}}^{\{\bar{\boldsymbol{u}}'(x',y')\}} \tag{A.26}$$

$$\{\boldsymbol{\varepsilon}'\} = \begin{Bmatrix} \varepsilon_{x'} \\ \varepsilon_{y'} \\ \gamma_{x'y'} \\ \gamma_{x'z'} \\ \gamma_{y'z'} \\ \theta_{z'} \end{Bmatrix} = \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & z' & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -z' & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & z' \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}^{[\mathbf{S}(z')]} \overbrace{\begin{Bmatrix} \bar{u}_{x',x'} \\ \bar{u}_{y',y'} \\ \bar{u}_{x',y'} + \bar{u}_{y',x'} \\ \bar{u}_{z',x'} + \theta_{y'} \\ \bar{u}_{z',y'} - \theta_{z'} \\ \theta_{z'} \\ \theta_{y',x'} \\ \theta_{x',y'} \\ \theta_{y',y'} - \theta_{x',x'} \end{Bmatrix}}^{\{\bar{\boldsymbol{\varepsilon}}'(x',y')\}} \tag{A.27}$$

A fictitious strain component $\theta_{z'}$ is added to the tensor.

The local stresses tensor in vector form is used in the next step so it is important to compute it.

$$\{\boldsymbol{\sigma}'\} = \begin{Bmatrix} \sigma_{x'x'} \\ \sigma_{y'y'} \\ \tau_{x'y'} \\ \tau_{x'z'} \\ \tau_{y'z'} \\ \sigma^*_{\mathrm{t}} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & 0 & 0 & 0 & 0 \\ C_{21} & C_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C^*_{\mathrm{t}} \end{bmatrix} \begin{Bmatrix} \varepsilon_{x'x'} \\ \varepsilon_{y'y'\mathrm{a}} \\ \gamma_{x'y'} \\ \gamma_{x'z'} \\ \gamma_{y'z'} \\ \theta^*_{z'} \end{Bmatrix} = [\mathbf{C}']\{\boldsymbol{\varepsilon}'\}$$
$$[\mathbf{C}'_{\mathrm{p}}] = \frac{E}{1-v^2}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & (1-v)/2 \end{bmatrix}; \quad [\mathbf{C}'_{\mathrm{s}}] = k_{\mathrm{s}} G \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad k_{\mathrm{s}} = \frac{5}{6} \tag{A.28}$$

Some important issues to mention are that σ^* refers to the fictitious stress component, $[\mathbf{C}'_p]$ to the plane-stress constitutive relation, $[\mathbf{C}'_s]$ to the shear stress-strain constitutive relation, and k_s to the shear correction factor[7].

Equilibrium equation and principle of virtual work (PVW)

The flat plate's strong form of the equilibrium equation and the PVW are the same as in the beam model (see Equations A.8 and A.9). The only things that change are the matrices of the displacements and strains derivatives.

For all $\delta\boldsymbol{u}'|_{x'\in\Gamma_u} = \mathbf{0}$, and such that $\{\delta\boldsymbol{\varepsilon}'\}^{\mathrm{T}} = \{\delta\varepsilon_{x',x'}, \delta\varepsilon_{y',y'}, \delta\gamma_{x',y'}, \delta\gamma_{x',z'}, \delta\gamma_{y',z'}, \delta\theta_{z'}\}$

and assuming $\{\delta\boldsymbol{u}'\} = [D(z')]\{\delta\bar{\boldsymbol{u}}'(x',y')\}$ and $\{\delta\boldsymbol{\varepsilon}'\} = [\mathbf{S}(z')]\{\delta\bar{\boldsymbol{\varepsilon}}'(x',y')\}$

The focus is again on the LHS of the PVW, having to compute the two terms, the stiffness and inertial. Therefore, the formulation of the integrals gives:

- Stiffness term: The first term of the PVW equation, it depends on material and cross-section parameters.

[7]The Reissner-Mindlin theory considers a constant distribution for the shear stresses while the exact distribution is parabolic.

$$\int_{\Omega}\{\delta\boldsymbol{\varepsilon}'\}^{\mathrm{T}}\{\boldsymbol{\sigma}'\}\,d\Omega=\int_{S}\{\delta\overline{\boldsymbol{\varepsilon}}'\}^{\mathrm{T}}\underbrace{\left(\int_{-h/2}^{h/2}[\mathbf{S}]^{\mathrm{T}}[\mathbf{C}'][\mathbf{S}]dz'\right)}_{[\overline{\mathbf{C}}']}\{\overline{\boldsymbol{\varepsilon}}'\}\,dS \tag{A.29}$$

$$\left[\overline{\mathbf{C}}'\right]=\int_{-h/2}^{h/2}[\mathbf{S}]^{\mathrm{T}}[\mathbf{C}'][\mathbf{S}]dz'$$

$$=h\begin{bmatrix} C_{11} & C_{12} & 0 & 0 & 0 & 0 & z_0'C_{11} & -z_0'C_{12} & 0\\ C_{21} & C_{22} & 0 & 0 & 0 & 0 & z_0'C_{21} & -z_0'C_{22} & 0\\ 0 & 0 & C_{33} & 0 & 0 & 0 & 0 & 0 & z_0'C_{33}\\ 0 & 0 & 0 & k_{\mathrm{s}}C_{44} & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & k_{\mathrm{s}}C_{55} & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & C_{\mathrm{t}}^{*} & 0 & 0 & 0\\ z_0'C_{11} & z_0'C_{12} & 0 & 0 & 0 & 0 & r_h'^2C_{11} & r_h'^2C_{12} & 0\\ -z_0'C_{21} & -z_0'C_{22} & 0 & 0 & 0 & 0 & r_h'^2C_{21} & r_h'^2C_{22} & 0\\ 0 & 0 & z_0'C_{33} & 0 & 0 & 0 & 0 & 0 & r_h'^2C_{33} \end{bmatrix} \tag{A.30}$$

Where $C_{11}=C_{22}=\frac{E}{1-\nu^2}$, $C_{12}=C_{21}=\frac{E\nu}{1-\nu^2}$, $C_{33}=C_{44}=C_{55}=\frac{E}{2\,(1+\nu)}=G=C_t^*$, $r_h'^2=\frac{1}{h}\int_{-h/2}^{h/2}z'^2dz'=\frac{h^2}{12}$ and $z_0'=\frac{1}{h}\int_{-h/2}^{h/2}z'dz'=0$.

Being $z_0'=0$ proves that the bending and membrane stresses and strains are uncoupled[8].

- Inertial term: The second term of the PVW equation, it depends also on material and cross-section parameters.

$$\int_{\Omega}\{\delta\boldsymbol{u}'\}^{\mathrm{T}}\rho\{\ddot{\boldsymbol{u}}'\}\,d\Omega=\int_{S}\{\delta\overline{\boldsymbol{u}}'\}^{\mathrm{T}}\underbrace{\left(\int_{-h/2}^{h/2}[\mathbf{S}]^{\mathrm{T}}\rho[\mathbf{S}]dz'\right)}_{[\overline{\boldsymbol{\rho}}']}\left\{\ddot{\overline{\boldsymbol{u}}}'\right\}dS \tag{A.31}$$

$$[\overline{\boldsymbol{\rho}}']=\int_{-h/2}^{h/2}[\mathbf{S}]^{\mathrm{T}}\rho[\mathbf{S}]dz'=\rho h\begin{bmatrix} 1 & 0 & 0 & 0 & z_0' & 0\\ 0 & 1 & 0 & -z_0' & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & -z_0' & 0 & r_h'^2 & 0 & 0\\ z_0' & 0 & 0 & 0 & r_h'^2 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.32}$$

The matrices erase the coupling terms ($z_0'=0$) because of the symmetry of the cross-section about its two access. The shear and mass centres coincide with the intersection of the symmetry axes and that is where the origin of the local reference system is placed.

The LHS of the PVW equation becomes:

$$\int_{\Omega}\{\delta\varepsilon'\}^{\mathrm{T}}\{\boldsymbol{\sigma}'\}\,d\Omega+\int_{\Omega}\{\delta\boldsymbol{u}'\}^{\mathrm{T}}\rho\{\ddot{\boldsymbol{u}}'\}\,d\Omega=\int_{S}\{\delta\overline{\varepsilon}'\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'\right]\{\overline{\varepsilon}'\}\,dS+\int_{S}\{\delta\overline{\boldsymbol{u}}'\}^{\mathrm{T}}[\overline{\boldsymbol{\rho}}']\left\{\ddot{\overline{\boldsymbol{u}}}'\right\}dS \tag{A.33}$$

[8]Remember this happens if the centre of mass is located at the origin of the local reference frame ($z'=0$).

Decomposition of stiffness term

Here again the stiffness term can be decomposed into the different components which affect the total deformation of the flat plate. In this case, the integral divides into

$$\begin{aligned}\int_S \{\delta\bar{\varepsilon}'\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'\right]\{\bar{\varepsilon}'\}\,dS = \int_S \{\delta\bar{\varepsilon}'_{\mathrm{m}}\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{m}}\right]\{\bar{\varepsilon}'_{\mathrm{m}}\}\,dS + \int_S \{\delta\bar{\varepsilon}'_{\mathrm{b}}\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{b}}\right]\{\bar{\varepsilon}'_{\mathrm{b}}\}\,dS + \\ \int_S \{\delta\bar{\varepsilon}'_{\mathrm{s}}\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{s}}\right]\{\bar{\varepsilon}'_{\mathrm{s}}\}\,dS + \int_S \{\delta\bar{\varepsilon}'_{\mathrm{t}}\}^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{t}}\right]\{\bar{\varepsilon}'_{\mathrm{t}}\}\,dS\end{aligned} \tag{A.34}$$

where

$$\begin{aligned}
&\{\bar{\varepsilon}'_{\mathrm{m}}\} = \left\{\begin{matrix}\bar{u}_{x',x'}\\ \bar{u}_{y',y'}\\ \bar{u}'_{x',y}+\bar{u}_{y',x'}\end{matrix}\right\} \quad \left[\overline{\mathbf{C}}'_{\mathrm{m}}\right] = \frac{hE}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & (1-\nu)/2\end{bmatrix} \longrightarrow \text{Membrane}\\
&\{\bar{\varepsilon}'_{\mathrm{b}}\} = \left\{\begin{matrix}\theta_{y',x'}\\ \theta_{x',y'}\\ \theta_{y',y'}-\theta_{x',x'}\end{matrix}\right\} \quad \left[\overline{\mathbf{C}}'_{\mathrm{b}}\right] = \frac{h^3E}{12\,(1-\nu^2)}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & (1-\nu)/2\end{bmatrix} \longrightarrow \text{Bending}\\
&\{\bar{\varepsilon}'_{\mathrm{s}}\} = \left\{\begin{matrix}\bar{u}_{z',x'}+\theta_{y'}\\ \bar{u}_{z',y'}-\theta_{x'}\end{matrix}\right\} \quad \left[\overline{\mathbf{C}}'_{\mathrm{s}}\right] = \frac{5hE}{12(1+\nu)}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \longrightarrow \text{Shear}\\
&\{\bar{\varepsilon}'_{\mathrm{t}}\} = \{\theta_{z'}\} \quad \left[\overline{\mathbf{C}}'_{\mathrm{t}}\right] = \frac{5hE}{12(1+\nu)}[1] \longrightarrow \text{Fictitious}
\end{aligned} \tag{A.35}$$

As in the case of the beam resolution, the decomposition of the stiffness term seeks to avoid shear locking and simplify the computational resolution.

Discretized displacements and strains

The discretization of the strains is analog to the one showed in Equation A.18. The only difference is that the flat shell considers the membrane and fictitious strains instead of the axial and twist for the beam. The displacements' discretization is the same for the beam and for the flat shell (Equation A.17).

$$\begin{aligned}
&\left\{\bar{\varepsilon}'^{(e)}_{\mathrm{m}}\right\} = \begin{bmatrix}\cdots & \mathbf{B}'^{(e,i)}_{\mathrm{m}} & \cdots\end{bmatrix}\left\{\begin{matrix}\vdots\\ \hat{u}^{(e,i)}\\ \vdots\end{matrix}\right\} \qquad \left\{\bar{\varepsilon}'^{(e)}_{\mathrm{s}}\right\} = \begin{bmatrix}\cdots & \mathbf{B}'^{(e,i)}_{\mathrm{s}} & \cdots\end{bmatrix}\left\{\begin{matrix}\vdots\\ \hat{u}^{(e,i)}\\ \vdots\end{matrix}\right\}\\
&\left\{\bar{\varepsilon}'^{(e)}_{\mathrm{b}}\right\} = \begin{bmatrix}\cdots & \mathbf{B}'^{(e,i)}_{\mathrm{b}} & \cdots\end{bmatrix}\left\{\begin{matrix}\vdots\\ \hat{u}^{(e,i)}\\ \vdots\end{matrix}\right\} \qquad \left\{\bar{\varepsilon}'^{(e)}_{\mathrm{t}}\right\} = \begin{bmatrix}\cdots & \mathbf{B}'^{(e,i)}_{\mathrm{t}} & \cdots\end{bmatrix}\left\{\begin{matrix}\vdots\\ \hat{u}^{(e,i)}\\ \vdots\end{matrix}\right\}
\end{aligned} \tag{A.36}$$

As the FEM DOFs for the beam and the flat shell are the same, equation A.19 can be applied in this case too. Both developments consider 3D cases, so it does not change. However, the first terms in the previous equation do change because now the element studied is 2D while in the other case it was 1D.

$$
\begin{aligned}
\begin{bmatrix} \cdots & \mathbf{N}^{(e,i)} & \cdots \end{bmatrix} &= \sum_i \begin{bmatrix} N^{(e,i)} & 0 & 0 & 0 & 0 & 0 \\ 0 & N^{(e,i)} & 0 & 0 & 0 & 0 \\ 0 & 0 & N^{(e,i)} & 0 & 0 & 0 \\ 0 & 0 & 0 & N^{(e,i)} & 0 & 0 \\ 0 & 0 & 0 & 0 & N^{(e,i)} & 0 \\ 0 & 0 & 0 & 0 & 0 & N^{(e,i)} \end{bmatrix} \\
\begin{bmatrix} \cdots & \mathbf{B}_{\mathrm{m}}'^{(e,i)} & \cdots \end{bmatrix} &= \sum_i \begin{bmatrix} N^{(e,i)}_{,x'} & 0 & 0 & 0 & 0 & 0 \\ 0 & N^{(e,i)}_{,y'} & 0 & 0 & 0 & 0 \\ N^{(e,i)}_{,y'} & N^{(e,i)}_{,x'} & 0 & 0 & 0 & 0 \end{bmatrix} \\
\begin{bmatrix} \cdots & \mathbf{B}_{\mathrm{b}}'^{(e,i)} & \cdots \end{bmatrix} &= \sum_i \begin{bmatrix} 0 & 0 & 0 & 0 & N^{(e,i)}_{,x'} & 0 \\ 0 & 0 & 0 & N^{(e,i)}_{,y'} & 0 & 0 \\ 0 & 0 & 0 & -N^{(e,i)}_{,x'} & N^{(e,i)}_{,y'} & 0 \end{bmatrix} \\
\begin{bmatrix} \cdots & \mathbf{B}_{\mathrm{s}}'^{(e,i)} & \cdots \end{bmatrix} &= \sum_i \begin{bmatrix} 0 & 0 & N^{(e,i)}_{,x'} & 0 & N^{(e,i)} & 0 \\ 0 & 0 & N^{(e,i)}_{,y'} & -N^{(e,i)} & 0 & 0 \end{bmatrix} \\
\begin{bmatrix} \cdots & \mathbf{B}_{\mathrm{t}}'^{(e,i)} & \cdots \end{bmatrix} &= \sum_i \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & N^{(e,i)} \end{bmatrix}
\end{aligned}
\tag{A.37}
$$

Again, the position of the shape functions and their derivatives depend on the DOFs which are activated in each contribution to the stiffness matrix.

Discretized FEM global matrix system

The scheme is the same as in the beam methodology, the rotation matrix is necessary to obtain the results in the global axes and the shape functions and their derivatives to define the FEM discretization.

$$
\left[\mathbf{M}^{(\mathbf{e})}\right] = \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{S^{(e)}} \left[\mathbf{N}^{(\mathbf{e})}\right]^{\mathbf{T}} \left[\overline{\boldsymbol{\rho}}'^{(e)}\right] \left[\mathbf{N}^{(\mathbf{e})}\right] dS \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]
\tag{A.38}
$$

$$
\begin{aligned}
\left[\mathbf{K}_{\mathrm{m}}^{(e)}\right] &= \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{S^{(e)}} \left[\mathbf{B}_{\mathrm{m}}'^{(e)}\right]^{\mathbf{T}} \left[\overline{\mathbf{C}}_{\mathrm{m}}'^{(e)}\right] \left[\mathbf{B}_{\mathrm{m}}'^{(e)}\right] dS \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right] \\
\left[\mathbf{K}_{\mathrm{b}}^{(e)}\right] &= \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{S^{(e)}} \left[\mathbf{B}_{\mathrm{b}}'^{(e)}\right]^{\mathbf{T}} \left[\overline{\mathbf{C}}_{\mathrm{b}}'^{(e)}\right] \left[\mathbf{B}_{\mathrm{b}}'^{(e)}\right] dS \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right] \\
\left[\mathbf{K}_{\mathrm{s}}^{(e)}\right] &= \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{S^{(e)}} \left[\mathbf{B}_{\mathrm{s}}'^{(e)}\right]^{\mathbf{T}} \left[\overline{\mathbf{C}}_{\mathrm{s}}'^{(e)}\right] \left[\mathbf{B}_{\mathrm{s}}'^{(e)}\right] dS \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right] \\
\left[\mathbf{K}_{\mathrm{t}}^{(e)}\right] &= \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]^{\mathbf{T}} \int_{S^{(e)}} \left[\mathbf{B}_{\mathrm{t}}'^{(e)}\right]^{\mathbf{T}} \left[\overline{\mathbf{C}}_{\mathrm{t}}'^{(e)}\right] \left[\mathbf{B}_{\mathrm{t}}'^{(e)}\right] dS \left[\widehat{\mathbf{R}}^{(\mathbf{e})}\right]
\end{aligned}
\tag{A.39}
$$

The objective regarding the structural theory ends in here. Later in this work, both static and dynamic aeroelasticity are studied. In the static case the mass matrix is not used because the corresponding matrix to run the modal analysis comes from the aerodynamic model. However, in the dynamic case, the mass matrix appears along with the aerodynamic one and that is why it has been also calculated,

A.1.3 Shear locking

Shear locking is an inconsistency that appears when linear interpolation is used for both lateral displacement and section rotation. The linear interpolation for lateral displacement causes the rotation to be constant

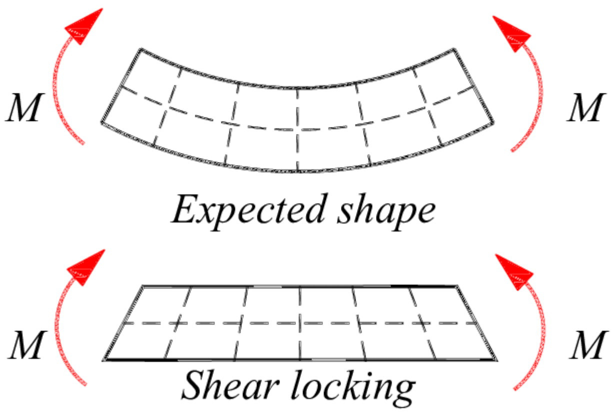


Figure A.10 Shear locking leads to over-stiff situations with wrong results in the FEM analysis. (Source: [4])

all over the beam[9], leading to zero curvature[10] as seen in Figure A.10. Zero curvature means only shear deformation applies to the beam, blocking the natural effects of bending deformation and bending stress energy [4].

It is important to know that shear locking appears in any bending element situation such as beams or flat shells, when Lagrange interpolation functions of the same order for deflection and rotation are used. The blocking increases specially when elements are thin.

The way to solve shear locking focuses on choosing consistent interpolation for both deflection (v) and rotation (θ). The derivative of the deflection needs to be the same as the section rotation, like the theory formulates.

Depending on the case, which can be the beam 1D or the flat shell 2D elements, shear locking is avoided by using different Gauss quadratures for the components of the stiffness matrix.

Referring to the beam computational formulation (see subsection A.2.1), although 2 Gauss points provide exact integration, for the shear component of the stiffness matrix a sub-integration with only 1 Gauss point avoids the problem with only a small loss in accuracy. The other components do not require Gauss quadrature to evaluate the integrals since the integrands are constant over the element.

On the other hand, for the flat shell computational formulation (see subsection A.2.2) something similar happens. The membrane (t), shear (s) and fictitious (t) components of the stiffness matrix are treated with 1 Gauss point while the bending (b) component and the mass matrix are integrated using 4 Gauss points.

A.2 Computational Approach

This section provides the mathematical formulation of the FEM method applied to structural beams (1D) and flat shells (2D). Although neither of the two elements are three-dimensional the whole computational definition considers the most extended case, the 3D one.

[9]For thin beams the shear strain is $\gamma_{xy} = v^{'} - \theta = 0$, where v' is the derivative of the lateral displacement and θ the section rotation.

[10]Curvature is the change in section rotation which is measured with the derivative $\theta^{'}$.

Each part has its own development and the main objective is to find the stiffness and mass global matrices. Apart from that, it is important to mention that the integrals are treated as discrete using the Gauss quadrature method.

A.2.1 Beam development

The beam computational model is based on the theory notes [1].

To begin with, the first section defines all the input data necessary to find the two global matrices, stiffness and mass. These data contains cross-section and material parameters, which are summarized in the following items:

- Material properties[11]
  - Young's modulus (E): 68.9 MPa
  - Shear modulus (G)[12]: 26 MPa
  - Density (ρ): 2700 kg/m^3
- Cross-section properties[13]
  - Orientation of y' axis ($\hat{\boldsymbol{\jmath}}'$) or z' axis ($\hat{\boldsymbol{k}}'$): The y' axis orientation depends on whether the beam is a spar or a rib and whether TR or SW is added. However, the z' axis orientation is always the same because the wing is flat so it is better to define this one.

$$\hat{\boldsymbol{k}}' = [\,0 \quad 0 \quad 1\,]$$

  - Area (A): The area depends on the height and width of the section. As linear taper is implemented, a constant value is not possible to give, it is different for each element. Being the shape rectangular the expression is

$$A = H \cdot W$$

    where H stands for *height* and W for *width.*

  - Polar inertia (J): It measures the ability to resist torsion and it depends on the shape. The expression to compute this parameter for a rectangular cross section is

$$\mathrm{J} = \mathrm{ab}^3 \left(\frac{16}{3} - 3.36\alpha \left(1 - \frac{1}{12}\alpha^4 \right) \right) \tag{A.40}$$

    where a is the section's height, b the width and α the parameter dividing the longest dimension by the shortest (always gives a value below 1).

[11] The materials used are isotropic, that is, materials whose properties do not change when tested in different directions.
[12] Alternatively, the shear modulus can be obtained via the Poisson ratio (ν) with the expression $G = E/(2\,(1+\nu))$
[13] All the development of the project deals only with cross-sections with symmetry about their two main axes thus parameters such as the cross area inertia ($I_{y'z'}$) are equal to zero.

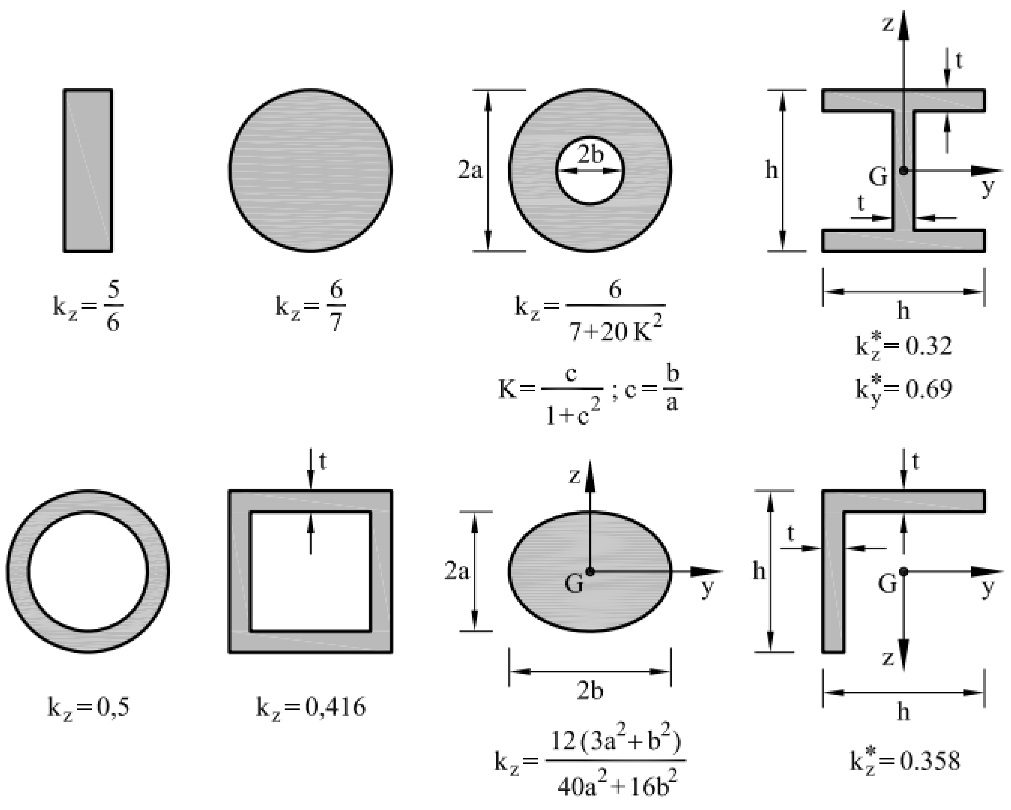


Figure A.11 Shear correction parameter k_z for typical cross sections. (Source: [2])
Asterisk (*) denotes values computed using the FEM

– Area inertia ($I_{y'} - I_{z'}$): The same happens as for the last two parameters, they depend on the section dimensions because of linear taper. The general expression for each axis is, taking as local reference system the one in Figure A.1),

$$\begin{aligned} I_{y'} &= \frac{1}{12} W H^3 \\ I_{z'} &= \frac{1}{12} H W^3 \end{aligned} \tag{A.41}$$

– Shear and twist correction factors ($k_y - k_z - k_t$): Looking at Figure A.11, for a rectangular cross-section there is only the shear correction parameter $k_z = \frac{5}{6}$.

Then, it is time to define the discretization data, which include the nodal coordinates and the nodal connectivities matrices.

$$[\mathbf{X}] = \begin{bmatrix} & \vdots & \\ \hat{x}^{(n)} & \hat{y}^{(n)} & \hat{z}^{(n)} \\ & \vdots & \end{bmatrix} \left.\vphantom{\begin{bmatrix} \vdots \\ x \\ \vdots \end{bmatrix}}\right\} \text{Number of nodes } N \qquad \begin{aligned} &\hat{x}^{(n)} : x\text{-coordinate of node } n \\ &\hat{y}^{(n)} : y\text{-coordinate of node } n \\ &\hat{z}^{(n)} : z\text{-coordinate of node } n \end{aligned} \tag{A.42}$$

$$[\mathbf{T}_{\mathrm{n}}] = \left[\begin{array}{ccc} & \vdots & \\ n_1^{(e)} & & n_2^{(e)} \\ & \vdots & \end{array}\right] \left.\vphantom{\begin{array}{c}\vdots\\n\\\vdots\end{array}}\right\} \text{Number of elements } N_e \qquad n_i^{(e)} : \begin{array}{c}\text{global node \# assigned to} \\ i\text{-th node in element } (e)\end{array} \tag{A.43}$$

Once the mesh nodes and connections between them are established, the boundary conditions are of utmost importance to define the problem one wants to solve. A matrix containing the fix nodes and DOFs simplifies the understanding and future implementation.

$$[\mathbf{U}_{\mathrm{p}}] = \left[\begin{array}{ccc} & \vdots & \\ u^{(p)} & n^{(p)} & j^{(p)} \\ & \vdots & \end{array}\right] \left.\vphantom{\begin{array}{c}\vdots\\n\\\vdots\end{array}}\right\} \text{Number of prescribed DOFs} \qquad \begin{array}{l} u^{(p)} : \text{value of prescribed displ/rot (p)} \\ n^{(p)} : \text{global node \# assigned to (p)} \\ j^{(p)} : \text{degree of freedom assigned to (p)} \end{array} \tag{A.44}$$

The DOFs indices characterize the three displacements first and the three rotations next.

$$\begin{array}{llll} j = 1: & \text{displacement in x-direction} & j = 4: & \text{rotation about x-direction} \\ j = 2: & \text{displacement in y-direction} & j = 5: & \text{rotation about y-direction} \\ j = 3: & \text{displacement in z-direction} & j = 6: & \text{rotation about z-direction} \end{array}$$

The next step is assembling both the stiffness and mass global matrices, which are square matrices of dimension N_{dof}. As the problem is 3D, the number of DOFs (N_{dof}) is six times the number of nodes (N).

The development shown below is performed for each element thus a *for* loop would be needed in the MATLAB code. Each element matrices would be added to the global matrices in the correct DOF positions in the last step.

- **Step 1:** Compute rotation matrix

  As the elements are defined in a local reference system it is necessary to first define the rotation matrix. This way, when arranging the stiffness and mass element matrices, that rotation matrix would transform from local to global coordinates.

  As the beam elements are 1D, their size can be easily computed,

$$\ell = \left\| \mathbf{X}\left(\mathbf{T}_{\mathrm{n}}(e,2),:\right) - \mathbf{X}\left(\mathbf{T}_{\mathrm{n}}(e,1),:\right) \right\| \tag{A.45}$$

Then, the three local axes of the beam element can be found with the element length and the input $\hat{\boldsymbol{\jmath}}'$,

$$\begin{aligned}\{\hat{\boldsymbol{\imath}}'\} &= \left(\{\mathbf{X}\left(\mathbf{T}_{\mathrm{n}}(e,2),:\right)\}^{\mathrm{T}} - \{\mathbf{X}\left(\mathbf{T}_{\mathrm{n}}(e,1),:\right)\}^{\mathrm{T}}\right)/\ell \\ \left\{\hat{\boldsymbol{k}}'\right\} &= \hat{\boldsymbol{k}} \\ \{\hat{\boldsymbol{\jmath}}'\} &= \left\{\hat{\boldsymbol{k}}'\right\} \times \{\hat{\boldsymbol{\imath}}'\}\end{aligned} \tag{A.46}$$

Arranging the local axes, each node rotation matrix and the corresponding one for the element are obtained,

$$\begin{aligned}[\mathbf{R}'] &= \begin{bmatrix} \{\hat{\boldsymbol{\imath}}'\} & \{\hat{\boldsymbol{\jmath}}'\} & \left\{\hat{\boldsymbol{k}}'\right\} & \cdots & [\mathbf{0}]_{3\times3} & \cdots \\ \cdots & [\mathbf{0}]_{3\times3} & \cdots & \{\hat{\boldsymbol{\imath}}'\} & \{\hat{\boldsymbol{\jmath}}'\} & \left\{\hat{\boldsymbol{k}}'\right\} \end{bmatrix}^{\mathrm{T}} \\ \mathbf{R}(:,i,e) &= \begin{bmatrix} [\mathbf{R}'] & [\mathbf{0}]_{6\times6} \\ [\mathbf{0}]_{6\times6} & [\mathbf{R}'] \end{bmatrix}\end{aligned} \tag{A.47}$$

As explained at the beginning of the development, this matrix is very important because all the following computations of the element mass and stiffness matrices would require of it.

- **Step 2:** Compute shape function derivatives

  The shape functions of the two-noded linear elements are those explained in subsection A.2.3. Their derivatives are easy to obtain,

$$N_{,x'}(1) = -1/\ell \qquad\qquad N_{,x'}(2) = 1/\ell \tag{A.48}$$

  The derivatives would be useful when computing the different contributions to the global stiffness matrix.

- **Step 3:** Compute each element matrix

  The matrices for each elements are divided into two categories, the contributions to the stiffness matrix and the mass matrix.

  The four contributions to the stiffness matrix are: the axial component, the bending component, the shear component and the twist component. Each one is computed following the same principle, which is to determine the matrix that characterizes each type of deformation (axial, bending, shear or twist) and apply the shape function derivatives matrix along with the element rotation matrix. This process results in the global element stiffness matrix for each contribution.

  – Axial component: The local axes of a beam are the x' axis in the longitudinal direction, the z' axis in the vertical direction and the y' where the cross product states.

    This way, the axial contribution affects the displacement in the longitudinal direction (x'). As the DOFs are organized, the latter corresponds to the first one.

Then, the characteristic matrix depends on the Young's modulus and the cross-sectional area.

$$\begin{aligned}
&\mathbf{B}'_{\mathrm{a}}(1,:,e) = \begin{bmatrix} N_{,x'}(1) & 0 & 0 & 0 & 0 & 0 & N_{,x'}(2) & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
&\overline{\mathbf{C}}'_{\mathrm{a}} = E\,A \\
&\mathbf{K}_{\mathrm{a}}(:,:,e) = \ell\,[\mathbf{R}(:,:,e)]^{\mathrm{T}}\,[\mathbf{B}'_{\mathrm{a}}(1,:,e)]^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{a}}\right][\mathbf{B}'_{\mathrm{a}}(1,:,e)]\,[\mathbf{R}(:,:,e)]
\end{aligned} \tag{A.49}$$

– <u>Bending component:</u> The second contribution is bending. In a 3D beam there are two types of bending, about y' or z' axes. The DOFs which correspond are rotation about y' or z' (DOFs 5 and 6).

  The characteristic matrix depends on the local inertia and the Young's modulus.

$$\begin{aligned}
&\mathbf{B}'_{\mathrm{b}}(:,:,e) = \begin{bmatrix} 0 & 0 & 0 & 0 & N_{,x'}(1) & 0 & 0 & 0 & 0 & 0 & N_{,x'}(2) & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{,x'}(1) & 0 & 0 & 0 & 0 & 0 & N_{,x'}(2) \end{bmatrix} \\
&\overline{\mathbf{C}}'_{\mathrm{b}} = E \begin{bmatrix} I_{y'}^{(\mathrm{T_m}(e))} & 0 \\ 0 & I_{z'}^{(\mathbf{T}_{\mathrm{m}}(e))} \end{bmatrix} \\
&\mathbf{K}_{\mathrm{b}}(:,:,e) = \ell\,[\mathbf{R}(:,:,e)]^{\mathrm{T}}\,[\mathbf{B}'_{\mathrm{b}}(:,:,e)]^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{b}}\right][\mathbf{B}'_{\mathrm{b}}(:,:,e)]\,[\mathbf{R}(:,:,e)]
\end{aligned} \tag{A.50}$$

– <u>Shear component:</u> The shear is probably the most complex deformation to understand because rotations and displacements are coupled. Following the above justification, both y' and z' are possible deformation axes. That said, a deformation in the y' direction is directly connected to a rotation about z' and vice versa.

  The matrix which defines the deformation phenomena strictly depends on the modulus of rigidity, the cross-sectional area and the shear correction factors.

  So to avoid shear locking[14] the shape functions assume only one Gauss point, giving $N = 1/2$.

$$\begin{aligned}
&\mathbf{B}'_{s}(:,:,e) = \begin{bmatrix} 0 & N_{,x'}(1) & 0 & 0 & 0 & -N & 0 & N_{,x'}(2) & 0 & 0 & 0 & -N \\ 0 & 0 & N_{,x'}(1) & 0 & N & 0 & 0 & 0 & N_{,x'}(2) & 0 & N & 0 \end{bmatrix} \\
&\overline{\mathbf{C}}'_{\mathrm{s}} = G\,A \begin{bmatrix} k_y & 0 \\ 0 & k_z \end{bmatrix} \\
&\mathbf{K}_{\mathrm{s}}(:,:,e) = \ell\,[\mathbf{R}(:,:e)]^{\mathrm{T}}\,[\mathbf{B}'_{\mathrm{s}}(:,:,e)]^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{s}}\right][\mathbf{B}'_{\mathrm{s}}(:,:,e)]\,[\mathbf{R}(:,:,e)]
\end{aligned} \tag{A.51}$$

– <u>twist component</u> Finally, the last contribution is due to the twist effect. It translates into a rotation about the longitudinal direction, x'.

  Its characteristic matrix depends on the modulus of rigidity and the polar inertia (directly related

[14]Shear locking is an error that appears because of the linear nature of the FEM. It leads to over-stiff situations for slender beams [2]. See subsection A.1.3

to the inertia concept).

$$\begin{aligned}
&\mathbf{B}'_{\mathrm{t}}(1,:,e) = \begin{bmatrix} 0 & 0 & 0 & N_{,x'}(1) & 0 & 0 & 0 & 0 & 0 & N_{,x'}(2) & 0 & 0 \end{bmatrix} \\
&\overline{\mathbf{C}}'_{\mathrm{t}} = G\,[J] \\
&\mathbf{K}_{\mathrm{t}}(:,:,e) = \ell\,[\mathbf{R}(:,:,e)]^{\mathrm{T}}\,[\mathbf{B}'_{\mathrm{t}}(1,:,e)]^{\mathrm{T}}\left[\overline{\mathbf{C}}'_{\mathrm{t}}\right][\mathbf{B}'_{\mathrm{t}}(1,:,e)]\,[\mathbf{R}(:,:,e)]
\end{aligned} \tag{A.52}$$

On the other hand, the mass matrix is computed directly with the cross-section and material input parameters. To compute the characteristic matrix one needs basically the density of the material, the cross-sectional area and the inertia modules including the polar one.

$$\begin{aligned}
&\overline{\boldsymbol{\rho}}' = \rho \begin{bmatrix} A & 0 & 0 & 0 & 0 & 0 \\ 0 & A & 0 & 0 & 0 & 0 \\ 0 & 0 & A & 0 & 0 & 0 \\ 0 & 0 & 0 & J & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{y'} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{z'} \end{bmatrix} \\
&\mathbf{M}_{\mathrm{e}}(:,:,e) = [\mathbf{0}]_{12\times12}
\end{aligned} \tag{A.53}$$

In this case, the mass matrix can be computed using two Gauss points, providing an exact integration result. For the 1D linear element the Gauss points coordinates ($\boldsymbol{\xi}$) and corresponding weights ($\boldsymbol{w}$) are

$$\begin{aligned}
&\{\boldsymbol{\xi}\} = \left\{-1/\sqrt{3}\ ;\ 1/\sqrt{3}\right\} \\
&\{\boldsymbol{w}\} = \{1\ ;\ 1\}
\end{aligned} \tag{A.54}$$

Then, applying the shape functions with its corresponding Gauss weights and rotation matrix, the element mass matrix is found.

$$\begin{aligned}
&N(1) = (1 - \xi(k))/2 \\
&N(2) = (1 + \xi(k))/2 \\
&\mathbf{N}(:,:,e,k) = [N(1)[\mathbf{1}]_{6\times6} \quad N(2)[\mathbf{1}]_{6\times6}] \\
&\mathbf{M}_e(:,:,e) = \mathbf{M}_e(:,i,e) + \boldsymbol{w}(k)\,\ell\,[\mathbf{R}(:,:,e)]^{\mathrm{T}}[\mathbf{N}(:,:,e,k)]^{\mathrm{T}}\,[\overline{\boldsymbol{\rho}}']\,[\mathbf{N}(:,i,e,k)][\mathbf{R}(:,i,e)]/2
\end{aligned} \tag{A.55}$$

The main step is this last one, as assembling the global matrices is simply organizing these element matrices, it has no mathematical formulation.

- **Step 4:** Assembly to global matrices

Finally, the step which keeps filling the global matrices by adding the contributions of each element consists of two stages.

The first stage identifies the position of the DOFs of each node so that they are added correctly to the global matrix. Considering six DOFs per node,

$$\begin{aligned} I_{\text{dof}}(j,1) &= 6\left(\mathbf{T}_{\text{n}}(e,1)-1\right)+j \\ I_{\text{dof}}(6+j,1) &= 6\left(\ \mathrm{T}_{\text{n}}(e,2)-1\right)+j \end{aligned} \tag{A.56}$$

And the second stage adds the previously computed element matrices to the global ones,

$$\begin{aligned} &\mathbf{K}\left(I_{\text{dof}},I_{\text{dof}}\right)=\mathbf{K}\left(I_{\text{dof}},I_{\text{dof}}\right)+\mathbf{K}_{\text{a}}(:,:,e)+\mathbf{K}_{\text{b}}(:,:,e)+\mathbf{K}_{\text{s}}(:,:,e)+\mathbf{K}_{\text{t}}(:,:,e) \\ &\mathbf{M}\left(I_{\text{dof}},I_{\text{dof}}\right)=\mathbf{M}\left(I_{\text{dof}},I_{\text{dof}}\right)+\mathbf{M}_{\text{e}}(:,:,e) \end{aligned} \tag{A.57}$$

Although both the stiffness and mass matrices are theoretically symmetric, due to machine precision when obtaining them computationally there are small differences in the last decimal digits.

A.2.2 Flat shell development

This case, which considers the computational approach for flat shells is more complex than the the one for beams in subsection A.2.1. It is based on the theory notes [3]. The idea and steps of the resolution are the same because both cases are based on the FEM. However, the elements are now bilinear quadrilateral with four nodes and the shell is 2D, not 1D as for the longitudinal beam.

There are less input parameters; the Young's modulus (E), the Poisson ratio (ν), the density (ρ) and the shell thickness (h).

Regarding the discretization data and boundary conditions the matrices are exactly the same than in the previous section except for the connectivities matrix, which now includes four columns (the finite elements are four-noded).

$$[\mathbf{T}_{\text{n}}]=\left[\begin{array}{cccc} n_1^{(e)} & n_2^{(e)} & n_3^{(e)} & n_4^{(e)} \end{array}\right]\Big\}\text{ Number of elements } N_e \qquad n_i^{(e)}:\ \begin{array}{c}\text{global node \# assigned to}\\ i\text{-th node in element (e)}\end{array} \tag{A.58}$$

Moving on to the computation of element matrices, the resolution process is identical. Only little complexity appears in the shape functions and matrices definition. Therefore, the development for each element is summarized below.

- **Step 1:** Compute rotation matrix

  It is necessary to define the rotation matrix to transform from the local reference system into the global one. First of all, longitudinal length and element surface vectors are defined,

$$\begin{aligned} \{\boldsymbol{S}\} &= \left(\{\mathbf{X}\left(\mathbf{T}_{\text{n}}(e,3),:\right)\}^{\text{T}}-\{\mathbf{X}\left(\mathbf{T}_{\text{n}}(e,1),:\right)\}^{\text{T}}\right)\times\left(\{\mathbf{X}\left(\mathbf{T}_{\mathbf{n}}(e,4),:\right)\}^{\text{T}}-\{\mathbf{X}\left(\mathbf{T}_{\text{n}}(e,2),:\right)\}^{\text{T}}\right)/2 \\ \{\boldsymbol{d}\} &= \left(\{\mathbf{X}\left(\mathbf{T}_{\text{n}}(e,2),:\right)\}^{\text{T}}+\{\mathbf{X}\left(\mathbf{T}_{\text{n}}(e,3),:\right)\}^{\text{T}}-\{\mathbf{X}\left(\mathbf{T}_{\text{n}}(e,4),:\right)\}^{\text{T}}-\{\mathbf{X}\left(\mathbf{T}_{\text{n}}(e,1),:\right)\}^{\text{T}}\right)/2 \end{aligned} \tag{A.59}$$

  The three local axes are obtained from the latter parameters. Notice that it is not necessary to give $\hat{\boldsymbol{\jmath}}'$

unit vector because the elements are 2D, so the normal unit vector is defined geometrically.

$$\begin{aligned}
&\left\{\widehat{\boldsymbol{k}}'\right\} = \{\boldsymbol{S}\}/\|\{\boldsymbol{S}\}\| \quad \text{(normal vector of the flat shell element)} \\
&\{\hat{\boldsymbol{\imath}}'\} = \{\boldsymbol{d}\}/\|\{\boldsymbol{d}\}\| \\
&\{\hat{\boldsymbol{\jmath}}'\} = \left\{\widehat{\boldsymbol{k}}'\right\} \times \{\hat{\boldsymbol{\imath}}'\}
\end{aligned} \tag{A.60}$$

Organizing the local unit vectors in a matrix the nodes' rotation matrix is found thus the element rotation matrix consists of the four contributions of each node in the corresponding element, giving a 24×24 square array.

$$\begin{aligned}
[\mathbf{R}'] &= \begin{bmatrix} \{\hat{\boldsymbol{\imath}}'\} & \{\hat{\boldsymbol{\jmath}}'\} & \left\{\widehat{\boldsymbol{k}}'\right\} & \cdots & [\mathbf{0}]_{3\times3} & \cdots \\ \cdots & [\mathbf{0}]_{3\times3} & \cdots & \{\hat{\boldsymbol{\imath}}'\} & \left\{\hat{\boldsymbol{\jmath}}'\right\} & \left\{\widehat{\boldsymbol{k}}'\right\} \end{bmatrix} \\
\mathbf{R}(:,:,e) &= \begin{bmatrix} [\mathbf{R}'] & [\mathbf{0}]_{6\times6} & [\mathbf{0}]_{6\times6} & [\mathbf{0}]_{6\times6} \\ [\mathbf{0}]_{6\times6} & [\mathbf{R}'] & [\mathbf{0}]_{6\times6} & [\mathbf{0}]_{6\times6} \\ [\mathbf{0}]_{6\times6} & [\mathbf{0}]_{6\times6} & [\mathbf{R}'] & [\mathbf{0}]_{6\times6} \\ [\mathbf{0}]_{6\times6} & [\mathbf{0}]_{6\times6} & [\mathbf{0}]_{6\times6} & [\mathbf{R}'] \end{bmatrix}
\end{aligned} \tag{A.61}$$

Throughout the development, this last matrix would appear continuously.

- **Step 2:** Get nodal coefficients for the shape functions

  Unlike the development for beams, now two arrays of four coefficients each are necessary. These coefficients define the contribution of each node in the element (see Table A.2).

$$\begin{aligned}
\{\boldsymbol{a}\} &= \{-1, 1, 1, -1\} \\
\{\boldsymbol{b}\} &= \{-1, -1, 1, 1\}
\end{aligned} \tag{A.62}$$

- **Step 3:** Compute each element matrix

  In this case, although not strictly the same as the 1D element case, there are also four contributions to the global stiffness matrix; the fictitious component, the shear component, the membrane component and the bending component.

  Relative to the mass matrix, the procedure does not change, it is only extended to four-noded elements.

  It is worth mentioning that, in order to avoid shear locking (see subsection A.1.3), two types of Gauss quadrature are used[15]. On the one hand, a 1 point Gauss quadrature involves the computation of the fictitious, shear and membranes components of the stiffness matrix while, on the other hand, a 4 point Gauss quadrature is used for the mass matrix and for the bending component of the stiffness matrix.

  - 1 Gauss point quadrature matrices:

    Before starting to compute the contributions to the stiffness global matrix, the shape functions in

[15]See Table A.3 for further information about Gauss quadrature for bilinear quadrilateral elements.

the parent or element domain[16] are defined, along with the initialization of the Jacobian matrix,

$$
\begin{aligned}
&\{\boldsymbol{N}_1\} = \{1,1,1,1\}^{\mathrm{T}}/4 \\
&\{\boldsymbol{N}_{1,\xi}\} = \{\boldsymbol{a}\}/4 \\
&\{\boldsymbol{N}_{1,\eta}\} = \{\boldsymbol{b}\}/4 \\
&[\boldsymbol{\mathcal{J}}_1] = [\mathbf{0}]_{2\times 2}
\end{aligned}
\tag{A.63}
$$

Once defined, the Jacobian matrix is filled, running a loop for all the nodes in the element studied. The mathematical formulation inside the *for* loop, for any node i is,

$$
\boldsymbol{\mathcal{J}}_1 = \boldsymbol{\mathcal{J}}_1 + \left\{ \begin{array}{c} \boldsymbol{N}_{1,\xi}(i) \\ \boldsymbol{N}_{1,\eta}(i) \end{array} \right\} \{\mathbf{X}\left(\mathbf{T}_{\mathrm{n}}(e,i),:\right)\} \; \begin{bmatrix} \boldsymbol{i}' & \boldsymbol{\jmath}' \end{bmatrix}
\tag{A.64}
$$

Afterwards, the transformation from the parent domain into the physical domain[17] is immediate,

$$
\begin{aligned}
&[\boldsymbol{N}_{1,x'}] = [\boldsymbol{\mathcal{J}}_1]^{-1} \begin{bmatrix} \boldsymbol{N}_{1,\xi} \\ \boldsymbol{N}_{1,\eta} \end{bmatrix} \\
&S_1 = 4 \cdot \det[\boldsymbol{\mathcal{J}}_1]
\end{aligned}
\tag{A.65}
$$

Being S_1 the area associated to the Gauss point considered for the element.

From this point, it is all set to define the computational approach to compute the fictitious, shear and membrane contributions to the stiffness matrix. The steps for each component do not change; compute the shape functions derivatives matrix (depending on the DOFs that are activated in each contribution) implementing a *for* loop for the nodes in the element, define the characteristic matrix which defines the type of deformation and apply the rotation from local to global axes along with the area associated to the Gauss point.

* **Fictitious component of stiffness matrix**

  The DOF affected is the rotation about the vertical axis (z direction) and the characteristic matrix depends on the plate's thickness, the Young's modulus and the Poisson coefficient of the material.

```
For each node $i$ (from 1 to 4) in the element:
    $\mathbf{B}_{\mathrm{t}}^{\prime(i)}(1,:,i) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \boldsymbol{N}_1(i) \end{bmatrix}$
End loop over nodes
$\overline{\mathbf{C}}_{\mathrm{t}}' = 5hE/\left(12\left(1+\nu\right)\right)$
$\mathbf{B}_{\mathrm{t}}'(1,:,e) = \left[\mathbf{B}_{\mathrm{t}}^{\prime(i)}(1,:,1), \mathbf{B}_{\mathrm{t}}^{\prime(i)}(1,:,2), \mathbf{B}_{\mathrm{t}}^{\prime(i)}(1,:,3), \mathbf{B}_{\mathrm{t}}^{\prime(i)}(1,:,4)\right]$
$\mathbf{K}_{\mathrm{t}}(:,:,e) = S_1[\mathbf{R}(:,:,e)]^{\mathrm{T}} \left[\mathbf{B}_{\mathrm{t}}'(1,:,e)\right]^{\mathrm{T}} \left[\overline{\mathbf{C}}_{\mathrm{t}}'\right] \left[\mathbf{B}_{\mathrm{t}}'(1,:,e)\right] \left[\mathbf{R}(:,:,e)\right]$
```
(A.66)

[16]Domain in which the nodes are defined using the local or element coordinates, those being ξ and η for 2D cases.
[17]Domain in which the nodes are defined using the global or physical coordinates, those being x and y for 2D cases.

* **Shear component of stiffness matrix**

  The DOF affected is the vertical displacement, coupled with the rotation about the x and y axes and the characteristic matrix depends on the plate's thickness, the Young's modulus and the Poisson coefficient of the material.

$$\begin{aligned}
&\text{For each node } i \text{ (from 1 to 4) in the element:}\\
&\qquad \mathbf{B}_{\mathrm{s}}^{\prime(i)}(:,:,i) = \begin{bmatrix} 0 & 0 & \boldsymbol{N}_{1,x'}(1,i) & 0 & \boldsymbol{N}_1(i) & 0 \\ 0 & 0 & \boldsymbol{N}_{1,x'}(2,i) & -\boldsymbol{N}_1(i) & 0 & 0 \end{bmatrix}\\
&\text{End loop over nodes}\\
&\overline{\mathbf{C}}_{\mathrm{s}}' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} 5hE/\left(12\left(1+\nu\right)\right)\\
&\mathbf{B}_{\mathrm{s}}'(:,:,e) = \left[\mathbf{B}_{\mathrm{s}}^{\prime(i)}(:,:,1), \mathbf{B}_{\mathrm{s}}^{\prime(i)}(:,:,2), \mathbf{B}_{\mathrm{s}}^{\prime(i)}(:,:,3), \mathbf{B}_{\mathrm{s}}^{\prime(i)}(:,:,4)\right]\\
&\mathbf{K}_{\mathrm{s}}(:,:,e) = S_1[\mathbf{R}(:,:,e)]^{\mathrm{T}}\left[\mathbf{B}_{\mathrm{s}}'(:,:,e)\right]^{\mathrm{T}}\left[\overline{\mathbf{C}}_{\mathrm{s}}'\right]\left[\mathbf{B}_{\mathrm{s}}'(:,:,e)\right]\left[\mathbf{R}(:,:,e)\right]
\end{aligned} \tag{A.67}$$

* **Membrane component of stiffness matrix**

  The DOFs affected are the longitudinal and transversal plane displacements (x and y directions) and the characteristic matrix depends on the plate's thickness, the Young's modulus and the Poisson coefficient of the material.

$$\begin{aligned}
&\text{For each node } i \text{ (from 1 to 4) in the element:}\\
&\qquad \mathbf{B}_{\mathrm{m}}^{\prime(i)}(:,:,i) = \begin{bmatrix} \boldsymbol{N}_{1,x'}(1,i) & 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{N}_{1,\boldsymbol{x}'}(2,i) & 0 & 0 & 0 & 0 \\ \boldsymbol{N}_{1,\boldsymbol{x}'}(2,i) & \boldsymbol{N}_{1,\boldsymbol{x}'}(1,i) & 0 & 0 & 0 & 0 \end{bmatrix}\\
&\text{End loop over nodes}\\
&\overline{\mathbf{C}}_{\mathrm{m}}' = \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \left(1-\nu\right)/2 \end{bmatrix} hE/\left(1-\nu^2\right)\\
&\mathbf{B}_{\mathrm{m}}'(:,:,e) = \left[\mathbf{B}_{\mathrm{m}}^{\prime(i)}(:,:,1), \mathbf{B}_{\mathrm{m}}^{\prime(i)}(:,:,2), \mathbf{B}_{\mathrm{m}}^{\prime(i)}(:,:,3), \mathbf{B}_{\mathrm{m}}^{\prime(i)}(:,:,4)\right]\\
&\mathbf{K}_{\mathrm{m}}(:,:,e) = S_1[\mathbf{R}(:,:,e)]^{\mathrm{T}}\left[\mathbf{B}_{\mathrm{m}}'(:,:,e)\right]^{\mathrm{T}}\left[\overline{\mathbf{C}}_{\mathrm{m}}'\right]\left[\mathbf{B}_{\mathrm{m}}'(:,:,e)\right]\left[\mathbf{R}(:,:,e)\right]
\end{aligned} \tag{A.68}$$

– 4 Gauss point quadrature matrices:

The first step is to initialize the matrices one wants to fulfill, the bending component and the mass matrices.

$$\begin{aligned}
\mathbf{K}_{\mathrm{b}}(:,:,e) &= [\mathbf{0}]_{24\times 24}\\
\mathbf{M}_{\mathrm{b}}(:,:,e) &= [\mathbf{0}]_{24\times 24}
\end{aligned} \tag{A.69}$$

Now the Gauss quadrature vectors are defined, being ξ and η the element coordinates and w the weight of each coordinate point affecting the integration.

$$\begin{aligned}\{\boldsymbol{\xi_4}\} &= \{-1,1,1,-1\}/\sqrt{3}\\ \{\boldsymbol{\eta}_4\} &= \{-1,-1,1,1\}/\sqrt{3}\\ \{\boldsymbol{w_4}\} &= \{1,1,1,1\}\end{aligned} \tag{A.70}$$

Then, the procedure that follows is repeated for every Gauss point k in a *for* loop. The Jacobian matrix is defined again (as in the beam development),

$$[\boldsymbol{\mathcal{J}}_4] = [\mathbf{0}]_{2\times 2} \tag{A.71}$$

Inside the loop for each Gauss point, another iterative process arises, this one being for each node in each bilinear quadrilateral element. The shape functions and their derivatives are defined, looking forward to filling the Jacobian matrix,

$$\begin{aligned}&\boldsymbol{N}_4(i) = (1+\boldsymbol{a}(i)\xi_4(k))\,(1+\boldsymbol{b}(i)\boldsymbol{\eta}_4(k))\,/4\\ &\boldsymbol{N}_{4,\xi}(1,i) = \boldsymbol{a}(i)\,(1+\boldsymbol{b}(i)\boldsymbol{\eta}_4(k))\,/4\\ &\boldsymbol{N}_{4,\eta}(1,i) = \boldsymbol{b}(i)\,(1+\boldsymbol{a}(i)\xi_4(k))\,/4\\ &\boldsymbol{\mathcal{J}}_4 = \boldsymbol{\mathcal{J}}_4 + \left\{\begin{matrix}\boldsymbol{N}_{4,\xi}(i)\\ \boldsymbol{N}_{4,\eta}(i)\end{matrix}\right\}\{\mathbf{X}\,(\mathbf{T}_{\rm n}(e,i),:)\}\ \begin{bmatrix}\hat{\boldsymbol{i}}' & \hat{\boldsymbol{j}}'\end{bmatrix}\end{aligned} \tag{A.72}$$

The loop for the nodes in the element finishes here, being the last step of this initial calculations the transformation from the parent domain into the physical one,

$$\begin{aligned}[\boldsymbol{N}_{4,x'}] &= [\boldsymbol{\mathcal{J}}_4]^{-1}\begin{bmatrix}\boldsymbol{N}_{4,\xi}\\ \boldsymbol{N}_{4,\eta}\end{bmatrix}\\ \boldsymbol{S_4}(e,k) &= \boldsymbol{w_4}(k)\cdot\det[\boldsymbol{\mathcal{J}}_4]\end{aligned} \tag{A.73}$$

S_4 is, again, the area associated to Gauss point k.

When this procedure related to the definition of the shape functions, the parent and physical domains and the Jacobian matrix is finished, one can move forward to the computation of the matrices stated at the beginning of the development.

$*$ **Bending component of stiffness matrix**

The first matrix is the bending component, which affects directly the stiffness of the flat plate, it is one of the four contributions.

The DOFs affected are the rotations in the base plane of the flat shell ($x-y$ plane). The characteristic matrix depends on the Poisson ratio, the plate's thickness and the Young's modulus. Those are the same parameters considered as in the other contributions of the stiffness matrix.

For each node i (from 1 to 4) in the element:

$$\mathbf{B}_{\mathrm{b}}^{\prime(i)}(:,:,i) = \begin{bmatrix} 0 & 0 & 0 & 0 & \boldsymbol{N}_{4,x'}(1,i) & 0 \\ 0 & 0 & 0 & \boldsymbol{N}_{4,x'}(2,i) & 0 & 0 \\ 0 & 0 & 0 & -\boldsymbol{N}_{4,x'}(1,i) & \boldsymbol{N}_{4,x'}(2,i) & 0 \end{bmatrix}$$

End loop over nodes

$$\overline{\mathbf{C}}_{\mathrm{b}}' = \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} h^3 E / \left(12\left(1-\nu^2\right)\right)$$

$$\mathbf{B}_{\mathrm{b}}'(:,:,e,k) = \left[\mathbf{B}_{\mathrm{b}}^{\prime(i)}(:,:,1), \mathbf{B}_{\mathrm{b}}^{\prime(i)}(:,:,2), \mathbf{B}_{\mathrm{b}}^{\prime(i)}(:,:,3), \mathbf{B}_{\mathrm{b}}^{\prime(i)}(:,:,4)\right]$$

$$\mathbf{K}_{\mathrm{b}}(:,:,e) = \mathbf{K}_{\mathrm{b}}(:,:,e) + \boldsymbol{S}_4(e,k)[\mathbf{R}(:,:,e)]^{\mathrm{T}} \left[\mathbf{B}_{\mathrm{b}}'(:,:,e,k)\right]^{\mathrm{T}} \left[\overline{\mathbf{C}}_{\mathrm{b}}'\right] \left[\mathbf{B}_{\mathrm{b}}'(:,:,e,k)\right] \left[\mathbf{R}(:,:,e)\right] \tag{A.74}$$

Once the stiffness element matrix is finished it is time to compute the mass element matrix. One needs, at first, to organize the shape functions of each node in an equivalent matrix,

For each node i (from 1 to 4) in the element:

$$\mathbf{N}^{(i)}(:,:,i) = \boldsymbol{N}_4(i)[\mathbf{1}]_{6\times 6} \quad ([\mathbf{1}]_{6\times 6} \equiv \text{ Identity matrix of } 6\times 6) \tag{A.75}$$

End loop over nodes

Next, with the density of the material and the plate's thickness, the characteristic matrix is found. And, multiplying the latter by the Gauss point area, the rotation matrix and the shape function matrix one obtains the element mass matrix contribution of Gauss point k.

$$\overline{\boldsymbol{\rho}}' = \rho\, h \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & h^2/12 & 0 & 0 \\ 0 & 0 & 0 & 0 & h^2/12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.76}$$

$$\mathbf{N}(:,:,e,k) = \left[\mathbf{N}^{(i)}(:,:,1), \mathbf{N}^{(i)}(:,:,2), \mathbf{N}^{(i)}(:,:,3), \mathbf{N}^{(i)}(:,:,4)\right]$$

$$\mathbf{M}_{\mathrm{e}}(:,:,e) = \mathbf{M}_{\mathrm{e}}(:,:,e) + \boldsymbol{S}_4(e,k)\,[\mathbf{R}(:,:,e)]^{\mathrm{T}}[\mathbf{N}(:,:,e,k)]^{\mathrm{T}}\,[\overline{\boldsymbol{\rho}}']\,[\mathbf{N}(:,:,e,k)][\mathbf{R}(:,:,e)]$$

The Gauss point loop ends in here. By repeating the exposed iterative process for the four Gauss points, the element bending and mass matrices are fulfilled.

- **Step 4:** Assembly to global matrices

Finally, the assembly of the stiffness and mass matrices follows the same procedure as for the beam. First of all, the position of the DOFs is identified and then each contribution is added to the global matrix.

$$\begin{aligned} I_{\text{dof}}(j,1) &= 6\left(\mathbf{T}_{\text{n}}(e,1)-1\right)+j \\ I_{\text{dof}}(6+j,1) &= 6\left(\ \mathrm{T}_{\text{n}}(e,2)-1\right)+j \end{aligned} \tag{A.77}$$

$$\begin{aligned} \mathbf{K}\left(I_{\text{dof}},I_{\text{dof}}\right) &= \mathbf{K}\left(I_{\text{dof}},I_{\text{dof}}\right)+\mathbf{K}_{\text{a}}(:,:,e)+\mathbf{K}_{\text{b}}(:,:,e)+\mathbf{K}_{\text{s}}(:,:,e)+\mathbf{K}_{\text{t}}(:,:,e) \\ \mathbf{M}\left(I_{\text{dof}},I_{\text{dof}}\right) &= \mathbf{M}\left(I_{\text{dof}},I_{\text{dof}}\right)+\mathbf{M}_{\text{e}}(:,:,e) \end{aligned} \tag{A.78}$$

Once again, both the stiffness and mass matrices should be symmetric but due to machine precision there are small differences in the last decimal digits (they are not symmetric).

A.2.3 Computational elements

Once the two FEM developments for the wing structure are explained, it is necessary to define which type of elements characterize each structural part.

So as to reach an accurate solution, it is convenient to use two-noded linear elements for the beams (spars and ribs) and bilinear quadrilateral elements for the flat shell.

Two-Noded Linear Elements

The spars and the ribs of the wing are studied as beams, so the finite element chosen is the two-noded linear one. It gives reliable results and allows to simplify the mathematical formulation.

Firstly, the shape functions in natural coordinates $\xi \in [-1,1]$ are

$$N^{(e,1)}(\xi)=\frac{1}{2}(1-\xi) \qquad N^{(e,2)}(\xi)=\frac{1}{2}(1+\xi) \tag{A.79}$$

Consequently, the derivatives of these shape functions in natural coordinates are

$$N^{(e,1)}_{,\xi}=-\frac{1}{2} \qquad N^{(e,2)}_{,\xi}=\frac{1}{2} \tag{A.80}$$

The characterization of this element is depicted in Figure A.12, one can see the element domain, the natural and local coordinates and the shape functions.

Next, the Jacobian of the element results as

$$\mathcal{J}'^{(e)}=\frac{\partial x'}{\partial \xi}=\sum_i N^{(e,i)}_{,\xi}\underbrace{\left\{\hat{\boldsymbol{x}}^{(e,i)}\right\}^{\mathrm{T}}\left\{\hat{\boldsymbol{i}}'^{(e)}\right\}}_{\hat{x}'(e,i)}=\frac{1}{2}\underbrace{\left(\hat{x}'^{(e,2)}-\hat{x}'^{(e,1)}\right)}_{\ell^{(e)}} \tag{A.81}$$

Finally, the shape function derivatives in local coordinates are like

$$N^{(e,1)}_{,x'}=\left(\mathcal{J}'^{(e)}\right)^{-1}N^{(e,1)}_{,\xi}=-\frac{1}{\ell^{(e)}} \qquad N^{(e,2)}_{,x'}=\left(\mathcal{J}'^{(e)}\right)^{-1}N^{(e,2)}_{,\xi}=+\frac{1}{\ell^{(e)}} \tag{A.82}$$

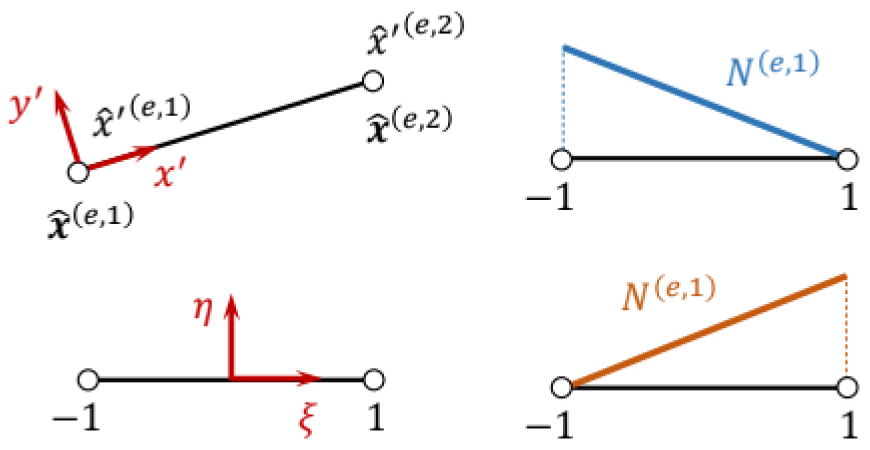


Figure A.12 Two-noded linear element variables and node ordering together with its shape functions. (Source: [1])

where $\ell^{(e)}$ is the element length.

At this point, when evaluating the integrals seen in subsection A.2.1, the two-noded linear works with either one or two Gauss points:

Table A.1 Gauss quadrature integration cases for two-noded linear elements. (Source: [1])

N_G	k	w_k	ξ_k
1	1	2	0
2	1	1	$-1/\sqrt{3}$
	2	1	$+1/\sqrt{3}$

In the latter table, N_G refers to the number of Gauss points, k to each Gauss point, w_k to the weight of the k-th Gauss point and ξ_k to the natural coordinate of the k-th Gauss point.

Bilinear Quadrilateral Element

As stated in the beginning of the section, the FEM shell discretization is based on the bilinear quadrilateral element. It is a four-noded element which allows to compute the structural properties by using a domain in where the element is a square of side two. Figure A.13 shows the domain representation and the node ordering of this element.

To begin with, the shape functions are

$$N^{(e,i)}(\xi,\eta) = \frac{1}{4}\left(1 + a_i\xi\right)\left(1 + b_i\eta\right) \tag{A.83}$$

where i refers to the number of the natural node and a_i and b_i are coefficients of the shape functions. Table A.2 shows the values depending on the number of node.

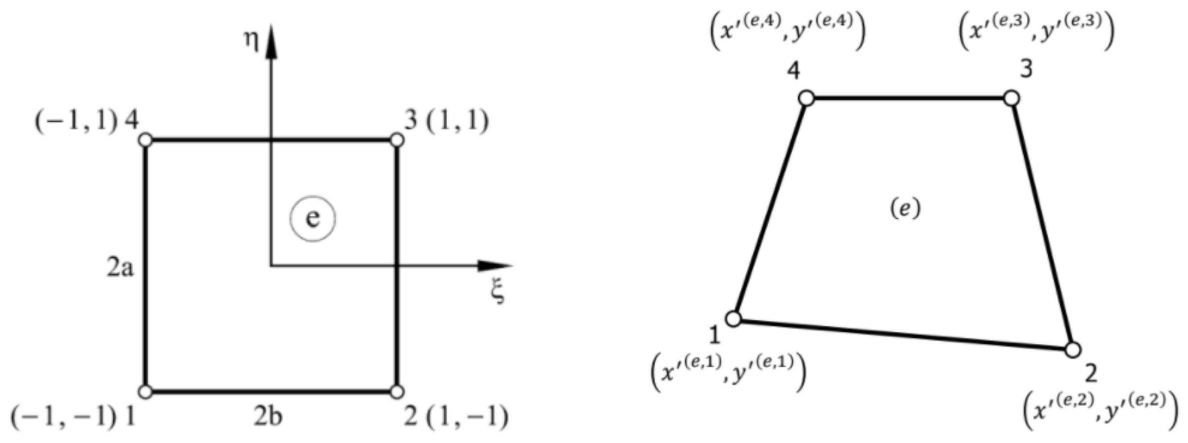

η
(−1, 1) 4
3 (1, 1)
e
2a
ξ
(−1, −1) 1
2b
2 (1, −1)
$(x'^{(e,4)}, y'^{(e,4)})$
4
$(x'^{(e,3)}, y'^{(e,3)})$
3
(e)
1
$(x'^{(e,1)}, y'^{(e,1)})$
2
$(x'^{(e,2)}, y'^{(e,2)})$

Figure A.13 Bilinear quadrilateral element variables and node ordering. (Source: [3])

Table A.2 Values of the coefficients for the shape functions at each node. (Source: [3])

i	**1**	**2**	**3**	**4**
a_i	-1	1	1	-1
b_i	-1	-1	1	1

Then, the shape functions natural derivatives and the Jacobian matrix of the element are

$$N^{(e,i)}_{,\xi}(\xi,\eta) = \frac{1}{4}a_i\left(1+b_i\eta\right) \qquad N^{(e,i)}_{,\eta}(\xi,\eta) = \frac{1}{4}b_i\left(1+a_i\xi\right) \tag{A.84}$$

$$\left[\mathcal{J}'^{(e)}\right] = \begin{bmatrix} \frac{\partial x'}{\partial \xi} & \frac{\partial y'}{\partial \xi} \\ \frac{\partial x'}{\partial \eta} & \frac{\partial y'}{\partial \eta} \end{bmatrix} = \sum_i \left\{ \begin{matrix} N^{(e,i)}_{,\xi} \\ N^{(e,i)}_{,\eta} \end{matrix} \right\} \overbrace{\left\{\hat{\boldsymbol{x}}^{(e,i)}\right\}^{\mathrm{T}} \left[\hat{\boldsymbol{i}}'^{(e)} \quad \hat{\boldsymbol{j}}'^{(e)}\right]}^{\left\{\widehat{\boldsymbol{x}'}^{(e,i)}\right\}^{\mathrm{T}}} \tag{A.85}$$

And, in the end, the local derivatives of the shape functions give

$$\left\{ \begin{matrix} N^{(e,i)}_{,x'} \\ N^{(e,i)}_{,y'} \end{matrix} \right\} = \left[\mathcal{J}'^{(e)}\right]^{-1} \left\{ \begin{matrix} N^{(e,i)}_{,\xi} \\ N^{(e,i)}_{,\eta} \end{matrix} \right\} \tag{A.86}$$

As seen in the two-noded linear element section, the integrals are evaluated using the Gauss quadrature integration. For this four-noded element, the procedure works with either one or four Gauss points. Table A.3 exposes the Gauss points weights and coordinates for each integration case.

Table A.3 Gauss quadrature integration cases for bilinear quadrilateral elements. (Source: [3])

N_G	k	w_k	ξ_k	η_k
1	1	4	0	0
4	1	1	$-1/\sqrt{3}$	$-1/\sqrt{3}$
	2	1	$+1/\sqrt{3}$	$-1/\sqrt{3}$
	3	1	$+1/\sqrt{3}$	$+1/\sqrt{3}$
	4	1	$-1/\sqrt{3}$	$+1/\sqrt{3}$

Notice that this table adds one column with respect to Table A.1. This is because the four-noded element uses a 2D domain while the two-noded one uses a 1D domain.

Appendix B

Aerodynamic Theory

The second appendix aims to define the aerodynamic model that will be coupled to the structural one, defined in the previous appendix. Both the theoretical and computational explanations are included.

B.1 Theory Basics

The following section aims to describe the basic concepts related to aerodynamics in this work. The fundamentals of the theory exposed apply for inviscid and incompressible flow, simplifying the model without losing accuracy in the results. Beginning with vortex filaments and the Biot-Savart law, the potential flow problem can be defined. However, the ultimate concept to be described is the Kutta-Joukowsky theorem and how to formulate the problem to analyse lifting surfaces.

B.1.1 Vortex quantities

Both subsection B.1.2 and this one are based on the theory slides [5].

To begin with, a vortex filament of constant strength Γ is presented in Figure B.1, where Γ is the circulation, $\boldsymbol{dl}$ the length differential of the filament and $\boldsymbol{r}$ the distance between the filament and point P (where the velocity wants to be found, thus the velocity differential $d\boldsymbol{V}$ is applied there).

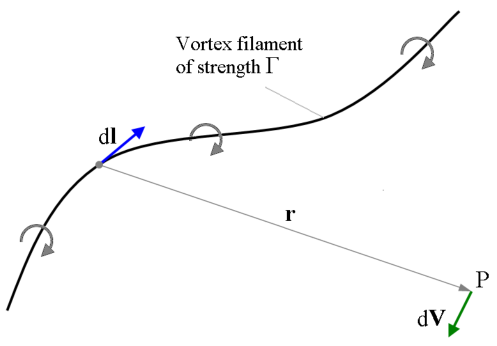


Figure B.1 Vortex filament of strength Γ. (Source: [6])

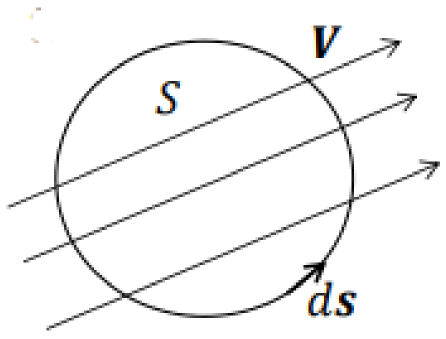


Figure B.2 Closed curve C enclosing a simply connected surface S in the flow field. The differential d$\boldsymbol{s}$ is analogous to d$\boldsymbol{l}$. (Source: [5])

Circulation is an important concept to introduce, Figure B.2 depicts a closed curved which can help to define the concept.

The equation defining circulation is, making use of Stoke's theorem,

$$\int_S (\nabla \times \boldsymbol{V}) \cdot \hat{\boldsymbol{n}}\, dS = \int_S \boldsymbol{w} \cdot \hat{\boldsymbol{n}}\, dS = \oint_C \boldsymbol{V} \cdot d\boldsymbol{l} = -\Gamma^1 \tag{B.1}$$

To model a lifting surface using vortex filaments, it is needed to study the 3D approach and applications of those. They can be defined as

$$\boldsymbol{w} \times d\boldsymbol{l} = 0 \tag{B.2}$$

where $\boldsymbol{w}$ refers to the vorticity[2]. As vorticity is equal to the rotational of the velocity field, the divergence of it gives

$$\nabla \cdot \boldsymbol{w} = \nabla \cdot (\nabla \times \boldsymbol{w}) = 0 \tag{B.3}$$

which shows that the vorticity is divergence free. Therefore, applying the divergence theorem

$$\int_V \nabla \cdot \boldsymbol{w}\, dV = \oint_{S=\partial V} \boldsymbol{w} \cdot \hat{\boldsymbol{n}}\, dS = 0 \tag{B.4}$$

one can demonstrate that the circulation is constant along a vortex filament following the example of a vortex tube:

- Consider the vortex tube in Figure B.3
- Apply equation B.4 to get

$$\oint_{S=\partial V} \boldsymbol{w} \cdot \hat{\boldsymbol{n}}\, dS = -\oint_{S=\partial V} \boldsymbol{w} \cdot \hat{\boldsymbol{n}}_1\, dS = +\oint_{S=\partial V} \boldsymbol{w} \cdot \hat{\boldsymbol{n}}_2\, dS = 0 \tag{B.5}$$

[1] By mathematical convention, the line integrals are positive in the counterclockwise direction. However, in aerodynamics it is convenient to consider circulation positive in the clockwise sense. Hence, the minus sign [5].

[2] In fluid mechanics vorticity is used to quantify the rotation of a fluid.

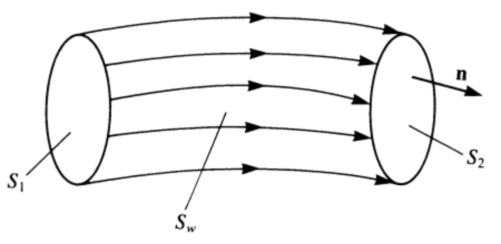


Figure B.3 Vortex tube made of vortex filaments. (Source: [7])

- Reorganize the terms

$$\oint_{S=\partial V} \boldsymbol{w} \cdot \hat{\boldsymbol{n}}_1 \, dS = + \oint_{S=\partial V} \boldsymbol{w} \cdot \hat{\boldsymbol{n}}_2 \, dS \tag{B.6}$$

- Define the circulation and prove

$$\Gamma = \oint_{S=\partial V} \boldsymbol{w} \cdot \hat{\boldsymbol{n}} \, dS = \Gamma_1 = \Gamma_2 = \text{constant} \tag{B.7}$$

Applying this last equation to a vortex filament, with $dS \to 0$, the following is obtained

$$\Gamma = w \, dS = \text{constant} \tag{B.8}$$

indicating that the vortex line cannot end in the fluid because if the circulation is constant then $w \to \infty$.

The results obtained above prove the first two out of three Helmholtz's theorems, named after Hermann von Helmholtz, an important German physician in the 19th century. These say:

- The strength of a vortex filament is constant along its length.
- A vortex filament cannot start or end in the fluid; it must form a closed path or extend to infinite.

B.1.2 The Biot-Savart law

In this subsection, the Biot-Savart law is demonstrated, obtaining a key result to apply in the aerodynamic vortex methods.

The mass conservation equation states

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i} (\rho \, u_i) = 0 \tag{B.9}$$

and if we make the hypothesis of steady incrompressible fluid it gives

$$\frac{\partial}{\partial x_i} (\rho \, u_i) = \frac{\partial \rho}{\partial x_i} u_i + \rho \frac{\partial u_i}{\partial x_i} = 0 \longrightarrow \frac{\partial u_i}{\partial x_i} = \nabla \cdot \boldsymbol{V} = 0 \tag{B.10}$$

Then, considering this steady incompressible fluid, the velocity field due to a vortex line comes from an unknown vector field, for example, **B**

$$\boldsymbol{V} = \nabla \times \boldsymbol{B} \tag{B.11}$$

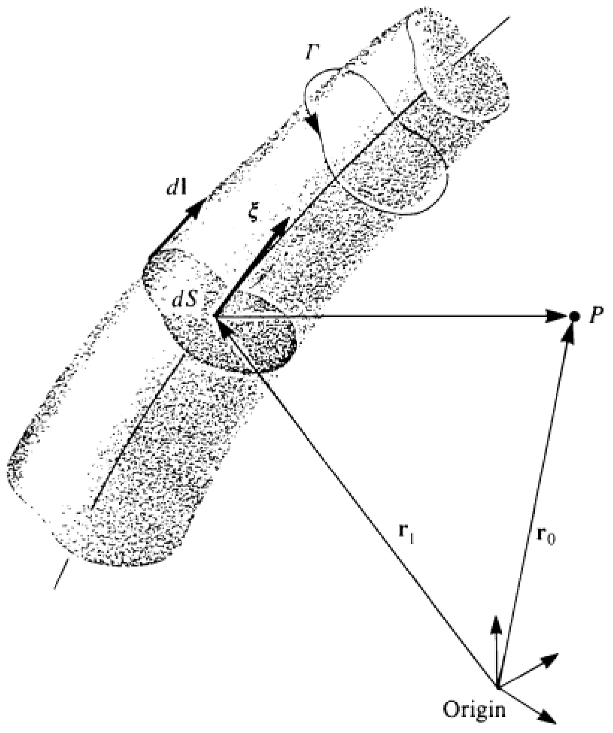


Figure B.4 Velocity at point P induced by a vortex segment. (Source: [7])

The vector field **B** is defined with the following condition

$$\nabla \cdot \boldsymbol{B} = 0 \tag{B.12}$$

Next, using the vorticity formula and applying the last equation in it, one obtains

$$\boldsymbol{w} = \nabla \times \boldsymbol{V} = \nabla \times (\nabla \times \boldsymbol{B}) = \nabla(\nabla \cdot \boldsymbol{B}) - \nabla^2 \boldsymbol{B} \longrightarrow \boldsymbol{w} = -\nabla^2 \boldsymbol{B} \tag{B.13}$$

which is a Poisson's problem[3] to find **B**. The equation can be solved using the Green's theorem (see [8], pp. 532-534), giving

$$\boldsymbol{B} = \frac{1}{4\pi} \int_V \frac{\boldsymbol{w}}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|} dV \tag{B.14}$$

where **B** is the value of the function at a point **P** due to vorticity and inside a volume containing the vortex line (see Figure B.4).

Now that **B** is known, making use of Equation B.11 one can find the induced velocity

$$\boldsymbol{V} = \frac{1}{4\pi} \int_V \nabla \times \frac{\boldsymbol{w}}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|} dV \tag{B.15}$$

[3]A Poisson's problem is defined as an equation in which the Lagrange operator of a vector, $\Delta \boldsymbol{g} = \nabla^2 \boldsymbol{g}$, equals another unknown function. It can be solved using the Green's theorem.

The last step to find the expression of the Biot-Savart law introduces the following assumptions

$$\boldsymbol{dl} = \frac{\boldsymbol{w}}{w} dl \qquad dV = dS\, dl \qquad \Gamma = w\, dS \tag{B.16}$$

which transform the integrand of the induced velocity equation into

$$\nabla \times \frac{\boldsymbol{w}}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|} dV = \nabla \times \frac{w\, \boldsymbol{dl}/dl\, dS dl}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|} = \nabla \times \frac{w\, \boldsymbol{dl}\, \Gamma/w}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|} = \nabla \times \frac{\Gamma\, \boldsymbol{dl}}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|} = \Gamma \frac{\boldsymbol{dl} \times (\boldsymbol{r}_0 - \boldsymbol{r}_1)}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|^3} \tag{B.17}$$

Finally, introducing the last equation into the integral computed previously, the Biot-Savart law states

$$\boldsymbol{V} = \frac{\Gamma}{4\pi} \int_V \frac{\boldsymbol{dl} \times (\boldsymbol{r}_0 - \boldsymbol{r}_1)}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|^3} \tag{B.18}$$

which in differential form reads

$$d\boldsymbol{V} = \frac{\Gamma}{4\pi} \frac{\boldsymbol{dl} \times (\boldsymbol{r}_0 - \boldsymbol{r}_1)}{|\boldsymbol{r}_0 - \boldsymbol{r}_1|^3} \tag{B.19}$$

This law is used in the computational code when calculating the aerodynamic influence coefficients matrix as the induced velocity at each collocation point is needed.

B.1.3 The potential flow problem

From this point to the end of the aerodynamic theory model the development can be followed in [7].

Definition of the Problem

Because of the type of flow which is selected in this aerodynamic formulation, it is necessary to define the potential flow problem.

On the one hand, as seen previously, the governing equations if the flow in the fluid region is considered to be steady, inviscid, incompressible and irrotational[4] are

$$\nabla \cdot \boldsymbol{V} = 0 \tag{B.20}$$

$$\boldsymbol{w} = (\nabla \times \boldsymbol{V}) = 0 \tag{B.21}$$

On the other hand, introducing a scalar potential function, Φ, from which the velocity, $\boldsymbol{V}$, is obtained thanks to the gradient operation

$$\boldsymbol{V} = \nabla\Phi \tag{B.22}$$

[4]The hypothesis does not impede the flow to be compressible.

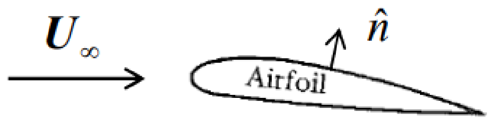


Figure B.5 Nomenclature used to define the potential flow problem. (Source: [5])

and combining it with Equation B.20, one gets a Laplace's equation[5]

$$\nabla^2\Phi = 0 \tag{B.23}$$

This gradient of the potential function also satisfies Equation B.21 because

$$\nabla \times \nabla\Phi = 0 \tag{B.24}$$

Equation B.23 can be solved defining suitable boundary conditions and, when the solution is found, the velocity field is easily computed. The pressure is obtained using Bernoulli's equation.

The boundary conditions can be of Dirichlet, Neumann or mixed. Typically, in aerodynamic problems the following are used:

- **Freesteam condition:** The flow approaches freestream conditions far away from the body[6].

$$\boldsymbol{x} \to \infty : \nabla\Phi = \boldsymbol{U}_\infty \tag{B.25}$$

- **Wall conditions:** For a body inside the fluid, the normal component (to the body's surface and to other solid boundaries) of the velocity must be zero.

$$\nabla\Phi \cdot \hat{\boldsymbol{n}} = 0 \tag{B.26}$$

Vortex Solution

The potential problem accepts other solutions apart from the general ones of source and doublet. In this case, the procedure to find one solution based on the vortex flow and using the Biot-Savart law is explained.

The singularity element, as shown in Figure B.6a, has only a tangential velocity component. Therefore

$$u_r = 0$$
$$u_\theta = u_\theta(r, \theta)$$

Substituting these velocity components in the continuity equation in cylindrical coordinates for an incom-

[5]A Laplace's equation is a 2nd order PDE (Partial Differential Equation) of elliptic type. It is a boundary value problem, boundary conditions must be defined for all the domain boundaries.
[6]It is not necessary that the flow is aligned with the x direction.

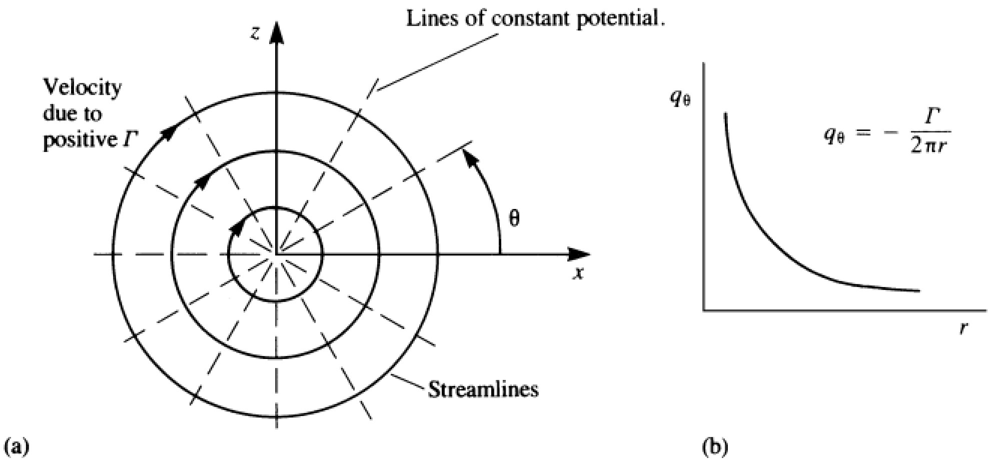

Lines of constant potential.
Velocity due to positive Γ
z
θ
x
Streamlines
(a)
q_θ
$q_\theta = -\frac{\Gamma}{2\pi r}$
r
(b)

Figure B.6 (a) Streamlines and equipotential lines for a 2D vortex. (b) Radial variation of the tangential velocity component induced by a vortex. (Source: [7])

pressible fluid

$$\frac{\partial u_r}{\partial r} + \frac{1}{r}\frac{\partial u_\theta}{\partial \theta} + \frac{\partial u_x}{\partial x} + \frac{u_r}{r} = 0 \tag{B.27}$$

it is proved that

$$u_\theta = u_\theta(r)$$

which means that the tangential velocity is constant along the θ variation.

Now, being the fluid irrotational, the vorticity equation reads

$$\boldsymbol{w} = \nabla \times \boldsymbol{V} = \begin{pmatrix} \frac{\partial}{\partial r} \\ \frac{1}{r}\frac{\partial}{\partial \theta} \\ \frac{\partial}{\partial x} \end{pmatrix} \times \begin{pmatrix} u_r \\ u_\theta \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{\partial u_\theta}{\partial x} \\ \frac{\partial u_r}{\partial x} \\ \frac{\partial u_\theta}{\partial r} - \frac{1}{r}\frac{\partial u_r}{\partial \theta} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{\partial u_\theta}{\partial r} - \frac{1}{r}\frac{\partial u_r}{\partial \theta} \end{pmatrix} = 0$$

which means

$$w_y = \frac{\partial u_\theta}{\partial r} - \frac{1}{r}\frac{\partial u_r}{\partial \theta} = -\frac{1}{r}\left(\frac{\partial}{\partial r}(r\,u_\theta) - \frac{\partial}{\partial \theta}(u_r)\right) = -\frac{1}{r}\frac{\partial}{\partial r}(r\,u_\theta) = 0$$

Integrating with respect to r one gets

$$r\,u_\theta = \text{constant} = A \tag{B.28}$$

meaning that the tangential velocity is inversely proportional to the radius, as depicted in Figure B.6b.

Thanks to the definition of the circulation (see Equation B.1), the value of the constant A can be found[7]

$$\Gamma = -\oint \boldsymbol{V} \cdot d\boldsymbol{l} = \int_{2\pi}^{0} = u_\theta \cdot r\,d\theta = -2\pi\,A \longrightarrow A = -\frac{\Gamma}{2\pi}$$

The velocity field is then

$$\begin{aligned} u_r &= 0 \\ u_\theta &= -\frac{\Gamma}{2\pi\,r} \end{aligned} \tag{B.29}$$

[7]Remember that the positive direction of the circulation is clockwise, contrary to θ evolution.

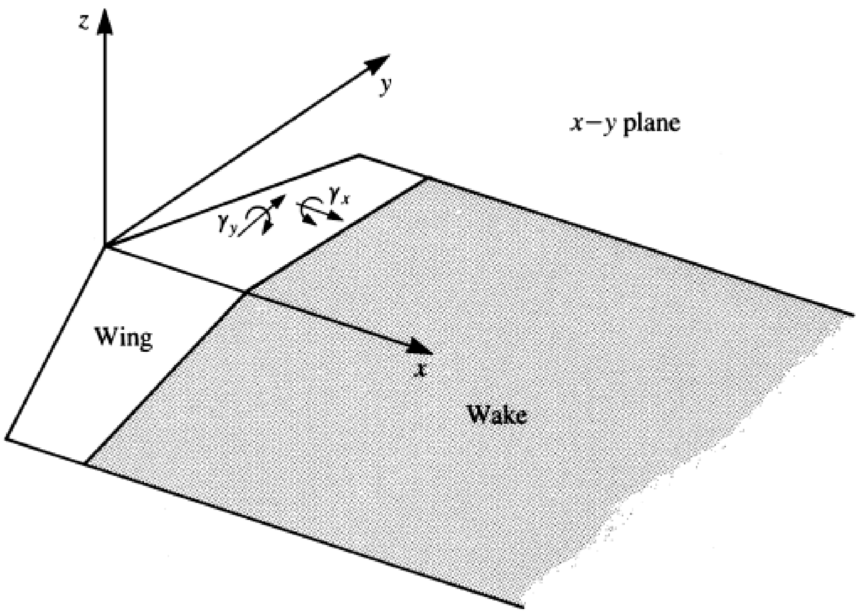


Figure B.7 Lifting surface model of a 3D wing. (Source: [7])

Finally, integrating the velocity field equations the velocity potential for a vortex element is found

$$\Phi = \int u_\theta \, r \, d\theta + C = -\frac{\Gamma}{2\pi}\theta + C \tag{B.30}$$

where the constant C is arbitrary and can be set to zero. Notice that this result is obtained using cylindrical coordinates and it is immediate to transform it into Cartesian ones. For a vortex located at (x_0, z_0), the velocity components look like

$$\begin{aligned} u_x &= \frac{\Gamma}{2\pi}\frac{z - z_0}{(z - z_0)^2 + (x - x_0)^2} \\ u_z &= -\frac{\Gamma}{2\pi}\frac{x - x_0}{(z - z_0)^2 + (x - x_0)^2} \end{aligned} \tag{B.31}$$

and the velocity potential

$$\Phi = -\frac{\Gamma}{2\pi}\tan^{-1}\frac{z - z_0}{x - x_0} + C \tag{B.32}$$

being C the arbitrary constant mentioned before, which can be set to zero.

B.1.4 Zero-thickness cambered wing at AoA–lifting surfaces

Now that the basic aerodynamic concepts are already introduced, it is time to formulate the problem adapted to a finite wing. The element treated as a wing is called lifting surface and it is depicted in Figure B.7.

The problem to be solved is

$$\nabla^2\Phi = 0 \tag{B.33}$$

with the boundary condition that requires no flow across the surface (in this cased evaluated at z = 0) shown in Figure B.8

$$\frac{\partial\Phi}{\partial z}(x, y, 0\pm) = Q_\infty\left(\frac{\partial\eta_c}{\partial x} - \alpha\right) \tag{B.34}$$

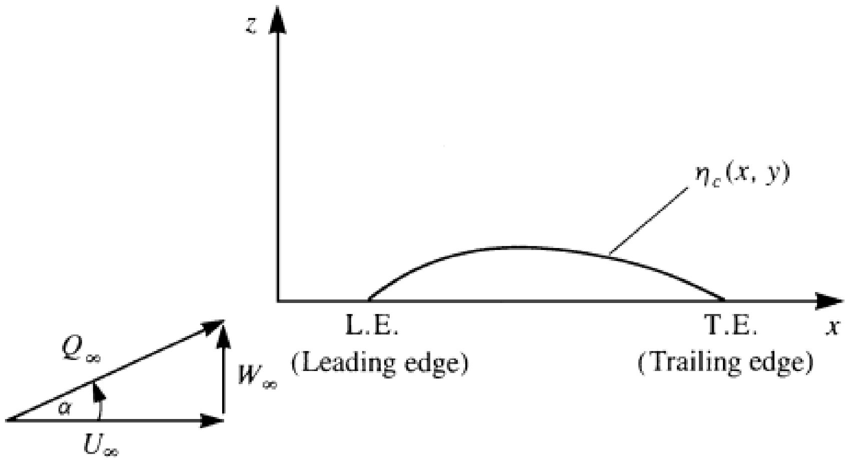


Figure B.8 Nomenclature used for the definition of the lifting wing problem. (Source: [7])

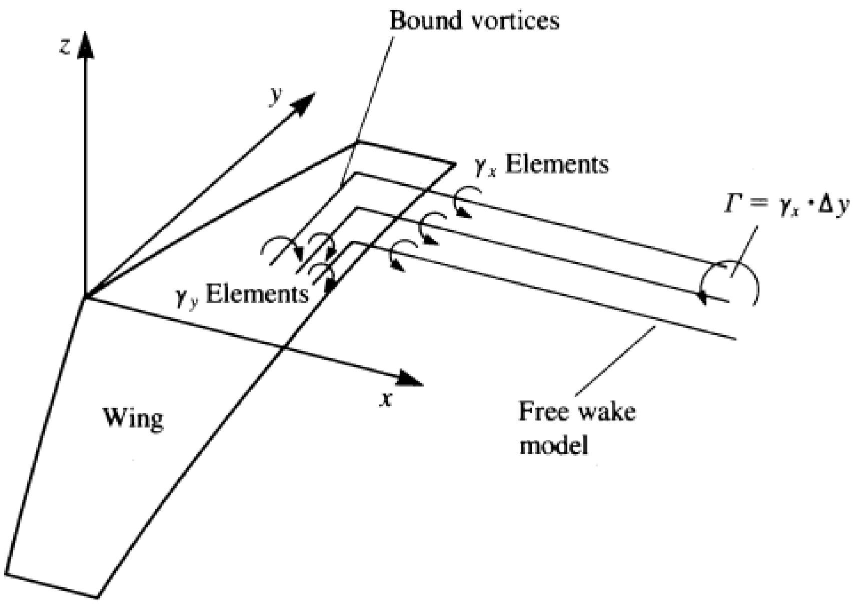


Figure B.9 Vortex representation for the lifting surface model. (Source: [7])

Because of the small-disturbance approximation the wake is also considered to be planar (placed on the z = 0 plane).

As the aerodynamic model is based on vortex distribution, this zero-thickness cambered wing at AoA lifting surface problem considers only the vortex solution. Vortex line distributions are used over the wing and the wake. By considering the Biot-Savart law in differential form,

$$d\boldsymbol{V} = \frac{-1}{4\pi} \frac{d\Gamma \boldsymbol{r} \times d\boldsymbol{l}}{r^3} \tag{B.35}$$

where $d\boldsymbol{V}$ is the velocity due to a vortex line element $d\boldsymbol{l}$ with a strength of $d\Gamma$, one can get the component of velocity normal to the wing (downwash).

This downwash is induced by vortices distributed over the wing and wake as in Figure B.9, where the elements

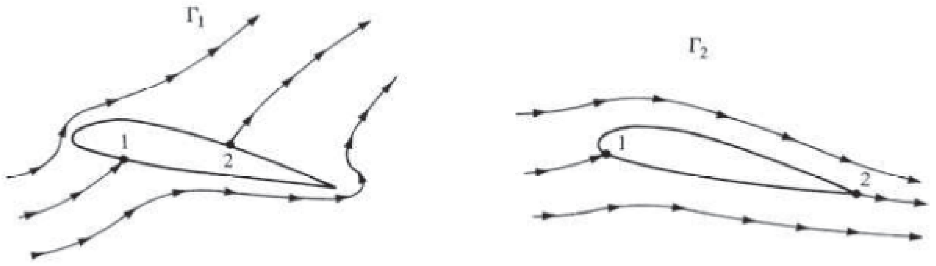


Figure B.10 Effect of different values of circulation on the potential flow over a given airfoil at a given angle of attack. (Source: [6])

that point in the y direction are denoted as γ_y[8] and the ones pointing in the x direction are denoted as γ_x.

$$u_z\left(x,y,z\right)=\frac{-1}{4\pi}\int_{\text{wing+wake}}\frac{\gamma_y(x-x_0)-\gamma_x(y-y_0)}{r^3}dx_0\,dy_0 \tag{B.36}$$

Notice that thee are two unknowns (γ_x and γ_y) and one equation. However, thanks to the first Helmholtz theorem in subsection B.1.1 vortex the strength of a vortex filament is constant along its length. This way, considering that the wing consist of numerous infinitesimal vortex lines then at any point

$$\left|\frac{\partial\gamma_x}{\partial x}\right|=\left|\frac{\partial\gamma_y}{\partial y}\right| \tag{B.37}$$

and the unknowns reduce to only one.

Finally, to construct the lifting surface equation for the unknown $\gamma=\gamma_x=\gamma_y$ the wing induced downwash (Equation B.36) must be equal and opposite in sign to to the normal component of the freestream velocity (Equation B.34)

$$\frac{-1}{4\pi}\int_{\text{wing+wake}}\frac{\gamma_y(x-x_0)-\gamma_x(y-y_0)}{\left[(x-x_0)^2+(y-y_0)^2\right]^{3/2}}dx_0\,dy_0=Q_\infty\left(\frac{\partial\eta_c}{\partial x}-\alpha\right) \tag{B.38}$$

This result allows to compute the velocity field distribution.

B.1.5 Vortex wake

Until now the aerodynamic formulation has focused on finding the zero normal flow boundary condition on solid surfaces such as the wing. However, in some cases the solution is not unique, different values of circulation can be selected thus the flow distribution varies (see Figure B.10). This is where the German mathematician Kutta appears, to define an additional restriction, with his well-know *Kutta condition*:

The flow leaves the sharp trailing edge of an airfoil smoothly and the velocity there is finite.

The last statement can be interpreted in several ways. One for example defines that the flow must leave the trailing edge (TE) smoothly. Figure B.11 depicts two shapes of the TE and their relation to the Kutta condition:

[8]The variable γ is known as the circulation density and it is computed dividing the total circulation of an airfoil, for example, by its length.

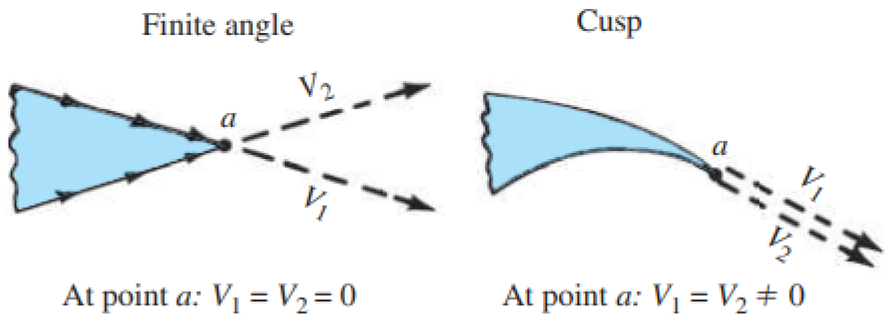

Finite angle
Cusp
V_2
a
V_1
a
V_1
V_2
At point a: $V_1 = V_2 = 0$
At point a: $V_1 = V_2 \neq 0$

Figure B.11 Different possible shapes of the trailing edge and their relation to the Kutta condition. (Source: [6])

- **Finite angle:** The flow leaves the TE along the bisector line there. At point a there needs to be a stagnation point (velocity equal to zero) so that the normal component of velocity vanishes adding both sides of the airfoil.
- **Cusp:** As both velocities at point a are in the same direction, they can be finite. Using Bernoulli's equation, as the pressure in a is unique, both velocities are the same.

The Kutta condition implies $\Delta p_{TE} = 0$ and if the circulation is modeled by vortex distribution then, for both the finite angle and the cusp,

$$\gamma_{TE} = V_1 - V_2 = 0 \tag{B.39}$$

Now, considering the wake is modeled by a vortex sheet, it must not create loads. The pressure difference across the sheet is obtained using

$$\Delta p = \rho \boldsymbol{V} \times \boldsymbol{\gamma} \tag{B.40}$$

where $\boldsymbol{\gamma} = (\gamma_x, \gamma_y, \gamma_z)$. The pressure difference equals to zero because of the Kutta condition, meaning that the velocity on the wake is parallel to the wake vortices.

This result will be very useful when introducing the computational approach of the aerodynamic model.

B.1.6 The Kutta-Joukowsky theorem

The KJ theorem relates the flow circulation to the lift produced by a body, so it is one of the most important theorems that apply to computational aerodynamics. When having the matrix of flow circulations it is immediate to compute the lift force and thus the induced drag[9].

This theorem is proved from the flow in a rotating cylinder, depicted in Figure B.12. If the cylinder is static, there is symmetry over the horizontal and vertical planes thus the lift and drag are zero (no aerodynamic force). However, if a point vortex of strength Γ is added by the principle of superposition, the symmetry over the horizontal plane disappears. Then, the drag continues to be zero but, thanks to the mentioned asymmetry, a lift net force appears.

This means that circulation added to a flow creates lift, which is the heart of the KJ theorem. Now, this

[9]Induced drag is the component of drag related to the generation of lift.

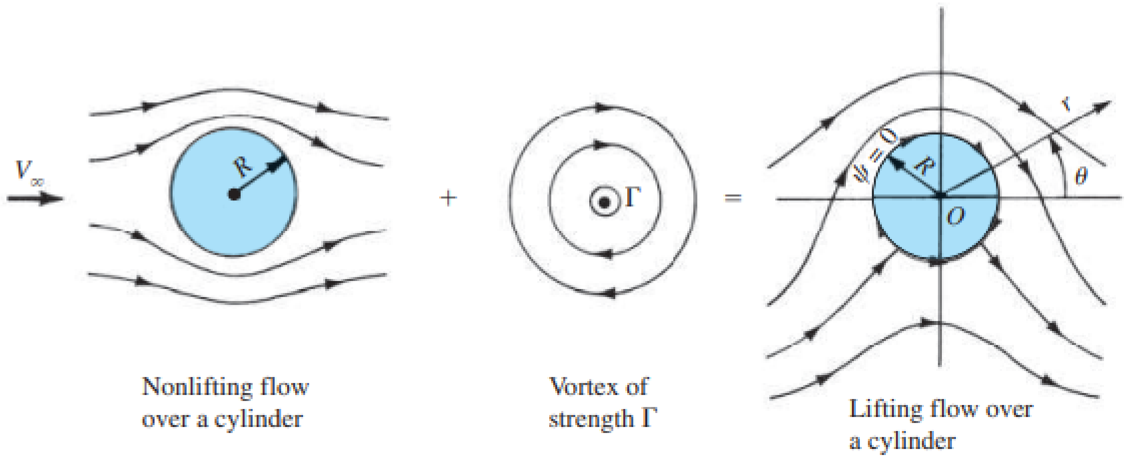

V_∞
R
+
Γ
=
$\psi = 0$
R
r
θ
O
Nonlifting flow over a cylinder
Vortex of strength Γ
Lifting flow over a cylinder

Figure B.12 Synthesis of lifting flow over a circular cylinder. (Source: [6])

relation between circulation and velocity is to be found using the rotating cylinder case.

The starting point are the flow equations of velocity over a rotating cylinder (see [6], pp. 291-293)

$$\begin{aligned} V_r &= \left(1 - \frac{R^2}{r^2}\right) V_\infty \cos\theta \\ V_\theta &= -\left(1 + \frac{R^2}{r^2}\right) V_\infty \sin\theta - \frac{\Gamma}{2\pi r} \end{aligned} \tag{B.41}$$

The idea is to obtain the lift from the pressure and the pressure from the velocity working with appropriate dimensionless coefficients.

- **Step 1:** Obtain pressure from velocity

  The pressure coefficient is defined as

  $$C_P = \frac{\Delta P}{\frac{1}{2}\rho V_\infty^2} \tag{B.42}$$

  and as the flow is elementary and inviscid the Bernoulli equation can be used to relate pressure with velocity. Between two points on a streamline, one in the free stream region and the other close to the cylinder, it states

  $$P_\infty + \frac{1}{2}\rho V_\infty^2 = P + \frac{1}{2}\rho V^2 \tag{B.43}$$

  and rearranging

  $$\Delta P = P - P_\infty = \frac{1}{2}\rho(V_\infty^2 - V^2)$$

  Going back to the pressure coefficient definition, it can be defined solely in terms of speed

  $$C_P = \frac{\frac{1}{2}\rho(V_\infty^2 - V^2)}{\frac{1}{2}\rho V_\infty^2} = \frac{V_\infty^2 - V^2)}{V_\infty^2} = 1 - \left(\frac{V}{V_\infty}\right)^2 \tag{B.44}$$

  Notice that the pressure distribution is defined over the cylinder surface so the velocity would be considered also over the same surface ($r = R$).

- **Step 2:** Turn the pressure distribution into a force, the lift force

Normal and axial aerodynamic forces are computed by integrating the pressure and shear distributions on the upper and lower surfaces of the body

$$\begin{aligned} N' &= -\int_{LE}^{TE} \left(P_u \cos\theta + \tau_u \sin\theta\right) dsu + \int_{LE}^{TE} \left(P_l \cos\theta - \tau_l \sin\theta\right) ds_l \\ A' &= \int_{LE}^{TE} \left(-P_u \sin\theta + \tau_u \cos\theta\right) ds_u + \int_{LE}^{TE} \left(P_l \sin\theta + \tau_l \cos\theta\right) ds_l \end{aligned} \tag{B.45}$$

Considering the flow is inviscid all the shear components can be removed, giving

$$\begin{aligned} N' &= -\int_{LE}^{TE} P_u \cos\theta \, ds_u + \int_{LE}^{TE} P_l \cos\theta \, ds_l \\ A' &= \int_{LE}^{TE} -P_u \sin\theta \, ds_u + \int_{LE}^{TE} P_l \sin\theta \, ds_l \end{aligned} \tag{B.46}$$

Now, by expressing the upper and lower surface differentials in Cartesian coordinates

$$dx = \cos\theta \, ds \qquad dy = -\sin\theta \, ds$$

and the axial and normal forces in terms of the aerodynamic ones

$$N' = L' \qquad A' = D'$$

one can write the integrals in Equation B.46 as

$$\begin{aligned} L' &= -\int_{LE}^{TE} P_u \, dx + \int_{LE}^{TE} P_l \, dx = \int_{LE}^{TE} \left(P_l - P_u\right) \, dx \\ D' &= \int_{LE}^{TE} P_u \, dy - \int_{LE}^{TE} P_l \, dy = \int_{LE}^{TE} \left(P_u - P_l\right) \, dy \end{aligned} \tag{B.47}$$

The idea is to work with dimensionless coefficients so, as the body is a cylinder, the characteristic length used will be its diameter D. This way[10]

$$\begin{aligned} c_l &= \frac{1}{D} \int_{LE}^{TE} \left(C_{P_l} - C_{P_u}\right) \, dx \\ d_d &= \frac{1}{D} \int_{LE}^{TE} \left(C_{P_u} - C_{P_l}\right) \, dy \end{aligned} \tag{B.48}$$

Transforming another time from Cartesian coordinates to cylindrical ones so to evaluate the integrals, considering $D = 2R$,

$$y = R \sin\theta \longrightarrow dy = R \cos\theta \, d\theta$$

$$x = R \cos\theta \longrightarrow dx = -R \sin\theta \, d\theta$$

[10]The lift and drag coefficients are defined per unit span.

the integrals state

$$\begin{aligned} c_l &= -\frac{1}{2}\int_{\pi}^{2\pi} C_{P_l}\sin\theta\, d\theta + \frac{1}{2}\int_{\pi}^{0} C_{P_u}\sin\theta\, d\theta \\ c_d &= \frac{1}{2}\int_{\pi}^{0} C_{P_u}\cos\theta\, d\theta - \frac{1}{2}\int_{\pi}^{2\pi} C_{P_l}\cos\theta\, d\theta \end{aligned} \tag{B.49}$$

At this point, noting that C_{P_u} and C_{P_l} are given by the same expression (Equation B.44 extended),

$$C_P = 1 - \left(\frac{V_{theta}(r=R)}{V_\infty}\right)^2 = 1 - \left(-2\sin\theta - \frac{\Gamma}{2\pi\, RV_\infty}\right)^2 \tag{B.50}$$

the integrals can be evaluated directly as one going from 0 to 2π,

$$\begin{aligned} c_l &= -\frac{1}{2}\int_{0}^{2\pi} C_{P_l}\sin\theta\, d\theta \\ c_d &= -\frac{1}{2}\int_{0}^{2\pi} C_{P_u}\cos\theta\, d\theta \end{aligned} \tag{B.51}$$

With several trigonometric integrals the results obtained are

$$\begin{aligned} c_l &= \frac{\Gamma}{RV_\infty} \\ c_d &= 0 \end{aligned} \tag{B.52}$$

which prove the hypothesis that, due to symmetry over the vertical axis, there is no drag force.

Finally, writing the dimensional lift force, the KJ theorem is defined[11]

$$c_l = \frac{L'}{\frac{1}{2}\rho V_\infty^2 2R} \longrightarrow L' = \frac{1}{2}\rho V_\infty^2 2RC_l = \frac{1}{2}\rho V_\infty^2 2R\frac{\Gamma}{RV_\infty} = \rho V_\infty \Gamma \tag{B.53}$$

Although this theorem is derived from the case of a rotating cylinder, it works for flows over bodies with arbitrary shape, making it a very powerful tool in aerodynamics.

B.2 Computational Approach

This next section explains how aerodynamics are programmed, putting focus on vortex methods. A general introduction to numerical panel methods opens the development, being followed by two singularity elements and their respective vortex computational formulations, the Horseshoe Vortex Method (HVM) and the Vortex Lattice Method (VLM).

B.2.1 Numerical panel methods

The numerical panel methods are introduced and developed in [7]. The basic methodology is picked from the corresponding chapters of the book.

In the theory section explained previously the solution to the potential flow problem is obtained using

[11]This result was found independently by Kutta, a German mathematician, and Joukowsky, a Russian physicist at the beginning of 20th century.

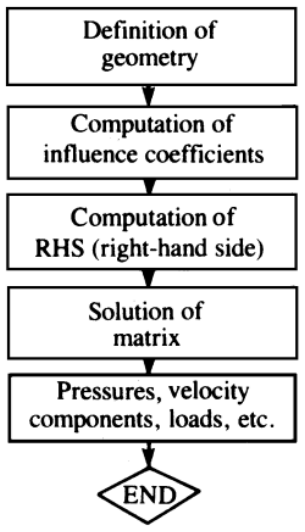

Definition of geometry
Computation of influence coefficients
Computation of RHS (right-hand side)
Solution of matrix
Pressures, velocity components, loads, etc.
END

Figure B.13 Flowchart for the numerical solution of the surface singularity distribution problem. (Source: [7])

analytical techniques. Therefore, a lot of simplifications are needed, both in the geometry and the boundary conditions. However, the application of numerical techniques allows expanding the cases to more real approaches, letting the analysis to be more precise.

The numerical methods that apply to this work are based on the surface distribution of singularity elements, with the objective of finding the strength of these elements. Figure B.13 defines the general steps to be followed when solving a numerical panel problem, once the singularity element is already chosen.

One by one, the steps can be explained shortly like:

- **Selection of the singularity element:** As mentioned previously, in this work, both of the methods exposed base their formulation on vortex flow. The order of the problem needs to be also defined and, once it is, an influence routine needs to be established. This routine outputs the velocity components and the potential induced by the element.
- **Definition of the geometry:** Once the basic solution element is selected, the geometry of the problem (the body) needs to be discretized so that it consists of those basic solution elements. The corner points of each element, along with its collocation point, are defined in this step too. It is important to remember that choosing an adequate grid can optimize the solution, enabling faster convergence.
- **Computation of influence coefficients:** For each of the elements, at the collocation point, the influence coefficient is calculated via a loop routine. A unit singularity strength is assumed and the equation is derived from the boundary conditions.
- **Computation of RHS:** The RHS of the matrix equations is known. It normally appears the free-stream velocity and other geometric data such as the AoA.
- **Solution of the linear set of equations:** Once the influence coefficients and the RHS of the matrix equation are known, the whole set can be calculated making use of standard numerical solvers.

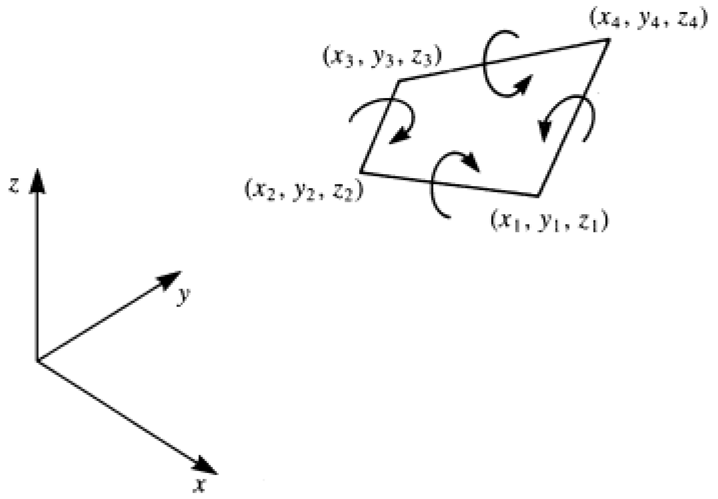


Figure B.14 Influence of a rectilinear vortex ring. (Source: [7])

- **Secondary computations:** The solution of the matrix equation gives the velocity field and the singularity strengths, which were assumed unitary at first. Secondary parameters can be computed then, like the pressure distribution using the Bernoulli equation or the aerodynamic loads by adding all the elements' values.

In the following subsections, the singularity elements considered in this work are to be further explained, as well as the vortex methods that apply to each of them.

B.2.2 Singularity elements

As previously mentioned, the solution of potential flow problems is based on the distribution of elementary solutions, which are obtained by imposing the zero normal flow condition on the solid boundaries. Therefore, this subsection puts emphasis on two typical numerical 3D elements, which are the key to the two numerical solutions exposed later.

Vortex Ring

On the one hand, the vortex ring is a quadrilateral element in which the induced velocity at a point is the addition of the four segments (see Figure B.14). The routine to compute the components of the just mentioned induced velocity inputs the coordinates of the element and the vortex line strength, which is assumed unitary at first.

Horseshoe Vortex

On the other hand, the horseshoe vortex is a simplified case of the vortex ring. As depicted in Figure B.15, the vortex line is placed on the xy plane; the two vortex segments are parallel to the x-axis while the finite-length vortex is parallel to the y-axis.

The induced velocity in the xy plane only has a component in the negative z direction. For a straight vortex

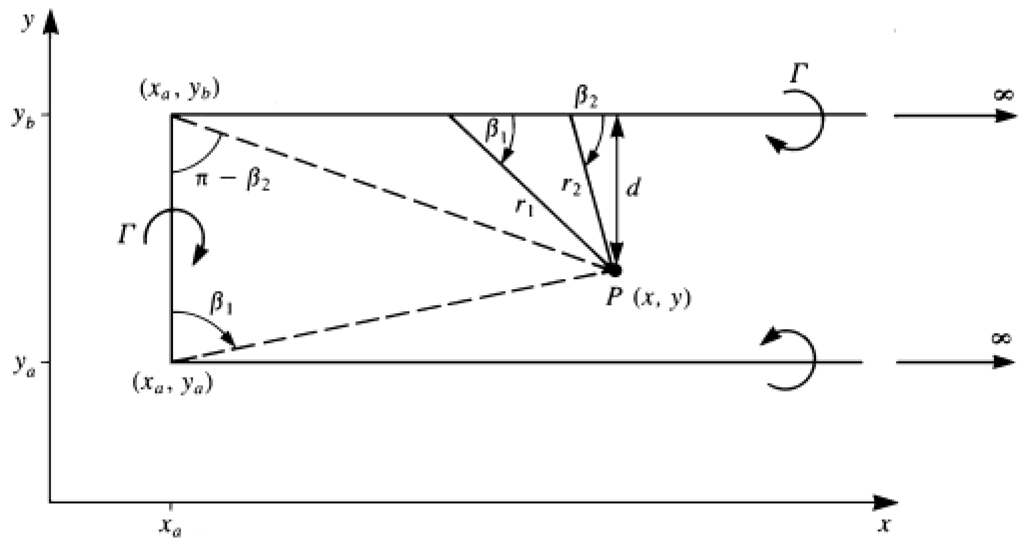


Figure B.15 Representation of a horseshoe vortex element. (Source: [7])

segment, as in this case, it can be computed using an extension of the Biot-Savart law (see [7], pp. 54-55)

$$u_z(x, y, 0) = -\frac{\Gamma}{4\pi\, d} (\cos\beta_1 - \cos\beta_2) \tag{B.54}$$

where the negative sign is a result of the velocity pointing in the $-z$ direction.

Using this last equation in each of the three straight segment one gets the total vertical induced velocity in the horseshoe element

$$\begin{aligned} u_z(x, y, 0) = &\frac{-\Gamma}{4\pi\,(y - y_a)} \left[1 + \frac{\sqrt{(x - x_a)^2 + (y - y_a)^2}}{x - x_a}\right] + \\ &\frac{\Gamma}{4\pi\,(y - y_b)} \left[1 + \frac{\sqrt{(x - x_a)^2 + (y - y_b)^2}}{x - x_a}\right] \end{aligned} \tag{B.55}$$

This result is programmed in a routine to compute the total induced velocity at each collocation point in the Horseshoe Vortex Method.

B.2.3 The Horseshoe Vortex Method

The HVM solves a lifting line problem of a finite wing by horseshoe elements. It is one of the simplest aerodynamic computational methods b, based on the analytical Prandtl's lifting line model), but it can include geometric effects such as wing taper, wing sweep or dihedral. Although the following resolution only considers one chordwise vortex, it can be easily extended to include more of them.

The hypothesis are:

- Small-disturbance assumption
- Thin lifting wing

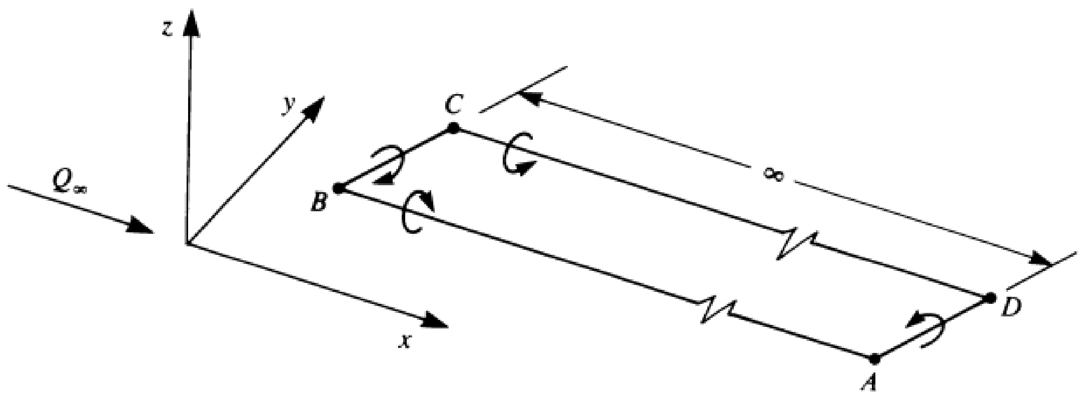


Figure B.16 Horseshoe vortex singularity element. (Source: [7])

- Large aspect ratio (AR > 4)

The problem aims to solve Laplace's equation, thus the vortex line is a solution of it. Moreover, the only boundary condition to be satisfied is the zero normal flow across the wing's solid surface

$$\nabla\left(\Phi+\Phi_\infty\right)\cdot\boldsymbol{n}=0 \tag{B.56}$$

Then, as the wing is placed on the xy plane, this boundary condition requires that the sum of the three normal components of the velocity (the one induced by the wing's bound vortices, by the wake and by the freestream velocity) is zero

$$u_{z,b}+u_{z,i}+U_\infty\alpha=0 \tag{B.57}$$

where the subscripts b and i refer to the bound vortices and the wake.

Now, following the six steps defined in subsection B.2.1, the numerical solution is built.

- **Singularity element:**

  To solve this problem the quadrilateral horseshoe vortex composed by four linear lifting lines depicted in Figure B.7 is selected. It consists of a straight bound vortex segment BC that models the lifting properties and of two semi-infinite trailing vortices representing the wake. The effect of segment AD is negligible because the wake segments extend far away from the TE of the wing.

  Since the small AoA approximation is considered, the trailing vortices are parallel to the wake. Therefore the model adopted, assuming a planar wing with a spanwise vortex distribution, is the one presented in Figure B.17, with all the vortices placed on the xy plane.

  The typical spanwise element appears in Figure B.18. The collocation point, where the normal surface vector of the element is located, is place at the center of the panel's three-quarter chord line whereas the bound vortex is placed at the panel quarter chord line. Apart from that, the positive circulation is also depicted.

  The velocity induced by one of these elements at an arbitrary point $P(x, y, z)$ can be computed by

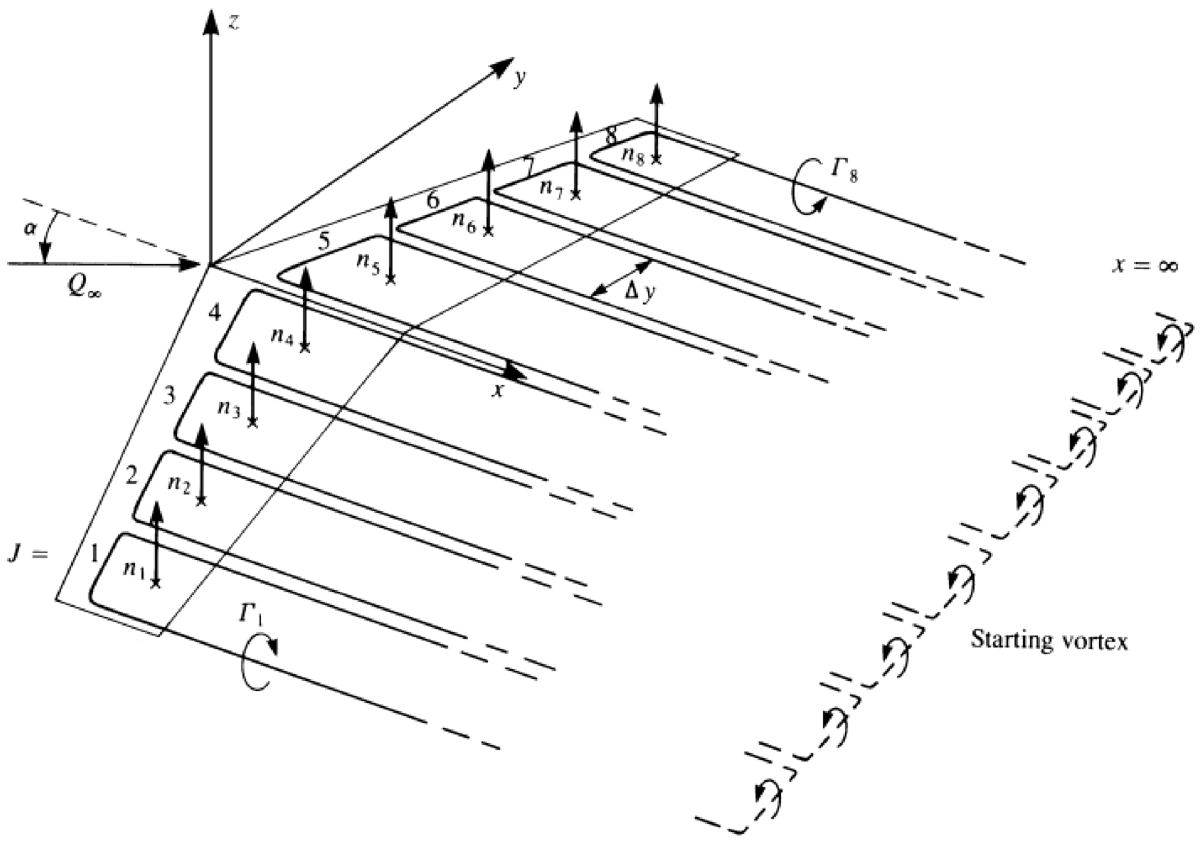


Figure B.17 Horseshoe vortex model for solving the lifting-line problem. (Source: [7])

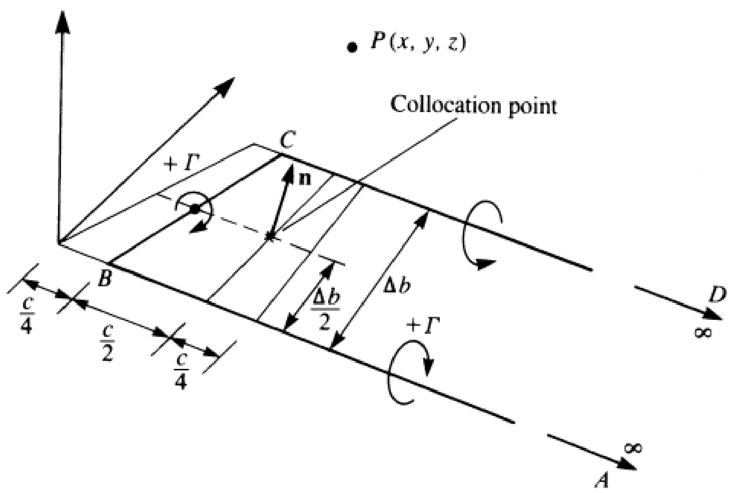


Figure B.18 Spanwise horseshoe vortex element with all the points and distances. (Source: [7])

applying three times the vortex line routine; for the AB, BC and CD segments. It is assumed that $y_A = y_B$, $y_C = y_D$ and $x_A = x_D \to \infty$, meaning that beyond A and D the influence is negligible[12]. The mentioned routine is summarized in the following:

– The formula to solve the velocity at an arbitrary point P is, if the vortex segment points from 1 to 2,

$$\mathbf{V}_{1,2} = \frac{\Gamma}{4\pi} \frac{\mathbf{r}_1 \times \mathbf{r}_2}{|\mathbf{r}_1 \times \mathbf{r}_2|^2} \mathbf{r}_0 \cdot \left(\frac{\mathbf{r}_1}{r_1} - \frac{\mathbf{r}_2}{r_2} \right) \tag{B.58}$$

– So the first step is to compute the cross product between $\mathbf{r}_1$ and $\mathbf{r}_2$,

$$\begin{aligned}
(\mathbf{r}_1 \times \mathbf{r}_2)_x &= (y_p - y_1) \cdot (z_p - z_2) - (z_p - z_1) \cdot (y_p - y_2) \\
(\mathbf{r}_1 \times \mathbf{r}_2)_y &= - (x_p - x_1) \cdot (z_p - z_2) + (z_p - z_1) \cdot (x_p - x_2) \\
(\mathbf{r}_1 \times \mathbf{r}_2)_z &= (x_p - x_1) \cdot (y_p - y_2) - (y_p - y_1) \cdot (x_p - x_2)
\end{aligned} \tag{B.59}$$

Its squared absolute value gives

$$|\mathbf{r}_1 \times \mathbf{r}_2|^2 = (\mathbf{r}_1 \times \mathbf{r}_2)_x^2 + (\mathbf{r}_1 \times \mathbf{r}_2)_y^2 + (\mathbf{r}_1 \times \mathbf{r}_2)_z^2 \tag{B.60}$$

– Then, the distances r_0, r_1 and r_2 need to be found,

$$\begin{aligned}
r_0 &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \\
r_1 &= \sqrt{(x_p - x_1)^2 + (y_p - y_1)^2 + (z_p - z_1)^2} \\
r_2 &= \sqrt{(x_p - x_2)^2 + (y_p - y_2)^2 + (z_p - z_2)^2}
\end{aligned} \tag{B.61}$$

– The singularities must be checked. In this case, the solution is singular when point P lies on the vortex thus a tolerance needs to be added. The segment is assumed to have a very small radius so that the singular condition is avoided. If any value is inferior to this radius, the velocity output is zero.

– Next, the computation of the dot product:

$$\begin{aligned}
\mathbf{r}_0 \cdot \mathbf{r}_1 &= (x_2 - x_1)(x_p - x_1) + (y_2 - y_1)(y_p - y_1) + (z_2 - z_1)(z_p - z_1) \\
\mathbf{r}_0 \cdot \mathbf{r}_2 &= (x_2 - x_1)(x_p - x_2) + (y_2 - y_1)(y_p - y_2) + (z_2 - z_1)(z_p - z_2)
\end{aligned} \tag{B.62}$$

– Finally, the initial formula is solved for each of the components of the velocity.

Once the routine is run for each of the three segments, the induced velocity is then

$$(u_x, u_y, u_z) = (u_{x_{AB}}, u_{y_{AB}}, u_{z_{AB}}) + (u_{x_{BC}}, u_{y_{BC}}, u_{z_{BC}}) + (u_{x_{CD}}, u_{y_{CD}}, u_{z_{CD}}) \tag{B.63}$$

As it will be seen in the loads computation, it is convenient to separate the wake-induced downwash

[12]From the potential point of view, infinite is assumed to be at least twenty wing spans behind the wing.

from the velocity induced by the bound vortex. Omitting the bound segment the equation states

$$(u_x, u_y, u_z)^* = (u_{x_{AB}}, u_{y_{AB}}, u_{z_{AB}}) + (u_{x_{CD}}, u_{y_{CD}}, u_{z_{CD}}) \tag{B.64}$$

- **Discretization and grid:**

  As show in Figure B.17, the wing is divided into several spanwise elements, with the wake segments parallel to the x axis and the bound ones that go through the quarter chord points of the panels.

  Then, Figure B.18 depicts all the data necessary to define in each element, highlighting the circulations, the collocation point (three-quarter chord) and the bound point (quarter chord). At the collocation point, assuming planar wing, the normal vector can be computed as

$$\boldsymbol{n}_j = (\sin\alpha_j, \cos\alpha_j) \tag{B.65}$$

  where j refers to an arbitrary panel, it is the counter that goes from the first one to the last.

  Other geometrical data such as the element's surface S_j or the mid-chord value in the spanwise direction are also calculated.

- **Influence coefficients:**

  It is probably the most important step in the development because the influence coefficients matrix is the key element to solve the aerodynamic system. So, to fulfill the normal velocity component boundary condition Equation B.56 must be true at each collocation point.

  Considering N discrete elements, the system of equations consists of finding, for each collocation point, the induced velocities due to all the horseshoe elements in the wing[13]. This can be written as

$$[(u_x, u_y, u_z)_{11}\Gamma_1 + (u_x, u_y, u_z)_{12}\Gamma_2 + \ldots + (u_x, u_y, u_z)_{1N}\Gamma_N + (U_{x,\infty}, U_{y,\infty}, U_{z,\infty})] \cdot \boldsymbol{n}_1 = 0 \tag{B.66}$$

  where the first subscript refers to the collocation point and the second to the element (each collocation point must consider all of the elements' contributions to the induced velocity) and the strengths of the vortices, Γ_j, are not known.

[13]The circulation to compute the induced velocities is assumed unitary, although the system solved next gives the real values.

Establishing the same procedure at each collocation point:

$$\begin{aligned}
a_{11}\Gamma_1 + a_{12}\Gamma_2 + a_{13}\Gamma_3 + \cdots + a_{1N}\Gamma_N &= -\mathbf{Q}_\infty \cdot \mathbf{n}_1 \\
a_{21}\Gamma_1 + a_{22}\Gamma_2 + a_{23}\Gamma_3 + \cdots + a_{2N}\Gamma_N &= -\mathbf{Q}_\infty \cdot \mathbf{n}_2 \\
a_{31}\Gamma_1 + a_{32}\Gamma_2 + a_{33}\Gamma_3 + \cdots + a_{3N}\Gamma_N &= -\mathbf{Q}_\infty \cdot \mathbf{n}_3 \\
\vdots \qquad\qquad & \qquad \vdots \\
a_{N1}\Gamma_1 + a_{N2}\Gamma_2 + a_{N3}\Gamma_3 + \cdots + a_{NN}\Gamma_N &= -\mathbf{Q}_\infty \cdot \mathbf{n}_N
\end{aligned} \tag{B.67}$$

where the influence coefficients are defined as

$$a_{ij} \equiv (u_x, u_y, u_z)_{ij} \cdot \boldsymbol{n}_i \tag{B.68}$$

and the subscript i refers to the collocation point.

- **RHS vector:**

At this point, the matrix system of equations is almost finished. The only element left is the RHS. As seen in the influence coefficients computations, it can be written as

$$RHS_i = -(U_{x,\infty}, U_{y,\infty}, U_{z,\infty}) \cdot \boldsymbol{n}_i \tag{B.69}$$

This way, a set of N linear algebraic equations with N unknown circulation can be defined in matrix form:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ a_{31} & a_{32} & \dots & a_{3N} \\ \vdots & & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \\ \vdots \\ \Gamma_N \end{pmatrix} = -V_\infty \sin\alpha \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{B.70}$$

When finding the influence coefficients matrix two loops are required, one for the collocation points and the other hand for the vortex elements that contribute to the induced velocity in the collocation point. The steps to follow are:

- Initialize a *for* loop for the collocation points considering the subscript i to refer to each of them.
- At this i collocation point, compute the RHS multiplying the negative components of the freestream velocity by the normal vector of the element.
- Initialize another *for* loop for the vortex elements, which contribute to the induced velocity at the collocation point. The subscript is *j*.
- Compute the coefficients a_{ij} and b_{ij} of the element.

Here $b_{ij} \equiv (u_x, u_y, u_z)^*_{ij} \cdot \boldsymbol{n}_i$ is the normal component of the wake-induced downwash. It will be used for the computation of the induced drag.

- **Solve set of equations:**

  The solution of the above problem can be obtained using standard methods for matrix systems of equations. The most direct case, although it is computationally expensive, is to calculate the inverse of the influence coefficients matrix and multiply it by the RHS of the matrix system of equations.

- **Secondary computations:**

  By solving the set of equations one obtains the circulations or vortex strengths of each element. Therefore, the lift of each bound vortex segment is obtained by using the KJ theorem:

$$\Delta L_j = \rho V_\infty \Gamma_j \Delta y_j \tag{B.71}$$

  where Δy_j is the projection of the bound vortex normal to the freestrem (Δb in Figure B.18).

  The induced drag computation is a little more complex as it appears due to the trailing vortices. That is why, when computing the b coefficients, the bound vortex segment contribution to the velocity is not considered. The formula states

$$\Delta D_j = -\rho\, u_{z,\mathrm{ind}_j} \Gamma_j \Delta y_j \tag{B.72}$$

  where the induced downwash u_{z,ind_j} at each collocation point j is computed by summing the induced velocities of all the trailing vortex segments. The matrix formulation of the procedure is

$$\begin{pmatrix} u_{z,\mathrm{ind}_1} \\ u_{z,\mathrm{ind}_2} \\ u_{z,\mathrm{ind}_3} \\ \vdots \\ u_{z,\mathrm{ind}_N} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1N} \\ b_{21} & b_{22} & \dots & b_{2N} \\ b_{31} & b_{32} & \dots & b_{3N} \\ \vdots & & \ddots & \vdots \\ b_{N1} & b_{N2} & \dots & b_{NN} \end{pmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \\ \vdots \\ \Gamma_N \end{pmatrix} \tag{B.73}$$

  Finally, the total lift and drag forces are obtained by adding all the panel contributions:

$$L = \sum_{j=1}^{N} \Delta L_j$$
$$D = \sum_{j=1}^{N} \Delta D_j$$

This is the most simple method to solve the aerodynamic potential problem. However, it is a very good base to understand the general concepts and steps to be followed.

B.2.4 The Vortex Lattice Method

The VLM method is an extension of the HVM. It uses a different singularity element, the vortex ring, and requires very little programming effort. The wing studied can have different shapes and add camber line as if a profile is considered thus the result is closer to a real case.

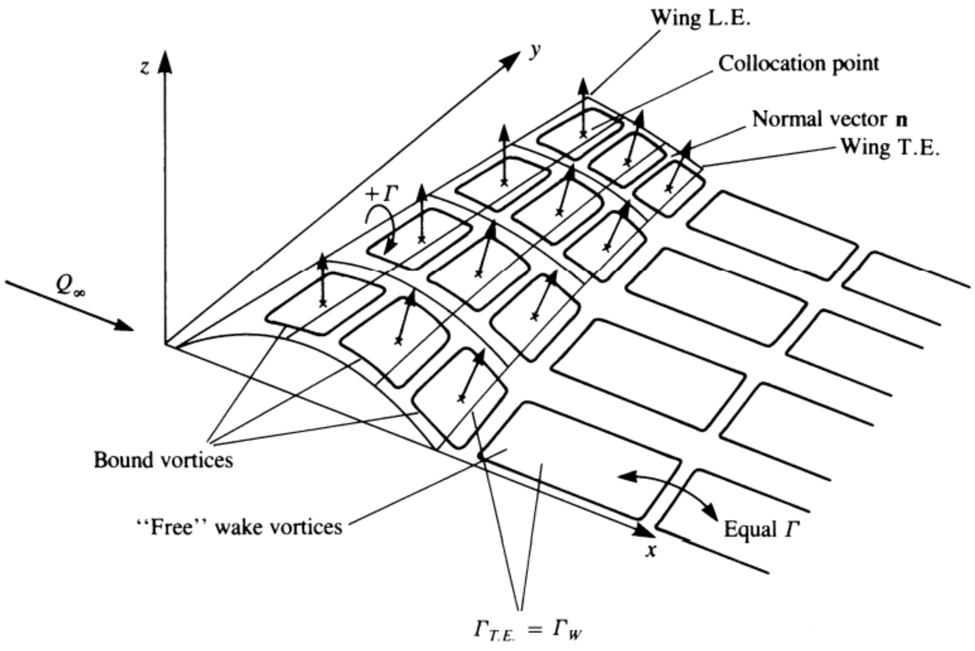


Figure B.19 Vortex ring model for a thin lifting surface. (Source: [7])

As in the previous method, the solution is based on the vortex line solution obtained via the incompressible continuity equation. Therefore, the boundary condition that needs to be satisfied is the zero normal component of the velocity across the wing's surface (see Equation B.56).

Then, following with the small-disturbance lifting line formulation (see subsection B.1.4), the boundary condition can be expressed as

$$\frac{-1}{4\pi}\int_{\text{wing+wake}} \frac{\gamma_y(x-x_0)-\gamma_x(y-y_0)}{[(x-x_0)^2+(y-y_0)^2]^{3/2}}dx_0\,dy_0 = Q_\infty\left(\frac{\partial\eta_c}{\partial x}-\alpha\right)$$

So to solve this lifting line problem numerically, the wing is divided into several vortex ring elements as in Figure B.19.

As for the HVM, the six steps to obtain a solution are followed.

- **Singularity element:**

  In this case, as stated previously, the singularity element is the vortex ring. Its notation is depicted in Figure B.20 and consists of four segments of strength Γ. Again, the leading segment of the vortex ring is placed on the quarter chord of the panel while the collocation point (where the normal vector of the element is defined) is at the center of the three-quarter chord line.

  Moving on to the numerical approach, Figure B.21 shows the indexation of the vortex rings or panels. The indices are i in the chordwise direction and j in the spanwise.

  The velocity induced by one of these elements (indices i, j) at an arbitrary point $P(x, y, z)$ can be

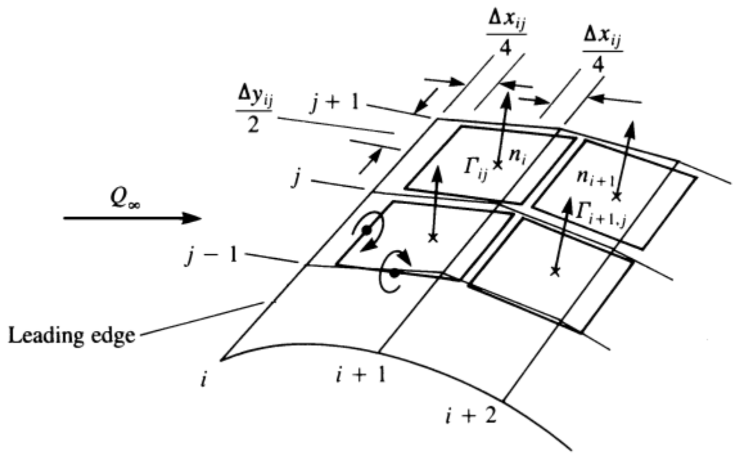


Figure B.20 Nomenclature for the vortex ring elements. (Source: [7])

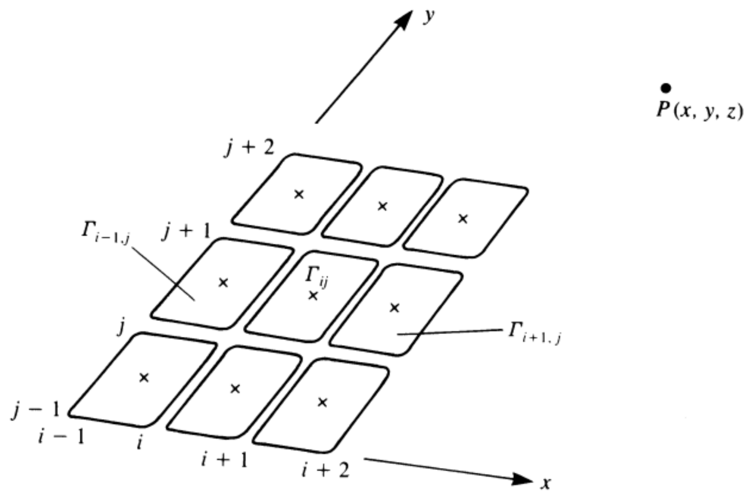


Figure B.21 Arrangement of vortex rings in a rectangular array. (Source: [7])

computed using the vortex line routine detailed in the previous section. It is then

$$(u_x, u_y, u_z) = (u_{x_1}, u_{y_1}, u_{z_1}) + (u_{x_2}, u_{y_2}, u_{z_2}) + (u_{x_3}, u_{y_3}, u_{z_3}) + (u_{x_4}, u_{y_4}, u_{z_4}) \tag{B.74}$$

where the subscripts 1, 2, 3 and 4 refer to the four segments that compose the vortex ring. Segment 1 is the leading segment and goes from (i, j) to $(i, j+1)$, segment 2 goes from $(i, j+1)$ to $(i+1, j+1)$, segment 3 from $(i+1, j+1)$ to $(i+1, j)$ and segment 4 from $(i+1, j)$ to (i, j).

Thanks to Figure B.21, when identifying the i and j indices of each panel, the coordinates of the four points of the element are known. Moreover, for the induced drag computation it is also convenient to find the induced velocity by the trailing vortex segments only. The trailing segments are 2 and 4 so omitting the two others

$$(u_x, u_y, u_z)^* = (u_{x_2}, u_{y_2}, u_{z_2}) + (u_{x_4}, u_{y_4}, u_{z_4}) \tag{B.75}$$

At this point, the definition of the singularity element is finished. The next steps are directly related to dividing the solid body in elements and building the system of equations.

- **Discretization and grid:**

  As depicted in Figure B.19, the wing is divided into several quadrilateral elements, now following a 2D distribution (in the chordwise and spanwise directions). Then, Figure B.20 shows the typical elements with all the distances and values to define, such as the collocations points at the center of the three-quarter chord line or the leading segment of the vortex ring on the panel's quarter chord line.

  The normal vector, which is key to the formulation of the set of equations, is also defined at the collocation point. It can be computed by doing the cross product of the vector from point (i, j) to $(i+1, j+1)$ and the one from $(i+1, j)$ to $(i, j+1)$.

  A positive circulation Γ is defined using the right-hand rule, as shown in Figure B.20. However, in this case there is an important difference with respect to the HVM. For the pressure or lift distributions the local circulation is needed, which for the LE panel is equal Γ_i but for the other panels it is equal to $\Gamma_i - \Gamma_{i-1}$.

- **Influence coefficients:**

  The influence coefficients calculation is similar to the method presented in last subsection. However, in here it is necessary to scan the panels in a 2D way, that is to say, one loop for each direction.

  This way, the computational routine consists of the following parts:

  – Scan the collocation points using two *for* loops, one for the chordwise direction (M divisions) and the other for the spanwise one (N divisions).

  – For each collocation point all the contributions of the vortex rings need to be computed so two more *for* loops are to be defined. Similarly to the previous step the loop are for both directions of the discretization.

  – Compute the induced velocities by the whole vortex ring and by the trailing vortices only.

  – Add all the contributions at each collocation points so to find the influence coefficients a_{ij} and b_{ij}.

  When computing the contribution to the induced velocity by a vortex ring in the TE, a free wake vortex ring with the same strength is added to cancel the circulation in the spanwise direction (see Figure B.22). Therefore, the influence coefficient formula is

$$a_{ij} \equiv [(u_x, u_y, u_z)_{ij} + (u_x, u_y, u_z)_{ijW}] \cdot \boldsymbol{n}_i \tag{B.76}$$

  where the subscript W refers to the free wake vortex added to the vortex ring in the TE.

  Apart from the influence coefficient, the components which contribute to the induced drag are obtained thanks to another coefficient, which is

$$b_{ij} \equiv \left[(u_x, u_y, u_z)^*_{ij} + (u_x, u_y, u_z)^*_{ijW}\right] \cdot \boldsymbol{n}_i \tag{B.77}$$

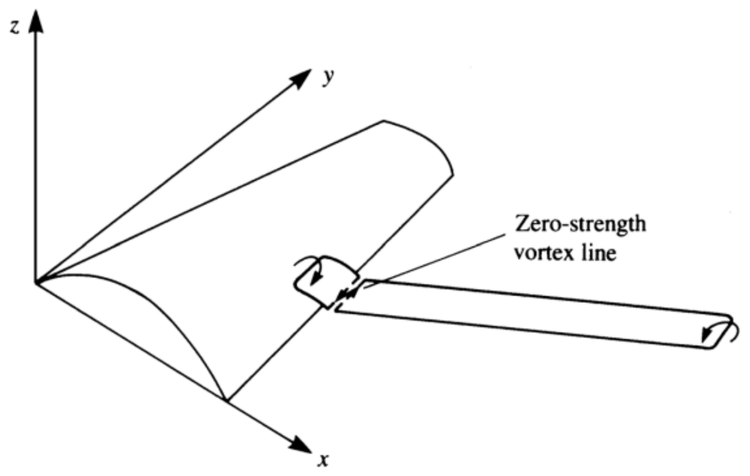


Figure B.22 Adding a vortex wake panel to fulfill the Kutta condition in the TE panels. (Source: [7])

- **RHS vector:**

  The RHS is calculated for each collocation point, giving as a result a vector, by only making use of the normal vector of the element and the three components of the freestream velocity. Equation B.69 shows the mathematical form.

- **Solve set of equations**

  When having both the influence coefficients matrix and the RHS vector the system of equations is immediately formulated (see Equation B.70).

  It comes from applying the zero normal flow boundary condition on each of the collocation points thus if there are N collocation points, the result will be a N-sized vector with the circulation at each point.

- **Secondary computations**

  Once the system of N algebraic equations is solved, the lift of each bound vortex segment is obtained by using the KJ theorem. It is important to notice that now the discretization is 2D, so the formulas depend on whether the panel is in the LE or not. That said, for the majority of panels ($i > 1$)

$$\Delta L_{ij} = \rho V_\infty \left(\Gamma_{i,j} - \Gamma_{i-1,j}\right) \Delta y_{ij} \tag{B.78}$$

  but for the LE ones ($i = 1$)

$$\Delta L_{ij} = \rho V_\infty \Gamma_{i,j} \Delta y_{ij} \tag{B.79}$$

  where Δy_{ij} is the panel width (see Figure B.20).

  Then, the induced drag computation follows the same principle as for the HVM thus it is due to the trailing vortices. The formulas state, for the middle panels ($i > 1$)

$$\Delta D_{ij} = -\rho \, u_{z,\text{ind}_{i,j}} \left(\Gamma_{i,j} - \Gamma_{i-1,j}\right) \Delta y_{ij} \tag{B.80}$$

and for the LE ones ($i = 1$)

$$\Delta D_{ij} = -\rho\, u_{z,\text{ind}_{i,j}} \Gamma_{i,j} \Delta y_{ij} \tag{B.81}$$

where the induced downwash u_{z,ind_j} at each collocation point j is computed by summing the induced velocities of all the trailing vortex segments. The matrix formulation of the procedure is described in Equation B.73.

Finally, the total lift and drag forces are obtained by adding all the panel contributions:

$$L = \sum_{i=1}^{M} \sum_{j=1}^{N} \Delta L_{ij}$$
$$D = \sum_{i=1}^{M} \sum_{j=1}^{N} \Delta D_{ij}$$

The VLM is a more detailed method to solve the potential flow problem because the discretization is 2D and the addition of the profile camber, for example, brings the result closer to realistic situations.

Appendix C

Validation of Separate Codes

Appendix C is divided into two parts. On the one hand, results of the two structural elements which make up the project mesh are validated. These elements are the beam and the plate.

On the other hand, steady aerodynamic results of two different methods are also compared with literature. The two methods are the Horseshoe Vortex Method (HVM) and the Vortex Lattice Method (VLM).

C.1 Structural Codes

Both codes are based on the Finite Element Method (FEM), using the Gauss quadrature to evaluate the integrals.

C.1.1 Beam element

The case which is studied in this section is a cantilever beam with uniform load along the longitudinal direction, see figure C.1. Although the beam can be studied in 1D with linear elements, the code is written in 3D, in case it needs to be extended for more complex analyses.

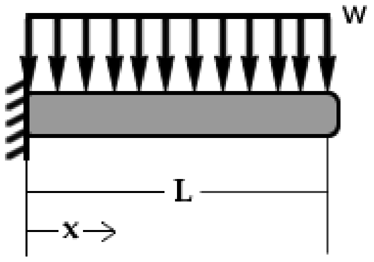


Figure C.1 Cantilever beam with uniform load along its longitudinal axis.

According to (Tejerizo, 2015) [9], the following analytical data can be extracted:

- Clamped reaction force $\longrightarrow R\,(x=0) = pL$
- Clamped reaction moment $\longrightarrow M\,(x=0) = \frac{pL^2}{2}$
- Shear stress distribution along longitudinal axis $\longrightarrow T\,(x) = p\,(L-x)$
- Bending moment distribution along longitudinal axis $\longrightarrow M\,(x) = -p\,\frac{(L-x)^2}{2}$

Then, applying the Mohr's second theorem, the expression of the vertical displacement at the tip of the beam is found.

$$w\,(x=L) = \frac{pL^4}{8EI} \tag{C.1}$$

Introducing numerical parameters so to obtain solutions of the equations, they can be compared to the solutions of the FEM MATLAB code. Table C.1 shows the comparison.

- Distributed load $\longrightarrow p = -10^5\,Pa$
- Beam length $\longrightarrow L = 2\,m$
- Young's modulus $\longrightarrow E = 69 \cdot 10^9\,Pa$
- Cross-sectional inertia $\longrightarrow I_z = 3.25 \cdot 10^{-3}\,m^4$

Table C.1 Analytical and computational comparison of the results of a cantilever beam with uniform load.

	Analytical	**FEM MATLAB**	**Error (%)**
Shear stress (x=0)	$-2 \cdot 10^5$ N	$-2 \cdot 10^5$ N	0
Shear stress (x=L)	0 N	0 N	0
Bending moment (x=0)	$2 \cdot 10^5$ Nm	$2 \cdot 10^5$ Nm	0
Bending moment (x=L)	0 Nm	0 Nm	0
Tip displacement	0.8918 mm	0.9069 mm	1.69

Finally, the deformed beam and the internal forces and moments in each direction are plotted in MATLAB, confirming they follow the same distributions as in (Tejerizo, 2015) [9].

N_e = 4
×10^-4
9.0693
0
N_e = 8
×10^-4
9.0693
0

Figure C.2 Vertical displacement along the longitudinal direction of the beam.

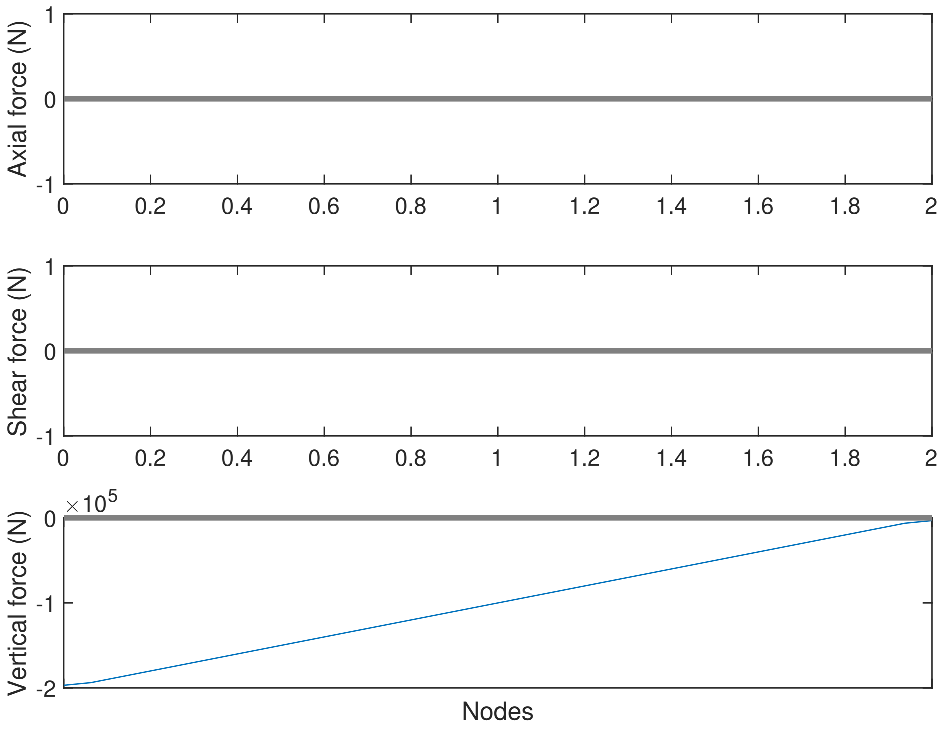


Figure C.3 Internal forces along the longitudinal direction of the beam in the three directions.

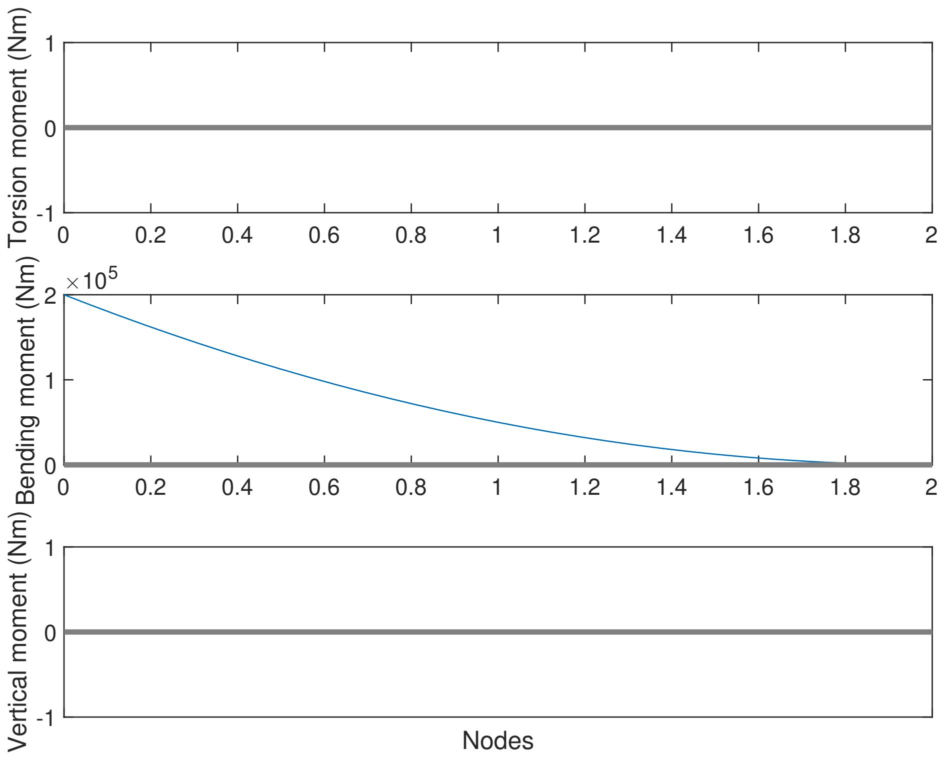


Figure C.4 Internal moments along the longitudinal direction of the beam in the three directions.

C.1.2 Plate element

The plate structure adds a dimension with respect to the beam one, the problem is considered in 2D. This second section of Annex A studies a squared flat plate with all the edges clamped.

According to (Young, Budynas and Sadegh, 2012) [10] the maximum displacement appears, naturally, at the center of the structure for the case with all edges fixed. Its value is found using equation C.2, for a Poisson coefficient $\nu = 0.3$.

$$w_{max} = \frac{\alpha q b^4}{E t^3} \tag{C.2}$$

E is the Young's modulus, b is the short edge length, q is the distributed load, t is the plate thickness and α is a dimensionless tabulated coefficient. The latter is obtained from the relation between the short and long edge, being 1 for a squared plate.

If one introduces numbers to each parameter in equation C.2, the maximum displacement can be obtained.

- Distributed load $\longrightarrow q = 10^6\,Pa$
- Plate edge length $\longrightarrow b = 2\,m$
- Young's modulus $\longrightarrow E = 69 \cdot 10^9\,Pa$
- Plate thickness $\longrightarrow t = 0.05\,m$
- Dimensionless coefficient $\longrightarrow \alpha = 0.0138$
- Poisson coefficient $\longrightarrow \nu = 0.3$

Once the analytical solution is settled, the code solution must be analyzed. Figure C.5 and figure C.6 represent the plate's behaviour when the distributed load is applied together with the convergence with the number of discrete elements used.

Finally, comparing the analytical and computational results, the code performance can be evaluated. Table C.2 summarizes the results, picking the numerical one for 32 elements in each direction.

Table C.2 Analytical and computational comparison of the results of an all-edges-clamped plate with uniform load.

	Analytical	**FEM MATLAB**	**Error (%)**
Maximum displacement	25.6 mm	30.1 mm	17.58

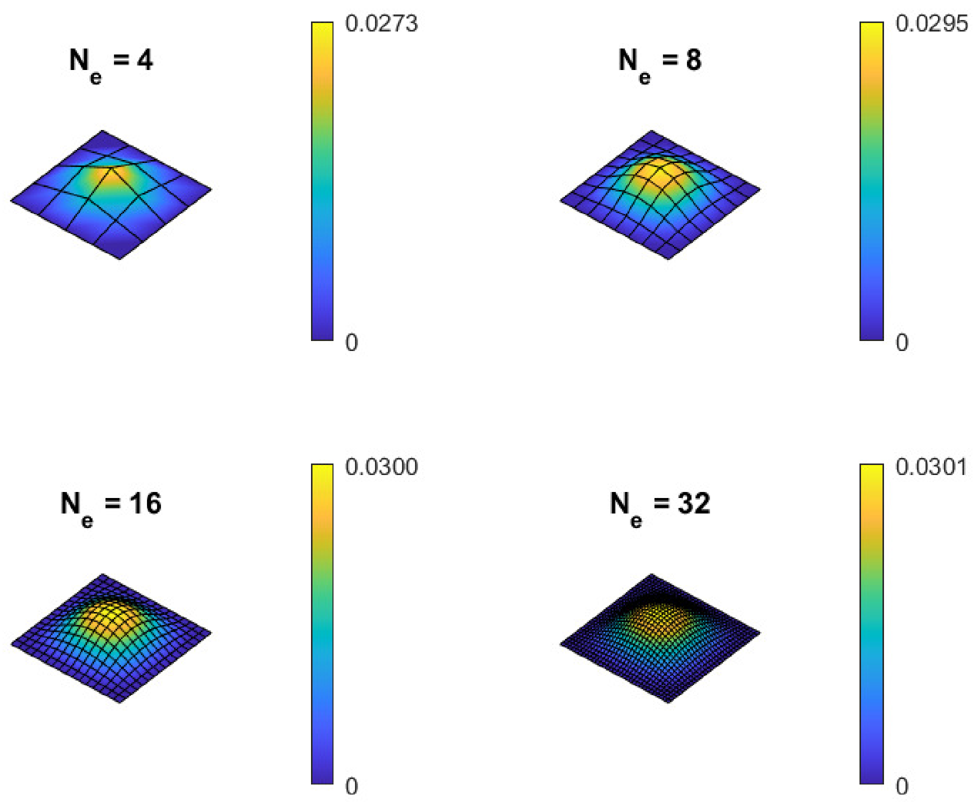


Figure C.5 Plate's vertical displacement distribution along its surface.

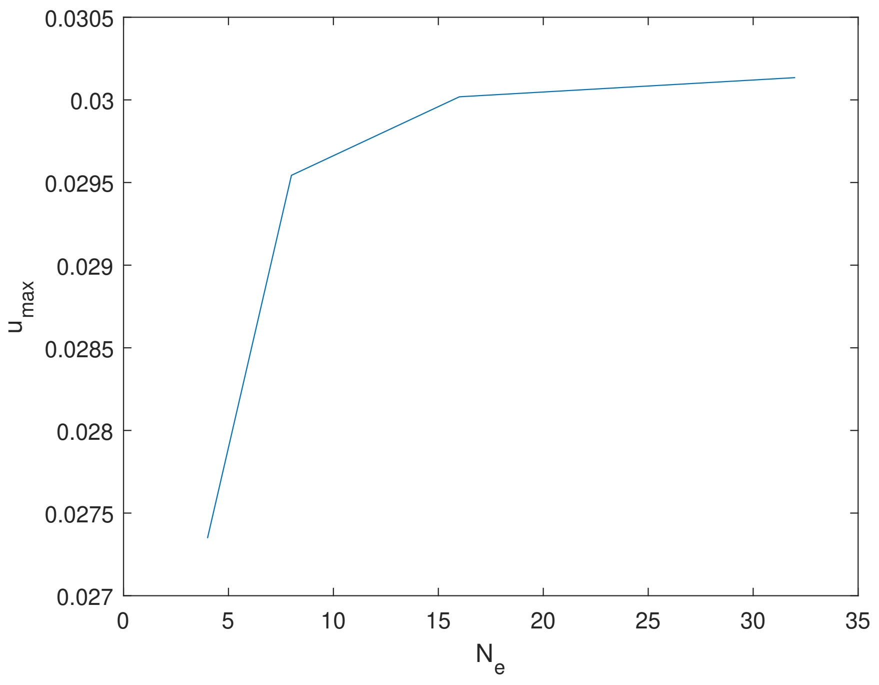


Figure C.6 Plate's maximum displacement as a function of the number of 2D elements.

Table C.3 EZASE and project codes comparison of the main aerodynamic forces together with the analytical values.

Solution	Lift (N)	Error L (%)	Induced Drag (N)	Error D_i (%)
EZASE VLM	13100.4013	1.45	259.4182	12.6
Analytical	**13293.2797**	**0**	**296.8210**	**0**
Project VLM	13111.6585	1.37	277.3525	6.6

C.2 Aerodynamic Codes

The computational aerodynamic methodology for the VLM is based on [7]. The HVM does not need to be validated because it is explained as an introduction to the VLM.

C.2.1 Vortex Lattice Method

The results to check in this part are the steady lift and induced drag obtained using the VLM development. The wing dimensions and flying condition are:

- Span $\longrightarrow b = 3.354\,m$
- Root chord $\longrightarrow c_r = 0.838\,m$
- Taper ratio $\longrightarrow TR = 1$
- Sweep angle $\longrightarrow SW = 0°$
- Angle of attack $\longrightarrow \alpha = 4°$
- Sideslip angle $\longrightarrow \beta = 0°$
- Freestream density $\longrightarrow \rho_\infty = 1.225\,kg/m^3$
- Freestream velocity $\longrightarrow V_\infty = 0.5\,V_{sound,SL}$[1] $= 170.15\,m/s$

The EZASE code created by NASA [11] plot the VLM lift and induced drag and compare them to the analytical values from the 3D Prandtl's lifting line theory. On the other hand, the code developed in this work is based on the methodology in [7]. Figures C.7 and C.8 correspond to the later while Figures C.9 and C.10 to the NASA code.

Then, in Table C.3 the analytical results are compared to the VLM ones obtained using both the EZASE and this work's code.

[1]It refers to the speed of sound at sea level, being 340.3 m/s.

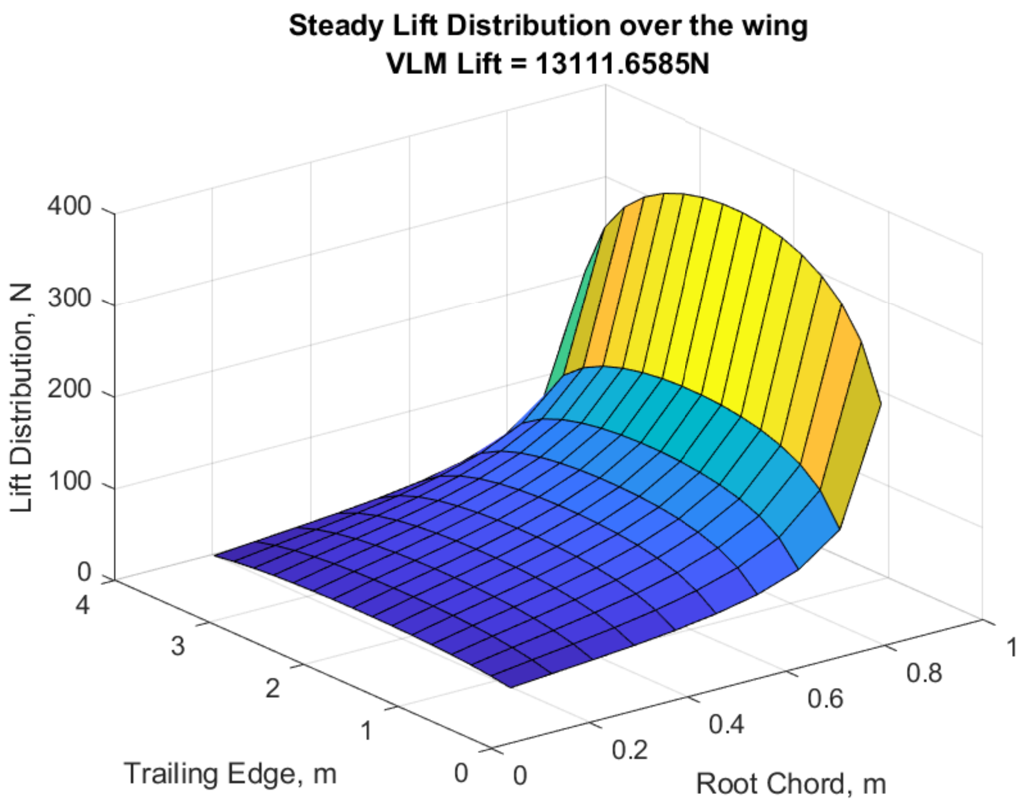


Figure C.7 Lift distribution along the wing computed using the developed VLM code.

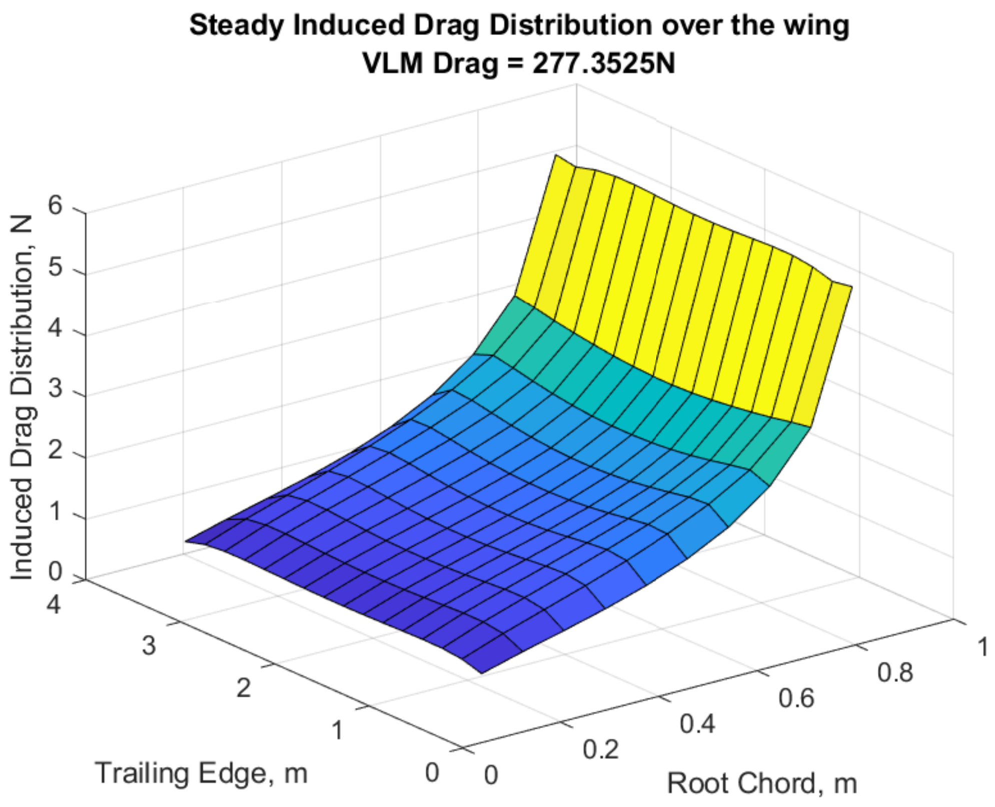


Figure C.8 Induced drag distribution along the wing computed using the developed VLM code.

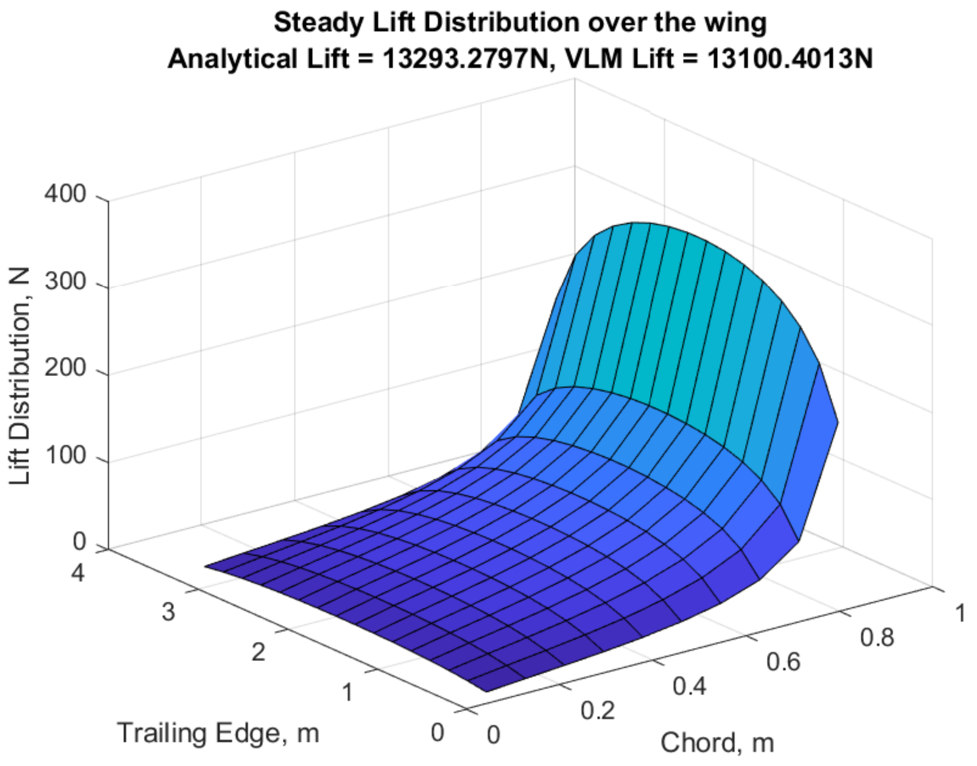


Figure C.9 Lift distribution along the wing computed using the VLM tutorial in EZASE NASA code. (Source: [11])

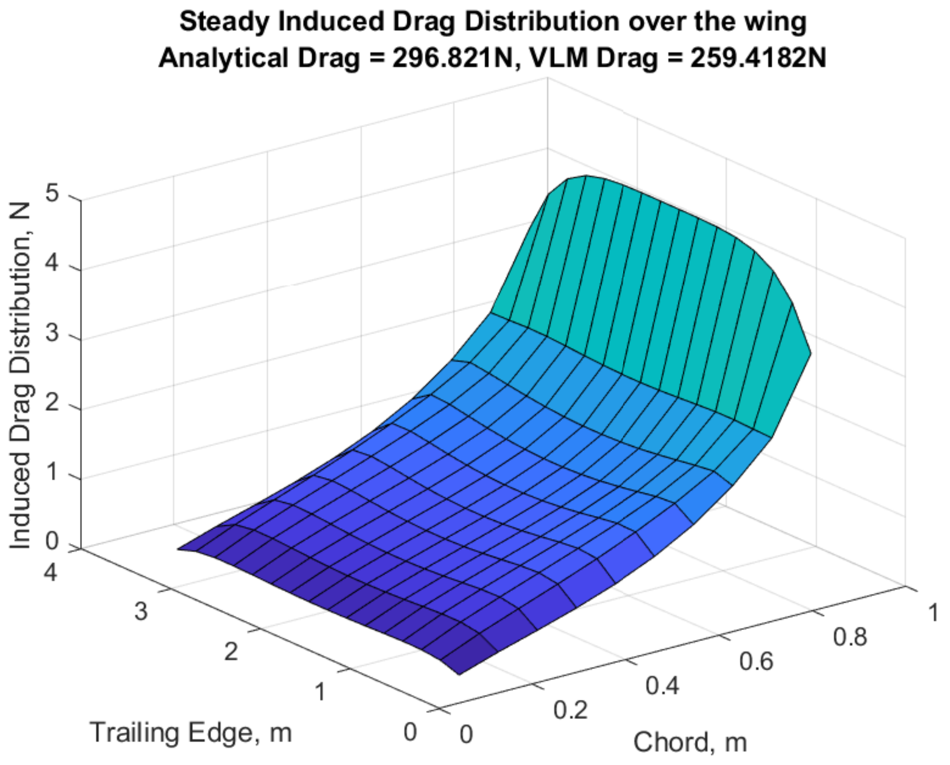


Figure C.10 Induced drag distribution along the wing computed using the VLM tutorial in EZASE NASA code. (Source: [11])

Appendix D

Aeroelastic Code

```
%-------------------------------------------------------------------------%
% TFG (Includes Aeroelastic Study modifying TR, SW, AR, Mp)
%-------------------------------------------------------------------------%
% Date: 15/02/2022
% Author/s: Guillermo Adroher Bolla

clc;
clear;
close all;

%% INPUT DATA

%_______
% STRUCTURAL INPUTS
%_______

% Material properties ----------------- Aluminum 6061-T6 (NASA paper)
rhoAl = 2700;   % density [kg/m^3]
Emod = 68.9e9;   % Young's modulus [Pa]
Gmod = 26e9;   % shear modulus [Pa]
nu = 0.33;  % poisson ratio
h = 3.06e-3;   % wing thickness [m]

%_______
% AERODYNAMIC INPUTS
%_______

% Geometric data
b = 2.2946;   % semi span length [m] Orig: 3.354 and 737: 2.2946
cr = 0.838;   % root chord length [m]
TR = 0.159;  % wing's taper ratio  737: 0.159
SW = 25;  % wing's sweep angle [deg]  737: 25
ct = cr*TR;   % tip chord length [m]
```

```
S = (cr+ct)*b/2;    % semi wing surface [m^2]
AR = b^2/S;   % wing's aspect ratio
mac = 2/3*cr.*(1+TR+TR.^2)./(1+TR) ;  % mean aerodynamic chord [m]

% Velocity and environment variables
M = 0.5;
Vsound = 340.3; % speed of sound at SL [m/s]
Qinf = M*Vsound;    % free-stream speed in wind axes [m/s]
mu = 1.983e-5;
rho = 1.225;    % density [kg/m^3]
Re = rho*Qinf*mac./(2*mu); % computed at the mean chord (cr+ct)/2

%_______
% PLOTTING INPUTS
%_______

mesh_plots = 0;     % 1 to plot them and 0 not to plot them
modes_plots = 0;
torsion_bending_plots = 0;
flutter_plot = 1;
modes_plots_dyn = 0;

%% MESH GENERATION MODULE - Bilinear Quadrilateral Elements

% Input mesh data
xel = 10;    % number of elements in the semi span direction (multiple of
             % 10, if not check the spars' position)
yel = 16;    % number of elements in the chord direction (20 EZASE)
xnod = xel+1;
ynod = yel+1;

% Mwing = 63.6146;   % computed analytically integrating (TR=1 and SW=0)
Mwing2 = 25.22; % (TR=0.159 and SW=25)
mp = 0.4454*Mwing2;  % point mass to add at some nodes  737: 0.4454
n_ribs = [5,6]; % in which ribs the mass is added  737 Orig: [4,5]
rib_nodes = [1,2];  % in which nodes of the rib the mass is added [1,2]
mp_nod = zeros(size(n_ribs,1),size(n_ribs,2)*size(rib_nodes,2));
for r = 1:size(n_ribs,2)
    mp_nod(1+(r-1)*size(rib_nodes,2):r*size(rib_nodes,2)) = ...
        rib_nodes +(n_ribs(r)-1)*xnod;
end
mp_nodes = mp_nod(:,[1 2 4 3]);

Dy = b/yel;  % element chordwise length [m]
Dx = zeros(ynod,1);
c = zeros(ynod,1);  % chord length at each y position [m]
c4 = zeros(ynod,1);  % quarter chord coordinate at each y position [m]
for i = 1:ynod
    c4(i) = 1/4*cr+Dy*(i-1)*tan(SW*pi/180);
    c(i) = (ct-cr)/b*Dy*(i-1)+cr;
    Dx(i) = c(i)/xel;
end

% Ribs and Spars cross-section parameters
    % Spars' properties (|| cross-sectional area)
        nspar = 2;  % number of spars
        ks_prime = [0 0 1]';
        % LE spar (bigger)
        Hos_LE = 0.0508;    % initial height LE spar
        Wos_LE = 0.0508*0.5; % initial width LE spar
        Hfs_LE = 0.0254;    % final height LE spar
        Wfs_LE = 0.0254*0.5; % final width LE spar
        for i = 1:ynod
            H_LE(i) = (Hfs_LE-Hos_LE)/b*Dy*(i-1) + Hos_LE;
            W_LE(i) = (Wfs_LE-Wos_LE)/b*Dy*(i-1) + Wos_LE;
        end
        As_LE = H_LE.*W_LE;  % cross-sectional area [m^2]
        Iys_primeLE = H_LE.^4/12;    % area inertias [m^4]
        Izs_primeLE = W_LE.^4/12;
        JsLE = H_LE.*W_LE.^3.*(16/3-3.36.*W_LE./H_LE.*(1-W_LE.^4./...
            (12*H_LE.^4)));
        IpsLE = Iys_primeLE+Izs_primeLE; % polar inertia [m^4]
        % TE spar (smaller)
        Hos_TE = 0.0381;    % initial height TE spar
        Wos_TE = 0.0381*0.5; % initial width TE spar
        Hfs_TE = 0.0254;    % final height TE spar
        Wfs_TE = 0.0254*0.5; % final width TE spar
        for i = 1:ynod
            H_TE(i) = (Hfs_TE-Hos_TE)/b*Dy*(i-1) + Hos_TE;
            W_TE(i) = (Wfs_TE-Wos_TE)/b*Dy*(i-1) + Wos_TE;
        end
        As_TE = H_TE.*W_TE;  % cross-sectional area [m^2]
        Iys_primeTE = H_TE.^4/12;    % area inertias [m^4]
        Izs_primeTE = W_TE.^4/12;
        JsTE = H_TE.*W_TE.^3.*(16/3-3.36.*W_TE./H_TE.*...
            (1-W_TE.^4./(12*H_TE.^4)));
        IpsTE = Iys_primeTE+Izs_primeTE; % polar inertia [m^4]

        kys = 5/6;          % shear and torsion correction factors
        kzs = 1;            % Onate Vol 2 page 42

    % Ribs' properties (|| cross-sectional area, H = 25.4mm and W = 25.4mm)
        jr_prime = [0 1 0]';
        Hor = 0.0508;   % initial height rib
        Wor = 0.0508*0.5;   % initial width rib
        Hfr = 0.0254;   % final height rib
        Wfr = 0.0254*0.5; % final width rib
        for i = 1:ynod
```

```
            for j = 1:xnod
                Hr(j,i) = (Hfr-Hor)/(c(i))*Dx(i)*(j-1)+Hor;
                Wr(j,i) = (Wfr-Wor)/(c(i))*Dx(i)*(j-1)+Wor;
            end
        end
        Ar = Hr.*Wr;  % cross-sectional area [m^2]
        Iyr_prime = Hr.^4/12;   % area inertias [m^4]
        Izr_prime = Wr.^4/12;
        Jr = Hr.*Wr.^3.*(16/3-3.36.*Wr./Hr.*(1-Wr.^4./(12*Hr.^4)));
        Ipr = Iyr_prime+Izr_prime; % polar inertia [m^4]
        kyr = 5/6;          % shear and torsion correction factors
        kzr = 1;            % Onate Vol 2 page 42

% Dimensions plate
dim.nd = 3;     % problem dimension
dim.nel = xel*yel;     % number of elements
dim.nnod = xnod*ynod;  % total number of nodes
dim.nne = 4;    % number of nodes in an element
dim.ni = 6;  % degrees of freedom per node (in case 3D extension)
dim.ndof = dim.nnod*dim.ni;  % total number of degrees of freedom


% Generating and plotting the mesh
nrib = ynod;  % number of ribs
[X,Xs,Xr,Tn,Tns,Tnr,Tnp,Tmp,Tdp,c75elem,c4corner,c4elem,normals,Sel,npan,...
    Up] = meshFunction_STRUCT2(dim,Dx,Dy,xnod,ynod,c,c4,b,nspar,nrib,...
    mesh_plots);


%% AERODYNAMIC MODULE

% Assembly of the influence coefficients matrix
[A,B] = infcoeff_VLM(npan,Tnp,Tmp,Tdp,xel,yel,c4corner,c75elem,normals);


%% STRUCTURAL MODULE

% Compute stiffness matrix of the shell elements
[Kplate,Mplate,R,N4,S4] = stiffnessPlate_STRUCT(X,Tn,dim,h,Emod,nu,rhoAl);

% Compute stiffness matrix of the spar elements
[KsparLE,MsparLE] = stiffnessSpars_STRUCT(X,Tns(1:16,:),dim,ks_prime,...
    Emod,Gmod,As_LE,JsLE,Iys_primeLE,Izs_primeLE,kys,kzs,yel,rhoAl);
[KsparTE,MsparTE] = stiffnessSpars_STRUCT(X,Tns(17:32,:),dim,ks_prime,...
    Emod,Gmod,As_TE,JsTE,Iys_primeTE,Izs_primeTE,kys,kzs,yel,rhoAl);

Kspars = KsparLE + KsparTE;
Mspars = MsparLE + MsparTE;

```

```
% Compute stiffness matrix of the rib elements
Kribs = zeros(dim.ndof,dim.ndof);
Mribs = zeros(dim.ndof,dim.ndof);
for i = 1:nrib
    [Krib,Mrib] = stiffnessRibs_STRUCT(X,Tnr(1+(i-1)*xel:i*xel,:),dim,...
        jr_prime,Emod,Gmod,Ar(:,i),Jr(:,i),Iyr_prime(:,i),Izr_prime(:,i),...
        kyr,kzr,xel,rhoAl);
    Kribs = Kribs + Krib;
    Mribs = Mribs + Mrib;
end

K = Kplate+Kribs+Kspars;
M = Mplate+Mribs+Mspars;

for i = mp_nodes
    M(dim.ni*(i-1)+(1:3),dim.ni*(i-1)+(1:3)) =  M(dim.ni*(i-1)+(1:3),...
        dim.ni*(i-1)+(1:3)) + mp/(size(rib_nodes,2)*size(n_ribs,2))*eye(3);
end

% K = Kplate;
% M = Mplate;


%% COUPLING MODULE

% D matrix --> Relate AoA with vertical (z) displacement of the nodes
D = matrixD_STRUCT(dim,Tn,X);

% Q matrix --> Relate pressure distribution with external forces
Q = matrixQ_STRUCT(dim,Tn,xel,X);

% E matrix --> Relate structural velocity in the collocation points ('z'
% direction) to velocity vector {u_point}
E = matrixE_STRUCT(dim,Tn);


%% SOLVER (Post-Process)

% Boundary conditions
u = zeros(dim.ndof,1);
[Ip,If] = dofVec(Up,dim);

% STATIC CASE
% Modal analysis (eigenvalues and eigenvectors)
neig = 18;

Kf = K;
invA = inv(A);
Mf = -(2*Q*invA*D);
```

```matlab

Kf = Kf(If,If);  % only picking the free nodes
Mf = Mf(If,If);

Phi = zeros(dim.ndof,neig);
[Phi(If,:),Lambda] = undampedFreq(Mf,Kf,neig); % natural frequencies and vibration modes
                                                % Lambda gives q_inf
Vdiv = sqrt(Lambda.*2./rho);
for i = 1:length(Vdiv)
    if imag(Vdiv(i)) ~= 0
        Vdiv(i) = 0;
    end
end

if modes_plots == 1
    plotModes(X,Tn,Phi,Lambda);
end

PhiAoA = D*Phi*180/pi;    % AoA matrix (Phi is the vertical displacements matrix)

if torsion_bending_plots == 1
    % Bending plots (along span per mode)
    sumZdisp = zeros(neig,ynod);    % Z displacements sum for each 'y' position divided by
                                    % the number of 'x' nodes for that 'y'
    ydof = dim.ni*ynod;
    xdof = dim.ni*xnod;
    figure()
    for mod = [1:6 15:18]
        for j = 1:ynod
            sumZdisp(mod,j) = real(sum(Phi(3+xdof*(j-1):6:j*xdof-3,mod)));
        end
        plot(X(1:xnod:dim.nnod,2)',sumZdisp(mod,:)./xnod,'DisplayName',...
            ['Mode ' num2str(mod)]); hold on;
        lgd = legend('show','Location','southwest','NumColumns',2);
        title(lgd,'Bending');
        xlabel('b (m)','Interpreter','Latex')
        ylabel('$u_{z}$ (m)','Interpreter','Latex')
    end

    % Torsion plots (along span per mode)
    torsY = zeros(neig,ynod);   % LE and TE 'z' displacement difference (LE-TE) for
                                % each 'y' position divided by the chord in that
                                % 'y' position (effective torsion angle)
    figure()
    for mod = 7:14
        for j = 1:ynod
            torsLE = real((Phi(3+xdof*(j-1),mod)));
            torsTE = real((Phi(j*xdof-3,mod)));
            torsY(mod,j) = (torsLE-torsTE)/c(j);
            torsY_deg(mod,j) = torsY(mod,j).*180/pi;
        end
        plot(X(1:xnod:dim.nnod,2)',torsY_deg(mod,:),'DisplayName',...
            ['Mode ' num2str(mod)]); hold on;
        lgd = legend('show','Location','southwest','NumColumns',2);
        title(lgd,'Torsion');
        xlabel('b (m)','Interpreter','Latex')
        ylabel('$\theta_{y}$ ($^{\circ}$)','Interpreter','Latex')
    end
end


%% DYNAMIC CASE
% Modal analysis (eigenvalues and eigenvectors)
Uinf = 0:1:60;  % 'speed sweep'
Vbar = zeros(2*dim.ndof,length(Uinf),neig);
LambdaDyn = zeros(neig,length(Uinf));
pR = zeros(length(Uinf),neig);
pI = zeros(length(Uinf),neig);
PhiDyn = zeros(dim.ndof,length(Uinf),neig);

for i = 1:length(Uinf)
    Ca = -1/2*rho*Uinf(i)*(2*Q*invA*E);
    Ka = K + 1/2*rho*Uinf(i)^2*(2*Q*invA*D);

    Mbar = [Ca(If,If) M(If,If); -eye(length(If)) zeros(length(If))];
    Kbar = [Ka(If,If) zeros(length(If)); zeros(length(If)) eye(length(If))];
    [Vbar([If If+dim.ndof],i,:),LambdaDyn(:,i)] = undampedFreq(Mbar,...
        Kbar,neig);

    pR(i,:) = real(-LambdaDyn(:,i));    % real part
    pI(i,:) = imag(-LambdaDyn(:,i));  % imaginary part
    PhiDyn(:,i,:) = Vbar(1:dim.ndof,i,:);

    [maxPr_vec, ind_vec] = maxk(pR(i,:),5);   % envelope
        if maxPr_vec(1) < 1e-4 && maxPr_vec(1) > -1e-4
            if maxPr_vec(2) < 1e-4 && maxPr_vec(2) > -1e-4
                if maxPr_vec(3) < 1e-4 && maxPr_vec(3) > -1e-4
                    if maxPr_vec(4) < 1e-4 && maxPr_vec(4) > -1e-4
                        maxPr(i) = maxPr_vec(5);
                        ind(i) = ind_vec(5);
                    else
                        maxPr(i) = maxPr_vec(4);
                        ind(i) = ind_vec(4);
                    end
                else
                    maxPr(i) = maxPr_vec(3);
                    ind(i) = ind_vec(3);
                end
            else
                maxPr(i) = maxPr_vec(2);
                ind(i) = ind_vec(2);
            end
        else
            maxPr(i) = maxPr_vec(1);
            ind(i) = ind_vec(1);
        end
end

if flutter_plot == 1
    figure()
    envel = plot((Uinf),maxPr,'-b','linewidth',2,...
        'DisplayName','Envelope'); hold on;
    for mod = 1:neig
        plot((Uinf),pR(:,mod),'.k','DisplayName',...
            ['Mode ' num2str(mod)]); hold on;
    end
    flut = yline(0,'-','Flutter Condition','DisplayName','Flutter',...
        'LabelHorizontalAlignment','left');
    legend([envel flut],'Location','Best');
    xlabel('$V_{\infty} (m/s)$','Interpreter','Latex')
    ylabel('Eigenvalue $p$','Interpreter','Latex')
%     title(sprintf('Flutter Condition for $m_p=12.87$ kg'),'Interpreter','Latex')
end

% Flutter Velocity
ind_sign = find(maxPr(2:end-1).*maxPr(3:end)<0)+1;
Vflutter = interp1(maxPr([ind_sign,ind_sign+1]),Uinf([ind_sign,...
    ind_sign+1]),0);

% Dynamic Modes
Ca_F = -1/2*rho*Vflutter*(2*Q*invA*E);
Ka_F = K + 1/2*rho*Vflutter^2*(2*Q*invA*D);

Vbar_F = zeros(2*dim.ndof,neig);
LambdaDyn_F = zeros(neig,1);
Mbar_F = [Ca(If,If) M(If,If); -eye(length(If)) zeros(length(If))];
Kbar_F = [Ka(If,If) zeros(length(If)); zeros(length(If)) eye(length(If))];
[Vbar_F([If If+dim.ndof],:),LambdaDyn_F(:,1)] = undampedFreq(Mbar,Kbar,...
    neig);

if modes_plots_dyn == 1
    plotModes(X,Tn,real(Vbar_F(1:dim.ndof,:)),real(LambdaDyn_F(:,1)));
end

 % END
```

Listing D.1 Main script of the aeroelastic code developed to study the flutter instability.

```
function [X,Xs,Xr,Tn,Tns,Tnr,Tnp,Tmp,Tdp,c75elem,c4corner,c4elem,normals,...
    Sel,npan,Up] = meshFunction_STRUCT2(dim,Dx,Dy,xnod,ynod,c,c4,b,...
    nspar,nrib,mesh_plots)

yel = ynod-1;
xel = xnod-1;
w_length = 20*b;    % wake limit points (theoretically infinite)

% Coordinates matrix X
X = zeros(xnod,ynod,dim.nd);
for i = 1:xnod
    for j = 1:ynod
        X(i,j,1) = c4(j) - 1/4*c(j) + (i-1)*Dx(j);
        X(i,j,2) = (j-1)*Dy;
    end
end
X = reshape(X,dim.nnod,dim.nd);

% Forward spars (along 'y' direction)
pspar = [0.2*xel,0.7*xel];  % xel in which each forward spar is located
                            % (for each ynod), at 20% and 70% of the chord
Xspars = zeros(ynod,dim.nd,nspar);
for nspar = 1:nspar
    for j = 1:ynod
        Xspars(j,1,nspar) = (c4(j)-1/4*c(j)) + pspar(nspar)*Dx(j);
        Xspars(j,2,nspar) = (j-1)*Dy;
    end
end

% Ribs (along 'x' direction)
Xribs = zeros(xnod,dim.nd,nrib);
for nrib = 1:nrib
    for i = 1:xnod
        Xribs(i,1,nrib) = (c4(nrib)-1/4*c(nrib)) + (i-1)*Dx(nrib);
        Xribs(i,2,nrib) = (nrib-1)*Dy;
    end
end

% Connectivities matrices Tn, Tnr, Tns
nod_num = 1:dim.nnod;
nod_num = reshape(nod_num,xnod,ynod);

Tn = zeros(dim.nel,dim.nne);
Tn(:,1) = reshape(nod_num(1:xnod-1,1:ynod-1),dim.nel,1);
Tn(:,2) = reshape(nod_num(2:xnod,1:ynod-1),dim.nel,1);
Tn(:,3) = reshape(nod_num(2:xnod,2:ynod),dim.nel,1);
Tn(:,4) = reshape(nod_num(1:xnod-1,2:ynod),dim.nel,1);

Tns = zeros(yel,2,nspar);    % Linear 2-noded elements
```

```
for nspar = 1:nspar
    for j = 1:yel
        Tns(j,1,nspar) = (pspar(nspar)+1)+(j-1)*xnod;
        Tns(j,2,nspar) = (pspar(nspar)+1)+j*xnod;
    end
end

Tnr = zeros(xel,2,nrib);    % Linear 2-noded elements
for nrib = 1:nrib
    for i = 1:xel
        Tnr(i,1,nrib) = i+xnod*(nrib-1);
        Tnr(i,2,nrib) = i+xnod*(nrib-1)+1;
    end
end

    X = reshape(X,xnod,ynod,dim.nd);

    % Panel collocation points (three-quarter chord of each element,
    % x-direction)
    c75elem = zeros(dim.nel,dim.nd);   % panel collocation points
    c_mid = zeros(dim.nel,dim.nd);   % chord length at elements midpoint
    cont2 = 1;
    for j = 1:yel
        for i = 1:xel
            c_mid(cont2,1) = (c(j)+c(j+1))/(2*xel);
            c75elem(cont2,1) = 1/2*(X(1,j,1)+X(1,j+1,1)) + 3/4*...
                c_mid(cont2)+ (i-1)*c_mid(cont2);
            c75elem(cont2,2) = 1/2*(X(1,j,2)+X(1,j+1,2));
            cont2 = cont2+1;
        end
    end

    % Panel corner points (quarter chord in each y-division)
    c4corner = zeros(ynod*(xnod+1),dim.nd); % panel corner points
    cont = 1;
    for j = 1:ynod
        for i = 1:xnod+1
            if i < xnod+1
                c4corner(cont,1) = X(1,j,1) + 1/4*Dx(j) + (i-1)*Dx(j);
                c4corner(cont,2) = X(1,j,2);
            elseif i == xnod+1
                c4corner(cont,1) = c4corner(cont-1,1) + w_length;
                c4corner(cont,2) = X(1,j,2);
            end
            cont = cont+1;
        end
    end

    % Connectivities matrix Tnp of the wing aerodynamic panels
    nod_num2 = 1:size(c4corner,1);
    nod_num2 = reshape(nod_num2,xnod+1,ynod);

    npan = (xel+1)*yel;   % number of aerodynamic panels
    Tnp = zeros(npan,dim.nne);
    Tnp(:,1) = reshape(nod_num2(1:xnod,1:ynod-1),npan,1);
    Tnp(:,2) = reshape(nod_num2(1:xnod,2:ynod),npan,1);
    Tnp(:,3) = reshape(nod_num2(2:xnod+1,2:ynod),npan,1);
    Tnp(:,4) = reshape(nod_num2(2:xnod+1,1:ynod-1),npan,1);

    c4elem = c4corner;
    wake_pos = (xnod+1):(xnod+1):size(c4corner,1);
    c4elem(wake_pos,:) = [];     % eliminating the corner points from the
                                 % freestream field

    % Materials connectivities matrix Tmp of the wing aerodynamic panels
    Tmp = ones(npan,1);
    cont = 1;
    for e = 1:npan
        if e == cont*xnod
            Tmp(e) = 0;
            cont = cont+1;
        end
    end

    % DOF connectivities matrix Tdp of the wing aerodynamic panels
    Tdp = zeros(npan,1);
    cont = 0;
    for e = 1:npan
        Tdp(e) = e-cont;
        if e == (cont+1)*xnod
            cont = cont+1;
            Tdp(e) = Tdp(e-1);
        end
    end

    %__________
    X = reshape(X,dim.nnod,dim.nd); % coordinates matrix is dim.ndof*dim.nd
    %__________

    % Normal vector computation (of each element at the collocation point)
    normals = zeros(dim.nel,dim.nd);
    Sel = zeros(dim.nel,dim.nd);
    for e = 1:dim.nel
        Ak = c4elem(Tn(e,3),:)' - c4elem(Tn(e,1),:)';
        Bk = c4elem(Tn(e,4),:)' - c4elem(Tn(e,2),:)';
        Sel(e,:) = cross(Ak,Bk)/2;    % element surface
        normals(e,:) = Sel(e,:)/norm(Sel(e,:));   % normal vector of the
                                                  % flat shell element
    end

% Plotting the Finite Element mesh
if mesh_plots == 1
    XX = X(:,1);
    YY = X(:,2);
    ZZ = X(:,3);

    figure()
    view(90,90)
    patch(XX(Tn'),YY(Tn'),ZZ(Tn'),'facecolor','none','edgecolor','k');
    for nspar = 1:nspar
        if nspar == 1
            patch(Xspars(:,1,nspar),Xspars(:,2,nspar),Xspars(:,3,nspar),...
                'facecolor','none','edgecolor','b','linewidth',1.5);
        else
            patch(Xspars(:,1,nspar),Xspars(:,2,nspar),Xspars(:,3,nspar),...
                'facecolor','none','edgecolor','b');
        end
    end
    for nrib = 1:nrib
        patch(Xribs(:,1,nrib),Xribs(:,2,nrib),Xribs(:,3,nrib),...
            'facecolor','none','edgecolor','r');
    end
%      title('Finite Element Structural Mesh of 2D Wing');
        set(gca,'DataAspectRatio',[1,1,1],'xcolor','none','ycolor',...
            'none','zcolor','none','color','none');
    legend('Wing Panels','LE Spar','TE Spar','Ribs','Location',...
        'southeast','Interpreter','Latex');


        % Plotting the Finite Element mesh
        % AA -> mesh    Acoll-> collocation     Acorn -> corner points
        Xcoll = c75elem(:,1);           Xcorn = c4elem(:,1);
        Ycoll = c75elem(:,2);           Ycorn = c4elem(:,2);
        Zcoll = c75elem(:,3);           Zcorn = c4elem(:,3);

        figure()
        view(40,50)
        patch(XX(Tn'),YY(Tn'),ZZ(Tn'),'facecolor','none','edgecolor','k');
        hold on
        plot3(Xcorn,Ycorn,Zcorn,'.b');
        plot3(Xcoll,Ycoll,Zcoll,'.r');
        quiver3(Xcoll,Ycoll,Zcoll,normals(:,1),normals(:,2),...
            normals(:,3),'g');
%          title('Finite Element Aerodynamic Mesh of 2D Wing');
        set(gca,'DataAspectRatio',[1,1,1],'xcolor','none','ycolor',...
            'none','zcolor','none','color','none');
        legend('Wing panels','Bounded points $\left(1/4 \cdot c\right)$',...
            'Collocation points $\left(3/4 \cdot c\right)$','Location',...
            'southeast','Interpreter','Latex');
end

Xspars = permute(Xspars,[1 3 2]);
Xspars = reshape(Xspars,ynod*nspar,dim.nd);
Xribs = permute(Xribs,[1 3 2]);
Xribs = reshape(Xribs,xnod*nrib,dim.nd);

Tns = permute(Tns,[1 3 2]);
Tns = reshape(Tns,yel*nspar,2);
Tnr = permute(Tnr,[1 3 2]);
Tnr = reshape(Tnr,xel*nrib,2);

Xs = zeros(dim.nnod,dim.nd);
n = 1;
for k = 1:(nspar*yel)
    Xs(Tns(k,1),:) = Xspars(k,:);
    if k == n*yel
        Xs(Tns(k,2),:) = Xspars(k+1,:);
        n = n+1;
    end
end

Xr = zeros(dim.nnod,dim.nd);
m = 1;
for k = 1:(nrib*xel)
    Xr(Tnr(k,1),:) = Xribs(k,:);
    if k == m*xel
        Xr(Tnr(k,2),:) = Xribs(k+1,:);
        m = m+1;
    end
end


% Fixed DOFs matrix Up
Up = zeros(xnod*dim.ni,3); % all left nodes are campled (y = 0)
cont = 0;
for nod = 1:dim.nnod
    if X(nod,2) == 0
        for ni = 1:dim.ni
            Up(cont+ni,2) = nod;
            Up(cont+ni,3) = ni;
        end
        cont = cont+dim.ni;
    end
end
```

Listing D.2 Function that discretizes the mesh and computes the boundary conditions.

```
function [A,B] = infcoeff_VLM(npan,Tnp,Tmp,Tdp,xel,yel,c4corner,c75elem,normals)

A = zeros(xel*yel,xel*yel);  % influence coefficient matrix
B = zeros(xel*yel,xel*yel);  % influence coefficient in the streamwise
                             % direction matrix
gamma = 1;  % unitary circulation

for i = 1:xel*yel
    np = normals(i,:)';
    PC = c75elem(i,:);     % collocation point 'i'
    for e = 1:npan
        P = c4corner(Tnp(e,:),:);
            P1 = P(1,:);
            P2 = P(2,:);
            P3 = P(3,:);
            P4 = P(4,:);

        if Tmp(e) == 1       % v_ind due to vortex ring
            u12 = vortxl_VLM(P1,P2,PC,gamma);    % Vortex Head 1-2
            u23 = vortxl_VLM(P2,P3,PC,gamma);    % Right side Vortex 2-3
            u34 = vortxl_VLM(P3,P4,PC,gamma);    % Free Stream Vortex 3-4
            u41 = vortxl_VLM(P4,P1,PC,gamma);    % Left Side Vortex 4-1

            u_ind = u12+u23+u34+u41;    % Induced Velocity
            u_ind_stream = u23+u41;     % Induced Velocity in the
                                        % freestream direction

        elseif Tmp(e) == 0   % v_ind due to horseshoe vortex
            u41 = vortxl_VLM(P4,P1,PC,gamma);    % Left Side Vortex 4-1
            u12 = vortxl_VLM(P1,P2,PC,gamma);    % Vortex Head 1-2
            u23 = vortxl_VLM(P2,P3,PC,gamma);    % Right Side Vortex 2-3

            u_ind = u41+u12+u23;
            u_ind_stream = u23+u41;

        end
        A(i,Tdp(e)) = A(i,Tdp(e)) + dot(u_ind,np);
        B(i,Tdp(e)) = B(i,Tdp(e)) + dot(u_ind_stream,np);
    end
end
```

Listing D.3 Function that computes the influence coefficients matrix.

```
function u = vortxl_VLM(X1,X2,XP,gamma)  % 'X2-X1' form the vortex line
                                         % 'XP' is the control point
                                         % 'gamma' is the circulation
                                         % 'u' is the velocity induced by
                                         % the vortex line
inv_4pi = 0.25/pi;

r0 = X2-X1;
r1 = XP-X1;  % defined as -(X-XP) to have beta in the right side
r2 = XP-X2;

norm_r1 = norm(r1);
norm_r2 = norm(r2);

r1xr2 = cross(r1,r2);
norm_r1xr2 = norm(r1xr2);

tol = 1.0e-6 ;

if (norm_r1>tol && norm_r2>tol && norm_r1xr2>tol)
    inv_r1xr2 = 1 / norm_r1xr2;
    inv_r1 = 1 / norm_r1;
    inv_r2 = 1 / norm_r2;

    a = r0 * inv_r1xr2;
    b = r1*inv_r1 - r2*inv_r2;
    c = dot(a,b);

    u = inv_4pi*gamma*c*r1xr2*inv_r1xr2;
else
    u = 0;
end
```

Listing D.4 Function that computes the induced velocity by a quadrilateral vortex element.

```
function [K,M,R,N4,S4] = stiffnessPlate_STRUCT(X,Tn,dim,h,E,nu,rho)
% Initialization of the global stiffness and mass matrices
K = zeros(dim.ndof,dim.ndof);
M = zeros(dim.ndof,dim.ndof);

% Assembly process
for el = 1:dim.nel
    % (a) Compute rotation matrix
    S = cross( (X(Tn(el,3),:)' - X(Tn(el,1),:)') , ...  % element surface
        (X(Tn(el,4),:)' - X(Tn(el,2),:)') )/2;
    k_prime = S/norm(S);   % normal vector of the flat shell element
    d = (X(Tn(el,2),:)' + X(Tn(el,3),:)' - X(Tn(el,4),:)' - X(Tn(el,1),:)');
    i_prime = d/norm(d); % i_prime = i'
    j_prime = cross(k_prime,i_prime);
    R_prime = [i_prime j_prime k_prime zeros(3,3);
        zeros(3,3) i_prime j_prime k_prime]';
    R(:,:,el) = [R_prime zeros(6,6) zeros(6,6) zeros(6,6);
        zeros(6,6) R_prime zeros(6,6) zeros(6,6);
        zeros(6,6) zeros(6,6) R_prime zeros(6,6);
        zeros(6,6) zeros(6,6) zeros(6,6) R_prime];

    % (b) Get nodal coefficients for the shape functions
    a = [-1 1 1 -1];
    b = [-1 -1 1 1];

    % (c) Compute element matrices
        % (c1) 1 Gauss point quadrature matrices
        N1 = [1 1 1 1]'/4;
        N1xi = a/4;
        N1eta = b/4;
        J1 = zeros(2,2);
        for nne = 1:dim.nne      % dim.nne = number of nodes per element
            J1 = J1 + [N1xi(nne);N1eta(nne)]*X(Tn(el,nne),:)*...
                [i_prime j_prime];
        end
        N1x_prime = J1\[N1xi;N1eta];
        S1 = 4*det(J1);

            % (c1.1) Ficticious
            for nne = 1:dim.nne
                Bt_prime_nod(1,:,nne) = [0 0 0 0 0 N1(nne)]; % each element
            end
            Ct_prime = 5*h*E/(12*(1+nu));
            Bt_prime(1,:,el) = [Bt_prime_nod(1,:,1) Bt_prime_nod(1,:,2)...
                Bt_prime_nod(1,:,3) Bt_prime_nod(1,:,4)];
            Kt(:,:,el) = S1*(R(:,:,el)'*Bt_prime(1,:,el)'*Ct_prime*...
                Bt_prime(1,:,el)*R(:,:,el));

            % (c1.2) Shear
            for nne = 1:dim.nne
                Bs_prime_nod(:,:,nne) = [0 0 N1x_prime(1,nne) 0 N1(nne) 0;
                    0 0 N1x_prime(2,nne) -N1(nne) 0 0];
            end
            Cs_prime = [1 0; 0 1]*5*h*E/(12*(1+nu));
            Bs_prime(:,:,el) = [Bs_prime_nod(:,:,1) Bs_prime_nod(:,:,2) ...
                Bs_prime_nod(:,:,3) Bs_prime_nod(:,:,4)];
            Ks(:,:,el) = S1*(R(:,:,el)'*Bs_prime(:,:,el)'*Cs_prime*...
                Bs_prime(:,:,el)*R(:,:,el));

            % (c1.3) Membrane
            for nne = 1:dim.nne
                Bm_prime_nod(:,:,nne) = [N1x_prime(1,nne) 0 0 0 0 0;
                    0 N1x_prime(2,nne) 0 0 0 0;
                    N1x_prime(2,nne) N1x_prime(1,nne) 0 0 0 0];
            end
            Cm_prime = [1 nu 0; nu 1 0; 0 0 (1-nu)/2]*h*E/(1-nu^2);
            Bm_prime(:,:,el) = [Bm_prime_nod(:,:,1) Bm_prime_nod(:,:,2) ...
                Bm_prime_nod(:,:,3) Bm_prime_nod(:,:,4)];
            Km(:,:,el) = S1*(R(:,:,el)'*Bm_prime(:,:,el)'*Cm_prime*...
                Bm_prime(:,:,el)*R(:,:,el));

        % (c2) 4 Gauss point quadrature matrices
        Kb = zeros(4*dim.ni,4*dim.ni,dim.nel);  % 4 Gauss points x dof/node
        Me = zeros(4*dim.ni,4*dim.ni,dim.nel);  % 4 Gauss points x dof/node
        xi4 = 1/(3^(1/2))*[-1 1 1 -1];
        eta4 = 1/(3^(1/2))*[-1 -1 1 1];
        w4 = [1 1 1 1];

        for k = 1:length(xi4)
            J4 = zeros(2,2);
            for nne = 1:dim.nne
                N4(nne) = (1+a(nne)*xi4(k))*(1+b(nne)*eta4(k))/4;
                N4xi(1,nne) = a(nne)*(1+b(nne)*eta4(k))/4;
                N4eta(1,nne) = b(nne)*(1+a(nne)*xi4(k))/4;
                J4 = J4 + [N4xi(nne); N4eta(nne)]*X(Tn(el,nne),:)*...
                    [i_prime j_prime];
            end
            N4x_prime = J4\[N4xi; N4eta];
            S4(el,k) = w4(k)*det(J4);

                % (c2.1) Bending
                for nne = 1:dim.nne
                    Bb_prime_nod(:,:,nne) = [0 0 0 0 N4x_prime(1,nne) 0;
                        0 0 0 N4x_prime(2,nne) 0 0;
                        0 0 0 -N4x_prime(1,nne) N4x_prime(2,nne) 0];
                end
                Cb_prime = [1 nu 0; nu 1 0; 0 0 (1-nu)/2]*h^3*E/...
                    (12*(1-nu^2));
                Bb_prime(:,:,el,k) = [Bb_prime_nod(:,:,1) ...
                    Bb_prime_nod(:,:,2) Bb_prime_nod(:,:,3) ...
                    Bb_prime_nod(:,:,4)];
                Kb(:,:,el) = Kb(:,:,el) + S4(el,k)*(R(:,:,el)'*...
                    Bb_prime(:,:,el,k)'*Cb_prime*Bb_prime(:,:,el,k)*R(:,:,el));

                % (c2.2) Mass matrix
                for nne = 1:dim.nne
                    Nnod(:,:,nne) = N4(nne)*eye(dim.ni);
                end
                rhoM = rho*h*[eye(3) zeros(3,3);
                    zeros(3,3) [h^2/12 0 0; 0 h^2/12 0; 0 0 0]];
                N(:,:,el,k) = [Nnod(:,:,1) Nnod(:,:,2) Nnod(:,:,3) Nnod(:,:,4)];
                Me(:,:,el) = Me(:,:,el)+S4(el,k)*(R(:,:,el)'*N(:,:,el,k)'*...
                    rhoM*N(:,:,el,k)*R(:,:,el));
        end

    % (d) Assembly to global matrices
    for ni = 1:dim.ni
        Idof(ni,1) = dim.ni*(Tn(el,1)-1) + ni;
        Idof(dim.ni + ni,1) = dim.ni*(Tn(el,2)-1) + ni;
        Idof(2*dim.ni + ni,1) = dim.ni*(Tn(el,3)-1) + ni;
        Idof(3*dim.ni + ni,1) = dim.ni*(Tn(el,4)-1) + ni;
    end
    K(Idof,Idof) = K(Idof,Idof)+Km(:,:,el)+Kb(:,:,el)+Ks(:,:,el)+Kt(:,:,el);
    M(Idof,Idof) = M(Idof,Idof)+Me(:,:,el);
end
```

Listing D.5 Function that computes the stiffness and mass matrices of the structural plate element.

```matlab
function [K,M] = stiffnessSpars_STRUCT(X,Tn,dim,k_prime,E,G,A,J,Iy_prime,...
    Iz_prime,ky,kz,yel,rho)

% Initialization of the global stiffness and mass matrices
K = zeros(dim.ndof,dim.ndof);
M = zeros(dim.ndof,dim.ndof);
el_spar = yel;  % elements in each spar

% Assembly process
for el = 1:el_spar
    % (a) Compute rotation matrix
    l = norm(X(Tn(el,2),:)-X(Tn(el,1),:));
    i_prime = (X(Tn(el,2),:)'-X(Tn(el,1),:)')/l;
    j_prime = cross(k_prime,i_prime);   % normal vector of the linear elem
    R_prime = [i_prime j_prime k_prime zeros(3,3);
        zeros(3,3) i_prime j_prime k_prime]';
    R(:,:,el) = [R_prime zeros(6,6);
        zeros(6,6) R_prime];

    % (b) Compute shape function derivatives
    Nx_prime(1) = -1/l;
    Nx_prime(2) = 1/l;

    % (c) Compute each element matrix
        % (c1) Axial component of stiffness matrix
        Ba_prime(1,:,el) = [Nx_prime(1) 0 0 0 0 0 Nx_prime(2) 0 0 0 0 0];
        Ca_prime = E*((A(el)*A(el+1))/2);
        Ka(:,:,el) = l*(R(:,:,el)'*Ba_prime(1,:,el)'*Ca_prime*...
            Ba_prime(1,:,el)*R(:,:,el));

        % (c2) Bending component of stiffness matrix
        Bb_prime(:,:,el) = [0 0 0 0 Nx_prime(1) 0 0 0 0 0 Nx_prime(2) 0;
            0 0 0 0 0 Nx_prime(1) 0 0 0 0 0 Nx_prime(2)];
        Cb_prime = E*[(Iy_prime(el)+Iy_prime(el+1))/2 0;
            0 (Iz_prime(el)+Iz_prime(el+1))/2];
        Kb(:,:,el) = l*(R(:,:,el)'*Bb_prime(:,:,el)'*Cb_prime*...
            Bb_prime(:,:,el)*R(:,:,el));

        % (c3) Shear component of stiffness matrix
        N = 1/2;    % shape functions assuming only 1 Gauss point
        Bs_prime(:,:,el) = [0 Nx_prime(1) 0 0 0 -N 0 Nx_prime(2) 0 0 0 -N;
            0 0 Nx_prime(1) 0 N 0 0 0 Nx_prime(2) 0 N 0];
        Cs_prime = G*((A(el)*A(el+1))/2)*[ky 0; 0 kz];
        Ks(:,:,el) = l*(R(:,:,el)'*Bs_prime(:,:,el)'*Cs_prime*...
            Bs_prime(:,:,el)*R(:,:,el));

        % (c4) Torsion component of stiffness matrix
        Bt_prime(1,:,el) = [0 0 0 Nx_prime(1) 0 0 0 0 0 Nx_prime(2) 0 0];
        Ct_prime = G*((J(el)*J(el+1))/2);
        Kt(:,:,el) = l*(R(:,:,el)'*Bt_prime(1,:,el)'*Ct_prime*...
            Bt_prime(1,:,el)*R(:,:,el));

        % (c5) Mass matrix
        xi = 1/(3^(1/2))*[1;1];     % Gauss points coordinates
        w = [1;1];  % Gauss points weights
        Ael = (A(el)*A(el+1))/2;
        Jel = (J(el)*J(el+1))/2;
        Iy_prime_el = (Iy_prime(el)+Iy_prime(el+1))/2;
        Iz_prime_el = (Iz_prime(el)+Iz_prime(el+1))/2;
        rhoM = rho.*[Ael;Ael;Ael;Jel;Iy_prime_el;Iz_prime_el].*eye(dim.ni);
        Me(:,:,el) = zeros(2*dim.ni,2*dim.ni);
        for k = 1:length(xi)
            Nnod(1) = (1-xi(k))/2;
            Nnod(2) = (1+xi(k))/2;
            Nmat(:,:,el,k) = [Nnod(1)*eye(dim.ni) Nnod(2)*eye(dim.ni)];
            Me(:,:,el) = Me(:,:,el)+w(k)*l*(R(:,:,el)'*Nmat(:,:,el,k)'*...
                rhoM*Nmat(:,:,el,k)*R(:,:,el))/2;
        end

    % (d) Assembly to global matrices
    for ni = 1:dim.ni
        Idof(ni,1) = dim.ni*(Tn(el,1)-1) + ni;
        Idof(dim.ni + ni,1) = dim.ni*(Tn(el,2)-1) + ni;
    end
    K(Idof,Idof) = K(Idof,Idof)+Ka(:,:,el)+Kb(:,:,el)+Ks(:,:,el)+Kt(:,:,el);
    M(Idof,Idof) = M(Idof,Idof)+Me(:,:,el);

end
```

Listing D.6 Function that computes the stiffness and mass matrices of the spar beam elements.

```matlab
function [K,M] = stiffnessRibs_STRUCT(X,Tn,dim,j_prime,E,G,A,J,Iy_prime,...
    Iz_prime,ky,kz,xel,rho)

% Initialization of the global stiffness and mass matrices
K = zeros(dim.ndof,dim.ndof);
M = zeros(dim.ndof,dim.ndof);
el_rib = xel;  % elements in each rib

% Assembly process
for el = 1:el_rib
    % (a) Compute rotation matrix
    l = norm(X(Tn(el,2),:)-X(Tn(el,1),:));
    i_prime = (X(Tn(el,2),:)'-X(Tn(el,1),:)')/l;
    k_prime = cross(i_prime,j_prime);   % normal vector of the linear elem
    R_prime = [i_prime j_prime k_prime zeros(3,3);
        zeros(3,3) i_prime j_prime k_prime]';
    R(:,:,el) = [R_prime zeros(6,6);
        zeros(6,6) R_prime];

    % (b) Compute shape function derivatives
    Nx_prime(1) = -1/l;
    Nx_prime(2) = 1/l;

    % (c) Compute each element matrix
        % (c1) Axial component of stiffness matrix
        Ba_prime(1,:,el) = [Nx_prime(1) 0 0 0 0 0 Nx_prime(2) 0 0 0 0 0];
        Ca_prime = E*((A(el)*A(el+1))/2);
        Ka(:,:,el) = l*(R(:,:,el)'*Ba_prime(1,:,el)'*Ca_prime*...
            Ba_prime(1,:,el)*R(:,:,el));

        % (c2) Bending component of stiffness matrix
        Bb_prime(:,:,el) = [0 0 0 0 Nx_prime(1) 0 0 0 0 0 Nx_prime(2) 0;
            0 0 0 0 0 Nx_prime(1) 0 0 0 0 0 Nx_prime(2)];
        Cb_prime = E*[(Iy_prime(el)+Iy_prime(el+1))/2 0;
            0 (Iz_prime(el)+Iz_prime(el+1))/2];
        Kb(:,:,el) = l*(R(:,:,el)'*Bb_prime(:,:,el)'*Cb_prime*...
            Bb_prime(:,:,el)*R(:,:,el));

        % (c3) Shear component of stiffness matrix
        N = 1/2;    % shape functions assuming only 1 Gauss point
        Bs_prime(:,:,el) = [0 Nx_prime(1) 0 0 0 -N 0 Nx_prime(2) 0 0 0 -N;
            0 0 Nx_prime(1) 0 N 0 0 0 Nx_prime(2) 0 N 0];
        Cs_prime = G*((A(el)*A(el+1))/2)*[ky 0; 0 kz];
        Ks(:,:,el) = l*(R(:,:,el)'*Bs_prime(:,:,el)'*Cs_prime*...
            Bs_prime(:,:,el)*R(:,:,el));

        % (c4) Torsion component of stiffness matrix
        Bt_prime(1,:,el) = [0 0 0 Nx_prime(1) 0 0 0 0 0 Nx_prime(2) 0 0];
        Ct_prime = G*((J(el)*J(el+1))/2);
        Kt(:,:,el) = l*(R(:,:,el)'*Bt_prime(1,:,el)'*Ct_prime*...
            Bt_prime(1,:,el)*R(:,:,el));

        % (c5) Mass matrix
        xi = 1/(3^(1/2))*[1;1];     % Gauss points coordinates
        w = [1;1];  % Gauss points weights
        Ael = (A(el)*A(el+1))/2;
        Jel = (J(el)*J(el+1))/2;
        Iy_prime_el = (Iy_prime(el)+Iy_prime(el+1))/2;
        Iz_prime_el = (Iz_prime(el)+Iz_prime(el+1))/2;
        rhoM = rho.*[Ael;Ael;Ael;Jel;Iy_prime_el;Iz_prime_el].*eye(dim.ni);
        Me(:,:,el) = zeros(2*dim.ni,2*dim.ni);
        for k = 1:length(xi)
            Nnod(1) = (1-xi(k))/2;
            Nnod(2) = (1+xi(k))/2;
            Nmat(:,:,el,k) = [Nnod(1)*eye(dim.ni) Nnod(2)*eye(dim.ni)];
            Me(:,:,el) = Me(:,:,el)+w(k)*l*(R(:,:,el)'*Nmat(:,:,el,k)'*...
                rhoM*Nmat(:,:,el,k)*R(:,:,el))/2;
        end

    % (d) Assembly to global matrices
    for ni = 1:dim.ni
        Idof(ni,1) = dim.ni*(Tn(el,1)-1) + ni;
        Idof(dim.ni + ni,1) = dim.ni*(Tn(el,2)-1) + ni;
    end
    K(Idof,Idof) = K(Idof,Idof)+Ka(:,:,el)+Kb(:,:,el)+Ks(:,:,el)+Kt(:,:,el);
    M(Idof,Idof) = M(Idof,Idof)+Me(:,:,el);

end
```

Listing D.7 Function that computes the stiffness and mass matrices of the rib beam elements.

```matlab
function D = matrixD_STRUCT(dim,Tn,X)

D = zeros(dim.nel,dim.ndof);

for e = 1:dim.nel
    d = (X(Tn(e,2),1)+X(Tn(e,3),1))/2 - (X(Tn(e,1),1)+X(Tn(e,4),1))/2;
    D(e,Tn(e,:)*6-3) = 1/(2*d).*[1 -1 -1 1];
end
```

Listing D.8 Function that computes matrix D.

```matlab
function Q = matrixQ_STRUCT(dim,Tn,xel,X)

Q = zeros(dim.ndof,dim.nel);
e2 = 1:dim.nel; % adjacent element to 'e' closer to LE
e2 = e2-1;
eLE = 1:xel:dim.nel;
e2(eLE) = 0;

for e = 1:dim.nel
    dy = (X(Tn(e,4),2)+X(Tn(e,3),2))/2 - (X(Tn(e,1),2)+X(Tn(e,2),2))/2;
    Idof = Tn(e,:)*6-3;

    Q(Idof,e) = Q(Idof,e) + dy/4;
    if e2(e) ~= 0
        Q(Idof,e2(e)) = Q(Idof,e2(e)) - dy/4;
    end
end
```

Listing D.9 Function that computes matrix Q.

```matlab
function E = matrixE_STRUCT(dim,Tn)

E = zeros(dim.nel,dim.ndof);

for e = 1:dim.nel
    xi_c = 0.5;     % isoparametric coordinates of the collocation point
    eta_c = 0;
    a = [-1 1 1 -1];
    b = [-1 -1 1 1];
    for i = 1:4
        E(e,Tn(e,i)*6-3) = 1/4*(1+xi_c*a(i))*(1+eta_c*b(i));
        % shape function at node 'i' of element 'e'
    end
end
```

Listing D.10 Function that computes matrix E.

```
function [Ip,If] = dofVec(Up,dim)
for p = 1:size(Up,1)
        Ip(p) = dim.ni*(Up(p,2)-1)+Up(p,3); % vector with prescribed DOFs
        up(Ip(p),1) = Up(p,1);
end
If = setdiff(1:dim.ndof,Ip);    % vector with free DOFs
```

Listing D.11 Function that computes the fixed and free DOFs vectors.

```
function plotModes(X,Tn_s,Phi,w2)  % Function to plot the modes shapes
% INPUTS:
% - X       Nodal coordinates matrix
% - Tn_s    Nodal connectivities matrix for shell elements
% - Phi     Matrix with global degrees of freedom for each mode
%           - each row correspond to a degree of freedom (same as 'u')
%           - each column corresponds to a different mode
% - w2      Squared natural frequencies vector

Nmodes = size(Phi,2);
for i = 1:ceil(Nmodes/6)
    figure()
    for j = 1:6
        k = 6*(i-1)+j;
        if k<=Nmodes
            scale = 0.3/max(abs(real([Phi(1:6:end,k);Phi(2:6:end,k);...
                Phi(3:6:end,k)])));
            x = X(:,1)+scale*Phi(1:6:end,k);
            y = X(:,2)+scale*Phi(2:6:end,k);
            z = X(:,3)+scale*Phi(3:6:end,k);
            subplot(3,2,j);
            hold on
            patch(x(Tn_s)',y(Tn_s)',z(Tn_s)',ones(size(Tn_s))',...
                'facecolor',[0.8,0.8,0.8],'edgecolor','k');
            set(gca,'DataAspectRatio',[1,1,1],'xcolor','none','ycolor','none','zcolor','none
    ','color','none');
            view(150,10)
            if nargin>3
%                title(sprintf('f = %g Hz',sqrt(w2(k))/2/pi));
%                % structural dynamics, vibrations

                title(sprintf('\\textbf{(M%g) } $q_{\\infty} = %g \\frac{kg}{m \\cdot s^2}
    $',(i-1)*6+j,w2(k)),'Interpreter','latex');
                % aerodynamics, dynamic response
            end
        end
    end
end
end
```

Listing D.12 Function that plot the eigenmodes.

```
function [MODES,EIGENVAL] = undampedFreq(M,K,neig)

  [MODES,EIGENVAL] = eigs(K,M,neig,'sm'); % 'sm' means smallest magnitude
                                          % 'lm' means largest magnitude
  EIGENVAL = diag(EIGENVAL);

  % Dynamic Response (Aerodynamics)
  [EIGENVAL,imodes] = sort(EIGENVAL);
  MODES = MODES(:,imodes);

%   % Vibrations (Structural Dynamics)
%   FREQ = sqrt(EIGENVAL);
%   [FREQ,imodes] = sort(FREQ);
```

Listing D.13 Function that computes the eigenvalues and eigenvectors of a matrix modal equation.

References

1. ROCA, David. *Computational Engineering: FEM Beam Elements.* Universitat Politècnica de Catalunya, February 2022.
2. OÑATE, Eugenio. *Structural analysis with the finite element method. Linear statics: volume 2: beams, plates and shells.* Springer Science & Business Media, 2013.
3. ROCA, David. *Computational Engineering: FEM Plate and Shell Elements.* Universitat Politècnica de Catalunya, February 2022.
4. MEGAHED, K. Introduction to Nonlinear Finite Element Analysis. [N.d.].
5. ORTEGA, Enrique. *Aerodynamics Module 2.1: Ideal Flow.* Universitat Politècnica de Catalunya, October 2020.
6. ANDERSON JR, John David. *Fundamentals of aerodynamics.* Tata McGraw-Hill Education, 2010.
7. KATZ, Joseph; PLOTKIN, Allen. *Low-speed aerodynamics.* Cambridge university press, 2001.
8. KARAMCHETI, K. *Principles of Ideal-Fluid Aerodynamics.* Wiley, 1966.
9. TEJERIZO FERNÁNDEZ, Maribel. Elaboración de fórmulas analíticas y tablas de cálculo para las estructuras metálicas de acero según la normativa Eurocódigo 3. 2015.
10. YOUNG, Warren C; BUDYNAS, Richard G; SADEGH, Ali M. *Roark's formulas for stress and strain.* McGraw-Hill Education, 2012.
11. SUH, PM. EZASE Easy Aeroelasticity: A Tool to Simulate Aircraft Wing Geometry. *NASA: Washington, DC, USA.* 2011.