UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Acoplamiento aeroelástico de una aeronave completa mediante interpolación con splines

Trabajo Fin de Máster

Máster Universitario en Ingeniería Aeronáutica

AUTOR/A: Ramírez Conca, Javier

Tutor/a: Lázaro Navarro, Mario

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

TRABAJO FIN DE MÁSTER

ACOPLAMIENTO AEROELÁSTICO DE UNA AERONAVE COMPLETA MEDIANTE LA INTERPOLACIÓN CON *SPLINES*

AUTOR

JAVIER RAMÍREZ CONCA

TUTOR

MARIO LÁZARO NAVARRO

MÁSTER EN INGENIERÍA AERONÁUTICA

CURSO ACADÉMICO 2022/2023

Resumen

El presente trabajo busca realizar la modelización de la estructura de una aeronave completa mediante un emparrillado de elementos viga (*Euler-Bernoulli*), y su acoplamiento con el modelo aerodinámico *Vortex Lattice Method* (método numérico 3D que modela las superficies aerodinámicas como una malla de paneles) a través de la interpolación con *splines*. De esta forma, se pretende obtener un modelo global con el objetivo de analizar tanto el problema aeroelástico estático (divergencia) como el dinámico (flameo) del avión completo.

Palabras clave: Método Malla de Torbellinos, interpolación, *splines*, modelización, estructura, aeroelasticidad, divergencia, flameo, aerodinámica

Abstract

The aim of this work is to model the structure of a complete aircraft by means of a grid of beam elements (*Euler-Bernoulli*), and its coupling with the aerodynamic model *Vortex Lattice Method* (a 3D numerical method that models the aerodynamic surfaces as a mesh of panels) through *splines* interpolation. In this way, the aim is to obtain a global model with the objective of analysing both the static aeroelastic problem (divergence) and the dynamic problem (flutter) of the entire aircraft.

Keywords: *Vortex Lattice Method*, interpolation, *splines*, modelling, structure, aeroelasticity, divergence, flutter, aerodynamics

Resum

El present treball busca realitzar la modelització de l'estructura d'una aeronau completa mitjançant un engraellat d'elements biga (*Euler-Bernoulli*), i el seu acoblament amb el model aerodinàmic *Vortex Lattice Method* (mètode numèric 3D que modela les superfícies aerodinàmiques com una malla de panells) a través de la interpolació amb *splines*. D'aquesta manera, es pretén obtindre un model global amb l'objectiu d'analitzar tant el problema aeroelèstic estàtic (divergència) com el dinàmic (flamege) de l'avió complet.

Paraules clau: Mètode Malla de Remolins, interpolació, *splines*, modelització, estructura, aeroelasticitat, divergència, flamege, aerodinàmica

Agradecimientos

En primer lugar, quería agradecer a mi tutor del proyecto, Mario Lázaro, por motivarme con sus clases del segundo curso del presente máster a escoger un tema relacionado con la Aeroelasticidad para mi trabajo final. Gracias a su propuesta, consejo y disponibilidad durante todo el curso ha sido posible la elaboración del mismo.

En segundo lugar, no puedo olvidarme de mi familia, mis padres, quienes siempre han estado apoyándome y animándome en todo este camino desde que empecé la carrera en 2017. A ellos les debo el hecho de poder haber estudiado aquello que me gusta. Y, por supuesto, a mi hermano, mi ejemplo a seguir en todos los ámbitos de la vida.

Por último, no quiero olvidarme de mis amigos del pueblo que, aunque no hayan estado conmigo en el día a día, siempre se han interesado por mi y me han mostrado su apoyo incondicional. Y también, a mis amigos de la carrera y las nuevas amistades del máster, gracias a ellos estos 2 últimos años han sido inolvidables.

Índice

Índice de figuras

Índice de tablas

Nomenclatura

α	Ángulo de ataque del ala	rad
$\boldsymbol{\lambda}$	Vector de frecuencias naturales de la estructura	
$\delta\mathcal{W}$	Trabajo virtual de las fuerzas exteriores	
Δc_{pj}	Incremento de coeficiente de presión entre extradós e intradós en el panel j	-
Δp_j	Incremento de presión entre extradós e intradós en el panel j	Pa
ΔY_j	Longitud del panel j proyectada en dirección de eje y	m
$\frac{\partial w}{\partial y}$	Grado de libertad de giro debido a la flexión	rad
Γ	Intensidad de torbellino	m^2/s
$\mathbf{a}$	Vector de ángulos de ataque de cada panel	rad
$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$	Matrices derivadas del problema aeroelástico numérico	
$\mathbf{B}_\sigma$, $\mathbf{f}_\sigma$, $\mathbf{R}_\sigma$, $\boldsymbol{\Psi}_\sigma$	Matrices derivadas de la interpolación por *splines* de superficie	
$\mathbf{D}_u$	Matriz que relaciona $\mathbf{a}$ con el vector de grados de libertad estructurales, $\mathbf{u}$	
$\mathbf{D}_v$	Matriz que relaciona $\mathbf{a}$ con la derivada respecto del tiempo del vector de grados de libertad estructurales, $\dot{\mathbf{u}}$	
$\mathbf{D}_z$	Matriz que relaciona $\mathbf{z}_A$ con el vector de grados de libertad estructurales, $\mathbf{u}$	
$\mathbf{D}_{eq}$	Matriz de amortiguamiento equivalente	
$\mathbf{H}$	Matriz de coeficientes de influencia aerodinámicos	m^{-1}
$\mathbf{i}$	Vector unitario en dirección x	
$\mathbf{j}$	Vector unitario en dirección y	
$\mathbf{k}$	Vector unitario en dirección z	
$\mathbf{K}^e$, $\mathbf{K}$	Matriz de rigidez de elemento y global	
$\mathbf{K}_{eq}$	Matriz de rigidez equivalente	

$\mathbf{M}$	Matriz de masa global	
$\mathbf{M}_{eq}$	Matriz de masa equivalente	
$\mathbf{n}$	Vector normal a la superficie definida por F	
$\mathbf{N}^e$	Matriz de funciones de forma de cada elemento estructural	
$\mathbf{N}_\sigma$	Matriz de relación entre los grados de libertad estructurales y el desplazamiento vertical de cualquier punto de la superficie σ	
$\mathbf{Q}$	Vector de las fuerzas generalizadas asociadas a los grados de libertad estructurales	
$\mathbf{R}_z$	Matriz de rotación alrededor del eje z	
$\mathbf{u}^e$	Vector de grados de libertad de cada elemento estructural	
$\mathbf{V}$	Vector velocidad total	m/s
$\mathbf{V}_\Gamma$	Vector velocidad inducida por un torbellino en herradura de intensidad Γ	m/s
$\mathbf{V}_\infty$	Vector velocidad del flujo libre	m/s
$\mathbf{V}_{aeronave}$	Vector velocidad debido a maniobras de la aeronave	m/s
$\mathbf{z}_A$	Vector con los desplazamientos verticales de los centros aerodinámicos de cada panel	m
$\mathcal{T}$	Energía cinética total	
$\mathcal{U}^e, \mathcal{U}$	Energía de deformación total de elemento y global	
$\mathcal{U}_f^e$	Energía de deformación por flexión de elemento	
$\mathcal{U}_t^e$	Energía de deformación por torsión de elemento	
ω_f	Frecuencia de flameo	Hz
ω_j	Parte imaginaria del autovalor s_j, frecuencias	rad/s
ω_n	Frecuencias naturales	Hz
ϕ	Función potencial	
ρ_∞	Densidad del flujo libre	$\mathrm{Kg/m^3}$

θ	Grado de libertad de torsión	rad
ξ	Coordenada adimensional en dirección y, $\xi = \frac{2y}{L^e}$	-
AR_e	Alargamiento del estabilizador	-
AR_w	Alargamiento del ala	-
b_e	Envergadura del estabilizador	m
b_w	Envergadura del ala	m
c_{re}	Cuerda en la raíz del estabilizador	m
c_{rw}	Cuerda en la raíz del ala	m
c_{te}	Cuerda en la punta del estabilizador	m
c_{tw}	Cuerda en la punta del ala	m
cma	Cuerda media aerodinámica del ala	m
cma_e	Cuerda media aerodinámica del estabilizador	m
F	Función que define la configuración geométrica de una superficie	
F_j	Fuerza debida al torbellino del panel j	N/m
g_j	Parte real del autovalor s_j, amortiguamientos	rad/s
L	Sustentación total	N
L^e	Longitud de cada elemento estructural	m
L_j	Sustentación en el panel j	N
m_i	Masa puntual del nodo i	Kg
$MTOW$	Máximo peso al despegue	Kg
N_e	Número de elementos	
N_n	Número de nodos	
N_p	Número de paneles	
$N_{\frac{\partial w}{\partial y}}$	Funciones de forma del giro de flexión $\frac{\partial w}{\partial y}$	

N_θ	Funciones de forma de torsión θ	
N_w	Funciones de forma de la flecha w	
OEW	Peso en vacío	Kg
q	Presión dinámica	Pa
S_j	Superficie del panel j	m^2
s_j	Autovalor j del problema aeroelástico dinámico, $s_j = g_j + i\ \omega_j$	
S_t	Superficie en planta del estabilizador	m^2
S_w	Superficie en planta del ala	m^2
U_D	Velocidad de divergencia	m/s
U_f	Velocidad de flameo	m/s
U_∞	Velocidad del flujo libre	m/s
VLM	*Vortex Lattice Method*	
w	Grado de libertad de desplazamiento vertical	m
x_{C_j}	Posición del punto de control del panel j	m
x_{CA_j}	Posición del centro aerodinámico del panel j	m
E	Módulo de rigidez a flexión del material	N/m^2
EI	Rigidez a flexión	Nm^2
G	Módulo de rigidez a torsión del material	N/m^2
GJ	Rigidez a torsión	Nm^2
I, J	Momentos de inercia de la sección del elemento	m^4

1. Introducción

1.1. Marco teórico y motivación

1.1.1. Aeroelasticidad: definición e introducción

El proyecto se centra en el desarrollo de un modelo numérico global capaz de modelizar la estructura de una aeronave completa en plano horizontal (fuselaje, ala y estabilizador) y su acoplamiento con la modelización aerodinámica de las superficies sustentadoras, con el objetivo de poder abarcar el problema aeroelástico, es decir, la relación entre las fuerzas aerodinámicas generadas debido a la acción del aire sobre la aeronave y su influencia en la deformación de la estructura, y viceversa.

De esta forma, se introduce el concepto *aeroelasticidad*, definido como la rama de la mecánica que estudia la respuesta de un sistema flexible en presencia de una corriente de aire. Este concepto se explica mejor de forma gráfica mediante el diagrama de *Collar*, el cual se muestra en la Figura 1. En ella se observa cómo la aeroelasticidad es la ciencia que estudia la interacción entre las fuerzas de inercia, las aerodinámicas y las elásticas. Esta interacción ocurre porque las estructuras reales no son completamente rígidas, sino que se deforman en presencia de acciones externas. En el caso de la elasticidad, el objetivo es obtener las deformaciones producidas en un cuerpo debido a acciones externas conocidas. En cuanto a las fuerzas de inercia, éstas son objeto de estudio en la dinámica de cuerpos, es decir, en el estudio del movimiento debido a acciones externas también conocidas. Por otro lado, la aerodinámica se encarga del estudio de las fuerzas que aparecen en cuerpos sólidos cuando existe movimiento relativo entre el fluido y el propio cuerpo.

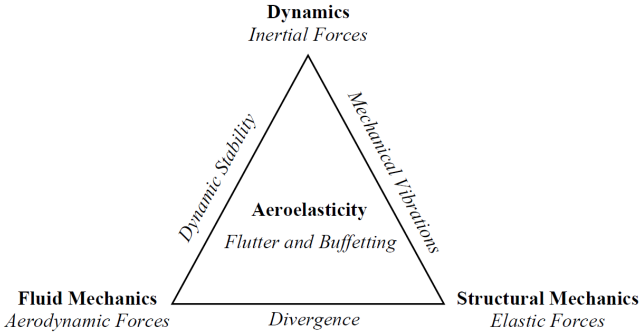


Figura 1: Triángulo de *Collar* [1]

Así, uniendo 2 a 2 estas disciplinas surgen nuevos campos de estudio:

- Vibraciones: interacción entre la elasticidad (deformaciones estructurales) y el movimiento de cuerpos (dinámica)
- Mecánica de vuelo: relación entre la aerodinámica y el movimiento de cuerpos sólidos
- Aeroelasticidad estática: interacción entre la mecánica de fluidos y la estructural

Finalmente, en el centro del triángulo y relacionando todas las disciplinas mencionadas anteriormente, surge la aeroelasticidad dinámica.

La importancia de esta disciplina reside en la propia interacción comentada anteriormente debido a la flexibilidad de los cuerpos reales ya que, cuando estos están inmersos en un fluido, las deformaciones estructurales inducen cambios en las fuerzas aerodinámicas, estas fuerzas adicionales generan un incremento de las deformaciones que, a su vez, provocan mayores fuerzas aerodinámicas. Este acoplamiento puede llegar a un estado de equilibrio provocando que se reduzca la interacción o, por el contrario, pueden dar lugar a inestabilidades que originan distintos fenómenos, cuya naturaleza genera la división de la aeroelasticidad en 2 grandes campos de estudio mencionados previamente: la estática, en la que las propiedades de masa no son significativas, y la dinámica, que engloba la interacción de todas las fuerzas (inercia, elásticas y aerodinámicas). De esta forma, en la Figura 2 se muestran los fenómenos más importantes derivados de esta problemática.

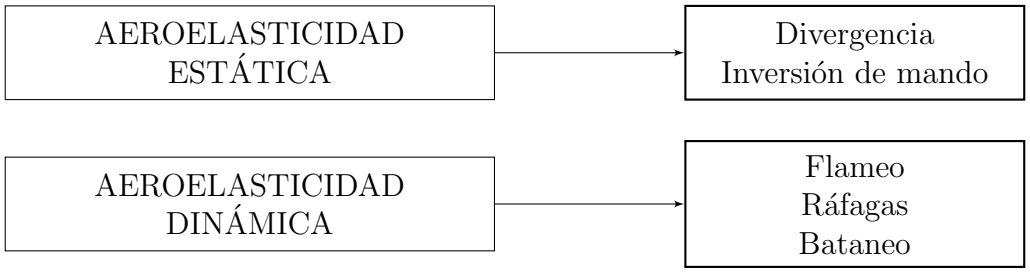


Figura 2: Clasificación de los principales fenómenos aeroelásticos en función de la naturaleza de su origen

En primer lugar, el fenómeno de inestabilidad más importante que se deriva de la aeroelasticidad estática es la *divergencia* de la estructura. Esta se produce cuando no se alcanza equilibrio entre las deformaciones estructurales y las fuerzas aerodinámicas generadas debidas a la velocidad de vuelo, por lo que la rigidez

de la estructura no es capaz de soportar las cargas aerodinámicas y se llega a la rotura. Este fenómeno es totalmente catastrófico para una aeronave y por este motivo es necesario conocer la velocidad de divergencia para no llegar a ese punto en la operación de la aeronave. Por otro lado, puede darse el caso de que, debido a la deformación estructural de la superficie de sustentación principal, la respuesta de la aeronave al deflectar las superficies de control no sea la esperada o, incluso puede darse el caso de que se obtenga la respuesta contraria. Esto se conoce como *inversión de mando*.

Finalmente, entre los fenómenos aeroelásticos dinámicos destaca el *flameo*, que es una vibración autoinducida y no amortiguada que da lugar a una deformación oscilatoria y continua de la estructura. Tras este punto, la aeronave es inestable y las deformaciones crecen hasta la rotura de la estructura. Un ejemplo de este fenómeno es la catástrofe del puente de *Tacoma*. Otros fenómenos objetos de estudio son el comportamiento frente a ráfagas, fenómeno totalmente no estacionario, y el *bataneo*, inestabilidad de alta frecuencia producida por la separación del flujo inducida por ondas de choque, motivo por el cual no aparece en aeronaves subsónicas y queda fuera del alcance del proyecto.

Por todos estos motivos, el estudio de esta rama de la mecánica es esencial en el ámbito de la aeronáutica ya que permite relacionar la las limitaciones aerodinámicas con las estructurales y definir así la operación segura de la aeronave dentro de su envolvente de vuelo.

1.1.2. Estado del arte y motivación

Una vez contextualizado el trabajo en su campo de investigación, es importante estudiar cómo se ha abarcado el problema a lo largo del tiempo. Para ello, hay que separar el modelado aerodinámico del estructural.

Modelado aerodinámico

En primer lugar, en cuanto a aerodinámica se refiere, el reto siempre ha sido el de disponer de un modelo capaz de calcular las fuerzas aerodinámicas generadas en una superficie sustentadora en el seno de un fluido (normalmente aire), es decir, conocidas las condiciones de vuelo, obtener la distribución de presiones y, consecuentemente, la sustentación del ala. Así, los modelos aerodinámicos se pueden dividir en estacionarios y no estacionarios (en función de si se tiene en cuenta la variable temporal para el cálculo de fenómenos transitorios, como puede ser el efecto de la estela), o según la dimensión del problema: 2D o 3D.

El modelado en dos dimensiones permite obtener la distribución de presiones, la sustentación global y el momento aerodinámico de un perfil alar a partir de su discretización en una serie de paneles y aplicando las ecuaciones de flujo potencial en 2D ([2] [3]). Por otro lado, entre los modelos 3D, en el ámbito académico, destacan 2 de ellos:

- Teoría de la Línea de Sustentación de *Prandtl* [4] [5]: modela el ala 3D como una sucesión de perfiles alares suponiendo un torbellino en cada uno de ellos (ver Figura 3). Considerando las ecuaciones de flujo potencial, se puede obtener la intensidad de cada torbellino y, consecuentemente, la distribución de presiones y sustentación a lo largo de la línea media del ala. Sin embargo, este modelo está limitado a alas planas, sin flecha y con gran alargamiento y, aunque sí tiene en cuenta la ley de cuerdas del ala, no determina la distribución de presiones sobre la superficie.

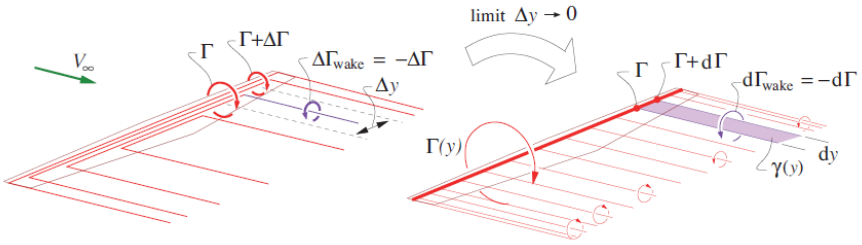


Figura 3: Representación del modelo de *Prandtl* [5]

- *Vortex Lattice Method*: modela las superficies sustentadoras mediante una malla finita de paneles en las 2 direcciones del ala (a lo largo de la cuerda y de la envergadura) suponiendo un vórtice en herradura en el centro aerodinámico de cada panel (ver Figura 4). En este caso sí se consigue obtener la distribución de sustentación en toda la superficie del ala, aunque, al igual que el modelo de *Prandtl*, utiliza las ecuaciones de flujo potencial (no viscoso, incompresible e irrotacional) por lo que no es capaz de modelar fenómenos viscosos o turbulentos.

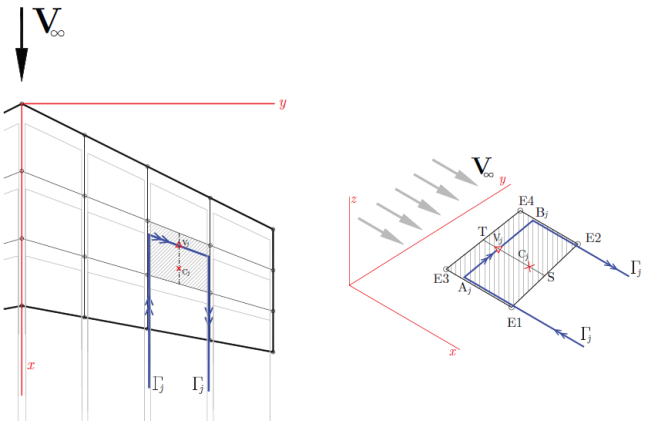


Figura 4: Representación del modelo *VLM* [6]

Comparando ambos modelos aerodinámicos, el modelo de *Prandtl* tiene importancia desde el punto de vista histórico y académico pero su uso se queda fuera de los objetivos del presente proyecto ya que se requiere obtener la distribución de presiones en toda la superficie del ala. Así, en el siguiente punto de la memoria se desarrollará en profundidad el *Vortex Lattice Method* y su implementación en *Matlab*.

Modelado estructural

Por otro lado, si se habla de análisis estructural es imprescindible mencionar el análisis modal, el cual tiene como objetivo la estimación de propiedades dinámicas de las estructuras. Este puede realizarse de forma teórica, como es el caso del presente proyecto, o experimental, a partir de ensayos de vibraciones. Este tipo de análisis es muy utilizado en la industria, sobre todo en la etapa de diseño, ya que permite obtener, entre otros, los modos propios y frecuencias naturales de vibración y ver cómo varían en función de la rigidez o la distribución de masa, por lo que se puede evitar con esto el rango de frecuencias de operación para que no se produzcan resonancias. También puede ser útil para comparar estados estructurales en función del desgaste con el tiempo comparando los modos de vibración de una estructura y otra, aunque para este caso sí son necesarias las técnicas experimentales.

Así, para desarrollar un modelo estructural de forma teórica, es necesario acudir a la teoría de vigas con el objetivo de modelar la estructura del ala mediante un emparrillado de elementos viga unidimensionales, obteniéndose un modelo de elementos finitos capaz de reproducir las propiedades elásticas y másicas de la estructura real. Para el modelado de estos elementos, se tienen en cuenta principalmente 2 modelos: el de *Euler-Bernoulli* y el de *Timoshenko*.

El primero de ellos establece que la deformación de una viga sometida a carga distribuida es proporcional a la cuarta potencia de su longitud e inversamente proporcional al módulo de elasticidad y al momento de inercia de la sección transversal. Además, considera que las tensiones son proporcionales a la distancia al eje neutro de la sección transversal y que la deformación es uniforme en cualquier sección transversal, es decir, al curvarse el material debido a la deformación, las secciones perpendiculares a la línea neutra permanecen perpendiculares [7]. Por otro lado, *Timoshenko* añade el efecto del cortante al modelo anterior ya que, aunque considera que las secciones perpendiculares a la viga tras la deformación siguen siendo planas, no seguirán siendo necesariamente perpendiculares [7], aspecto representado en la Figura 5.

Por último, en cuanto a la implementación en el *software* de cálculo, para cumplir con los objetivos del trabajo, es suficiente con considerar el modelo de *Euler-Bernoulli* ya que la introducción del de *Timoshenko* no supondría una mejora sustanciable en los resultados comparada con la complejidad añadida de su programación. De esta forma, al igual que con el modelo aerodinámico escogido, en la siguiente sección se presentará de forma detallada el modelo estructural, así como su implementación numérica en *Matlab* con el fin de calcular las matrices de masa y rigidez.

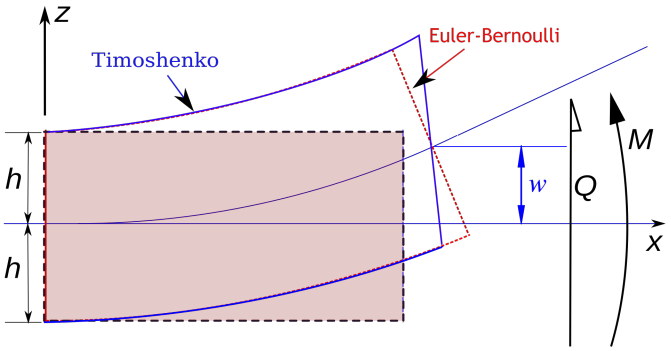


Figura 5: Comparativa gráfica entre el modelo de viga de *Euler-Bernoulli* y *Timoshenko* [8]

Acoplamiento aeroelástico

Finalmente, para el estudio aeroelástico es necesario acoplar tanto estructura como aerodinámica. Típicamente se hace uso del concepto de *eje elástico*, el cual modela la estructura del ala como 2 vigas (una para cada semiala, Figura 6) donde se concentra su rigidez y cuyos grados de libertad son el giro a torsión y

el desplazamiento vertical, mientras que la masa se distribuye entre la malla de paneles de forma puntual. Así, a partir de la deformación de este eje, se obtiene el desplazamiento de los paneles del modelo aerodinámico. Lo que se busca en este proyecto es desarrollar un modelo estructural más complejo, como se ha descrito previamente, formado por un emparrillado de elementos viga unidimensionales con 3 grados de libertad por nodo: giro en dirección longitudinal (a lo largo de la envergadura) y transversal (a lo largo de la cuerda), y desplazamiento vertical, en los que se definirán las propiedades de rigidez según su sección transversal. De esta forma, para acoplar la deformación de este modelo con los paneles del modelo aerodinámico, es necesario la introducción de una interpolación de superficie o *splines*, basada en la teoría de placas [9], desarrollado en la siguiente sección junto con los modelos estructural y aerodinámico.

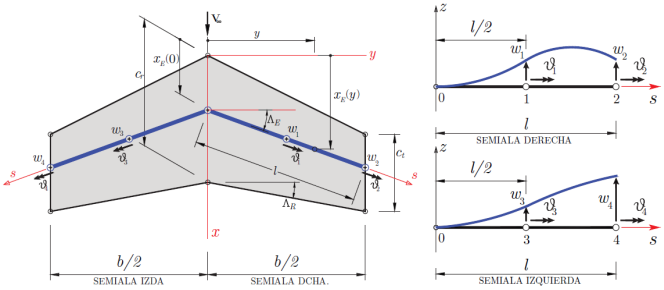


Figura 6: Modelización estructural mediante eje elástico [10]

Como resumen, la motivación de este proyecto reside en implementar un modelo de elementos finitos en plano horizontal de la estructura de la aeronave y acoplarlo con el modelo aerodinámico de la malla de torbellinos, teniendo así un modelo completo en el que no se tenga que suponer un eje elástico donde se concentre la rigidez de la estructura y, por tanto, permita un estudio más realista del comportamiento de la aeronave. Con esto, se pretende abarcar con el mismo modelo, el problema de vibraciones, aeroelasticidad estática y dinámica del avión completo, obteniendo así los modos de vibración y frecuencias naturales, la velocidad de divergencia y el punto de flameo de la aeronave, en primera aproximación.

1.1.3. Relación con los *ODS* (Objetivos de Desarrollo Sostenible)

Para finalizar el marco teórico del trabajo, es muy importante contextualizarlo dentro de los objetivos a corto y largo plazo dentro del ámbito aeronáutico a nivel mundial. Así, cualquier proyecto actual debe incidir sobre alguno de los 17 Objetivos de Desarrollo Sostenible definidos por Naciones Unidas con el propósito de lograr un futuro mejor y más sostenible [11].

Figura 7: Objetivos de Desarrollo Sostenible [11]

De esta forma, de entre los 17 objetivos mostrados en la Figura 7, el trabajo abarca aspectos de 5 de ellos[1]:

- 7. Energía asequible y no contaminante
- 9. Industria, innovación e infraestructura
- 11. Ciudades y comunidades sostenibles
- 13. Acción por el clima
- 15. Vida de ecosistemas terrestres

Como se ha introducido previamente, la aeroelasticidad estudia la interacción entre la estructura de una aeronave y las fuerzas que actúan sobre ella, por lo que comprender, controlar y predecir estos fenómenos permite optimizar diseños y reducir el consumo de combustible, lo que conduce a una menor contaminación y una mayor sostenibilidad en el uso de energía. Además, por la propia reducción de emisiones, se constribuye a la sostenibilidad del transporte aéreo, teniendo un impacto positivo en las comunidades mediante la reducción de la contaminación del aire y el ruido causado por el tráfico aéreo, mejorando así la calidad de vida de las personas que viven cerca de los aeropuertos y rutas aéreas. En esta misma línea de reducción de emisiones contaminantes, la mayor eficiencia de las aeronaves ayuda a mitigar el cambio climático reduciendo la huella de carbono de la industria aeroespacial así como, de forma indirecta, se reduce el impacto negativo en los ecosistemas cercanos a los aeropuertos y en las áreas que sobre vuelan los

[1]En el anexo A, se adjunta una tabla resumen con el grado de relación del trabajo con los *ODS*.

aviones, preservando de esta forma la biodiversidad y la vida silvestre que habita estas zonas. Por último, el proyecto implica el desarrollo y la aplicación de técnicas de modelado y simulación avanzadas con lo que se fomenta la innovación en el campo de la ingeniería aeroespacial y contribuye al desarrollo de infraestructuras y tecnologías más eficientes. Además, puede llegar a impulsar la industria aeronáutica generando empleo y promoviendo el crecimiento económico sostenible.

En resumen, el estudio teórico aeroelástico de un avión a partir del modelado y simulación puede influir positivamente en los *ODS* relacionados con la energía, la innovación, las comunidades sostenibles, la acción climática y, de forma indirecta, con la conservación de los ecosistemas. Al mejorar la eficiencia y reducir las emisiones de los aviones, se promueve un transporte aéreo más sostenible y se contribuye a la mitigación del cambio climático y a la protección del medio ambiente.

1.2. Objetivos

El principal objetivo del trabajo es el desarrollo de un modelo de elementos finitos en plano horizontal que permita definir la estructura de una aeronave completa, formada por el fuselaje, ala principal y estabilizador horizontal, así como su acoplamiento con el modelo aerodinámico de paneles *Vortex Lattice Method* utilizado para modelar el comportamiento aerodinámico de las superficies sustentadoras de la aeronave (ala y estabilizador). De esta forma, el objetivo del proyecto se puede dividir en 3 grandes bloques:

- Desarrollo del modelo estructural
- Acoplamiento con el modelo aerodinámico
- Aplicación: modelización de una aeronave completa

Para alcanzar este objetivo se empleará la metodología expuesta en el siguiente apartado pero es necesario definir algunos objetivos secundarios para abordar el problema. En primer lugar, es necesario conocer los fundamentos de la aerodinámica, tanto estacionaria como no estacionaria, para comprender el modelo aerodinámico propuesto y poder implementarlo en *Matlab*. En segundo lugar, para el desarrollo del modelo estructural de elementos finitos, es imprescindible revisar los principios del análisis modal así como entender la teoría de estructuras con el objetivo de seleccionar los elementos adecuados para la modelización de la estructura y que el modelo presentado sea lo más realista posible. Además, al igual que con el modelo aerodinámico, es vital entender el modelo utilizado desde el punto de vista numérico para su implantación en *Matlab*.

De esta forma, el siguiente objetivo es realizar el acoplamiento de los 2 modelos para poder obtener resultados en términos de aeroelasticidad. Para ello, es necesario estudiar los fundamentos de la aeroelasticidad y revisar los distintos modelos existentes de acoplamiento.

Con todo ello, se pretende obtener un modelo global (estructura-aerodinámica) de una aeronave completa que modelice su comportamiento estático y dinámico de una manera realista, así como determinar la influencia que tienen cada uno de los modelos en el modelo global de la aeronave.

1.3. Metodología

Una vez especificado el tema del trabajo así como el marco teórico del mismo, es necesario definir la correcta metodología que permita alcanzar los objetivos descritos en el apartado anterior, la cual se muestra de forma esquematizada en la Figura 8.

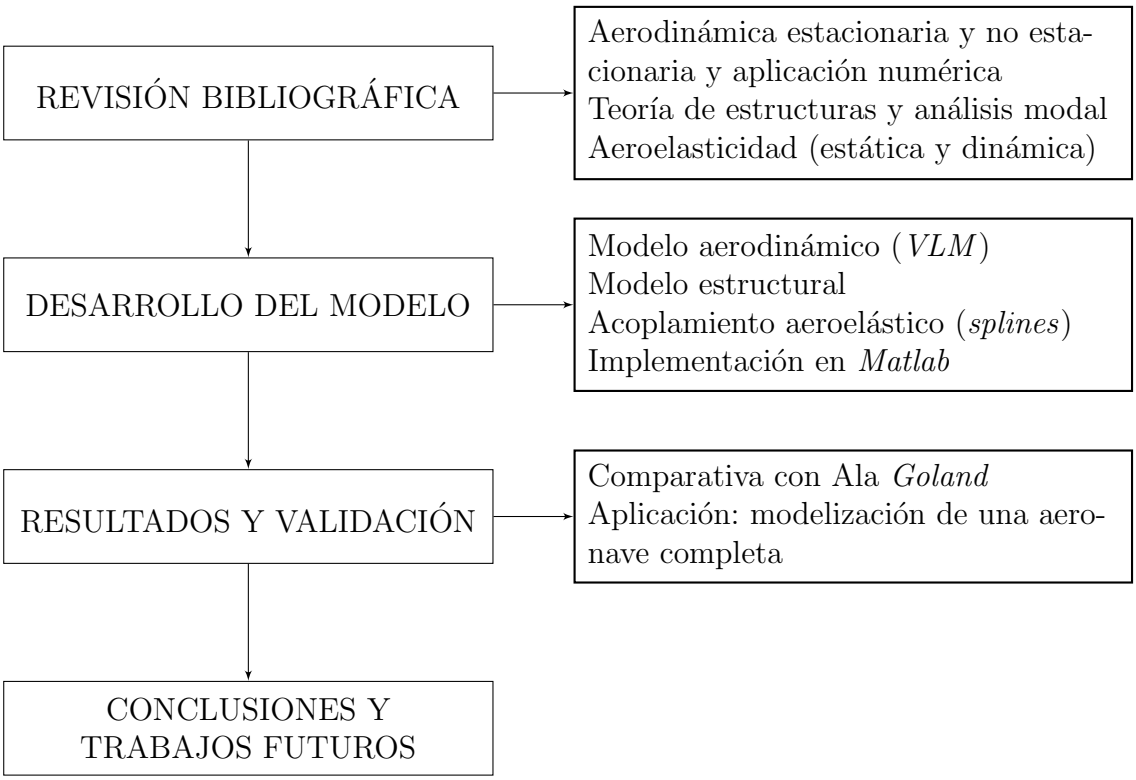


Figura 8: Metodología del trabajo

En primer lugar y como se ha mencionado en los objetivos secundarios, es necesario conocer los fundamentos de la aerodinámica y la teoría de estructuras así como su aplicación numérica para desarrollar el modelo global objetivo del proyecto. Es por ello por lo que el primer paso del trabajo es realizar una revisión bibliográfica pertinente para comprender estos aspectos y poder implementarlos en el programa de simulación, en este caso, *Matlab*.

El siguiente paso, tal como se ha comentado, es el desarrollo numérico del modelo en *Matlab*. Para ello, se divide la programación en los 2 sub-modelos, estructura y aerodinámica, para, posteriormente, unificarlos en el modelo global a partir de su acoplamiento mediante la interpolación por *splines*.

Por último, se pretende validar el modelo con el objetivo de verificar que su uso permite modelizar correctamente fenómenos reales aeroelásticos, por lo que se valida con los resultados del ala *Goland*, la cual está muy contrastada por diversos estudios ([12] [13] [14] [15] [16]) y se puede tomar como punto de partida. Así, se pasa a la implementación para una aeronave completa tomando la geometría de una real y obteniendo resultados en cuanto a vibraciones libres, divergencia y flameo.

Finalmente, a partir de los resultados obtenidos en el punto anterior, se puede verificar el cumplimiento o no de los objetivos propuestos así como definir una serie de trabajos futuros para mejorar y/o ampliar el modelo actual desarrollado.

2. Fundamentos teóricos

2.1. Modelo aerodinámico

Para explicar el modelo aerodinámico utilizado en el proyecto es necesario partir de la teoría de flujo potencial, la cual asume flujo irrotacional e incompresible, en el que el campo de velocidades del fluido deriva de una función potencial [4] [2]:

$$\mathbf{V}(x, y, x, t) = \nabla\phi(x, y, z, t) \tag{1}$$

donde $\phi(x, y, z, t)$ representa el potencial no-estacionario y $\nabla(\cdot)$ es el operador gradiente. Para el presente trabajo, se aplicará un modelo casi-estacionario, por lo que la dependencia con el tiempo se elimina y el potencial solo dependerá de las 3 coordenadas espaciales. Aplicando las ecuaciones fundamentales de la mecánica de fluidos se puede demostrar que el potencial estacionario verifica la ecuación de *Laplace*[4] [6]:

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} \tag{2}$$

En este punto cabe destacar que el objetivo del modelo no es el de calcular el potencial en cada punto, ya que no interesan las líneas de corriente, sino las fuerzas generadas en el ala. Por este motivo, para la obtención de la distribución de presiones y, consecuentemente, la distribución de fuerzas, es necesario el cálculo del campo de velocidades y la aplicación de las condiciones de contorno, que no es más que imponer que el flujo se mantenga tangente al ala.

Para el cálculo del campo de velocidades, se discretiza la superficie media del ala en paneles, obteniendo cuadriláteros en los que se definirán 2 puntos característicos (punto de control y de vórtice). En el punto de vórtice se ubica un filamento de torbellino en herradura[3], mientras que en el punto de control se impone la condición de contorno. Con ello, se obtiene el sistema de ecuaciones que permite obtener la intensidad de cada torbellino, la cual está relacionada con la sustentación a través del teorema de *Kutta-Joukowski* en su versión vectorial.

Así, el primer paso es describir el método para calcular el campo de velocidades de un filamento de torbellino en herradura para finalmente plantear la discretización geométrica de la superficie y resolver el sistema de ecuaciones, obteniendo, de esta forma, la distribución de fuerzas sobre el ala.

[2] A partir de este punto, en todos los desarrollos posteriores, se utilizarán las variables en negrita para definir vectores y matrices.

[3] Esta elección es debida a que éstos verifican la ecuación de *Laplace*.

2.1.1. Ley de *Biot-Savart*

Como se ha descrito previamente, el primer paso es calcular el campo de velocidades debido a un filamento de torbellino en herradura. Para ello, se hace uso de la ley de *Biot-Savart* ya que, aunque se definió con el objetivo de obtener el campo magnético debido a corrientes eléctricas estacionarias, si se aplica a aerodinámica 3D, por analogía[4], es posible calcular el campo de velocidades debido a un filamento en el cual está definido un torbellino de intensidad Γ.

Para la aplicación en cuestión, se desea obtener la velocidad debida a un filamento recto de longitud semi-infinita ya que la herradura se compone de 3 segmentos: 2 semi-infinitos y 1 finito, el cual se puede calcular como 2 semi-infinitos. En la Figura 9 se muestra el esquema comentado:

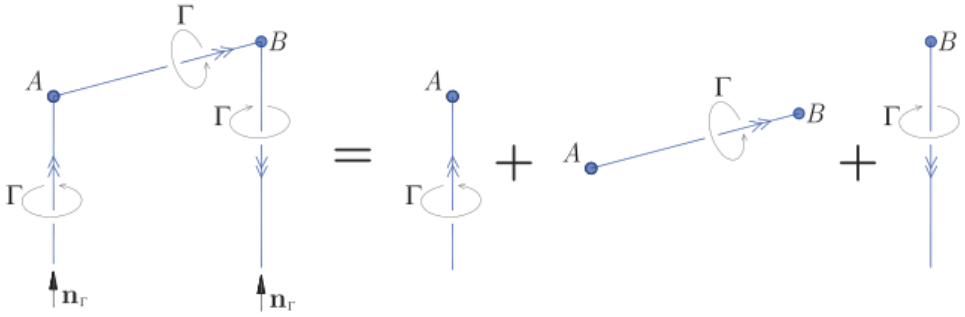


Figura 9: Torbellino en herradura [6]

Por tanto, atendiendo a la analogía con la ley de *Biot-Savart*, la velocidad inducida en un punto genérico P debida a este tipo de filamento se puede calcular como [4]:

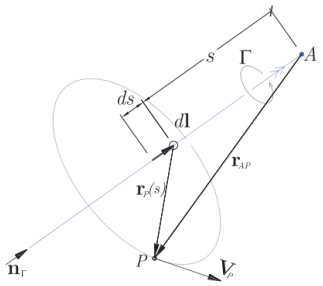


Figura 10: Filamento semi-infinito de intensidad Γ [6]

[4]Esto se debe a que el potencial del que deriva el campo magnético también satisface la ecuación de *Laplace*.

$$\mathbf{V}_{semi}(\mathbf{R}_P, \mathbf{R}_A, \mathbf{n}_\Gamma, \Gamma) = \frac{\Gamma}{4\pi||\mathbf{r}_{AP}||} \cdot \frac{\mathbf{n}_\Gamma \times \mathbf{n}_{AP}}{1 + \mathbf{n}_\Gamma \cdot \mathbf{n}_{AP}} \tag{3}$$

donde Γ es la intensidad del torbellino, $\mathbf{r}_{AP}$ es el vector que define la recta entre los puntos A y P, siendo $\mathbf{n}_{AP}$ su vector unitario[5], y $\mathbf{n}_\Gamma$ es el vector unitario que define la dirección de Γ. Finalmente, de esta forma, atendiendo a la Figura 9, la velocidad debida al torbellino en herradura será:

$$\begin{aligned}\mathbf{V}_{herr}(\mathbf{R}_P, \mathbf{R}_A, \mathbf{R}_B, \mathbf{n}_\Gamma, \Gamma) = \mathbf{V}_{semi}(\mathbf{R}_P, \mathbf{R}_A, \mathbf{n}_\Gamma, \Gamma) + \mathbf{V}_{semi}(\mathbf{R}_P, \mathbf{R}_B, \mathbf{n}_{AB}, \Gamma) \\ + \mathbf{V}_{semi}(\mathbf{R}_P, \mathbf{R}_A, \mathbf{n}_{AB}, -\Gamma) + \mathbf{V}_{semi}(\mathbf{R}_P, \mathbf{R}_A, \mathbf{n}_\Gamma, -\Gamma)\end{aligned} \tag{4}$$

2.1.2. Modelo numérico de la malla de torbellinos

Una vez obtenida la expresión de la velocidad inducida en un punto cualquiera del espacio debida a un filamento de torbellino en herradura, el siguiente paso es la definición de la malla de paneles que modela la superficie de sustentación y la aplicación de la condición de contorno de flujo tangencial que derivará en el sistema de ecuaciones lineal con el que se obtiene el vector de intensidades de cada torbellino.

Así, como se ha descrito en la introducción de la sección 2.1, la modelización de la superficie se realiza mediante singularidades en forma de vórtices 3D o filamentos de torbellino en forma de herradura por lo que, en primer lugar, es necesario conocer los *Teoremas de Vorticidad*:

1. La circulación Γ se mantiene constante a lo largo de una línea de vórtice.
2. Los filamentos de torbellino no pueden acabar o empezar de forma brusca en un fluido sino que deben ser cerrados, prolongarse hasta el infinito o terminar en un contorno sólido. Además, la circulación en cualquier sección del filamento es igual a la intensidad del vórtice.
3. Si el fluido inicialmente es irrotacional y no-viscoso, se mantiene irrotacional.

De estos 3 teoremas se deriva la utilidad que tienen los torbellinos para representar superficies de sustentación ya que la velocidad normal a la malla es continua y, además, permiten modelar variaciones en las velocidades tangenciales y, consecuentemente, fuerzas o presiones.

[5]El vector unitario se calcula como: $\mathbf{n}_{AP} = \frac{\mathbf{r}_{AP}}{||\mathbf{r}_{AP}||} = \frac{\mathbf{R}_P - \mathbf{R}_A}{||\mathbf{R}_P - \mathbf{R}_A||}$.

Siguiendo la implementación clásica, los torbellinos se sitúan en a 1/4 del borde de ataque de cada panel en dirección de la cuerda (dirección X en este caso) y en el centro del mismo en dirección de la envergadura, es decir, Y, mientras que el punto de control también se sitúa en el centro en dirección Y pero a 3/4 en X. Este punto es necesario ya que el sistema numérico lineal se contruirá de forma que se satisfaga la condición de velocidad tangencial en dicho punto.

De esta forma, el primer paso es definir la geometría de las superficies de sustentación en plano horizontal xOy. Tras esto, se divide la superficie en cuadriláteros y se determina el segmento acotado de la herradura A_jB_j a 1/4 del frente del panel y el punto de control como se ha mencionado anteriormente. Por otro lado, los segmentos semi-infinitos de los torbellinos serán siempre paralelos a la velocidad del fluido sin perturbar $\mathbf{V}_\infty = U_\infty \mathbf{i}$, es decir, llevarán la dirección $+x$. Así, cada panel tendrá asociado un torbellino en herradura de intensidad Γ_j con las características descritas.

En la Figura 11 se muestra una representación de la discretización de un ala genérica con la ubicación del torbellino en herradura y los puntos característicos de un panel.

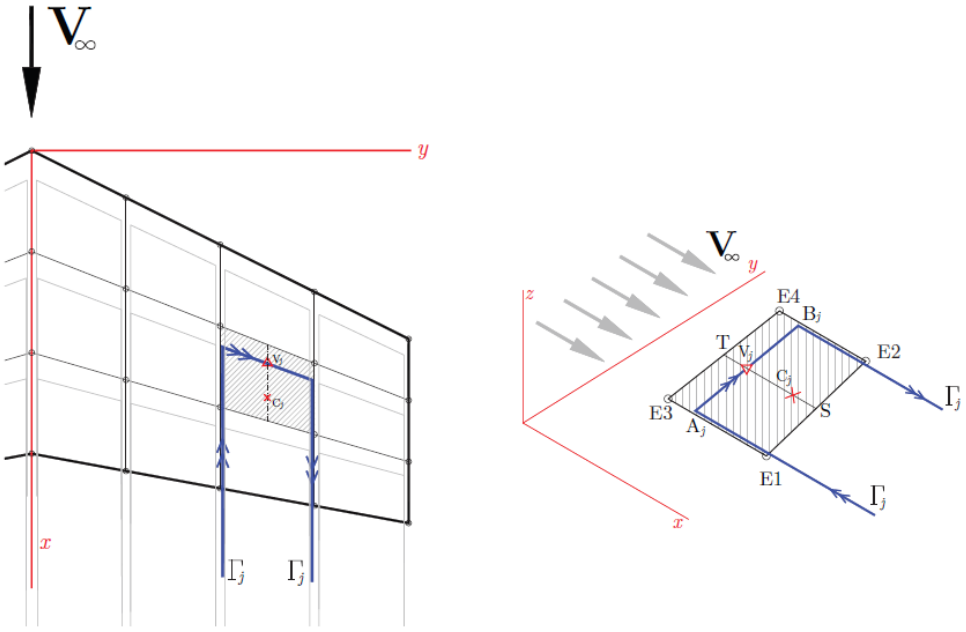


Figura 11: Representación de la construcción de la malla de torbellinos sobre un ala en flecha genérica [6]

Por tanto, para la implementación numérica en *Matlab* es importante definir estos puntos de una forma fácil de programar. Atendiendo a la figura anterior, primero se calculan los vectores de posición de los puntos A_j y B_j ya que, como se ha visto en la sección 2.1.1, son necesarios para el cálculo de la velocidad inducida por el torbellino:

$$\mathbf{R}_{A_j} = \frac{1}{4}\mathbf{R}_{E1} + \frac{3}{4}\mathbf{R}_{E3},\ \mathbf{R}_{B_j} = \frac{1}{4}\mathbf{R}_{E2} + \frac{3}{4}\mathbf{R}_{E4} \tag{5}$$

Seguidamente, para el cálculo del punto donde se ubica el torbellino y el de control se hace uso de los puntos T y S, mostrados también en la Figura 11:

$$\mathbf{R}_S = \frac{1}{2}\mathbf{R}_{E1} + \frac{1}{2}\mathbf{R}_{E2},\ \mathbf{R}_T = \frac{1}{2}\mathbf{R}_{E3} + \frac{1}{2}\mathbf{R}_{E4} \tag{6}$$

$$\mathbf{R}_{V_j} = \frac{1}{4}\mathbf{R}_S + \frac{3}{4}\mathbf{R}_T,\ \mathbf{R}_{C_j} = \frac{3}{4}\mathbf{R}_S + \frac{1}{4}\mathbf{R}_T \tag{7}$$

Finalmente, solo queda aplicar la condición de contorno de flujo tangencial en el punto de control de cada panel para formar el sistema de ecuaciones lineales que permita obtener las intensidades de los torbellinos. Para ello, se define la configuración geométrica del ala mediante la ecuación de superficie $F(x, y, z) = 0$, que puede reescribirse como $F(x, y, z) = z - f(x, y)$ si la superficie media del ala está contenida en el plano horizontal xOy como es el caso de estudio.

Así, para la aplicación de las condiciones de contorno es necesario obtener el vector normal a dicha superficie $\mathbf{n}(x, y, z)$ ya que para imponer que la velocidad total del flujo es tangente a la superficie se establece que el vector velocidad sea normal a este vector. Para su cálculo, se consideran 2 puntos de la superficie infinitamente cercanos: $x' = x+dx$, $y' = y+dy$, $z' = z+dz$ siendo $d\mathbf{t} = (dx, dy, dz)$ un vector tangente a la superficie. Expandiendo en series de *Taylor* la función $F(x', y', z')$ y sabiendo que, como ambos puntos pertenecen a la superficie verifican $F(x, y, z) = F(x', y', z') = 0$, se llega a la ecuación 9, donde $\nabla F(x, y, z)$ es el gradiente de la función $F(x, y, z)$, el cual es normal a la superficie definida por $F(x, y, z)$.

$$F(x', y', z') = F(x + dx, y + dy, z + dz) = F(x, y, z) + \frac{\partial F}{\partial x}dx + \frac{\partial F}{\partial y}dy + \frac{\partial F}{\partial z}dz \tag{8}$$

$$\nabla F(x, y, z) \cdot d\mathbf{t} = 0 \tag{9}$$

Con este resultado ya se puede pasar a la aplicación de las condiciones de contorno y la formulación del sistema de ecuaciones numérico.

En primer lugar, la velocidad total $\mathbf{V}(x,y,z)$ es la suma de la velocidad del flujo libre $\mathbf{V}_\infty = U_\infty \mathbf{i}$, la velocidad relativa del fluido respecto de la aeronave debida a maniobras estacionarias como puede ser el cabeceo, alabeo o guiñada del avión, representada por $-\mathbf{v}_{aeronave}(x,y,z)$, y la velocidad inducida por todos los torbellinos, denotada por $\mathbf{v}_\Gamma(x,y,z)$:

$$\mathbf{V}(x,y,z) = \mathbf{V}_\infty(x,y,z) - \mathbf{v}_{aeronave}(x,y,z) + \mathbf{v}_\Gamma(x,y,z) \tag{10}$$

En este punto ya se puede imponer la condición de flujo tangente a la superficie en cada panel j, esto es, la proyección de la velocidad total sobre la normal de la superficie debe ser nula, por lo que, en los puntos de control $\mathbf{x}_{C_j} = (x_{C_j}, y_{C_j}, z_{C_j})$ debe verificarse que el producto escalar entre la velocidad total y el gradiente de la función de superficie debe ser nulo:

$$\mathbf{V}(\mathbf{x}_{C_j}) \cdot \nabla F(\mathbf{x}_{C_j}) = \left[\mathbf{V}_\infty(\mathbf{x}_{C_j}) - \mathbf{v}_{aeronave}(\mathbf{x}_{C_j}) + \mathbf{v}_\Gamma(\mathbf{x}_{C_j})\right] \cdot \nabla F(\mathbf{x}_{C_j}) = 0, \quad 1 \leq j \leq N \tag{11}$$

siendo $\nabla F(\mathbf{x}_{C_j}) = \left[\frac{\partial F}{\partial x}\mathbf{i} + \frac{\partial F}{\partial y}\mathbf{j} + \frac{\partial F}{\partial z}\mathbf{k}\right]$ y N el número de paneles total. Por tanto, se tienen N ecuaciones para obtener las N intensidades de los torbellinos: $\{\Gamma_j\}_{j=1}^N$.

Aplicando ahora el cálculo de la velocidad para un torbellino en herradura, se puede demostrar que el sistema de ecuaciones anterior es lineal en la intensidad del torbellino, lo que facilita su resolución e implementación en *Matlab*. Primeramente, se puede expresar la velocidad inducida por el torbellino en herradura de forma lineal con la intensidad del torbellino Γ:

$$\mathbf{V}_{herr}(\mathbf{R}_P, \mathbf{R}_A, \mathbf{R}_B, \mathbf{n}_\infty, \Gamma) = \mathbf{V}_{herr}(\mathbf{R}_P, \mathbf{R}_A, \mathbf{R}_B, \mathbf{n}_\infty, 1) \cdot \Gamma \tag{12}$$

siendo $\mathbf{n}_\infty$ el vector unitario que lleva la dirección de la estela, paralela al flujo libre. Aplicando este resultado, se puede expresar la velocidad inducida por todos los torbellinos en el punto de control C_j de la siguiente forma:

$$\sum_{k=1}^{N} \mathbf{V}_{herr}(\mathbf{R}_{C_j}, \mathbf{R}_{A_k}, \mathbf{R}_{B_k}, \mathbf{n}_\infty, 1) \cdot \Gamma_k = \sum_{k=1}^{N} \nu_{jk} \cdot \Gamma_k \tag{13}$$

Introduciendo este resultado en la ecuación 11, se obtiene el sistema de ecuaciones global que permite obtener las intensidades de todos los torbellinos de los paneles que discretizan el ala:

$$\begin{aligned}&\left[\mathbf{V}_\infty(\mathbf{x}_{C_j}) - \mathbf{v}_{aeronave}(\mathbf{x}_{C_j}) + \mathbf{v}_\Gamma(\mathbf{x}_{C_j})\right] \cdot \nabla F(\mathbf{x}_{C_j}) = \\ &\quad \sum_{k=1}^{N} \left(\nu_{jk} \cdot \nabla F(\mathbf{x}_{C_j})\right) \cdot \Gamma_k + \left[\mathbf{V}_\infty(\mathbf{x}_{C_j}) - \mathbf{v}_{aeronave}(\mathbf{x}_{C_j})\right] \cdot \nabla F(\mathbf{x}_{C_j}) = \\ &\quad \sum_{k=1}^{N} H_{jk} \cdot \Gamma_k + \left[\mathbf{V}_\infty(\mathbf{x}_{C_j}) - \mathbf{v}_{aeronave}(\mathbf{x}_{C_j})\right] \cdot \nabla F(\mathbf{x}_{C_j}) = 0,\ 1 \leq j \leq N \end{aligned} \quad (14)$$

El último paso es organizar los coeficientes H_{jk} en forma de matriz cuadrada (pero no simétrica), donde cada elemento es el efecto del torbellino Γ_k en el panel j, cuya dimensión es $[L^{-1}]$. A esta matriz se la denomina *Matriz de coeficientes de influencia aerodinámicos*[6]. y sus elementos únicamente dependen de la geometría de la malla de paneles que modeliza el ala. Así, se puede expresar el sistema de ecuaciones en forma matricial (15), donde **a** representa el ángulo con el que cada panel ve incidir el flujo, es decir, a_j es el ángulo de ataque de cada panel.

$$\mathbf{H}\ \mathbf{\Gamma} = U_\infty\ \mathbf{a} \quad (15)$$

$$a_j = -\left[\frac{\mathbf{V}_\infty(\mathbf{x}_{C_j}) - \mathbf{v}_{aeronave}(\mathbf{x}_{C_j})}{U_\infty}\right] \cdot \nabla F(\mathbf{x}_{C_j}) \quad (16)$$

De esta forma, es posible programar el desarrollo descrito de la malla de torbellinos en *Matlab*. Este proceso se detalla en el anexo B.2, donde se adjuntan todas las funciones creadas para formar el modelo global aerodinámico. Además, se detallará su uso en la sección 3.2.2.

2.1.3. Distribución de presiones, distribución de sustentación y sustentación total

Por último, para acabar con el modelo aerodinámico, se describe el cálculo de la distribución de presiones, de sustentación y la sustentación total a partir de las intensidades de los torbellinos $\mathbf{\Gamma}$ obtenidas del sistema de ecuaciones lineal desarrollado en la sección anterior, ya que es necesario para el desarrollo del problema aeroelástico (2.4). En primer lugar, a partir del teorema de *Kutta-Joukovski* en su versión vectorial se puede calcular la magnitud y dirección de la fuerza de sustentación generada en un panel cualquiera j a partir de la ecuación 17, siendo $\mathbf{\Gamma}_j = \Gamma_j \mathbf{n}_{AB_j}$.

$$\mathbf{F}_j = \rho_\infty \mathbf{V}_\infty \times \mathbf{\Gamma}_j \quad (17)$$

[6] *AIC* en sus siglas en inglés

A la vista de la expresión, según el modelo, solo aparecen fuerzas en el panel a lo largo del segmento AB por lo que la fuerza calculada es por unidad de longitud $[FL^{-1}]$, es decir, se calculan fuerzas distribuidas a lo largo del segmento A_jB_j que serán perpendiculares al plano formado por los vectores $\mathbf{n}_{AB_j}$ y $\mathbf{V}_\infty$. En el caso particular de que las superficies de cálculo estén contenidas en el plano xOy, el producto vectorial anterior deriva en una fuerza en dirección $+z$, tal y como cabría esperar. Dado que este caso es el de estudio, ya que se quiere modelar el ala principal y el estabilizador horizontal, el desarrollo del cálculo de fuerzas se realiza para el mismo:

$$\mathbf{n}_{AB_j} = \sin\lambda_j\mathbf{i}+\cos\lambda_j\mathbf{j} \rightarrow \mathbf{F}_j = \rho_\infty\left(U_\infty\mathbf{i}\right)\times\Gamma_j\left(\sin\lambda_j\mathbf{i}+\cos\lambda_j\mathbf{j}\right) = \rho_\infty U_\infty\Gamma_j\cos\lambda_j\mathbf{k} \tag{18}$$

Para el cálculo de la fuerza de sustentación total en cada panel L_j, solo es necesario multiplicar la magnitud de la fuerza calculada en 18 F_j por la longitud del segmento AB del panel j:

$$L_j = F_j\Delta s_j = \rho_\infty U_\infty\Gamma_j\cos\lambda_j\Delta s_j = \rho_\infty U_\infty\Gamma_j\Delta y \tag{19}$$

siendo Δy la proyección de la longitud del segmento AB_j a lo largo de la envergadura del ala.

Obtenida la sustentación por panel, es trivial el cálculo de la distribución de presiones (20) y la sustentación total del ala (21):

$$\Delta p_j = \frac{L_j}{S_j} = \frac{\rho_\infty U_\infty\Gamma_j\Delta y}{\Delta x_j\Delta y} = \frac{\rho_\infty U_\infty\Gamma_j}{\Delta x_j} \rightarrow \Delta c_{p_j} = \frac{\Delta p_j}{\frac{1}{2}\rho_\infty U_\infty^2} \tag{20}$$

$$L = \sum_{j=1}^{N} L_j \rightarrow C_L = \frac{L}{\frac{1}{2}\rho_\infty U_\infty^2 S_w} \tag{21}$$

Finalmente, mencionar que también se puede calcular la sustentación a lo largo de la envergadura del ala sumando la contribución de sustentación de cada panel que se encuentra en la misma posición y, tal y como se esquematiza en la Figura 12.

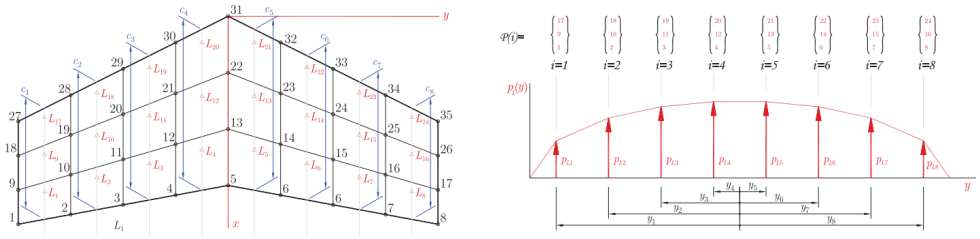


Figura 12: Cálculo de la distribución de sustentación en el ala [6]

2.2. Modelo estructural

2.2.1. Modelización numérica de la estructura y matriz de rigidez

El modelo estructural desarrollado en el proyecto consiste en un emparrillado de elementos finitos en plano horizontal de tipo viga (1D), cuyo objetivo es el de modelar la estructura del ala principal, estabilizador horizontal y fuselaje de la aeronave.

En primer lugar, es necesario introducir la tipología de viga a modelar. Para ello, acudiendo a la teoría de vigas y como se ha mencionado en la introducción, los elementos utilizados serán tipo *Euler-Bernoulli*, denominación que hace referencia a los autores *Leonhard Euler* y *Daniel Bernoulli* que desarrollaron esta teoría en el siglo XVIII. En ella, se considera un sistema de coordenadas en el que el eje X es tangente al eje baricéntrico de la viga y los ejes Y y Z coinciden con los ejes principales de inercia. Así, las principales hipótesis de este modelo para la flexión simple en plano XY son [7]:

- El material del elemento es homogéneo, isotrópico y tiene comportamiento elástico lineal (sigue la ley de *Hooke*). El coeficiente de *Poisson* es despreciable.
- El desplazamiento vertical solo depende de X.
- Los puntos que pertenecen a la fibra neutra solo tienen desplazamiento vertical y giro.
- La tensión perpendicular a la fibra neutra es nula.
- Las secciones planas perpendiculares al eje de la viga, permanecen perpendiculares una vez deformada la viga.

El último de los puntos anteriores es la hipótesis de *Bernoulli*, la cual diferencia esta teoría de la *Timoshenko*, ya que este considera que, tras la deformación, las superficies planas siguen siendo planas pero no necesariamente perpendiculares al eje de la viga, por lo que la teoría de vigas utilizada es más simplificada y, por tanto, reduce el coste de su programación.

Una vez introducido el elemento unidimensional, se definen los grados de libertad. Cada elemento estará formado por 2 nodos, los cuales pueden desplazarse verticalmente y girar (Figura 13). Así, los elementos estarán sometidos a flexión y torsión, con lo que los grados de libertad de cada nodo serán:

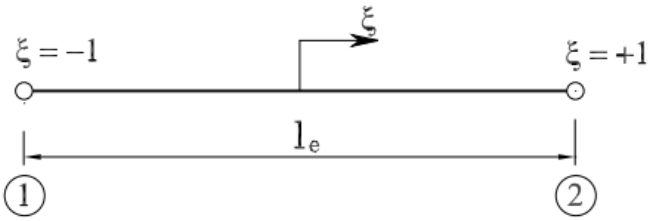


Figura 13: Elemento unidimensional tipo viga

$$\mathbf{u}^e = \{\theta_1,\ w_1,\ \frac{\partial w_1}{\partial y}, \theta_2,\ w_2,\ \frac{\partial w_2}{\partial y}\}^T \tag{22}$$

siendo θ_i el giro de torsión, w_i el desplazamiento vertical y $\frac{\partial w_i}{\partial y}$ el giro debido a la flexión del elemento. Así, ya se pueden obtener las funciones de forma, las cuales definen la deformada en otros puntos distintos a los nodos suponiendo variaciones polinómicas. En este caso, se supone una variación lineal para la torsión del elemento mientras que para la flexión, se utiliza una cúbica, siendo $\xi = \frac{2}{L^e} y$:

$$\theta(\xi) = a_0 + a_1\xi, \quad w(\xi) = b_0 + b_1\xi + b_2\xi^2 + b_3\xi^3 \tag{23}$$

donde a_0, a_1, b_0, b_1, b_2 y b_3 son coeficientes que se obtienen de aplicar las condiciones de contorno en los nodos del elemento (24 y 25), obteniéndose de esta forma la matriz de funciones de forma del elemento (27), necesaria para el cálculo de la matriz de rigidez del mismo.

De esta forma, se aplican las condiciones de contorno en cada nodo:

$$\text{Nodo 1} \ \rightarrow\ \theta(-1) = \theta_1, \quad w(-1) = w_1, \quad \frac{\partial w}{\partial \xi}(-1) = \frac{\partial w_1}{\partial \xi} \tag{24}$$

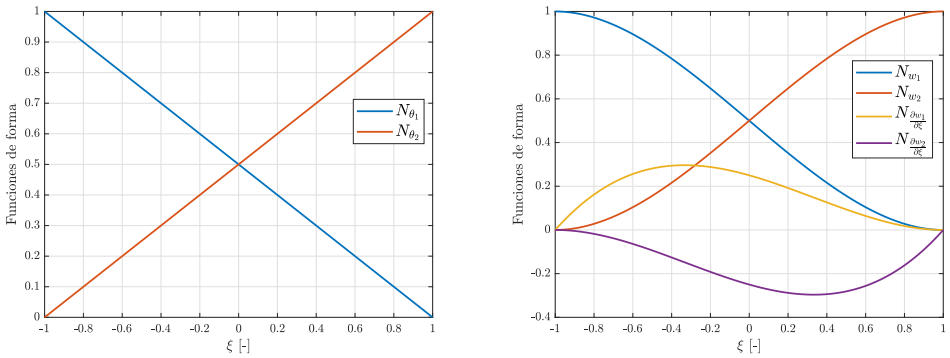
$$\text{Nodo 2} \ \rightarrow\ \theta(+1) = \theta_2, \quad w(+1) = w_2, \quad \frac{\partial w}{\partial \xi}(+1) = \frac{\partial w_2}{\partial \xi} \tag{25}$$

por lo que los polinomios que definen la torsión y flexión del elemento quedarán de la siguiente forma:

$$\theta(\xi) = \frac{1-\xi}{2}\ \theta_1 + \frac{1+\xi}{2}\ \theta_2 \ \rightarrow\ \theta(\xi) = N_{\theta_1}\theta_1 + N_{\theta_2}\theta_2 \tag{26}$$

$$w(\xi) = \tfrac{2-3\xi+\xi^3}{4} w_1 + \tfrac{1-\xi-\xi^2+\xi^3}{4} \tfrac{\partial w_1}{\partial \xi} \tfrac{L^e}{2} + \tfrac{2+3\xi-\xi^3}{4} w_2 + \tfrac{-1-\xi+\xi^2+\xi^3}{4} \tfrac{\partial w_2}{\partial \xi} \tfrac{L^e}{2}$$

$$w(\xi) = N_{w_1} w_1 + N_{\frac{\partial w_1}{\partial \xi}} \frac{\partial w_1}{\partial \xi} \frac{L^e}{2} + N_{w_2} w_2 + N_{\frac{\partial w_2}{\partial \xi}} \frac{\partial w_2}{\partial \xi} \frac{L^e}{2} \tag{27}$$



(a) Torsión (b) Flexión

Figura 14: Representación de las funciones de forma del elemento unidimensional

Finalmente, reescribiendo el resultado en forma matricial, se obtiene la matriz de funciones de forma de elemento, donde L^e es la longitud del elemento:

$$\mathbf{N}^e(\xi) = \begin{bmatrix} N_{\theta_1} & 0 & 0 & N_{\theta_2} & 0 & 0 \\ 0 & N_{w_1} & \frac{L^e}{2} N_{\frac{\partial w_1}{\partial \xi}} & 0 & N_{w_2} & \frac{L^e}{2} N_{\frac{\partial w_2}{\partial \xi}} \\ 0 & \frac{2}{L^e}\frac{\partial N_{w_1}}{\partial \xi} & \frac{\partial N_{\frac{\partial w_1}{\partial \xi}}}{\partial \xi} & 0 & \frac{2}{L^e}\frac{\partial N_{w_2}}{\partial \xi} & \frac{\partial N_{\frac{\partial w_2}{\partial \xi}}}{\partial \xi} \end{bmatrix} \tag{28}$$

Así, de acuerdo a los grados de libertad nodales, la matriz de rigidez se construye a partir de la energía de deformación, la cual se divide en dos componentes, una asociada a la deformación por flexión, y otra a la deformación por torsión:

$$\mathcal{U}^e = \mathcal{U}^e_f + \mathcal{U}^e_t = \frac{1}{2}\int_e EI(y)\left(\frac{d^2w}{dy^2}\right)^2 dy + \frac{1}{2}\int_e GJ(y)\left(\frac{d\theta}{dy}\right)^2 dy \tag{29}$$

siendo E y G los módulos de rigidez a flexión y torsión del material de la viga, e I y J los momentos de inercia a flexión y torsión, los cuales dependen de la sección de cálculo. Al ser un modelo de elementos finitos, tanto I como J se suponen constantes en cada elemento y su cálculo dependerá de la sección del mismo[7].

De esta forma, a la vista de 29, dado que en el vector de grados de libertad aparece el giro a torsión y la primera derivada de la flecha, basta con derivar una vez la matriz de funciones de forma obtenida previamente respecto a la coordenada y,

[7]En el siguiente apartado se detalla el cálculo de estos parámetros en función de si el elemento pertenece a un larguero o costilla, en el caso de ala y estabilizador horizontal, o de si forma parte del fuselaje, ya que la sección supuesta será variable y, por tanto, la rigidez del elemento también.

y definir una matriz de rigideces $\mathbf{D}^e$ para poder expresar la energía de deformación total de forma matricial en función de los grados de libertad del elemento:

$$\mathcal{U}^e = \frac{1}{2}\mathbf{u}^{eT}\left(\int_{-1}^{1}\frac{d\mathbf{N}(\xi)}{d\xi}^T\mathbf{D}^e\frac{d\mathbf{N}(\xi)}{d\xi}\frac{L^e}{2}d\xi\right)\mathbf{u}^e = \frac{1}{2}\mathbf{u}^{eT}\mathbf{K}^e\mathbf{u}^e \tag{30}$$

siendo:

$$\mathbf{D}^e = \begin{bmatrix} GJ & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & EI \end{bmatrix} \tag{31}$$

Finalmente, se obtiene la matriz de rigidez del elemento:

$$\mathbf{K}^e = \begin{bmatrix} \frac{GJ}{L^e} & 0 & 0 & -\frac{GJ}{L^e} & 0 & 0 \\ 0 & \frac{12EI}{L^{e3}} & \frac{6EI}{L^{e2}} & 0 & -\frac{12EI}{L^{e3}} & \frac{6EI}{L^{e2}} \\ 0 & \frac{6EI}{L^{e2}} & \frac{4EI}{L^e} & 0 & -\frac{6EI}{L^{e2}} & \frac{2EI}{L^e} \\ -\frac{GJ}{L^e} & 0 & 0 & \frac{GJ}{L^e} & 0 & 0 \\ 0 & -\frac{12EI}{L^{e3}} & -\frac{6EI}{L^{e2}} & 0 & \frac{12EI}{L^{e3}} & -\frac{6EI}{L^{e2}} \\ 0 & \frac{6EI}{L^{e2}} & \frac{2EI}{L^e} & 0 & -\frac{6EI}{L^{e2}} & \frac{4EI}{L^e} \end{bmatrix} \tag{32}$$

Por último, solo queda ensamblar la matriz de rigidez de cada elemento para obtener la matriz de rigidez global de toda la estructura. Para ello, es necesario, en primer lugar, transformar la matriz $\mathbf{K}^e$ a coordenadas globales mediante una rotación del sistema de referencia local del elemento al global de la estructura. Dicha transformación se realiza a partir de una matriz de rotación $\mathbf{R}$ que, para el caso de barras en plano horizontal, coincide con la matriz $\mathbf{R}_z$ de rotación alrededor del eje vertical:

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos\alpha & -sin\alpha & 0 \\ sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{33}$$

siendo α es ángulo de giro del elemento respecto del sistema de referencia global. Así, todos los giros quedan expresados en el mismo sistema de referencia y se pueden sumar, alcanzando la compatibilidad de giros.

Sin embargo, si se quiere utilizar esta matriz de rotación, es necesario cambiar el vector de grados de libertad visto anteriormente para que el desplazamiento vertical ocupe la tercera posición y así la rotación sea coherente. De esta forma, el vector $\mathbf{u}^e$ quedará de la siguiente forma:

$$\mathbf{u}^e = \{\theta_1,\ \frac{\partial w_1}{\partial y},\ w_1, \theta_2,\ \frac{\partial w_2}{\partial y},\ w_2\}^T \tag{34}$$

Por tanto, para que todo el desarrollo matemático sea consistente, hay que recalcular las funciones de forma y, a su vez, la matriz de rigidez[8], aunque simplemente basta con intercambiar las filas 2 y 3, y 5 y 6, quedando $\mathbf{K}^e$ de la siguiente forma:

$$\mathbf{K}^e = \begin{bmatrix} \frac{GJ}{L^e} & 0 & 0 & -\frac{GJ}{L^e} & 0 & 0 \\ 0 & \frac{6EI}{L^{e2}} & \frac{4EI}{L^e} & 0 & -\frac{6EI}{L^{e2}} & \frac{2EI}{L^e} \\ 0 & \frac{12EI}{L^{e3}} & \frac{6EI}{L^{e2}} & 0 & -\frac{12EI}{L^{e3}} & \frac{6EI}{L^{e2}} \\ -\frac{GJ}{L^e} & 0 & 0 & \frac{GJ}{L^e} & 0 & 0 \\ 0 & \frac{6EI}{L^{e2}} & \frac{2EI}{L^e} & 0 & -\frac{6EI}{L^{e2}} & \frac{4EI}{L^e} \\ 0 & -\frac{12EI}{L^{e3}} & -\frac{6EI}{L^{e2}} & 0 & \frac{12EI}{L^{e3}} & -\frac{6EI}{L^{e2}} \end{bmatrix} \tag{35}$$

Así, finalmente, ya se puede realizar el ensamblaje en la matriz de rigidez global de la estructura $\mathbf{K}$. Esto dependerá de los nodos que conecta cada elemento, por lo que es necesario definir una matriz de conectividad de la estructura. Esta matriz, de dimensiones $(N_{elementos} \times 2)$, ya que cada elemento une 2 nodos, contendrá la información de los nodos que une cada elemento para colocar los elementos de la matriz de rigidez local de forma correcta en la global. De esta forma, $\mathbf{K}$ será una matriz cuadrada de dimensiones $(N_{gdl} \times N_{gdl})$ que permite expresar la energía de deformación total en función del vector de grados de libertad global de la estructura, $\mathbf{U}$:

$$\mathcal{U} = \frac{1}{2}\mathbf{U}^T\ \mathbf{K}\ \mathbf{U} \tag{36}$$

Por ejemplo, si el elemento 1 une los nodos 1 y 2 y el elemento 2, los nodos 2 y 3, el ensamblaje en la matriz global sería el que sigue. En primer lugar, las matrices de rigidez de cada elemento serán $\mathbf{K}^{(1)}$ y $\mathbf{K}^{(2)}$:

$$\mathbf{K}^{(1)} = \begin{bmatrix} k_{11}^{(1)} & 0 & 0 & k_{14}^{(1)} & 0 & 0 \\ 0 & k_{22}^{(1)} & k_{23}^{(1)} & 0 & k_{25}^{(1)} & k_{26}^{(1)} \\ 0 & k_{32}^{(1)} & k_{33}^{(1)} & 0 & k_{35}^{(1)} & k_{36}^{(1)} \\ k_{41}^{(1)} & 0 & 0 & k_{44}^{(1)} & 0 & 0 \\ 0 & k_{52}^{(1)} & k_{53}^{(1)} & 0 & k_{55}^{(1)} & k_{56}^{(1)} \\ 0 & k_{62}^{(1)} & k_{63}^{(1)} & 0 & k_{65}^{(1)} & k_{66}^{(1)} \end{bmatrix}, \mathbf{K}^{(2)} = \begin{bmatrix} k_{11}^{(2)} & 0 & 0 & k_{14}^{(2)} & 0 & 0 \\ 0 & k_{22}^{(2)} & k_{23}^{(2)} & 0 & k_{25}^{(2)} & k_{26}^{(2)} \\ 0 & k_{32}^{(2)} & k_{33}^{(2)} & 0 & k_{35}^{(2)} & k_{36}^{(2)} \\ k_{41}^{(2)} & 0 & 0 & k_{44}^{(2)} & 0 & 0 \\ 0 & k_{52}^{(2)} & k_{53}^{(2)} & 0 & k_{55}^{(2)} & k_{56}^{(2)} \\ 0 & k_{62}^{(2)} & k_{63}^{(2)} & 0 & k_{65}^{(2)} & k_{66}^{(2)} \end{bmatrix}$$

Denotando numéricamente los grados de libertad del nodo 1 por (1,2,3), los del 2 por (4,5,6) y los del 3 por (7,8,9), el ensamblaje en la matriz global de estos 2 elementos será:

[8]También podría haberse optado por utilizar la matriz de rotación $\mathbf{R}_y$ y mantener las matrices anteriores.

$$\mathbf{K} = \begin{bmatrix} k_{11}^{(1)} & 0 & 0 & k_{14}^{(1)} & 0 & 0 & \cdots & \cdots & \cdots \\ 0 & k_{22}^{(1)} & k_{23}^{(1)} & 0 & k_{25}^{(1)} & k_{26}^{(1)} & \cdots & \cdots & \cdots \\ 0 & k_{32}^{(1)} & k_{33}^{(1)} & 0 & k_{35}^{(1)} & k_{36}^{(1)} & \cdots & \cdots & \cdots \\ k_{41}^{(1)} & 0 & 0 & k_{44}^{(1)} + k_{11}^{(2)} & 0 & 0 & k_{14}^{(2)} & 0 & 0 \\ 0 & k_{52}^{(1)} & k_{53}^{(1)} & 0 & k_{55}^{(1)} + k_{22}^{(2)} & k_{56}^{(1)} + k_{23}^{(2)} & 0 & k_{25}^{(2)} & k_{26}^{(2)} \\ 0 & k_{62}^{(1)} & k_{63}^{(1)} & 0 & k_{65}^{(1)} + k_{32}^{(2)} & k_{66}^{(1)} + k_{33}^{(2)} & 0 & k_{35}^{(2)} & k_{36}^{(2)} \\ \vdots & \vdots & \vdots & k_{41}^{(2)} & 0 & 0 & k_{44}^{(2)} & 0 & 0 \\ \vdots & \vdots & \vdots & 0 & k_{52}^{(2)} & k_{53}^{(2)} & 0 & k_{55}^{(2)} & k_{56}^{(2)} \\ \vdots & \vdots & \vdots & 0 & k_{62}^{(2)} & k_{63}^{(2)} & 0 & k_{65}^{(2)} & k_{66}^{(2)} \end{bmatrix}$$

Atendiendo al modelo numérico descrito para la estructura, para su implementación en *Matlab* será necesario el cálculo de las matrices de conectividad y grados de libertad, las cuales definirán qué nodos conecta cada elemento y qué grados de libertad están asociados a cada nodo, así como la definición de parámetros como el número de nodos, el número de elementos, geometría de la estructura y, al igual que en el modelo aerodinámico, el número de divisiones en eje x e y, las cuales definirán los parámetros anteriores. Así, en el anexo B.3, se muestran todas las funciones desarrolladas para la implementación del modelo estructural, cuyo uso se describirá en el apartado 3.2.2 para el caso práctico propuesto.

2.2.2. Rigidez a flexión y torsión de los elementos

Para finalizar con el modelo estructural, es necesario detallar el cálculo de las rigideces a flexión y torsión de cada elemento ya que son determinantes en el cálculo de la matriz de rigidez global, tal y como se ha visto en la sección 2.2.1. Así, los módulos de rigidez E y G solo dependen del material y se suponen constantes, la diferencia reside en el cálculo de los momentos de inercia. Para ello, hay que diferenciar los elementos que pertenecen a ala y estabilizador, y los del fuselaje, ya que la sección estructural de los elementos es muy diferente y, por tanto, su rigidez también lo será.

En primer lugar, para el fuselaje, se supone una sección circular de pared delgada que incluye refuerzos mediante largueros longitudinales en Z (Figura 15). Además, el radio de la sección será variable para simular mejor la estructura del fuselaje de la aeronave real. Para este tipo de sección, los momentos de inercia respecto al centro del círculo son [7]:

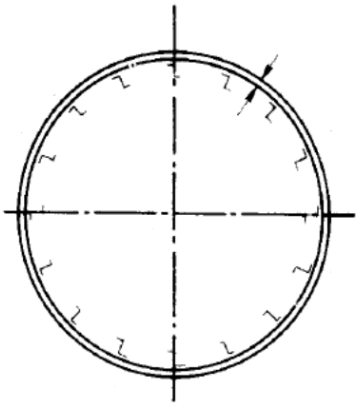


Figura 15: Sección circular de pared delgada con largueros en Z como refuerzos

$$I_0 = 2\pi t R^3 + \sum_{i=1}^{N_l} A_i R^2, \quad I_x = I_y = I = \frac{I_0}{2} \tag{37}$$

$$J = \frac{4\left(\pi R^2\right)^2}{\frac{(2\pi R)}{t}} + \sum_{i=1}^{N_l} \frac{l_i t_i^3}{3} \tag{38}$$

siendo R el radio de la sección del fuselaje, t el espesor, N_l el número de largueros de refuerzo y A_i, l_i y t_i el área, longitud de cada tramo y espesor de estos. Al ser largueros en Z, se componen de 3 segmentos, los cuales se suponen del mismo espesor t_l, por lo que siendo b la longitud de las alas y h la del alma, $A_i = 2bt_l + ht_l$.

Por otro lado, para el caso del ala y estabilizador horizontal, la estructura está formada por largueros y larguerillos, y costillas, que definen la forma del perfil alar. Así, se simplifica la estructura mediante estos elementos, como se muestra en la Figura 16.

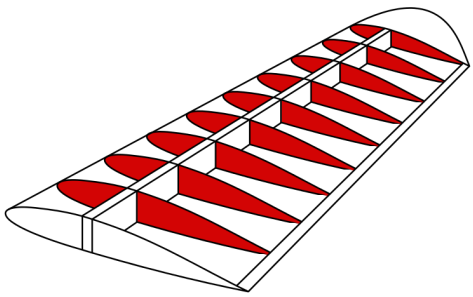


Figura 16: Sección simplificada para ala y estabilizador horizontal

La sección de los largueros y larguerillos será en forma de I, mientras que las costillas se suponen macizas pero de bajo espesor. De esta forma, la longitud de las costillas dependerá de la cuerda en cada punto del ala, y su forma, para simplificar los cálculos, se supone rectangular con altura igual al espesor medio del perfil real del ala. Las dimensiones de la sección de los largueros se supone de forma que la hipótesis de pared delgada sea aceptable. Así, el cálculo de I y J para estas secciones será [7]:

- Largueros:

$$I = \frac{1}{12}ht^3 + 2\left(\frac{1}{12}tb^3\right) \tag{39}$$

$$J = 2\frac{bt^3}{3} + \frac{ht^3}{3} \tag{40}$$

- Costillas:

$$I = \frac{1}{12}t_c^3\ h \tag{41}$$

$$J = \frac{1}{3}t_c^3\ h \tag{42}$$

siendo h y b las longitudes de alma y alas de los largueros, y t y t_c los espesores de largueros y costillas.

Debido a esta distinción en el cálculo de la rigidez, en la programación se comprueba para cada elemento su pertenencia dentro de la estructura y se calcula su rigidez de la forma descrita para introducirla en el cálculo de la matriz de rigidez local y su posterior ensamblaje en la matriz global.

2.3. Acoplamiento aeroelástico

2.3.1. Interpolación mediante *splines*

Para desarrollar el método de acoplamiento entre los modelos aerodinámico y estructural se hace uso del trabajo realizado por *Rodden* y *Harder* en [17] y [18]. El objetivo es obtener un modelo capaz de calcular los desplazamientos verticales de cualquier punto de la superficie de paneles del modelo aerodinámico a partir de la solución de desplazamientos de los nodos estructurales. Para ello, se utiliza la interpolación mediante *splines* de superficie descrita en los artículos mencionados previamente.

Partiendo de la ecuación diferencial que rige la flexión de placas de la teoría de *Kirchoff*[9] ([9]) y suponiendo la aplicación de una carga puntual, se puede llegar a una forma explícita de $w(x, y)$ que define el desplazamiento vertical de cualquier punto de la superficie (43) y que, además, es diferenciable:

$$w(x, y) = a_0 + a_1\ x + a_2\ y + \sum_{i=1}^{N} F_i r_i^2 \log(r_i^2) \tag{43}$$

donde a_0, a_1 y a_2 son coeficientes desconocidos, $F_i = \frac{P_i}{16\pi D}$, siendo P_i la carga puntual aplicada, y $r_i^2 = (x - x_i)^2 + (y - y_i)^2$. El subíndice i hace referencia a los puntos donde se conoce el desplazamiento, que, aplicado al caso de estudio, se corresponde con los nodos estructurales que pertenecen a la superficie aerodinámica.

Por tanto, la deformada w para un punto cualquiera de la superficie se puede expresar en forma matricial como:

$$w(x, y) = \{a_0, a_1, a_2, F_1, \cdots, F_N\} \begin{Bmatrix} 1 \\ x \\ y \\ \psi_1(x, y) \\ \vdots \\ \psi_N(x, y) \end{Bmatrix} = \mathbf{\Psi}_\sigma^T(x, y)\ \mathbf{f}_\sigma \tag{44}$$

donde $\psi_i(x, y) = r_i^2(x, y) \log(r_i^2(x, y))$ y σ hace referencia a la superficie aerodinámica. Así, para calcular el desplazamiento vertical de cualquier punto de la superficie definido por las coordenadas (x, y), basta con aplicar las ecuaciones de equilibrio (fuerzas y momentos nulos en la placa) y la ecuación anterior en los puntos conocidos i, y relacionarlos con los grados de libertad estructurales:

$$\sum_{i=1}^{N} F_i = \mathbf{0}, \quad \{0, 0, 0, 1, \cdots, 1\}\ \mathbf{f}_\sigma = \mathbf{0} \tag{45}$$

$$\sum_{i=1}^{N} x_i\ F_i = \mathbf{0}, \quad \{0, 0, 0, x_1, \cdots, x_N\}\ \mathbf{f}_\sigma = \mathbf{0} \tag{46}$$

$$\sum_{i=1}^{N} y_i\ F_i = \mathbf{0}, \quad \{0, 0, 0, y_1, \cdots, y_N\}\ \mathbf{f}_\sigma = \mathbf{0} \tag{47}$$

$$\mathbf{\Psi}_\sigma^T(x_i, y_i)\ \mathbf{f}_\sigma = w_i, \quad 1 \leq i \leq N \tag{48}$$

[9] $D\nabla^4 w = q$, donde D es la rigidez a flexión de la placa, $w(x, y)$ es la deformada a flexión de cualquier punto de la superficie y q la carga por unidad de área aplicada.

Expresando las $(N+3)$ ecuaciones anteriores en forma matricial e igualando a los grados de libertad estructurales[10] se tiene:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 0 & x_1 & \cdots & x_N \\ 0 & 0 & 0 & y_1 & \cdots & y_N \\ 1 & x_1 & y_1 & \psi_1(x_1,y_1) & \cdots & \psi_N(x_1,y_1) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & y_N & \psi_1(x_N,y_N) & \cdots & \psi_N(x_N,y_N) \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ F_1 \\ \vdots \\ F_N \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ w_1 \\ \vdots \\ w_N \end{Bmatrix} \tag{49}$$

$$\begin{Bmatrix} 0 \\ 0 \\ 0 \\ w_1 \\ \vdots \\ w_N \end{Bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 1s & en & pos. & w_1 \\ 1s & en & pos. & w_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1s & en & pos. & w_N \end{bmatrix} \begin{Bmatrix} u_1 \\ \vdots \\ u_{N_{gdl}} \end{Bmatrix} \tag{50}$$

Igualando ambos resultados y definiendo nuevas matrices para simplificar el desarrollo, se obtiene:

$$\mathbf{B}_\sigma \ \mathbf{f}_\sigma = \mathbf{R}_\sigma \ \mathbf{u} \tag{51}$$

$$w_\sigma(x,y) = \mathbf{\Psi}_\sigma^T(x,y) \ \mathbf{B}_\sigma^{-1} \ \mathbf{R}_\sigma \ \mathbf{u} = \mathbf{N}_\sigma^T(x,y) \ \mathbf{u} \tag{52}$$

De esta forma, la expresión 52 permite relacionar el desplazamiento vertical de cualquier punto de la superficie aerodinámica σ con los grados de libertad estructurales a partir de la matriz $\mathbf{N}_\sigma(x,y)$, cuyas dimensiones son $(1 \times N_{gdl})$[11].

Finalmente, estas nuevas matrices se programan en *Matlab* como se muestra en el anexo B.4, y, a partir del resultado de la matriz de acoplamiento $\mathbf{N}_\sigma(x,y)$, se desarrolla el cálculo de la matriz de masas y de las matrices de acoplamiento dinámicas, necesarias para la resolución del problema aeroelástico.

[10] Según el modelo estructural, el desplazamiento vertical de un nodo contenido en la superficie σ se corresponde con el tercer término del vector de grados de libertad estructurales del nodo (34), por lo que se puede obtener la matriz $\mathbf{R}_\sigma$ de relación.

[11] $\mathbf{\Psi}_\sigma^T(x,y) \equiv (1 \times (N+3))$ $\mathbf{B}_\sigma \equiv ((N+3) \times (N+3))$, $\mathbf{R}_\sigma \equiv ((N+3) \times (N_{gdl}))$ y $\mathbf{u} \equiv (N_{gdl} \times 1)$.

2.3.2. Matriz de masas

Dado que la distribución de masa de un avión real es muy compleja, una práctica común en este tipo de análisis es asignar masas puntuales en diversos puntos de la malla estructural, normalmente los nodos, y obtener la matriz de masas como la suma de la contribución de cada uno de estos puntos. Así, la matriz de masa se puede calcular mediante la expresión 53, la cual hace uso de la matriz $\mathbf{N}_\sigma(x, y)$ obtenida del acoplamiento aeroelástico. Además, aparece el término m_i que hace referencia a cada masa puntual considerada en el punto (x_i, y_i).

$$\mathbf{M} = \sum_{i=1}^{N} \mathbf{N}_\sigma(x_i, y_i)\ \mathbf{N}_\sigma^T(x_i, y_i)\ m_i \tag{53}$$

Esta forma de cálculo se debe a que la distribución de masas de una estructura real es muy irregular. Por ejemplo, las alas suelen ser de geometría variable, con un elevado número de elementos estructurales (largueros y costillas) que no se distribuyen de forma continua a lo largo de la misma, y otros elementos que suponen masas puntuales como los de control, los depósitos de combustible, motores, tren de aterrizaje, etc. Por otro lado, en el fuselaje hay que tener en cuenta la bodega, la cabina o el estabilizador vertical, elementos que también hace que la parametrización matemática de la masa no sea sencilla.

De esta forma, al igual que para el cálculo de la rigidez estructural, para la matriz de masa se divide el modelo en superficies sustentadoras y fuselaje.

En primer lugar, en el caso de ala y estabilizador horizontal se divide la masa total de estas superficies entre los nodos de la estructura que pertenecen a estas. Sin embargo, esta distribución no se realiza de forma uniforme, sino que, para obtener un modelo más realista, se supone una distribución tal que los nodos centrales contengan más masa que los de las esquinas o los de los extremos. Así, un nodo que une 4 elementos tendrá el doble de masa que uno que une 2 elementos (esquinas), y 1.5 veces más masa que uno que une 3 elementos. Con ello, se puede determinar el valor de m_i de cada nodo y calcular su contribución a la matriz de masa global.

Por otro lado, para el fuselaje se supone una distribución entre sus nodos de forma que se mantenga el centro de gravedad real del mismo, ya que este parámetro influye de manera muy importante a la estabilidad del mismo y hace el modelo más realista.

Con todo ello, *MatrizMasa* y *dist_fuselaje* son las funciones desarrolladas en *Matlab*, ambas en el anexo B.4.2. La primera calcula la matriz de masa de ala y estabilizador (M_{w+s}) a partir de la información extraída de los modelos estructural y aerodinámico, para, posteriormente, sumar la contribución del fuselaje (M_f). La función *dist_fuselaje* calcula el valor m_i de los nodos del fuselaje para el cálculo de M_f.

2.3.3. Matrices de acoplamiento fluido-estructura

Para finalizar el acoplamiento entre modelos, además de definir $\mathbf{N}_\sigma^T(x, y)$, es necesario el cálculo de matrices adicionales para realizar el cálculo numérico del problema aeroelástico, como se verá en la sección 2.4. Así, son necesarias 3 matrices adicionales, conocidas como *matrices de acoplamiento dinámicas*:

- $\mathbf{D}_z$: matriz de dimensiones $(N_p \times N_{gdl})$, que relaciona los desplazamientos verticales de los centros aerodinámicos de cada panel, $\mathbf{z}_A$, con los desplazamientos verticales de los grados de libertad estructurales, $\mathbf{u}$.

$$\mathbf{z}_A = \mathbf{D}_z\ \mathbf{u} \tag{54}$$

- $\mathbf{D}_u$: matriz de dimensiones $(N_p \times N_{gdl})$, que relaciona el ángulo de ataque de cada panel evaluado en los puntos de control, $\mathbf{a}$, con los grados de libertad estructurales, $\mathbf{u}$.

$$\mathbf{a} = \mathbf{D}_u\ \mathbf{u} \tag{55}$$

- $\mathbf{D}_v$: matriz de dimensiones $(N_p \times N_{gdl})$, que relaciona el ángulo de ataque de cada panel de la malla aerodinámica, $\mathbf{a}$, con la variación en el tiempo de los grados de libertad estructurales, $\dot{\mathbf{u}}$, por lo que tiene naturaleza dinámica.

$$\mathbf{a} = \frac{1}{U_\infty}\ \mathbf{D}_v\ \dot{\mathbf{u}} \tag{56}$$

En primer lugar, para la obtención de la matriz $\mathbf{D}_z$, se parte de la ecuación 52, donde se tiene el desplazamiento vertical de cualquier punto de una superficie cualquiera σ, por lo que, evaluando la matriz $\mathbf{N}_\sigma$ en la posición de los centros aerodinámicos de los paneles que pertenecen a cada superficie, se obtiene la matriz $\mathbf{D}_z$, donde S es el número total de superficies consideradas en el modelo.

$$\mathbf{z}_A = \begin{bmatrix} \mathbf{N}_1^T(x_{CA_j}, y_{CA_j}) & si\ (x_{CA_j}, y_{CA_j}) \in\ Sup.\ 1 \\ \vdots & \vdots \\ \mathbf{N}_S^T(x_{CA_j}, y_{CA_j}) & si\ (x_{CA_j}, y_{CA_j}) \in\ Sup.\ S \end{bmatrix} \mathbf{u} \tag{57}$$

Por otro lado, para el cálculo de las matrices $\mathbf{D}_u$ y $\mathbf{D}_v$ es necesario partir de la condición de contorno de aerodinámica no-estacionaria [4][3]:

$$\frac{\partial \varphi}{\partial z} = \frac{\partial z_p}{\partial t} + U_\infty \frac{\partial z_p}{\partial x} \;\rightarrow\; a_\sigma(t) = \frac{\partial w_\sigma}{\partial x} + \frac{1}{U_\infty}\frac{\partial w_\sigma}{\partial t} \tag{58}$$

Introduciendo en este punto el resultado de 52, se obtiene:

$$a_\sigma(x,y,t) = \frac{\partial \mathbf{N}_\sigma^T(x,y)}{\partial x}\ \mathbf{u}(t) + \frac{1}{U_\infty}\mathbf{N}_\sigma^T(x,y)\ \dot{\mathbf{u}}(t) \tag{59}$$

y, finalmente, por comparación, se pueden calcular las matrices $\mathbf{D}_u$ y $\mathbf{D}_v$ de forma similar a $\mathbf{D}_z$:

$$\mathbf{a}_C^{(1)} = \begin{bmatrix} \frac{\partial \mathbf{N}_1^T(x_{C_j}, y_{C_j})}{\partial x} & si\ (x_{C_j}, y_{C_j}) \in\ Sup.\ 1 \\ \vdots & \vdots \\ \frac{\partial \mathbf{N}_S^T(x_{C_j}, y_{C_j})}{\partial x} & si\ (x_{C_j}, y_{C_j}) \in\ Sup.\ S \end{bmatrix} \mathbf{u} \tag{60}$$

$$\mathbf{a}_C^{(2)} = \frac{1}{U_\infty}\begin{bmatrix} \mathbf{N}_1^T(x_{C_j}, y_{C_j}) & si\ (x_{C_j}, y_{C_j}) \in Sup.\ 1 \\ \vdots & \vdots \\ \mathbf{N}_S^T(x_{C_j}, y_{C_j}) & si\ (x_{C_j}, y_{C_j}) \in Sup.\ S \end{bmatrix} \dot{\mathbf{u}} \tag{61}$$

De esta forma, el vector de ángulos de ataque de cada panel evaluado en los puntos de control vendrá determinado por la suma de las ecuaciones 60 y 61.

$$\mathbf{a}_C = \mathbf{a}_C^{(1)} + \mathbf{a}_C^{(2)} = \mathbf{D}_u\ \mathbf{u} + \frac{1}{U_\infty}\ \mathbf{D}_v\ \dot{\mathbf{u}} \tag{62}$$

Por último, se desarrolla la función *MatricesAcoplamiento_Dinamicas* (B.4.3) en *Matlab* que realiza el proceso descrito para el cálculo de las matrices $\mathbf{D}_z$, $\mathbf{D}_u$ y $\mathbf{D}_v$.

2.4. Aeroelasticidad estática y dinámica

Una vez definidos los modelos estructural y aerodinámico así como su acoplamiento, ya es posible abordar el problema aeroelástico de forma numérica. Para ello, se hace uso de las ecuaciones de *Lagrange* en los grados de libertad estructurales que permite obtener la geometría deformada del avión cuando se encuentra sometida a fuerzas externas, en este caso, aerodinámicas. En primer lugar, se desarrolla el problema estático, es decir, no se tienen en cuenta las fuerzas de inercia ni los efectos no-estacionarios de las fuerzas aerodinámicas, para luego, extenderlo al dominio temporal y abordar el problema dinámico.

Aeroelasticidad estática

Las ecuaciones de *Lagrange* en su forma estática son:

$$\frac{\partial \mathcal{U}}{\partial \mathbf{u}} = \mathbf{Q} \tag{63}$$

donde $\mathcal{U}$ representa la energía de deformación, $\mathbf{u}$ el vector de grados de libertad estructurales y $\mathbf{Q}$ las fuerzas generalizadas asociadas a los grados de libertad estructurales. Expresando la energía de deformación elástica en función del vector de grados de libertad a partir del modelo estructural, se llega a la ecuación 64, en la que aparece la matriz de rigidez $\mathbf{K}$ del modelo.

$$\mathcal{U} = \frac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} \rightarrow \mathbf{K}\mathbf{u} = \mathbf{Q} \tag{64}$$

Por otro lado, para la obtención de $\mathbf{Q}$, se aplica el principio del trabajo virtual, es decir, el trabajo debido a las fuerzas exteriores actuantes sobre la aeronave debido a un desplzamiento virtual de los grados de libertad de $\mathbf{u}$. Así, el trabajo virtual, se puede expresar siempre como una combinación lineal de coeficientes por la variación virtual de $\mathbf{u}$ (65), en la que dichos coeficientes se corresponden con los elementos del vector $\mathbf{Q}$ de las fuerzas generalizadas.

$$\delta\mathcal{W} = \delta\mathbf{u}^T\mathbf{Q} \tag{65}$$

En este punto, entra en juego el modelo aerodinámico por un lado, ya que se necesita del cálculo de las fuerzas aerodinámicas sobre cada panel, y, por otro, el acoplamiento fluido-estructura, para la obtención de los desplazamientos virtuales de los centros aerodinámicos producidos por las fuerzas actuantes. Así, a partir de la ecuación 19 obtenida en 2.1 y de la 57 descrita en 2.3 se puede calcular el trabajo virtual y, consecuentemente, el vector de fuerzas generalizadas:

1. Fuerzas $\rightarrow \mathbf{L} = \rho_\infty U_\infty \boldsymbol{\Delta Y \Gamma}$
2. Desplazamientos $\rightarrow \mathbf{z}_A = \mathbf{D}_z\mathbf{u}$
3. Desplazamiento virtual $\rightarrow \delta\mathbf{z}_A = \mathbf{D}_z\delta\mathbf{u}$
4. Trabajo virtual $\rightarrow \delta\mathcal{W} = \delta\mathbf{z}_A^T\,\mathbf{L} = (\delta\mathbf{u}^T\mathbf{D}_z^T)\,\mathbf{L}$
5. Fuerzas generalizadas $\rightarrow \mathbf{Q} = \mathbf{D}_z^T\mathbf{L}$

Finalmente, las ecuaciones de *Lagrange* para el problema estático quedarán:

$$\mathbf{K}\mathbf{u} = \mathbf{D}_z^T\mathbf{L} \tag{66}$$

donde, aplicando el sistema de ecuaciones lineal obtenido también del modelo aerodinámico (15) e introduciendo la parte estacionaria del vector **a** desarrollado en 2.3, se llega al sistema de ecuaciones final que permite resolver el problema aeroelástico estático:

$$\mathbf{K}\mathbf{u} = \mathbf{D}_z^T\mathbf{L} = \rho_\infty U_\infty^2 \mathbf{D}_z^T \boldsymbol{\Delta Y} \mathbf{H}^{-1} \left(\mathbf{a}_r + \mathbf{D}_u \mathbf{u}\right) \tag{67}$$

Para compactar el problema, de 67 se pueden agrupar los términos que multiplican al vector **u** e introduciendo la matriz $\mathbf{A}_s$, de dimensiones $(N_{gdl} \times N_{gdl})$[12], el vector $\mathbf{f}_r$, $(N_{gdl} \times 1)$[13] y la definición de presión dinámica, $q = \frac{1}{2}\rho_\infty U_\infty^2$:

$$\left(\mathbf{K} - \rho_\infty U_\infty^2 \mathbf{D}_z^T \boldsymbol{\Delta Y} \mathbf{H}^{-1} \mathbf{D}_u\right) \mathbf{u} = \rho_\infty U_\infty^2 \mathbf{D}_z^T \boldsymbol{\Delta Y} \mathbf{H}^{-1} \mathbf{a}_r \tag{68}$$

$$\mathbf{A}_s = 2\ \mathbf{D}_z^T \boldsymbol{\Delta Y} \mathbf{H}^{-1} \mathbf{D}_u, \quad \mathbf{f}_r = 2\ \mathbf{D}_z^T \boldsymbol{\Delta Y} \mathbf{H}^{-1} \mathbf{a}_r \tag{69}$$

$$\left(\mathbf{K} - q\ \mathbf{A}_s\right) \mathbf{u} = q\ \mathbf{f}_r \tag{70}$$

La ecuación 70 permite obtener la deformación estacionaria de cada grado de libertad de la aeronave para cualquier velocidad de vuelo: $\mathbf{u} = q\left(\mathbf{K} - q\ \mathbf{A}_s\right)^{-1} \mathbf{f}_r$. Además, se puede apreciar la inestabilidad por divergencia, ya que la respuesta es inversamente proporcional al determinante de la matriz $[\mathbf{K} - q\ \mathbf{A}_s]$, por lo que si dicho determinante es nulo, la respuesta tenderá al infinito. Así, para el cálculo de la velocidad de divergencia, basta con resolver el problema de valores y vectores propios asociado (71), donde los autovalores se corresponden con valores de presión que anulan el determinante anterior, mientras que los autovectores se corresponden con la deformación de los grados de libertad asociada a cada presión. Por tanto, la presión de divergencia será el menor valor positivo de los autovalores calculados, ya que, a partir de ese momento, la aeronave será inestable.

$$\left(\mathbf{K} - q\ \mathbf{A}_s\right) \mathbf{u} = \mathbf{0} \tag{71}$$

Aeroelasticidad dinámica

Para el problema dinámico, se parte de las ecuaciones de *Lagrange* en su forma completa:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{T}}{\partial \dot{\mathbf{u}}}\right) + \frac{\partial \mathcal{U}}{\partial \mathbf{u}} = \mathbf{Q} \tag{72}$$

[12] $(N_{gdl} \times N_p) \cdot (N_p \times N_p) \cdot (N_p \times N_{gdl}) = (N_{gdl} \times N_{gdl})$.
[13] $(N_{gdl} \times N_p) \cdot (N_p \times N_p) \cdot (N_p \times 1) = (N_{gdl} \times 1)$.

donde la única diferencia con las ecuaciones estáticas es la dependencia con el tiempo de los grados de libertad y, por tanto, la aparición de la energía cinética $\mathcal{T}$.

Recurriendo de nuevo a las definiciones de las energías de deformación y cinética, se puede obtener la parte izquierda de la ecuación 72 en función de las matrices de masa y rigidez del modelo:

$$\mathcal{U} = \frac{1}{2}\mathbf{u}^T\mathbf{K}\ \mathbf{u} \rightarrow \frac{\partial \mathcal{U}}{\partial \mathbf{u}} = \mathbf{K}\ \mathbf{u} \tag{73}$$

$$\mathcal{T} = \frac{1}{2}\ddot{\mathbf{u}}^T\mathbf{M}\ \ddot{\mathbf{u}} \rightarrow \frac{d}{dt}\left(\frac{\partial \mathcal{T}}{\partial \dot{\mathbf{u}}}\right) = \mathbf{M}\ \ddot{\mathbf{u}} \tag{74}$$

por lo que, la ecuación 72 se puede reescribir como:

$$\mathbf{M}\ \ddot{\mathbf{u}} + \mathbf{K}\ \mathbf{u} = \mathbf{Q} \tag{75}$$

El cálculo del vector de fuerzas generalizadas se aborda de la misma forma que en el caso estático, a partir del principio del trabajo virtual, siendo el desarrollo el mismo que en el caso anterior pero teniendo en cuenta que tanto la sustentación como los desplazamientos de los centros aerodinámicos de los paneles dependerán del tiempo. Sin embargo, en el modelo aerodinámico utilizado, la dependencia de la sustentación con tiempo no aparece ya que en el mismo no se incluyen los efectos no-estacionarios, como por ejemplo, el efecto de la estela del ala, para lo que sería necesario implementar el *Unsteady Vortex Lattice Method*, que, además de la malla de paneles de las superficies de sustentación, genera una malla para las estelas ya que en cada instante de tiempo se generará un torbellino aguas abajo de dichas superficies. Este aspecto queda pendiente como trabajo futuro dentro del proyecto, ya que para cumplir con el objetivo principal, es suficiente con plantear un modelo casi-estacionario y obtener resultados preliminares de la velocidad de flameo.

De esta forma, se mantiene el modelo aerodinámico desarrollado en 2.1, aunque introduciendo la condición de contorno propia de aerodinámica no estacionaria: $\frac{\partial \varphi}{\partial z} = \frac{\partial z_p}{\partial t} + U_\infty \frac{\partial z_p}{\partial x}$ ([4][19]), la cual genera nuevos términos tanto en el vector $\mathbf{a}$[14] como en el cálculo de la fuerza de sustentación:

$$\mathbf{a} = \mathbf{D}_u\ \mathbf{u} + \frac{1}{U_\infty}\mathbf{D}_v\ \dot{\mathbf{u}} \tag{76}$$

[14]Expresión obtenida en 2.3, en la que no se tiene en cuenta la parte rígida, ya que no influye en el cálculo de las curvas de flameo, como tampoco lo hacía en la velocidad de divergencia.

$$\mathbf{L} = \rho_\infty U_\infty \boldsymbol{\Delta Y}\ \boldsymbol{\Gamma} + \rho_\infty \boldsymbol{\Delta S}\ \mathbf{T}\ \dot{\boldsymbol{\Gamma}} \tag{77}$$

donde $\boldsymbol{\Gamma}(t)$ y $\dot{\boldsymbol{\Gamma}}(t)$ se obtienen a partir de la aplicación del sistema de ecuaciones lineal del modelo aerodinámico (78 y 79), la matriz $\boldsymbol{\Delta Y}$ es ya conocida, matriz diagonal $(N_p \times N_p)$ que contiene la proyección a lo largo de la envergadura del segmento AB de cada panel, la matriz $\boldsymbol{\Delta S}$ es también diagonal $(N_p \times N_p)$ y contiene el área de cada panel, y la $\mathbf{T}$ es una matriz cuadrada de la misma dimensión que las anteriores que contiene 3/4 en los elementos de la diagonal y 1s en los elementos (j,k) si se cumple que $x_{Ck} < x_{Cj}$ y $y_{Ck} = y_{Cj}$. Esto es debido a que al aplicar las ecuaciones de aerodinámica no estacionaria, la sustentación de un panel depende de su propia circulación pero también de la variación de circulación $\dot{\Gamma}_j$ de los paneles aguas arriba [19].

$$\boldsymbol{\Gamma}(t) = U_\infty\ \mathbf{H}^{-1}\ \mathbf{a} = U_\infty\ \mathbf{H}^{-1} \left(\mathbf{D}_u\ \mathbf{u} + \frac{1}{U_\infty} \mathbf{D}_v\ \dot{\mathbf{u}} \right) \tag{78}$$

$$\dot{\boldsymbol{\Gamma}} = U_\infty\ \mathbf{H}^{-1}\ \dot{\mathbf{a}} = U_\infty\ \mathbf{H}^{-1} \left(\mathbf{D}_u\ \dot{\mathbf{u}} + \frac{1}{U_\infty} \mathbf{D}_v\ \ddot{\mathbf{u}} \right) \tag{79}$$

A partir de estas expresiones, se puede desarrollar la fuerza de sustentación, en la que se distinguen 4 términos (80). Los 2 primeros tienen naturaleza circulatoria ya que provienen de la parte estacionaria de la ecuación 77, mientras que los otros 2 son debidos a fuerzas de inercia. El tercero de ellos es debido a que el perfil, al sufrir una torsión, gira con velocidad de rotación respecto del sistema de referencia global que se mueve con velocidad U_∞, por lo que se trata de fuerzas tipo *coriolis*. Por último, el cuarto término refleja las fuerzas de inercia por aceleraciones del aire alrededor del perfil, por lo que estas pueden surgir incluso en ausencia de velocidad.

$$\begin{aligned} \mathbf{L} = \rho_\infty\ U_\infty^2\ \boldsymbol{\Delta Y}\ \mathbf{H}^{-1}\mathbf{D}_u\ \mathbf{u}\ +\ \rho_\infty\ U_\infty\ \boldsymbol{\Delta Y}\ \mathbf{H}^{-1}\mathbf{D}_v\ \dot{\mathbf{u}}\ + \\ \rho_\infty\ U_\infty\ \boldsymbol{\Delta S}\ \mathbf{T}\ \mathbf{H}^{-1}\mathbf{D}_u\ \dot{\mathbf{u}}\ +\ \rho_\infty\ \boldsymbol{\Delta S}\ \mathbf{T}\ \mathbf{H}^{-1}\mathbf{D}_v\ \ddot{\mathbf{u}} \end{aligned} \tag{80}$$

Finalmente, el vector de fuerzas generalizadas se puede expresar de la misma forma que en el caso estático, pero con la dependencia temporal: $\mathbf{Q} = \mathbf{D}_z^T\ \mathbf{L}$. Así, introduciendo los resultados anteriores y compactando los términos de $\mathbf{u}$, $\dot{\mathbf{u}}$ y $\ddot{\mathbf{u}}$, se llega al sistema de ecuaciones que permite obtener las vibraciones de la aeronave ante cualquier perturbación exterior:

$$\mathbf{M}\ \ddot{\mathbf{u}} + \mathbf{K}\ \mathbf{u} = \rho_\infty\ U_\infty^2\ \mathbf{A}\ \mathbf{u} + \rho_\infty\ U_\infty\ \mathbf{B}\ \dot{\mathbf{u}} + \rho_\infty\ \mathbf{C}\ \ddot{\mathbf{u}} \tag{81}$$

siendo las matrices $\mathbf{A}$, $\mathbf{B}$ y $\mathbf{C}$ cuadradas y de dimensiones $(N_{gdl} \times N_{gdl})$:

$$\mathbf{A} = \mathbf{D}_z^T \boldsymbol{\Delta Y}\ \mathbf{H}^{-1}\mathbf{D}_u \tag{82}$$

$$\mathbf{B} = \mathbf{D}_z^T \boldsymbol{\Delta Y}\ \mathbf{H}^{-1}\mathbf{D}_v + \mathbf{D}_z^T \boldsymbol{\Delta S}\ \mathbf{T}\ \mathbf{H}^{-1}\mathbf{D}_u \tag{83}$$

$$\mathbf{C} = \mathbf{D}_z^T \boldsymbol{\Delta S}\ \mathbf{T}\ \mathbf{H}^{-1}\mathbf{D}_v \tag{84}$$

Por último, queda describir el método de resolución de este sistema para la obtención de la velocidad de flameo. Para ello, se desarrolla el método espacio-estado.

En primer lugar, se reordenan los términos para compactar la ecuación:

$$\mathbf{M}_{eq}\ \ddot{\mathbf{u}} + \mathbf{D}_{eq}\ \dot{\mathbf{u}} + \mathbf{K}_{eq}\ \mathbf{u} = \mathbf{0} \tag{85}$$

$$\mathbf{M}_{eq} = \mathbf{M} - \rho_\infty\ \mathbf{C}, \quad \mathbf{D}_{eq} = -\ \rho_\infty\ U_\infty\ \mathbf{B}, \quad \mathbf{K}_{eq} = \mathbf{K} - \rho_\infty\ U_\infty^2\ \mathbf{A} \tag{86}$$

Introduciendo ahora la solución $\mathbf{u}(t)$ como superposición modal: $\mathbf{u}(t) = \mathbf{u}\ e^{st}$, donde $\mathbf{u}$ representa el autovector y s el autovalor para cada velocidad de vuelo, se tiene:

$$s^2\ \mathbf{M}_{eq} + s\ \mathbf{D}_{eq} + \mathbf{K}_{eq} = \mathbf{0} \tag{87}$$

Para la resolución del problema, como se ha comentado previamente, se transforma la ecuación anterior en su versión espacio-estado, con el objetivo de obtener un sistema lineal. Para ello, se definen $\mathbf{w}_1 = \mathbf{u}$ y $\mathbf{w}_2 = s\ \mathbf{u}$, por lo que el problema cuadrático se reduce a un sistema lineal de 2 ecuaciones, aunque de tamaño dos veces el inicial ($2 \times N_{gdl}$):

$$s\ \mathbf{M}_{eq}\ \mathbf{w}_2 + \mathbf{D}_{eq}\ \mathbf{w}_2 + \mathbf{K}_{eq}\ \mathbf{w}_1 = \mathbf{0} \tag{88}$$

$$s\ \mathbf{w}_1 - \mathbf{w}_2 = \mathbf{0} \tag{89}$$

Finalmente se reescriben las ecuaciones anteriores en forma matricial:

$$\left(s \begin{bmatrix} \mathbf{I}_{N_{gdl}} & \mathbf{0}_{N_{gdl}} \\ \mathbf{0}_{N_{gdl}} & \mathbf{M}_{eq} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{N_{gdl}} & -\mathbf{I}_{N_{gdl}} \\ \mathbf{K}_{eq} & \mathbf{D}_{eq} \end{bmatrix} \right) \begin{Bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{Bmatrix} = \begin{bmatrix} \mathbf{0}_{N_{gdl}} \\ \mathbf{0}_{N_{gdl}} \end{bmatrix} \tag{90}$$

De esta forma, se obtendrán $2N_{gdl}$ autovalores y sus correspondientes autovectores, que aparecerán en parejas complejo-conjugadas. Así, la inestabilidad del movimiento oscilatoria dependerá de la naturaleza de los autovalores s_j. Por tanto, descomponiendo el autovalor en su parte real y compleja, $s_j = -g_j + i\omega_j$, la forma en el tiempo asociada al modo j será del tipo $e^{s_j t} = e^{-g_j t}\left(\cos\omega_j t + i\sin\omega_j t\right)$, por lo que se pueden distinguir 6 tipos de respuesta:

- $g_j > 0$, $\omega_j \neq 0 \rightarrow$ decrecimiento exponencial armónico
- $g_j > 0$, $\omega_j = 0 \rightarrow$ decrecimiento exponencial puro
- $g_j < 0$, $\omega_j \neq 0 \rightarrow$ crecimiento exponencial armónico
- $g_j < 0$, $\omega_j = 0 \rightarrow$ crecimiento exponencial puro
- $g_j = 0$, $\omega_j \neq 0 \rightarrow$ límite de estabilidad dinámico, autovalor asociado a la velocidad de flameo (movimiento oscilatorio no amortiguado)
- $g_j > 0$, $\omega_j = 0 \rightarrow$ límite de estabilidad estático, autovalor asociado a la velocidad de divergencia

Así, resolviendo el sistema anterior para cada velocidad de vuelo, se puede representar la parte real e imaginaria de cada autovalor, obteniendo así las curvas de flameo, en las que se observa de forma gráfica las velocidades para las que se dan dichas inestabilidades. Además, es especialmente importante remarcar que para velocidad nula, es decir, avión en tierra, se tiene el problema clásico de vibraciones libres no amortiguadas cuya solución son las frecuencias naturales del avión y sus modos propios de vibración, mientras que si se elimina la parte no-estacionaria de la ecuación 81, se tiene el problema estático, lo que es consistente con que en los tipos de respuesta aparezca la inestabilidad por divergencia.

Implementación en *Matlab*

El último paso es implementar los desarrollos mostrados en esta sección en *Matlab*, al igual que se ha hecho con los modelos aerodinámico y estructural. En los anexos B.6 y B.7 se detallan las funciones donde se ha programado el problema estático y dinámico.

Para el caso estático, solo se desarrolla una función: *ModosDivergencia*, la cual resuelve el problema de autovalores y autovectores de la ecuación 71 obteniendo la velocidad de divergencia y, además, representando el modo de deformación del avión con el que se alcanza la divergencia.

Por otro lado, para el problema dinámico se desarrollan 4 funciones:

- *CurvasFlameo*: obtiene las curvas de flameo resolviendo el sistema matricial de la ecuación 90 para cada velocidad de vuelo definidas como un vector. Para ello, hace uso de la función *SolucionAutovalores_Metodo_EspacioEstado* en cada iteración.
- *SolucionAutovalores_Metodo_EspacioEstado*: obtiene la solución del sistema matricial de la ecuación 90 dada la velocidad de vuelo.
- *Calcula_Divergencia_y_Flameo*: calcula las velocidades de divergencia y flameo a partir de las curvas de flameo.
- *DibujaCurvasFlameo*: representa las curvas de flameo a partir de los resultados de frecuencias (ω_j) y amortiguamientos (g_j) obtenidos de *CurvasFlameo*.

Así, en la siguiente sección, se hará uso de estas funciones, además de las del modelo estructural y aerodinámico, para la obtención de resultados a partir de su implementación en el *script* general del programa descrito en el anexo B.1.

3. Aplicación práctica

3.1. Validación del modelo: Ala Goland

Una vez detallados los modelos estructural y aerodinámico utilizados en el trabajo, el primer paso de cualquier proyecto ingenieril consiste en verificar que dichos modelos son correctos y pueden usarse para la aplicación para la que se han desarrollado. Así, tal y como se comentó en la introducción, se hace uso del ala *Goland* para la verificación del modelo global ya que está altamente contrastada en numerosos estudios de esta índole, entre los cuales, se destaca el del propio *Goland* en 1945 [12], el de *Raghavan* y *Patil* [15] y el de *Kumar et al.* [13]. El primero de los mencionados, se centra en el estudio de flameo del ala, mientras que en los otros 2 se obtienen los modos y frecuencias propias de la semi-ala empotrada. Además, en [15] se obtiene también la velocidad de divergencia y el punto de flameo del ala, es decir, velocidad y frecuencia de vibración. En este caso, al hacer uso de un modelo aerodinámico casi-estacionario, el flameo no puede utilizarse como verificación, por lo que este apartado está centrado en el cálculo del problema de vibraciones, es decir, en la obtención de frecuencias naturales y modos propios, y en el de la divergencia, dado que es un fenómeno aeroelástico estático.

En primer lugar, en la Tabla 1 se resumen los parámetros principales del ala *Goland* utilizada en los artículos mencionados, aunque para la validación se hará uso de [15]. De estos, se puede derivar que se utiliza un modelo de eje elástico ya que se definen rigideces a flexión y torsión. Por el contrario, en el modelo actual, las rigideces se calculan por elemento según la sección del mismo, por lo que es necesario ajustar los parámetros de las secciones así como el número de costillas y largueros para obtener una rigidez global similar y poder comparar los resultados. Así, en la Figura 17 se muestra la geometría en planta y la discretización en paneles y nodos del modelo global desarrollado.

Parámetro	**Descripción**	**Valor**
b	Semi-envergadura [m]	6.096
c	Cuerda [m]	1.829
EI	Rigidez a flexión [Nm^2]	$9.77 \cdot 10^6$
GJ	Rigidez a torsión [Nm^2]	$0.987 \cdot 10^6$
x_c	Centro de gravedad seccional [% c]	43
x_a	Posición del eje elástico [% c]	33
m	Masa por unidad de envergadura [kg/m]	35.719

Tabla 1: Parámetros geométricos, elásticos y másicos del ala *Goland* [15]

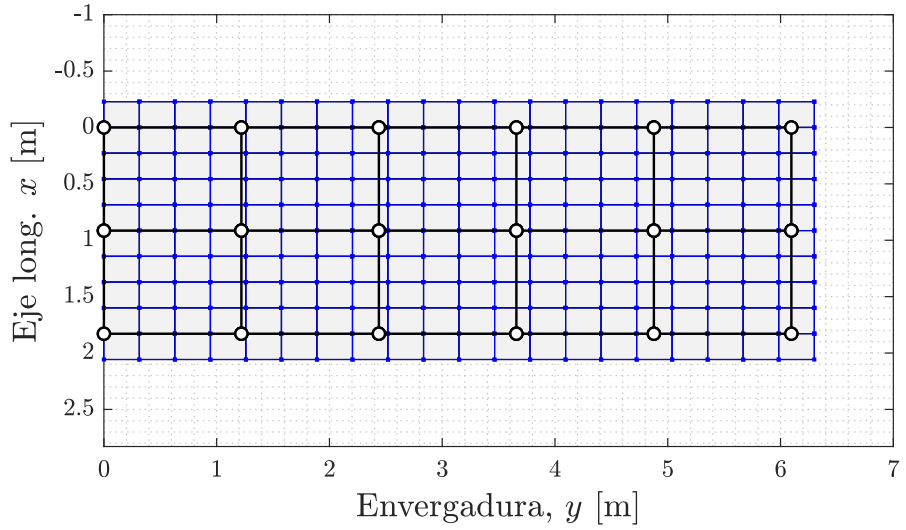


Figura 17: Superposición del modelo estructural (negro) y de paneles (azul)

Además, es especialmente importante comentar que la discretización de paneles es tal que la resolución de la malla sea suficiente como para que no varíen los resultados pero teniendo en cuenta el coste computacional.

3.1.1. Frecuencias naturales y modos propios

Definida la geometría en el programa, se pueden calcular las matrices de masa y rigidez como se ha visto en la sección 2.2, dando paso a la resolución del problema de vibraciones libres (91), siendo $\mathbf{u}$ el vector de grados de libertad y $\mathbf{M}$ y $\mathbf{K}$ las matrices de masa y rigidez del ala respectivamente [15].

$$\mathbf{M\ddot{u}} + \mathbf{Ku} = \mathbf{0} \tag{91}$$

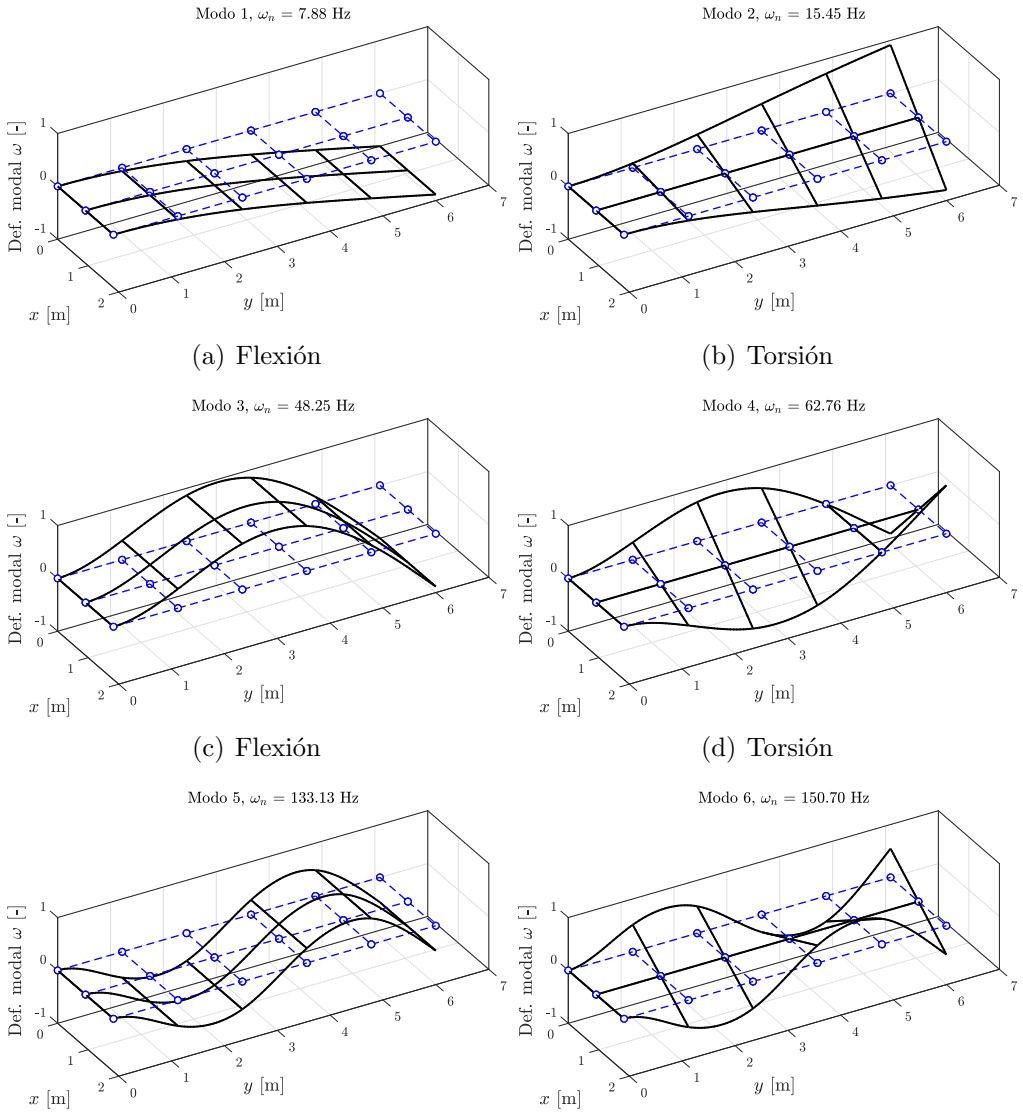
Proponiendo una solución armónica del tipo $u(t) = \bar{u}\ e^{\lambda t}$, se tiene[16]:

$$\left(\mathbf{K} + \lambda^2\mathbf{M}\right)\bar{\mathbf{u}}\ e^{\lambda t} = \mathbf{0} \quad \leftrightarrow \quad \left(\mathbf{K} + \lambda^2\mathbf{M}\right)\bar{\mathbf{u}} = \mathbf{0} \tag{92}$$

Obteniendo una expresión que corresponde a un problema generalizado de valores y vectores propios, de cuya resolución (debe cumplirse que $\det\left(\mathbf{K} + \lambda^2\mathbf{M}\right) = 0$) se obtienen las frecuencias naturales (autovalores) y los modos de vibración de la estructura (autovectores).

[15] Se emplea la nomenclatura de Newton. Así para una variable genérica Ω; $\frac{\mathrm{d}\Omega}{\mathrm{d}t} \rightarrow \dot{\Omega}$, $\frac{\mathrm{d}^2\Omega}{\mathrm{d}t^2} \rightarrow \ddot{\Omega}$.
[16] Donde $\lambda \in \mathbb{C}$, y $\bar{u}$ denotará la amplitud del movimiento vibratorio.

De esta forma, se obtienen tanto las frecuencias naturales de la estructura como la deformada de los modos de vibración. En la Figura 18 se muestra la deformada de los 6 primeros modos, que se corresponden con modos de flexión y torsión pura.



(a) Flexión (b) Torsión

(c) Flexión (d) Torsión

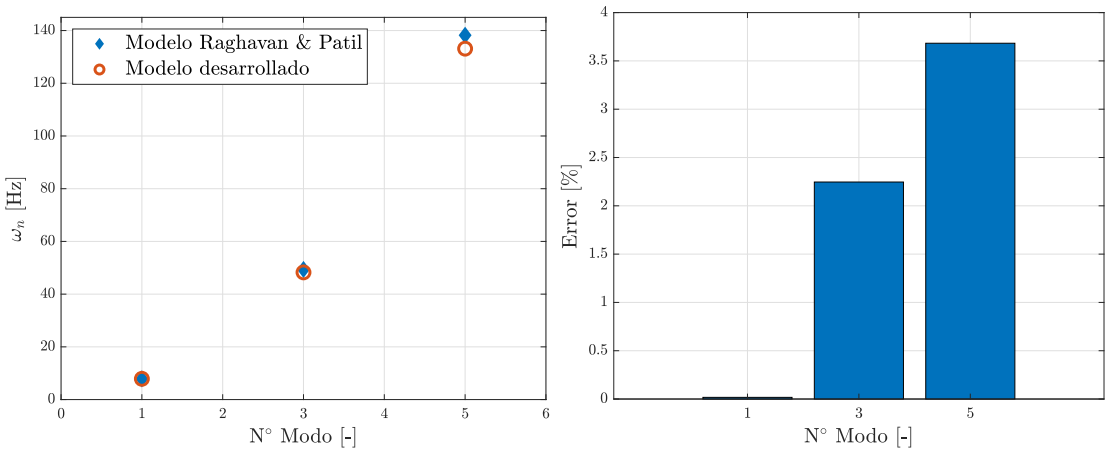
(e) Flexión (f) Torsión

Figura 18: Primeros 6 modos de vibración de la estructura analizada

En este punto ya se puede comparar con los resultados de [15]. En la Tabla 2 y en la Figura 19 se presenta la comparativa de resultados entre los 2 modelos. Así, se puede determinar que el modelo reproduce de forma muy aproximada el comportamiento del ala, aumentando el error a mayores frecuencias.

Modo	Frecuencia natural [Hz]		Error [%]
	Raghavan & Patil	**Modelo actual**	
1	7.88	7.88	0.017
3	49.36	48.25	2.25
5	138.22	133.13	3.68

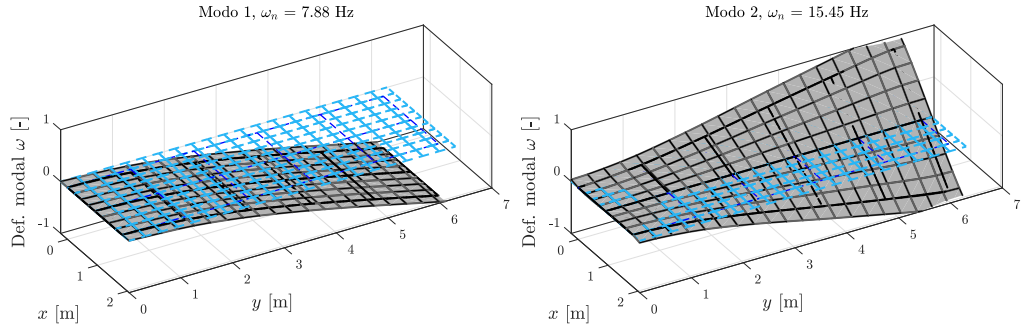
Tabla 2: Frecuencias naturales de los 3 primeros modos de flexión obtenidas en [15] y en el modelo del proyecto



(a) ω_n vs Nº Modo (b) Error

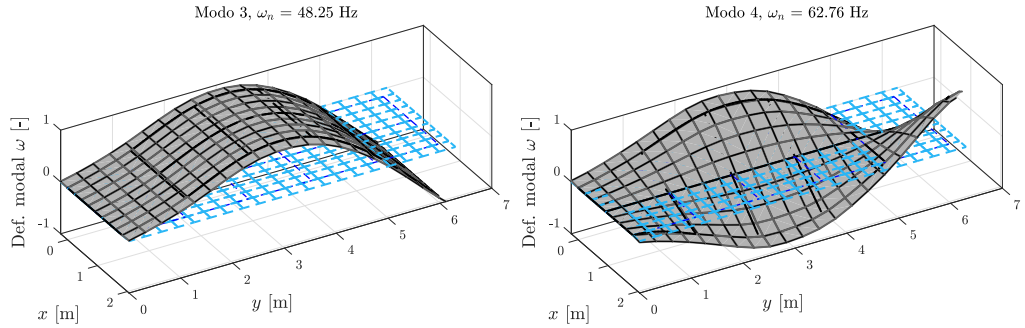
Figura 19: Comparativa gráfica del error en el cálculo de las frecuencias naturales entre [15] y el modelo desarrollado

Finalmente se comprueba que el modelo de acoplamiento mediante *splines* funciona correctamente, esto es, la interpolación a partir de la solución de desplazamientos de los nodos estructurales genera una deformada correcta de los paneles del modelo aerodinámico. Así, se muestran en la Figura 20 los 6 modos anteriores pero representando además la deformada de los paneles, donde se puede observar el correcto acoplamiento entre modelos.



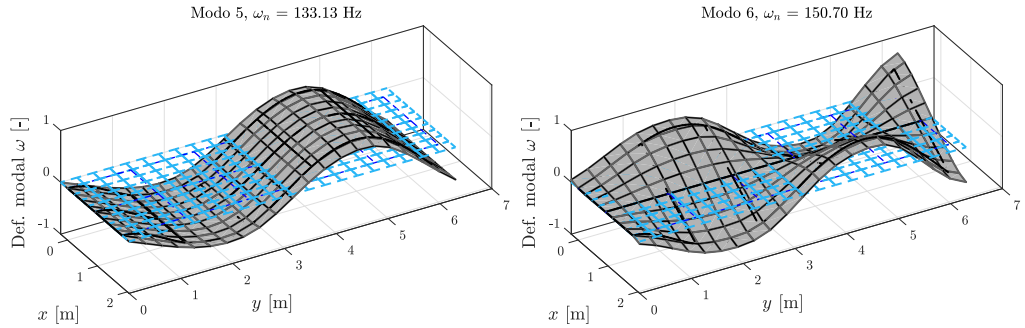
(a) Flexión

(b) Torsión



(c) Flexión

(d) Torsión



(e) Flexión

(f) Torsión

Figura 20: Representación conjunta de la estructura y los paneles del modelo aerodinámico de los 6 primeros modos de vibración del ala *Goland*

3.1.2. Aeroelasticidad estática: velocidad de divergencia

Además de la comparación en términos de frecuencias naturales y modos propios, se completa la validación del modelo con el cálculo de la velocidad de divergencia del ala ya que es un fenómeno independiente de la variable temporal. Así, siguiendo el desarrollo visto en el apartado 2.4 y a partir del cálculo de las matrices de acoplamiento fluido-estructura (2.3), se puede obtener la matriz $\mathbf{A}_s$ llegando, al igual que en el caso de vibraciones libres, a un problema de autovalores y autovectores, donde los autovalores son valores de presión dinámica y los autovectores los modos de divergencia del ala. Así, la presión de divergencia y, consecuentemente la velocidad de divergencia, es la mínima de los autovalores obtenidos. En la Tabla 3, se muestra el valor de dicha velocidad en condiciones a nivel del mar ($\rho_\infty = 1.225$ kg/m^3), donde se puede observar que el modelo se aproxima razonablemente bien al valor obtenido en [15], con un error de 1.61 %.

Parámetro	***Raghavan & Patil***	**Modelo actual**	**Error**
Velocidad de divergencia	252.39 m/s	256.47 m/s	1.61 %

Tabla 3: Velocidad de divergencia obtenida en [15] y en el modelo del proyecto

Por último, para finalizar esta sección y pasar al modelado de la aeronave completa, se muestra en la Figura 21 el modo de divergencia del ala, donde se visualiza claramente el acoplamiento existente entre flexión y torsión.

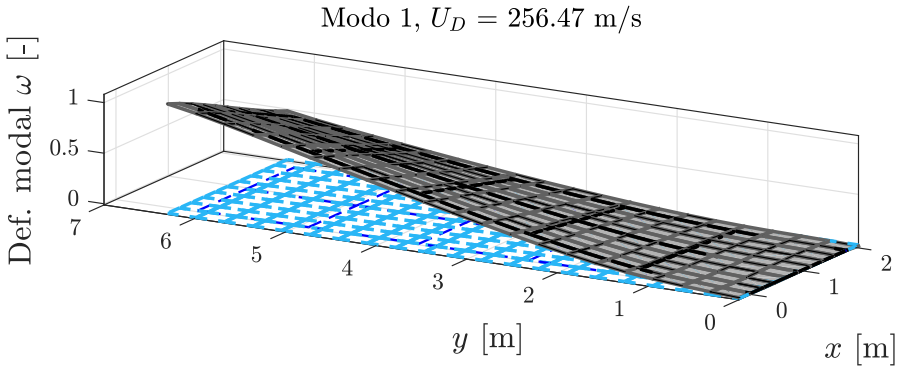


Figura 21: Modo de divergencia obtenido

3.2. Aplicación en un caso real: *Cessna 172 Skyhawk*

3.2.1. Descripción y características de la aeronave

Tras la validación del modelo desarrollado, se busca ponerlo en práctica modelando un avión completo, objetivo principal del trabajo. Para ello, se escoge una aeronave real para implementar su geometría y características estructurales en el programa.

En cuanto a la elección de avión a modelar, hay que tener en cuenta las limitaciones de los modelos utilizados, sobre todo se debe prestar especial atención al aerodinámico, ya que en su desarrollo se suponía flujo no viscoso, irrotacional e incompresible. De esta forma, si se pretende que el modelo obtenga resultados realistas, la aeronave a escoger debe ser de baja velocidad, donde estos efectos sean despreciables.

Así, se elige la *Cessna 172 Skyhawk*, avión de ala alta con capacidad para 4 personas, distinguido por ser la aeronave más fabricada de la historia[17] y, probablemente, una de las aeronaves de entretenimiento más populares del mundo. Se caracteriza por su diseño de ala alta, su estabilidad ante viento de baja velocidad y su baja velocidad de entrada en pérdida (85 km/h), por lo que sus usos principales son el entrenamiento, el recreo, el transporte ligero o las operaciones de búsqueda y rescate.

Figura 22: *Cessna 172 Skyhawk* [20]

[17]Los primeros modelos fueron entregados en 1956 y, desde entonces, la cifra de unidades fabricadas asciende a más de 43000 [20].

En cuanto a geometría y prestaciones principales, en la Tabla 4 se resumen los principales parámetros del avión mientras que en la Figura 24 se puede observar el plano 3 vistas del mismo con alguna de las dimensiones indicadas en la Tabla 4. Cabe destacar la máxima velocidad de vuelo, ya que, como se ha comentado previamente, para que se puedan obtener conclusiones realistas, se deben cumplir las hipótesis del modelo aerodinámico. Así, sabiendo que la velocidad del sonido a nivel del mar es de 343.20 m/s, el mach de vuelo del avión si se encuentra volando a máxima velocidad será de 0.24, por lo que se puede tomar como correcta la hipótesis de flujo incompresible.

Envergadura [m]	11
Cuerda en la raíz [m]	1.63
Alargamiento [-]	7.50
Longitud [m]	8.28
Altura [m]	2.72
Superficie alar [m^2]	16.17
Superficie del estabilizador horizontal [m^2]	3.35
Diámetro de la hélice [m]	1.90
OEW (*Overall Empty Weight*) [kg]	745
MTOW (*Maximum Take Off Weight*) [kg]	1111
Máxima velocidad (SL, IAS) [km/h]	302
Velocidad de crucero [km/h]	226
Techo de vuelo [m]	4115
Máximo alcance [km]	1074
Autonomía [h]	6.60

Tabla 4: Características principales de la aeronave de estudio [20]

Por otro lado, en relación a la estructura y los materiales utilizados, es un avión totalmente metálico en el que la construcción del fuselaje es convencional, formada por mamparos de chapa metálica, refuerzos y recubrimiento, conocida como estructura semimonocasco. Los elementos principales de la estructura son los largueros de carga delantero y trasero a los que se sujetan las alas, y otro mamparo con anclajes a los que se sujetan los montantes de las alas.

Las alas, que contienen los depósitos de combustible, están constituidas por 2 largueros, 3 larguerillos y 18 costillas conformadas en chapa metálica, estando toda la estructura recubierta de chapa de aluminio. Cabe destacar que el larguero posterior solo se sujeta al fuselaje y no llega hasta la punta del ala ya que a él se sujetan los alerones y los flaps.

El empenaje consiste en un estabilizador vertical convencional , un timón de dirección, estabilizador horizontal y un timón de profundidad. Su construcción es similar a la del resto del avión.

Todos estos aspectos se pueden ver en la Figura 23, donde se muestra de forma esquemática la estructura interna del avión.

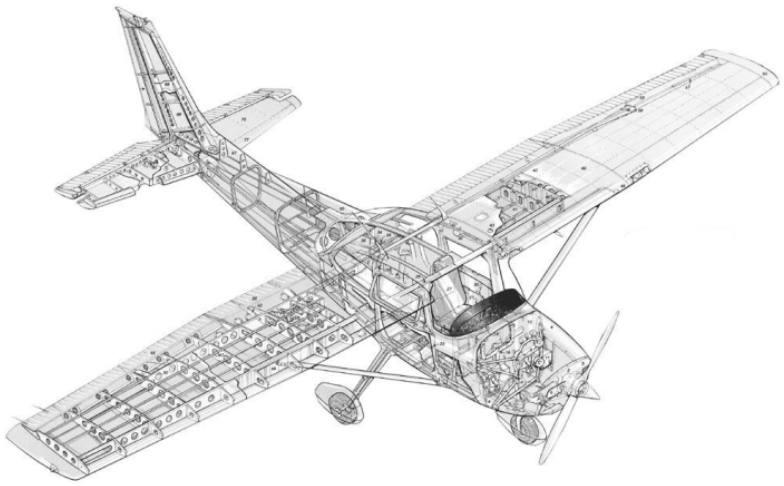


Figura 23: Esquema de la estructura de la *Cessna 172 Skyhawk* [21]

Por último, en cuanto a la planta motriz, el avión se propulsa a partir de un motor de pistón de 4 cilindros *Lycoming IO-360-L2A* con una hélice de 2 palas, que proporciona 120 KW de potencia.

3.2.2. Implementación en el modelo

Para la implementación de la aeronave en el modelo, son necesarias las dimensiones de las superficies alares así como de la longitud del fuselaje, además de datos acerca de la estructura de ala, estabilizador horizontal y fuselaje.

En primer lugar, para el modelo aerodinámico de paneles, solamente se necesita de las dimensiones en planta de las superficies alares, incluyendo elevadores, alerones y flaps. A partir de la Figura 24 se pueden definir las coordenadas exteriores de las 2 superficies para llevar a cabo la discretización en paneles. En el anexo B.2 se detallan todas las funciones utilizadas para la generación de la malla de paneles, así como el cálculo de la matriz **H**, aunque a continuación se describen algunas de ellas.

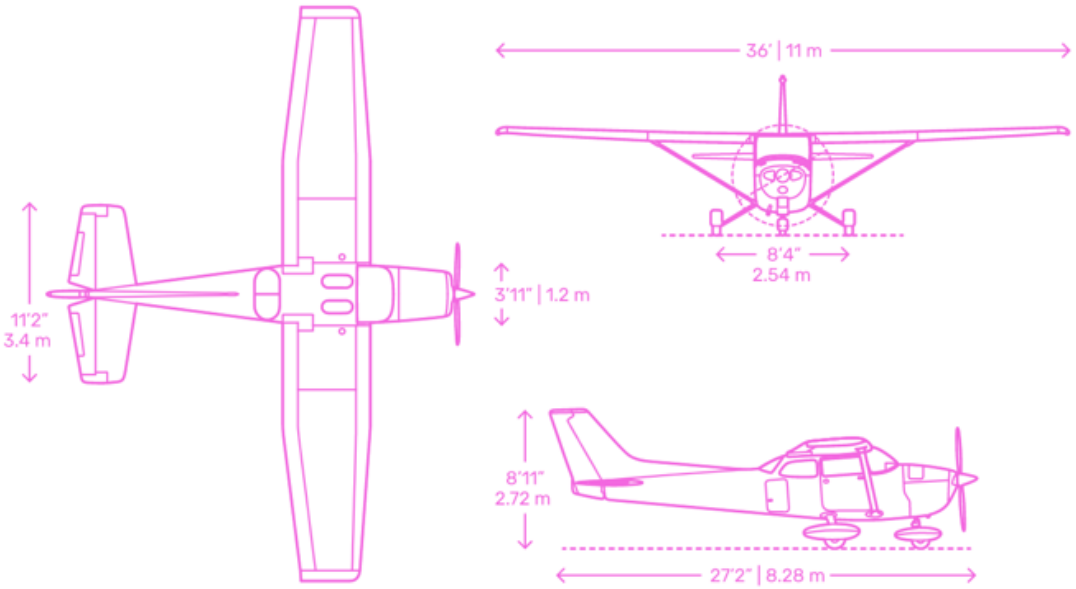


Figura 24: Plano 3 vistas de la aeronave [22]

Para la generación de la malla se usa la función *GeometriaPaneles* (B.2.1), la cual genera una estructura de datos llamada *PANELES* con las coordenadas de las esquinas, de los puntos de control y de los centros aerodinámicos de cada panel, obtenidas a partir de las coordenadas de los vértices de las superficies de sustentación[18], y la discretización que se quiera en dirección x (*nx_vector*) e y (*ny_vector*). Además, en la estructura *PANELES* también se obtienen algunas matrices útiles para el cálculo de fuerzas aerodinámicas y necesarias para el problema aeroelástico, como pueden ser la matriz $\mathbf{R}$[19] o la $\mathbf{T}$[20].

Seguidamente, para el cálculo de la matriz de coeficientes de influencia aerodinámicos se sigue el procedimiento visto en 2.1.2 mediante la función *MatrizH* (B.2.2), la cual implementa *TorbellinoSemiInfinito* (B.2.4) y *TorbellinoHerradura* (B.2.3) para el cálculo de la velocidad vertical inducida por los torbellinos en herradura desarrollado en 2.1.1.

[18]Para la definición de coordenadas se utiliza la estructura de datos *COORD*, donde se definen tres elementos: la parte de cuerda constante del ala, la parte de cuerda variable y el estabilizador. Además, para diferenciar las superficies se utiliza el vector *SUPERFICIE*, donde el identificador 1 se utiliza para el ala y el 2 para el estabilizador.

[19]Matriz de 1s y 0s, de tamaño $ny \times N_P$ (discretización en dirección y = número de filas de paneles), en la que los 1s aparecen en los índices de los paneles que se encuentran en cada fila.

[20]Descrita en 2.4.

Por otro lado, para el modelo estructural, son necesarios otros datos de la aeronave además de la geometría, como pueden ser las propiedades de los materiales de fabricación y los elementos estructurales de ala, estabilizador y fuselaje, mencionados en la sección anterior.

Primeramente, de forma similar a la función *GeometriaPaneles* del modelo aerodinámico, se utiliza *GeometriaEstructura* (B.3.1), la cual genera la discretización en nodos y elementos del ala y estabilizador horizontal a partir de sus coordenadas exteriores y la división en eje x (número de largueros - 1) e y (número de costillas). Una vez se tienen estas 2 estructuras de datos, la creación del fuselaje se hace a partir de la función *GeometriaAvion* (B.3.2), que, a su vez, lo acopla con las estructuras del ala y estabilizador generando una única estructura de datos llamada *ESTRUCTURA*[21]. Así, se obtienen los siguientes parámetros, necesarios para el cálculo de las matrices de masa y rigidez:

- Número total de nodos (N_n), de elementos (N_e), y de grados de libertad ($N_n \times 3$)
- Matriz con las coordenadas de los nodos ($N_n \times 3$)
- Matriz de grados de libertad ($N_n \times 3$)
- Matriz de conectividad ($N_e \times 2$)

Además, es necesario definir en este punto las propiedades de los materiales utilizados, concretamente módulo elástico E y módulo a torsión G, para el cálculo de la matriz de rigidez. Dado que la estructura es de aluminio, se tiene $E = 6.3 \cdot 10^{10}$ y $G = 2.63 \cdot 10^{10}$, pero se podría definir cualquier otro material. También se deben definir las propiedades másicas de ala y estabilizador en forma de masa por unidad de longitud así como el peso del fuselaje, para el cálculo de la matriz de masa.

Así, ya se pueden obtener las matrices de masa, $\mathbf{M}$, y rigidez, $\mathbf{K}$, así como las de acoplamiento fluido-estructura, $\mathbf{D_u}$, $\mathbf{D_v}$ y $\mathbf{D_z}$, paso previo a la obtención de resultados. Para ello, se desarrollan 3 funciones más:

- *RigidezEstructura* (B.3.3): obtiene la matriz $\mathbf{K}$ a partir de *ESTRUCTURA*. Para ello, implementa la función *RigidezElemento*, la cual calcula la matriz de rigidez a nivel local para luego expandirla en los ejes globales. Esta función requiere de las rigideces a flexión y torsión de cada elemento, por lo que se comprueba, en el caso de ala y estabilizador, si el elemento es parte de un

[21] Cabe destacar, que el fuselaje se introduce en esta función dividido en 3 partes: nariz, hiato y cola, y se puede discretizar de forma independiente cada una de ellas.

larguero o de una costilla para así calcular los momentos de inercia I y J, tal como se ha visto en 2.2.2. Para los elementos del fuselaje, se considera una sección circular de pared delgada y radio variable para tener en cuenta las dimensiones reales[22] de la aeronave.

- *MatrizMasa* (B.4.2): calcula la matriz $\mathbf{M}$ a partir de *ESTRUCTURA* y *PANELES* aplicando el desarrollo 2.3.2. Al igual que con la rigidez, se distingue el cálculo para ala y estabilizador, y para el fuselaje. En el caso de ala y estabilizador, a partir de la masa distribuida, se calcula la masa total y se reparte entre los nodos en función de los elementos que conectan, es decir, los nodos centrales, que conectan 4 elementos, tendrán el doble de masa que los de las esquinas, que conectan 2. Por otro lado, para el fuselaje, se distribuye la masa total en función de la localización de los nodos para mantener la posición real del centro de gravedad mediante la función *dist_fuselaje*.

- *MatricesAcoplamiento_Dinamicas* (B.4.3): obtiene las matrices de acoplamiento a partir de las funciones *MatrizRs*, *Spline*, *SplineFunction* y *DSplineFunction* que implementan el modelo de acoplamiento por *splines* visto en 2.3[23].

Finalmente, en la Figura 25 se muestra la vista en planta de los 2 modelos en *Matlab* para el avión de estudio que se utilizará en las secciones siguientes para la obtención de resultados[24].

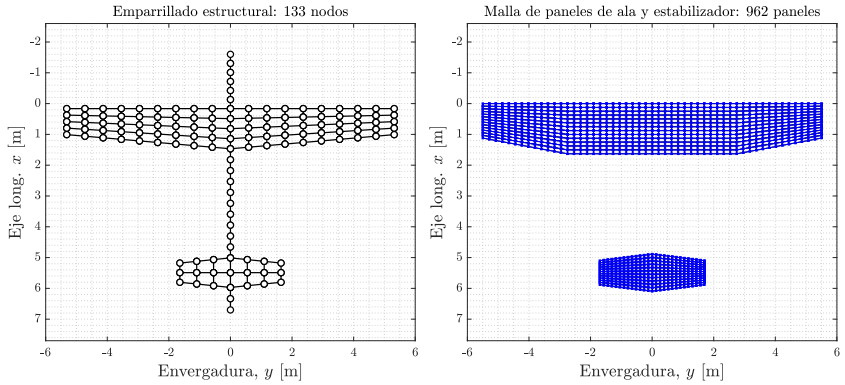


Figura 25: Representación del modelo estructural y aerodinámico de la *Cessna 172 Skyhawk* en *Matlab*

[22]Se utiliza la función *dim_fuselaje*, la cual divide el fuselaje en 3 secciones: de la nariz a la cabina aumenta el diámetro desde 0.5 a 1.2 m, la sección de la cabina permanece constante, y de esta hasta la cola desciende linealmente a 0.3 m.

[23]*MatrizRs* $\equiv \mathbf{R}_\sigma$, *Spline* $\equiv \mathbf{B}_\sigma$, *SplineFunction* $\equiv \boldsymbol{\psi}_{\boldsymbol{\sigma}}$ y *DSplineFunction* $\equiv \frac{\partial \boldsymbol{\psi}_{\boldsymbol{\sigma}}}{\partial x}$ (2.3.1).

[24]El número de paneles depende de la discretización elegida, la cual se hace teniendo en cuenta el *trade-off* entre coste computacional y resolución de la malla.

3.2.3. Vibraciones libres: frecuencias naturales y modos propios de vibración

Partiendo de las mismas ecuaciones vistas en 3.1.1, se obtienen las frecuencias y modos propios de vibración para la aeronave de estudio. En la Figura 26 se representan los 6 primeros modos de vibración con las frecuencias a las que se producen.

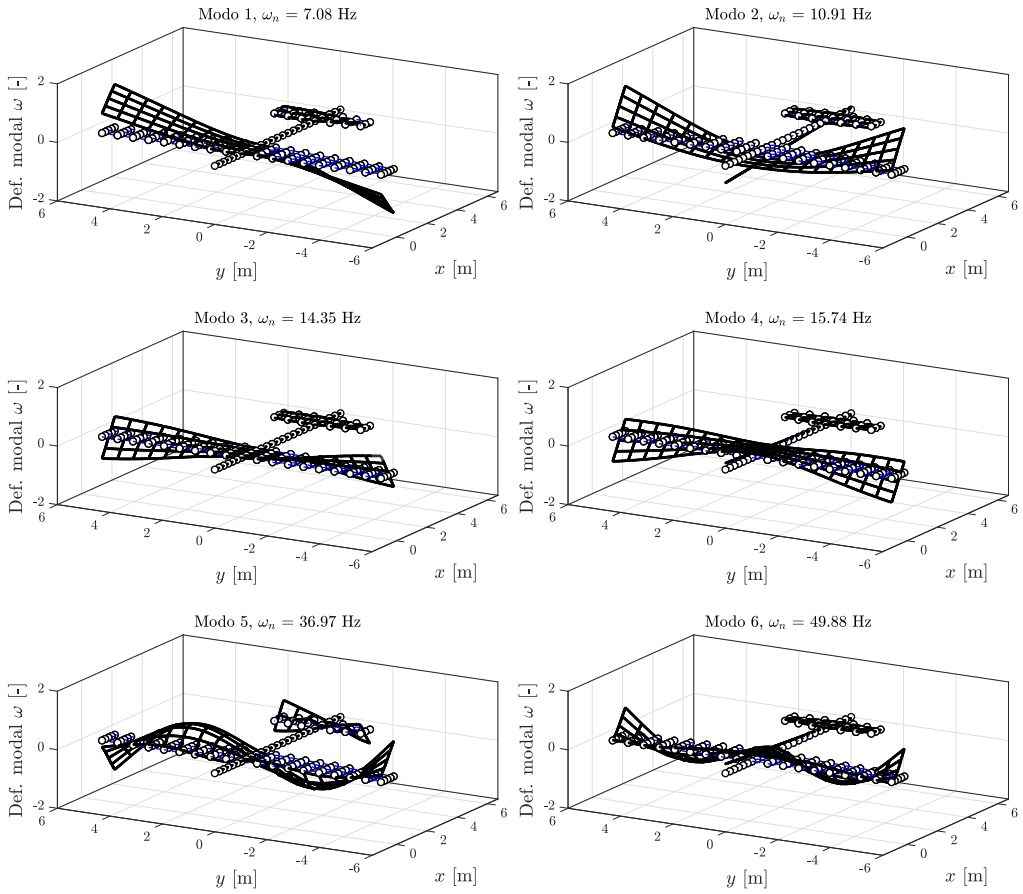


Figura 26: Primeros 6 modos de vibración de la estructura del avión completo

A diferencia del caso de validación, en el que solo teníamos un ala y se distinguían claramente los modos de torsión y flexión, para la aeronave completa existe acoplamiento de estos 2 movimientos en todos los modos, obteniendo configuraciones mucho más complejas. Además, es especialmente importante destacar la deformación del fuselaje, ya que es de órdenes de magnitud inferior a las superficies sustentadoras dado que su rigidez es muy superior.

Imponiéndose misma rigidez de elemento en todas las partes del modelo, cosa que no es realista pero sirve para comprender mejor el problema, se observa cómo las deformaciones en fuselaje, ala y estabilizador son del mismo orden de magnitud y la forma de los modos se simplifica, apareciendo modos de flexión y torsión puros[25] (Figura 27).

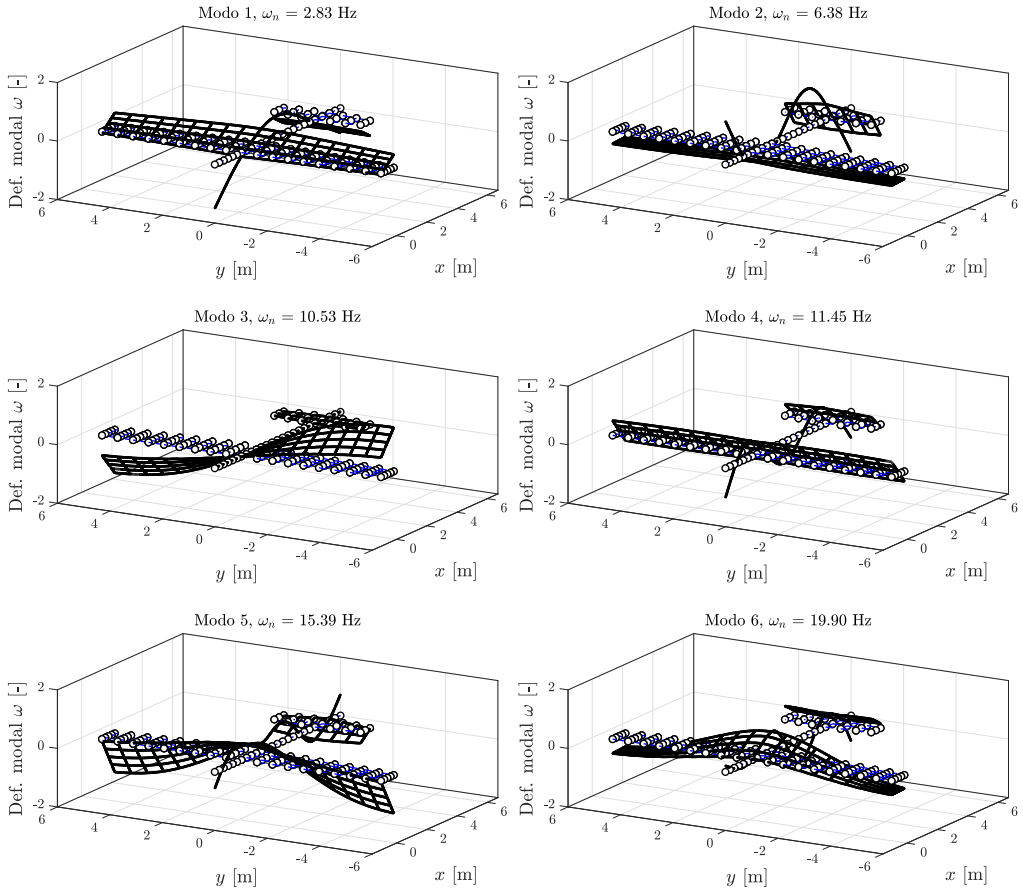


Figura 27: Primeros 6 modos de vibración de la estructura del avión completo con elementos de igual rigidez

Por último, al igual que en el caso del ala *Goland*, se busca comprobar que el modelo de *splines* funciona correctamente y la deformada de los paneles contiene a la de los nodos de la estructura. A la vista de la Figura 28, donde se muestran los mismos modos que en la Figura 26 pero añadiendo los paneles del modelo aerodinámico, se observa que la deformación de los paneles coincide con la de la estructura, por lo que la interpolación es correcta.

[25]Las frecuencias naturales son bajas debido a que la rigidez de los elementos supuesta lo es.

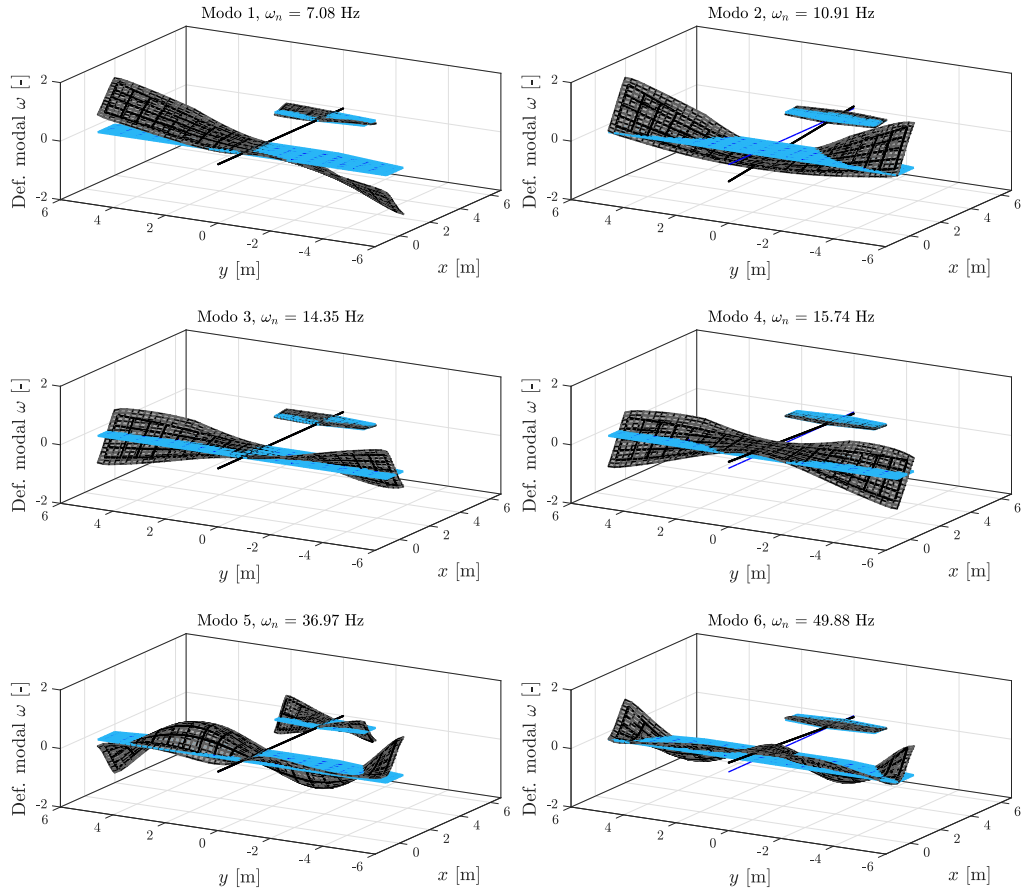


Figura 28: Representación conjunta de la estructura y los paneles del modelo aerodinámico de los 6 primeros modos de vibración del avión completo

3.2.4. Aeroelasticidad estática: velocidad de divergencia

Para abordar el problema estático, como se ha descrito en 2.4, se calculan, en primer lugar, las matrices de acoplamiento $\mathbf{D}_z$ y $\mathbf{D}_u$ y, con ellas, la matriz $\mathbf{A}_s$. Así, se puede resolver el problema de valores y vectores propios de la ecuación 71, con la que se obtiene una velocidad de divergencia de 174.37 m/s, siendo el modo de divergencia el representado en la Figura 29. De nuevo se observa cómo la divergencia de la aeronave aparece de forma acoplada por la torsión y flexión del ala.

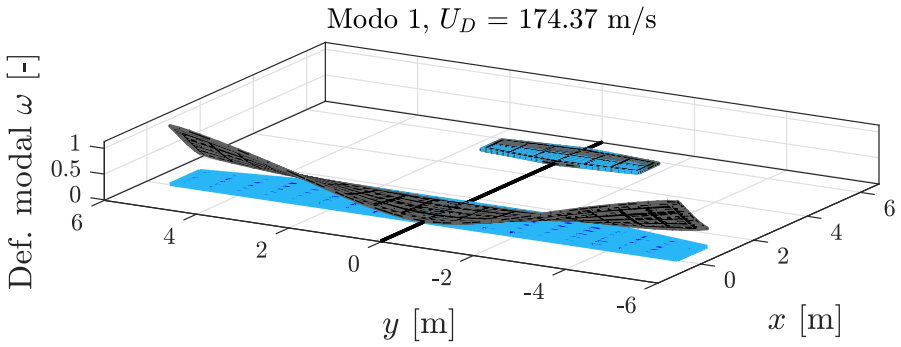
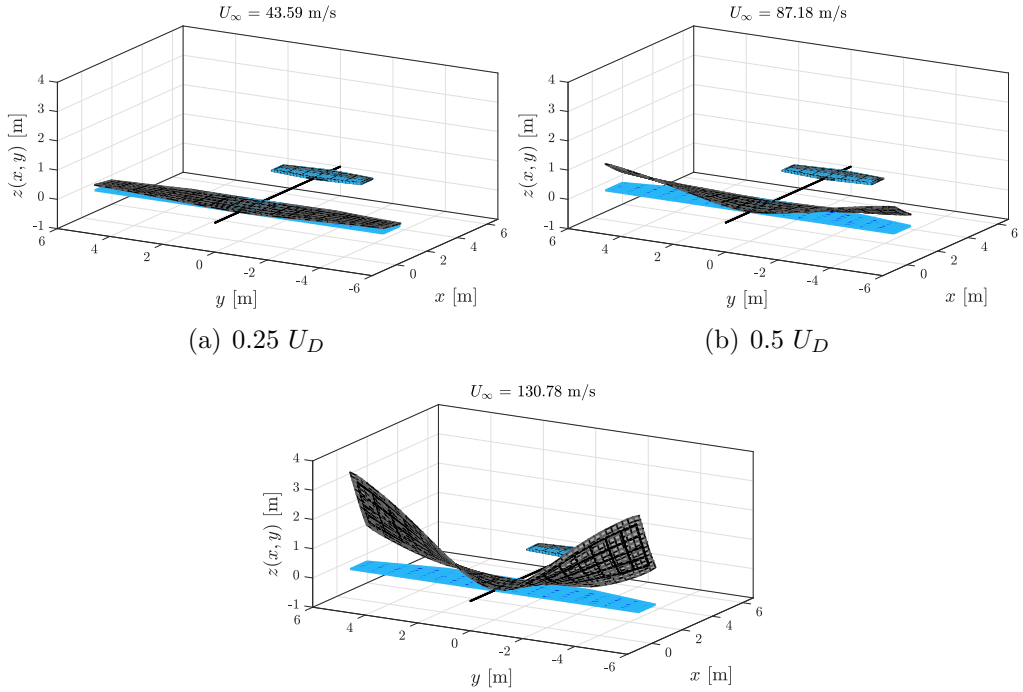


Figura 29: Modo de divergencia del avión completo

Para completar el análisis estático, se muestra en la Figura 30 la solución estática del avión para diferentes velocidades de vuelo hasta aproximarse a la divergencia, donde se observa el aumento de la deformación hasta la inestabilidad y el límite estructural.



(a) 0.25 U_D (b) 0.5 U_D

(c) 0.75 U_D

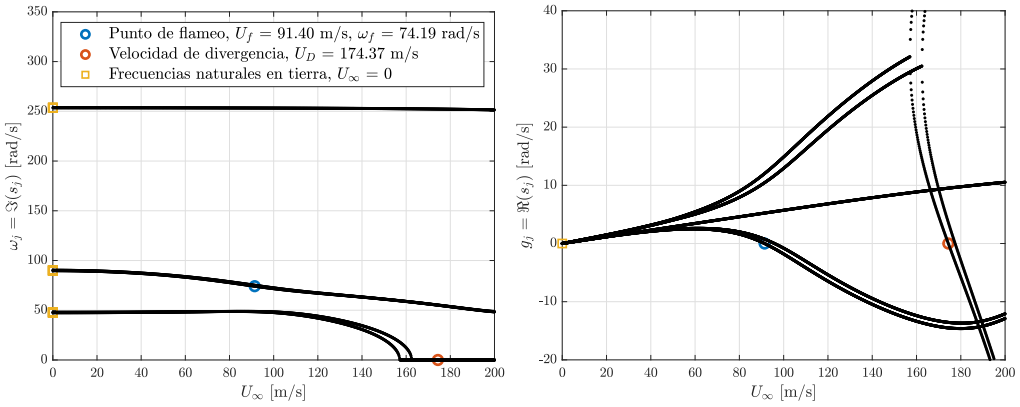
Figura 30: Solución estática para diferentes velocidades de vuelo

3.2.5. Aeroelasticidad dinámica: velocidad y frecuencia de flameo

Por último, se procede al análisis del problema dinámico. Partiendo de las matrices de acoplamiento dinámicas $\mathbf{D}_z$, $\mathbf{D}_u$ y $\mathbf{D}_v$, se pueden calcular las matrices $\mathbf{A}$, $\mathbf{B}$ y $\mathbf{C}$ de la ecuación 81 y resolver el problema en su versión *espacio-estado* como se detalla en 2.4.

De esta forma, se pueden obtener las curvas de flameo (Figura 31), donde se representa la parte real (amortiguamiento) e imaginaria (frecuencia) de los autovalores del problema para cada velocidad de vuelo, es decir, $g_j(U_\infty)$ vs U_∞ y $\omega_j(U_\infty)$ vs U_∞. Así, cuando $U_\infty = 0$, se obtiene la solución estática, por tanto el amortiguamiento es nulo, y aparecen las frecuencias naturales calculadas en el problema de vibraciones libres, aunque con pequeñas diferencias ya que en este caso se tiene en cuenta el efecto de la masa aparente del aire alrededor del perfil, $\rho_\infty \mathbf{C}\ddot{\mathbf{u}}$, que afecta a la matriz de masa global.

Por otro lado, al aumentar la velocidad de vuelo, los autovalores varían teniendo ya parte real e imaginaria no nulas, hasta llegar a los límites de estabilidad: punto de flameo ($g_j = 0$, $\omega_j > 0$) o divergencia ($g_j = 0$, $\omega_j = 0$). La aparición de uno u otro a una menor velocidad de vuelo solo depende de las características másicas y estructurales de la aeronave, pudiendo aparecer antes el flameo o viceversa, aunque normalmente el fenómeno de flameo surge a menores velocidades que la inestabilidad por divergencia.



(a) Frecuencias (b) Amortiguamientos

Figura 31: Curvas de flameo de la aeronave completa

En la figura anterior, se muestran las curvas de flameo de los 5 modos de menor frecuencia para el caso de estudio, en las que se observa lo comentado anteriormente:

- Las frecuencias naturales difieren a las del problema de vibraciones libres dado que la matriz de masa en este caso es $\mathbf{M}_{eq} = \mathbf{M} - \rho_\infty \mathbf{C}$ y, además, porque para la obtención de las curvas se han restringido los grados de libertad de uno de los nodos del fuselaje eliminando así los modos de sólido rígido, por lo que las frecuencias naturales cambian.
- La inestabilidad por flameo se da antes que la divergencia de la aeronave, cosa que ocurre normalmente en aeronaves reales. Además, comparando esta velocidad obtenida (91.40 m/s) con el de la máxima velocidad de vuelo dada por el fabricante (83.89 m/s), se observa la coherencia de los resultados, ya que la aeronave nunca llega a la velocidad en que se vuelve inestable.
- La velocidad de divergencia coincide exactamente con la calculada en el apartado anterior, por lo que el desarrollo del problema dinámico es consistente con el estático.
- Aparecen modos muy similares, esto se debe a que, por la simetría del problema, existen modos de deformación simétricos y antisimétricos. En este caso, tanto el flameo como la divergencia se dan ligeramente antes en los modos simétricos.

Por último, al igual que en el problema estático, se representa en la Figura 32 el modo de flameo del avión, en la que se puede ver que es el ala la que alcanza la inestabilidad dinámica.

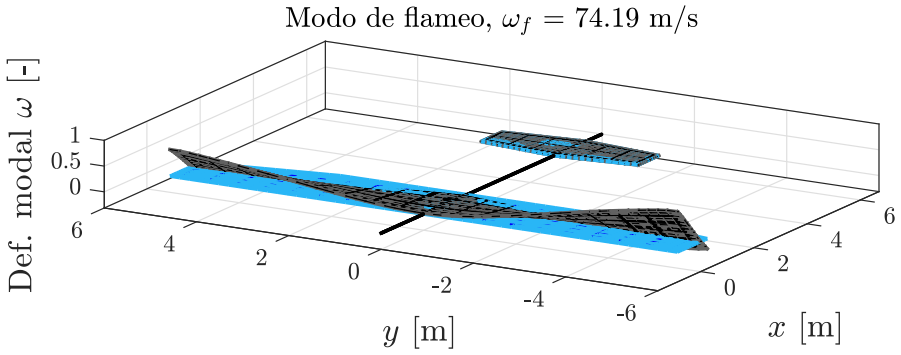


Figura 32: Modo de flameo del avión completo

4. Conclusiones

4.1. Objetivos alcanzados y valoración

En el presenta trabajo se ha abordado el estudio de la aeroelasticidad en el ámbito aeronáutico de forma numérica mediante el modelado estructural y aerodinámico de una aeronave completa. Dicho estudio ha permitido obtener las siguientes conclusiones.

En primer lugar, en cuanto a los objetivos definidos al inicio del proyecto, se ha conseguido desarrollar un modelo matemático capaz de simular el comportamiento estructural de una aeronave así como su acoplamiento con el modelo aerodinámico de paneles mediante la interpolación con *splines* de superficie, método muy utilizado en diversas aplicaciones industriales como es el caso de los códigos de elementos finitos.

Con ello, se ha realizado su validación a partir de resultados contrastados en otras investigaciones y definido un caso práctico de aplicación para describir el uso del modelo así como de los resultados que se pueden obtener a partir de él. De esta forma, se ha analizado tanto el problema de vibraciones libres como el de aeroelasticidad, estático y dinámico, consiguiendo una mejor compresión de estos y obteniendo resultados preliminares útiles en fases de diseño, aplicados al ámbito aeroespacial, concretamente en aeronaves de baja velocidad.

Por último, en cuanto a aspectos destacables de los modelos desarrollados en el proyecto, es especialmente importante la influencia de parámetros como el número de paneles del modelo aerodinámico y el número de nodos y elementos del estructural. Así, el número de paneles es elección del usuario e influye en la resolución de la malla de paneles, cosa que afecta a la precisión de los resultados y al coste computacional. Como se ha comentado, es necesario tener en cuenta el *trade-off* entre estos 2 aspectos. Por otro lado, el número de nodos y elementos viene definido por la estructura a analizar así como la distribución de masa y la geometría de esta. Así, a mayor número de elementos estructurales, mayor número de elementos en el modelo y mayor rigidez de la estructura.

4.2. Implicaciones de los resultados obtenidos

Una vez comentadas las principales conclusiones, se pretende ver el alcance de los resultados obtenidos con la realización del proyecto.

Así, el desarrollo del modelo estructural de elementos finitos en plano horizontal y su acoplamiento con el aerodinámico de paneles permite la aplicación al caso de un avión completo, cosa que no puede realizarse con la suposición de eje elástico, típica en el modelado de superficies de sustentación. Esto supone una importante ventaja ya que ya no es necesario suponer una rigidez concentrada en un eje, sino que permite distribuir la rigidez de las distintas partes de la aeronave de forma realista.

Gracias a este aspecto, se puede obtener un conocimiento más amplio del comportamiento aeroelástico de una aeronave completa en su envolvente de vuelo, así como resultados preliminares importantes en diseño como son las velocidades de divergencia y flameo, o las frecuencias naturales de vibración, y su influencia con las distribuciones de masa y rigidez de la aeronave, cosa que puede suponer una mejora sustancial en la optimización del proceso de diseño de aeronaves.

4.3. Trabajos futuros

Por último, se podrían introducir una serie de trabajos a abordar en el futuro a partir del estudio realizado con el objetivo de ampliar el alcance del mismo.

- En primer lugar, en cuanto al modelo aerodinámico se podría plantear la implementación del *Vortex Lattice Method* no-estacionario con el objetivo de completar el problema aeroelástico dinámico al tener en cuenta fenómenos no estacionarios. Además, un avance importante sería la implementación del *Doublet Lattice Method*, modelo aerodinámico con utilidad industrial que, al cambiar torbellinos por dobletes, permite simular efectos de compresibilidad, lo que mejoraría enormemente las limitaciones del modelo actual.

- Por otro lado, en cuanto al modelo estructural, el modelo de elementos finitos estructurales se podría mejorar implementando elementos de *Timoshenko*, que se ajustan más al comportamiento real de vigas ya que no consideran la hipótesis de perpendicularidad de las secciones planas perpendiculares al eje neutro tras la deformación de la viga. Además, se podría mejorar el modelado de la distribución de masa a partir de datos estructurales reales.

- Por último, además del modelado en plano horizontal, la implementación de superficies verticales, como el estabilizador vertical, y dispositivos móviles, como flaps, elevadores y alerones, implicaría una mejora sustancial del modelo al poder considerar efectos en los 3 ejes espaciales mediante maniobras de alabeo, cabeceo y guiñada del avión.

5. Pliego de condiciones

5.1. Objeto

El pliego de condiciones tiene como objetivo establecer las especificaciones y requisitos necesarios para el correcto desarrollo del presente proyecto fin de máster. Dado que en la totalidad del tiempo empleado en la realización del trabajo se ha hecho uso del ordenador, es objeto del pliego de condiciones asegurar el uso adecuado de este con tal de prevenir los riesgos que esto conlleva. Así, se toman las medidas descritas según el *Real Decreto 488/1997, de 14 de abril, sobre disposiciones mínimas de seguridad y salud relativas al trabajo con equipos que incluyen pantallas de visualización.*

5.2. Condiciones de *software*

Por último, además de las medidas preventivas en el uso del ordenador, comentadas en el objeto de este pliego de condiciones, es necesario definir las condiciones del *software* utilizado en el proyecto. De esta manera, se listan a continuación los diferentes *softwares* utilizados durante el trabajo, describiendo su utilización y sus requerimientos:

- ***Microsoft Teams***
  - Uso: reuniones no presenciales con el tutor del proyecto.
  - Condiciones: licencia de *Microsoft 365*.
- ***Matlab***:
  - Uso: desarrollo numérico del modelo y post-proceso de resultados.
  - Condiciones: licencia de *MathWorks* para *Matlab*.
- ***Excel***:
  - Uso: recopilación de datos de la aeronave implementada en el modelo.
  - Condiciones: licencia de *Microsoft 365*.
- ***Overleaf***:
  - Uso: redacción de la memoria del trabajo mediante el sistema de composición de textos LaTeX.
  - Condiciones: uso libre desde el navegador, se ejecuta *online*.

- ***PowerPoint***:
  - Uso: creación de la presentación final del trabajo.
  - Condiciones: licencia de *Microsoft 365*.

6. Presupuesto

6.1. Introducción

El propósito del presente apartado es detallar el presupuesto necesario para llevar a cabo el proyecto. Para ello, con el objetivo de obtener una adecuada valoración de los costes del trabajo, se ha decidido dividir en varias etapas o fases, cuyos costes se detallan individualmente formando los presupuestos parciales. De esta forma, las fases del proyecto son:

- Fase I. Documentación y revisión bibliográfica
- Fase II. Desarrollo del modelo numérico. Programación en *Matlab*
- Fase III. Validación y análisis de resultados
- Fase IV. Redacción de la memoria

Por otro lado, se definen los diferentes conceptos imputables en los costes del proyecto:

- Personal: coste de los recursos humanos involucrados en el trabajo.
- Material inventariable: coste de los materiales y herramientas tanto físicas como software empleado durante la realización del trabajo.
- Material fungible: coste del material de oficina utilizado.
- Otros: costes no incluidos en los grupos anteriores.

En primer lugar, los costes debidos al personal se han obtenido de las tablas de retribuciones salariales del profesorado de la *Universitat Politècnica de València* y de las escalas salariales en concepto de ingenieros.

Concepto	**Salario Anual [€/año]**	**Salario Unitario [€/h]**
Contratado doctor	60000	33.90
Ingeniero junior	25000	14.12

Tabla 5: Salarios anuales ingeniero junior y contratado doctor

Estos costes representan el salario del alumno que realiza el proyecto, representado como ingeniero junior, así como del profesor responsable de la supervisión del trabajo, en este caso el tutor del mismo, representado como contratado doctor, cuya retribución anual es de 34121.74€, a lo que hay que añadir trienios y sexenios además del coste empresa (cotizaciones de la seguridad social, etc.). Con todo ello, se obtiene un salario anual promedio de 60000€. Por otro lado, para el ingeniero se ha tomado un salario medio en función de su categoría. Por último, comentar que para hallar el coste unitario por hora de trabajo se ha extraído del *BOE-A-2023-10117* la duración de la jornada laboral de 1770 horas, en vigor a partir del 1 de enero de 2023.

En cuanto al tiempo invertido de cada uno de ellos dependerá de las horas de tutorías o cuestiones así como de la preparación de la documentación y *software* necesarios, por lo que será variable dependiendo de las fases del proyecto.

Por otro lado, entre el material inventariable y fungible utilizado se considera el ordenador, la impresora y el material de oficina. En el caso del ordenador e impresora, atendiendo al consumo en electricidad y tinta (en el caso de la impresora) se considera un precio unitario de 0.40€/h y 12.00€/h respectivamente.

6.2. Presupuestos parciales

6.2.1. Fase I: Documentación

En esta primera etapa se considera el tiempo desde el inicio del trabajo hasta el fin de la revisión bibliográfica realizada para contextualizar el proyecto en el correcto marco teórico y comprender la problemática asociada al mismo.

Concepto	**Precio unitario [€/h]**	**Cantidad [h]**	**Coste total [€]**
Personal			**9083.05**
Contratado doctor	33.90	4.00	135.59
Ingeniero junior	14.12	60.00	847.46
Material inventariable			**28.00**
Ordenador	0.40	40.00	16.00
Impresora	12.00	1.00	12.00
Material fungible			**15.00**
Material de oficina			15.00
Total			**1026.05**

Tabla 6: Costes Fase I

6.2.2. Fase II: Desarrollo del modelo

En esta fase se considera todo el proceso de programación del modelo. En cuanto a los costes incluidos, cabe destacar el coste de *software*, para el cual se ha tomado el precio de licencia estándar de *Matlab* anual de 876€y se divide entre las horas anuales totales según el *BOE*, para obtener el coste unitario del programa.

Concepto	**Precio unitario [€/h]**	**Cantidad [h]**	**Coste total [€]**
Personal			**3333.33**
Contratado doctor	33.90	15.00	508.47
Ingeniero junior	14.12	200.00	2824.86
Material inventariable			**177.18**
Ordenador	0.40	200.00	80.00
Impresora	12.00	0.00	0.00
Software (*Matlab*)	0.49	200.00	97.18
Material fungible			**5.00**
Material de oficina			5.00
Total			**3515.51**

Tabla 7: Costes Fase II

6.2.3. Fase III: Validación y análisis de resultados

Una vez realizada la programación del modelo, se pasa al post-procesado de los resultados obtenidos y su validación. En esta etapa del proyecto se tiene en cuenta el tiempo invertido en la búsqueda de resultados de otros trabajos de investigación similares para comparar resultados, así como en el cálculo y post-proceso de los resultados finales del avión completo.

Concepto	**Precio unitario [€/h]**	**Cantidad [h]**	**Coste total [€]**
Personal			**1581.92**
Contratado doctor	33.90	5.00	169.49
Ingeniero junior	14.12	100.00	1412.43
Material inventariable			**61.72**
Ordenador	0.40	100.00	40.00
Impresora	12.00	1.00	12.00
Software (*Matlab*)	0.49	20.00	9.72
Material fungible			**5.00**
Material de oficina			5.00
Total			**1648.64**

Tabla 8: Costes Fase III

6.2.4. Fase IV: Redacción

Por último, durante esta última fase se ha elaborado la memoria del trabajo. Para ello, se ha recogido toda la documentación vista en la revisión bibliográfica inicial y se han ordenado todos los resultados para la correcta lectura y compresión del proyecto. Además, se ha elaborado la presentación del mismo para su posterior defensa.

Concepto	**Precio unitario [€/h]**	**Cantidad [h]**	**Coste total [€]**
Personal			**1401.13**
Contratado doctor	33.90	8.00	271.19
Ingeniero junior	14.12	80.00	1129.94
Material inventariable			**44.00**
Ordenador	0.40	80.00	40.00
Impresora	12.00	1.00	12.00
Material fungible			**10.00**
Material de oficina			10.00
Total			**1455.13**

Tabla 9: Costes Fase IV

6.3. Presupuesto total

A partir de los presupuestos parciales se obtiene el presupuesto de ejecución material (P.E.M.) del proyecto. A esto hay que aplicar un 21 % de IVA, un 13 % debido a gastos generales y un 8 % de beneficio industrial. Además, es necesario añadir el coste debido a la depreciación del ordenador debido a su funcionamiento. Para ello se ha tomado una vida útil de 7 años y una pérdida de valor lineal con el tiempo. Sabiendo que el valor inicial del mismo es de 900€, la depreciación anual es de 128.57€, con lo que en las 420 horas de funcionamiento, la pérdida de valor es de 5.28€.

Con todo ello se obtiene el presupuesto total del estudio realizado.

Fase	**Descripción**	**Importe [€]**
I	Documentación	1026.05
II	Programación	3515.51
III	Validación y análisis de resultados	1648.64
IV	Redacción	1455.13
P.E.M.		**7645.33**
Amortización del ordenador		5.28
8 % Beneficio industrial		611.63
13 % Gastos generales		993.89
21 % IVA		1605.52
TOTAL		**10861.65**

Tabla 10: Presupuesto total

El presupuesto total asciende a la expresada cantidad de DIEZ MIL OCHOCIENTOS SESENTA Y UN EUROS CON SESENTA Y CINCO CÉNTIMOS.

Valencia, 11 de julio de 2023

Referencias

[1] Aerospace Engineering Blog. *Aeroelasticity, composites and the Grumman X-29*. 2021. URL: https://aerospaceengineeringblog.com/aeroelasticity-composites-and-the-grumman-x-29/ (visitado 03-06-2023).

[2] Dr. Mario Lázaro. *Tema 1. Aerodinámica Estacionaria de Perfiles (Fundamentos de Aerodinámica Estacionaria, Método de los Paneles Estacionario)*. Dto. Mecánica de los Medios Continuos y Teoría de Estructuras.

[3] Dr. Mario Lázaro. *Tema 1. Aerodinámica No-Estacionaria de Perfiles (Fundamentos de Aerodinámica No-Estacionaria, Método de los Paneles No-Estacionario)*. Dto. Mecánica de los Medios Continuos y Teoría de Estructuras.

[4] J. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill, 2001.

[5] Dr. Mario Lázaro. *Tema 3. Aerodinámica Alas Rectas (Solución Integral de Prandtl, Distribución de Sustentación del Ala Rígida)*. Dto. Mecánica de los Medios Continuos y Teoría de Estructuras.

[6] Dr. Mario Lázaro. *Tema 7. Aerodinámica de Superficies de Sustentación 3D (Método de la malla de torbellinos, Distribución de presiones, Distribución de Sustentación)*. Dto. Mecánica de los Medios Continuos y Teoría de Estructuras.

[7] J.L. Pérez y M. Lázaro. *Teoría de vigas en Ingeniería Aeroespacial*. Universitat Politècnica de València, 2019.

[8] Wikipedia. *Teoría de vigas de Timoshenko*. URL: https://es.wikipedia.org/wiki/Teor%5C%C3%5C%ADa_de_vigas_de_Timoshenko (visitado 05-06-2023).

[9] J.L. Pérez. *Teoría de placas y láminas en Ingeniería Aeroespacial*. Universitat Politècnica de València, 2019.

[10] Dr. Mario Lázaro. *Tema 8. Modelo Estructural de Alas 3D (Modelización Estructural Alas en Flecha, Matrices de Acoplamiento Fluido-Estructura)*. Dto. Mecánica de los Medios Continuos y Teoría de Estructuras.

[11] United Nations. *Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible*. URL: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ (visitado 03-06-2023).

[12] M. Goland. «The futter of a uniform cantilevered wing». En: *Journal of Applied Mechanics* (1945).

[13] S. Kumar, A. Kumar y M. Manjuprasad. «Probabilistic free vibration analysis of Goland wing». En: *International Journal of Aerospace System Engineering* (2019).

[14] R. Palacios y B. Epureanu. «An Intrinsic Description of the Nonlinear Aeroelasticity of Very Flexible Wings». En: *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference* (2011).

[15] B. Raghavan y P. Patil. «Flight Dynamics of High Aspect-Ratio Flying Wings: Effect of Large Trim Deformation». En: *Journal of Aircraft* (2009).

[16] J. Murua et al. «Stability and Open-Loop Dynamics of Very Flexible Aircraft Including Free-Wake Effects». En: *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference* (2011).

[17] W. Rodden, J. McGrew y T. Kálman. «Comment on Ïnterpolation Using Surface Splines"». En: *Journal of Aircraft* (1972).

[18] R. Harder. «Interpolation Using Surface Splines». En: *Journal of Aircraft* (1972).

[19] Dr. Mario Lázaro. *Tema 11. Método No-Estacionario de la Malla de Torbellinos*. Dto. Mecánica de los Medios Continuos y Teoría de Estructuras.

[20] Wikipedia. *Cessna 172 Skyhawk*. URL: https://en.wikipedia.org/wiki/Cessna_172 (visitado 11-06-2023).

[21] *FLY AT HOME EP-2: AIRCRAFT PARTS*. URL: https://pilotahmad.wordpress.com/2021/04/07/fly-at-home-ep-2-aircraft-parts/ (visitado 20-05-2023).

[22] *Cessna 172 Skyhawk Aircraft*. URL: https://www.dimensions.com/element/cessna-172-skyhawk-aircraft (visitado 20-05-2023).

Anexo

A. Grado de relación del proyecto con los *ODS*

Objetivos de desarrollo sostenible	Alto	Medio	Bajo	No procede
ODS 1. Fin de la pobreza				X
ODS 2. Hambre cero				X
ODS 3. Salud y bienestar				X
ODS 4. Educación de calidad				X
ODS 5. Igualdad de género				X
ODS 6. Agua limpia y saneamiento				X
ODS 7. Energía asequible y no contaminante		X		
ODS 8. Trabajo decente y crecimiento económico				X
ODS 9. Industria, innovación e infraestructura	X			
ODS 10. Reducción de las desigualdades				X
ODS 11. Ciudades y comunidades sostenibles		X		
ODS 12. Producción y consumo responsables				X
ODS 13. Acción por el clima	X			
ODS 14. Vida submarina				X
ODS 15. Vida de ecosistemas terrestres		X		
ODS 16. Paz, justicia e instituciones sólidas				X
ODS 17. Alianzas para lograr objetivos				X

Tabla 11: Tabla resumen del grado de relación del trabajo con los *ODS*

B. Código en *Matlab*

B.1. Programa principal

B.1.1. Programa principal

```
%%%% ANALISIS MODAL Y AEROELASTICO DE UN AVION COMPLETO %%%%

%% DESCRIPCION DEL PROGRAMA PRINCIPAL
% main (script)
% Descripcion general: En este script se realiza un analisis de
    vibraciones libres y aeroelastico (estatico y dinamico) de un
    modelo de avion completo en plano horizontal. En primer lugar, se
    define la geometria de ala, estabilizador y fuselaje para llevar a
     cabo el modelado estructural a partir del modelo de vigas y el
    aerodinamico mediante el VLM.

% A partir de esto, se resuleve el problema de vibraciones libres,
    obteniendose los modos de vibracion y frecuencias naturales de la
    estructura completa.

% Seguidamente se resuleve el problema aeroelastico estatico
    calculando la velocidad de divergencia y el modo en que diverge la
     aeronave.

% Finalmente se aborda el problema aeroelastico dinamico desde un
    desarrollo casi estacionario ya que el VLM utilizado para modelar
    las superficies aerodinamicas es estacionario. En este punto, se
    calculan las curvas de flameo y se representa el modo de flameo.

%==========================================================================
% Universitat Politecnica de Valencia
% Master Universitario en Ingenieria Aeronautica
% Escuela Tecnica de Ingenieria del Diseno
% Dpto. de Mecanica de los Medios Continuos y Teoria de Estructuras
% Trabajo Fin de Master
% Autor: Ramirez Conca, Javier
% Tutor: Lazaro Navarro, Mario
%==========================================================================

%% INICIALIZACION
clc
clear
close all

addpath('Modelo Estructural')
addpath('Modelo Aerodinamico')
addpath('Acoplamiento')
```

```
addpath('Vibraciones libres')
addpath('Aeroelasticidad Estatica')
addpath('Aeroelasticidad Dinamica')
addpath('Distribucion Fuerzas')
addpath('Representaciones')

%% GEOMETRIA (CESSNA 172 SKYHAWK)

% ALA
bw  = 11;... envergadura [m]
crw = 1.63;... cuerda en la raiz [m]
ctw = 1.12;... cuerda en punta de ala [m]
l_w = ctw/crw;... estrechamiento del ala [-]
cma = 2*crw*(1+l_w+l_w^2) / (3*(1+l_w));... cma [m]
N_costillas_w = 9;... numero de costillas del ala
N_largueros_w = 5;... numero de largueros  y larguerillos del ala

% ESTABILIZADOR HORIZONTAL
be  = 3.4;... envergadura [m]
cre = 1.2;... cuerda en la raiz [m]
cte = 0.78;... cuerda en punta de ala [m]
l_e = cte/cre;... estrechamiento del ala [-]
cma_e = 2*cre*(1+l_e+l_e^2) / (3*(1+l_e));... cma [m]
N_costillas_e = 3;... numero de costillas del estabilizador
N_largueros_e = 3;... numero de largueros del estabilizador

% FUSELAJE
nariz = 1.6;... longitud de la nariz [m]
hiato = 2*crw;.
h_estr = hiato + (1/10)*crw + (1/10)*cre;
cola = 8.3 - (nariz + crw + hiato + cre);... longtitud de la cola [m]
lf = nariz + crw + hiato + cre + cola;... longitud del fuselaje [m]
df = [6 10 2];... divisiones del fuselaje en elementos
FUSELAJE = [nariz+1/10*crw h_estr cola+1/10*cre];

% PESOS
OEW = 745;... peso en vacio [kg]
MTOW = 1111;... peso maximo al despegue [kg]

%% MODELO ESTRUCTURAL (ELEMENTOS VIGA EULER-BERNOULLI)

% Coordenadas de contorno
% Ala
NW = [ (1/10)*crw,  0];
SW = [ (9/10)*crw,  0];
NE = [ (1/10)*crw, bw/2-bw/60];
SE = [ ctw - (1/10)*ctw, bw/2-bw/60];
COORD_ESTR_ALA = [NW;SW;SE;NE];

```

```
% Estabilizador
NW = [ crw + hiato + (1/10)*cre, 0];
SW = [ crw + hiato + (9/10)*cre, 0];
NE = [ crw + hiato + cre/2 - cte/2 + (1/10)*cte, be/2-be/60];
SE = [ crw + hiato + cre/2 - cte/2 + (9/10)*cte, be/2-be/60];
COORD_ESTR_ESTAB = [NW;SW;SE;NE];

% Discretizacion del emparrillado estructural de las superficies
% sustentadoras

% Ala
nx_w = N_largueros_w - 1;... discretizacion eje long. "x"
ny_w = N_costillas_w;... discretizacion envergadura. "y"

% Estabilizador
nx_e = N_largueros_e - 1;... discretizacion eje long. "x"
ny_e = N_costillas_e;... discretizacion envergadura. "y"

% Generamos la geometria de la estructura
[ESTRUCTURA_ALA] = GeometriaEstructura(COORD_ESTR_ALA,nx_w,ny_w);
[ESTRUCTURA_ESTAB] = GeometriaEstructura(COORD_ESTR_ESTAB,nx_e,ny_e);
[ESTRUCTURA] = GeometriaAvion(ESTRUCTURA_ALA,ESTRUCTURA_ESTAB,
    FUSELAJE,df);

Nn = ESTRUCTURA.NumNodos;... numero de nodos
Ne = ESTRUCTURA.NumElem;... numero de elementos
Ngdl = ESTRUCTURA.NumGDL;... numero de grados de libertad
CoordNodos = ESTRUCTURA.CoordNodos;... matriz coord. nodos (Nn x 3)
GDL = ESTRUCTURA.GDL;... matriz de grados de libertad (Nn x 3)
Conectivity = ESTRUCTURA.Conect;... matriz de conectividad (Ne x 2)
ESTRUCTURA.crw = crw;
ESTRUCTURA.crt = cre;

%% Dibujo del emparrillado 2D (plano XY)
% t = [-1,1]';... vector de discretizacion de cada elemento
% DibujoEstructura(ESTRUCTURA,t)

% Propiedades estructurales (materiales)
% Modulo de Young del acero = 2.1E11 [N/m2]
% Modulo de Young del aluminio = 7E10 [N/m2]
% Modulo de torsion del acero = 8.5E10 [N/m2]
% Modulo de torsion del aluminio = 2.63E10 [N/m2]

% Ala
E_w = 7E10;
G_w = 2.63E10;
% Estabilizador
E_e = 7E10;
G_e = 2.63E10;
```

```
% Fuselaje
E_f = 7E10;
G_f = 2.63E10;

ESTRUCTURA.E = [E_w E_e E_f];
ESTRUCTURA.G = [G_w G_e G_f];

%% MODELO AERODINAMICO DE PANELES (VLM)

% Coordenadas contorno
% Ala
NW = [ 0, 0];
SW = [ crw, 0];
NE = [ 0, bw/4];
SE = [ crw, bw/4];
COORD_VLM{1} = [NW;SW;SE;NE];

NW = [ 0, bw/4];
SW = [ crw, bw/4];
NE = [ 0, bw/2];
SE = [ ctw, bw/2];
COORD_VLM{2} = [NW;SW;SE;NE];

% Estabilizador
NW = [ crw + hiato, 0];
SW = [ crw + hiato + cre, 0];
NE = [ crw + hiato + cre/2 - cte/2, be/2];
SE = [ crw + hiato + cre/2 + cte/2, be/2];
COORD_VLM{3} = [NW;SW;SE;NE];

% Identificacion de la superficie de sustentacion
IdSuperficie(1) = 1;... seccion recta del ala
IdSuperficie(2) = 1;... seccion variable del ala
IdSuperficie(3) = 2;... estabilizador

% Discretizacion de cada celda en paneles
nx = [13, 13, 11];... discretizacion eje long. "x"
ny = [13, 13, 13];... discretizacion envergadura. "y"

% Generamos la geometria de paneles
[PANELES] = GeometriaPaneles(COORD_VLM,nx,ny,IdSuperficie);
PANELES.N_sup = max(PANELES.IdSuperficie);

Np = PANELES.NumPaneles;
R = PANELES.MatrizR;
T = PANELES.MatrizT;
yl = PANELES.Vector_y;
S = PANELES.Area;
DS = diag(S);
```

```
177 idAla = (PANELES.IdSuperficie == 1);
    idEstab = (PANELES.IdSuperficie == 2);
179 Sw = sum(S(idAla));
    St = sum(S(idEstab));
181 dX = PANELES.dX;
    dY = PANELES.dY;
183 DY = diag(dY);
    xCA = PANELES.CoordCA(:,1);
185 yCA = PANELES.CoordCA(:,2);
    zCA = zeros(Np,1);
187 xCj = PANELES.CoordControl(:,1);
    yCj = PANELES.CoordControl(:,2);
189 zCj = zeros(Np,1);

191 %% Dibujo ala
    % DibujoMalla(PANELES);
193
    % Propiedades masicas
195 % Ala
    PANELES.bw = bw;
197 PANELES.m_w = 27.1;... masa distribuida del ala [kg/m]
    masa_dist_w = PANELES.m_w;
199 % Estabilizador
    PANELES.be = be;
201 PANELES.m_e = 21.9;... masa distribuida del estabilizador [kg/m]
    masa_dist_e = PANELES.m_e;
203 % Fuselaje
    PANELES.lf = lf;
205 PANELES.m_f = OEW - (masa_dist_w*bw + masa_dist_e*be);

207 %% VISUALIZACION PANELES-ESTRUCTURA

209 t = [-1,1]';... vector de discretizacion de cada elemento
    DibujoGlobal(ESTRUCTURA,t,PANELES)
211
    %% CONDICIONES DE CONTORNO
213
    ESTRUCTURA.CondContorno = [];... vector con los GDL nulos
215
    %% CALCULO DE MATRICES
217
    % MATRICES DE MASA Y RIGIDEZ
219 K = RigidezEstructura(ESTRUCTURA);... matriz de rigidez
    M = MatrizMasa(PANELES,ESTRUCTURA);... matriz de masa del ala
221
    % MATRIZ DE COEFICIENTES AERODINAMICOS
223 H = MatrizH(PANELES);
    invH = inv(H);
225
```

```
%% MATRICES DE ACOPLAMIENTO FLUIDO-ESTRUCTURA
[Du,Dv,Dz] = MatricesAcoplamiento_Dinamicas(ESTRUCTURA,PANELES);

%% FRECUENCIAS NATURALES Y MODOS DE VIBRACION

% FRECUENCIAS NATURALES Y MODOS DE LA ESTRUCTURA
m = 12;... numero de modos a calcular
h = 1.5;... factor de escala
[wnat,V] = ModosVibracionLibre(h,m,ESTRUCTURA,M,K);
DibujoModosEstPaneles(ESTRUCTURA,PANELES,wnat,V,h,1)

%% VISUALIZACION DE MODOS EN MOVIMIENTO
% modo = 9;... modo que se quiera representar
% w_m = wnat(modo);... frecuencia natural del modo a representar
% vec_m = V(:,modo);... autovector del modo a representar
% DibujaModoVibracion(modo,ESTRUCTURA,w_m,vec_m,h)

%% AEROELASTICIDAD ESTATICA -> MODOS DE DIVERGENCIA

% ANGULO DE ATAQUE RIGIDO
% Ala
alpha_w_0 = 1;... angulo de ataque del ala
ar_w = -alpha_w_0*ones(Np_w,1);
% Estabilizador
alpha_e_0 = 0;... angulo de ataque del estabilizador
ar_e = -alpha_e_0*ones(Np_e,1);
% Vector de angulos de ataque rigidos
ar = [ar_w; ar_e];

% MATRICES ADICIONALES
As = 2 * Dz' * DY * invH * Du;
fr = 2 * Dz'* DY * invH * ar;

% CALCULO Y REPRESENTACION DE LOS MODOS DE INESTABILIDAD POR
    DIVERGENCIA
m = 1;... numero de modos a calcular
h = 1;... factor de escala
rho = 1.225;...densidad del aire sin perturbar [kg/m3]
[uD,~] = ModosDivergencia(h,m,K,As,ESTRUCTURA,PANELES,rho);
Ud = uD(1);

%% SOLUCION ESTATICA PARA U < Udivergencia
% U = 0.5*Ud;
% q = 0.5*rho*U^2;
% u_sol = (K - q * As) \ (q * fr);
% DibujoModosEstPaneles(ESTRUCTURA,PANELES,U,u_sol,h,2);

%% AEROELASTICIDAD DINAMICA -> CURVAS DE FLAMEO

```

```
% MATRICES ADICIONALES (MATRICES DE LAS FUERZAS GENERALIZADAS)
A = Dz' * DY * invH * Du;
B = (Dz' * DY * invH * Dv + Dz' * DS * T * invH * Du);
C = Dz' * DS * T * invH * Dv;

% CALCULO FRECUENCIAS Y AMORTIGUAMIENTOS DE LAS CURVAS DE FLAMEO
Vmax = 300;
dV = Vmax/5000;
V = (0:dV:Vmax);
zeta = 0;... amortiguamiento artificial
m = 10;
[w,g,~] = CurvasFlameo(A,B,C,K,M,V,zeta,rho,m);

% CALCULO VELOCIDADES DE DIVERGENCIA Y FLAMEO
[Ud,Uf,wf] = Calcula_Divergencia_y_Flameo(w,g,V);

% REPRESENTACION GRAFICA CURVAS FLAMEO
[w_plot,g_plot] = DibujaCurvasFlameo(w,g,V);

% wj vs V
figure
% scatter(Uf(end),wf(end),50,'LineWidth',2)
% hold on
scatter(uD(1),0,50,'LineWidth',2)
hold on
scatter(uD(2),0,50,'LineWidth',2)
hold on
scatter(V,w_plot(1:10,:),5,'k','filled')
xlabel('$U_{\infty}$ [m/s]','Interpreter','LaTex','FontSize',12)
ylabel('$\omega_j = \Im(s_j)$ [rad/s]','Interpreter','LaTex','
    FontSize',12)
title('Frecuencias Propias','Interpreter','LaTex','FontSize',14)
ax = gca;
ax.TickLabelInterpreter = 'latex';
grid on
box on
txt_legend_div1 = ['Velocidad de divergencia, $U_D$ = ' num2str(uD(1)
    ,'%.2f') ' m/s'];
%txt_legend_freq1 = 'Frecuencias naturales en tierra, $U_\infty$ =
    0';
txt_legend_flameo = ['Punto de flameo, $U_f$ = ' num2str(Uf(end),'%.2
    f') ' m/s, $\omega_f$ = ' num2str(wf(end),'%.2f') ' rad/s'];
legend(txt_legend_flameo,txt_legend_div1,'Interpreter','LaTex','
    FontSize',10)
ylim([0 350])
% gj vs V
figure
% scatter(vec_flameo(1),0,'filled')
% hold on
```

```
scatter(uD(1),0,50,'LineWidth',2)
hold on
scatter(uD(2),0,50,'LineWidth',2)
hold on
scatter(V,g_plot(1:10,:),5,'k','filled')
xlabel('$U_{\infty}$ [m/s]','Interpreter','LaTex','FontSize',12)
ylabel('$g_j = \Re(s_j)$ [rad/s]','Interpreter','LaTex','FontSize'
    ,12)
title('Amortiguamientos','Interpreter','LaTex','FontSize',14)
ax = gca;
ax.TickLabelInterpreter = 'latex';
grid on
box on
txt_legend_flameo = ['Punto de flameo, $U_f$ = ' num2str(Uf(end),'%.2
    f') ' m/s, $\omega_f$ = ' num2str(wf(end),'%.2f') ' rad/s'];
txt_legend_div1 = ['Velocidad de divergencia, $U_D$ = ' num2str(uD(1)
    ,'%.2f') ' m/s'];
legend(txt_legend_flameo,txt_legend_div1,'Interpreter','LaTex','
    FontSize',10)
ylim([-20 40])
```

Código 1: *Script* del programa principal

B.2. Modelo aerodinámico

B.2.1. Generación y discretización en paneles de la geometría de las superficies de sustentación

```
function [PANELES] = GeometriaPaneles(COORD, nx_vector, ny_vector,
    SUPERFICIE)

%% DESCRIPCION DE LA FUNCION ==============================================
%
% GeometriaPaneles genera la discretizacion en paneles de una
% superficie sustentadora y calcula los puntos de interes de
% cada panel a partir de un vector de coordenadas que definen
% los limites de las superficies a modelar, las discretizaciones
% en X e Y de cada superficie y un vector de identificacion de
% superficie
%
% COORD: array con las coordenadas de los nodos de cada celda
%
% INPUTS =================================================================
% y
% |           NE ............ SE
% |             \                \
% |               \                \
% |                 \                \
% |                   \                \
% |                     \                \
% |                       NW..............SW
% |
% |
% ---------------------------------------> x
%
% INPUTS =================================================================
% condiciones: yNW = ySW; yNE = ySE; 0<yNW < yNE; 0<ySW < ySE
% condiciones: xNW < xSW; xNE < xSE;
% nx_vector, ny_vector: numero de paneles en direccion "x" y en "y"
% Extraemos coordenadas
% 1 <= j <= NumCeldas
% NW = COORD{j}(1,:); Coordenadas [X,Y]
% SW = COORD{j}(2,:); Coordenadas [X,Y]
% SE = COORD{j}(3,:); Coordenadas [X,Y]
% NE = COORD{j}(4,:); Coordenadas [X,Y]
% SUPERFICIE es un vector del tamano igual al NumCeldas = COORD.
% Se usa para identificar el tipo de superficie cuando hay varias,
% por ejemplo
% SUPERFICIE = [1 1 1, 2, 2]: Las tres primeras celdas forma parte
% la Sup 1 que puede ser el ala, mientras que las dos ultimas forma
% otra superficie independiente (Sup 2) que puede ser el estabiliz.
% Tambien se pueden distinguir ala y alerones.
```

```
43 %
   % OUTPUTS ==========================================================
45 % PANELES.CoordNodos,...      = [xNW,yNW,xSW,ySW,xSE,ySE,xNE,yNE]
   % PANELES.CoordControl,...   = [xC, yC];
47 % PANELES.CoordCA,... = [xV, yV]
   % PANELES.CoordVertices,... = [xA, yA, xB, yB]
49 % PANELES.IdSuperficie = identificacion de la superficie del panel
   % PANELES.dX...          = dX: "cuerda" de cada panel
51 % PANELES.dY...          = dY: "envergadura" de cada panel
   % PANELES.Area...        = Ap: "area panel"
53 % PANELES.MatrizR...     = Tamano ny x N: En cada fila "nu", 1<=nu<=ny
   %                          hay "1" en elvindice de los paneles que
55 %                          estan en dicha fila y "0" en los que estan
   %                          fuera.
57 % PANELES.MatrizT         = (tamano N x N) En el elemento (j,j) hay
   %                          "3/4" y el elemento (j,k) es "1" si x_Ck
59 %                          < x_Cj AND y_Ck == y_Cj.
   %                          En caso contrario es "0".
61 % PANELES.Vector_y        = Vector de tamano ny con las coordenadas
   %                           yVj=yCj de cada columna de paneles
63 % PANELES.Cuerdas         = Vector de tamano ny con las cuerdas asociad
   %                          a cada coordenada y_nu, -b/2 <= y_nu <=+b/2
65 % PANELES.NumPaneles      = numero total de paneles
   % PANELES.nxtotal         = numero total de paneles en x
67 % PANELES.nytotal         = numero total de paneles en y
   %
69 %
   % CROQUIS
71 %
   % ------------------------------------------> Y
73 % |            (NW)-----------------(NE)
   % |              |                   |
75 % |             (A).....(V).....(B)
   % |              |                   |
77 % |              |                   |
   % |              |    panel j        |
79 % |              |                   |
   % |              |         C         |
81 % |              |                   |
   % |            (SW)-----------------(SE)
83 % |

85 NumCeldas = numel(COORD);

87 %% INFORMACIoN COORDENADAS DE LOS PANELES ============================
   PANELES.CoordNodos = [];
89 PANELES.CoordControl = [];
   PANELES.CoordCA = [];
91 PANELES.CoordCG = [];
```

```
PANELES.CoordVertices = [];
PANELES.IdSuperficie = [];
PANELES.dX = [];
PANELES.dY = [];
PANELES.Area = [];
PANELES.MatrizR = [];
PANELES.MatrizT = [];

%figure
%hold on

for celda = 1:NumCeldas
    % Extraemos las coordenadas de los paneles
    nx = nx_vector(celda);
    ny = ny_vector(celda);
    [X,Y] = GeneracionCelda(COORD{celda},nx,ny);
    id = SUPERFICIE(celda);

    % PANELES DE LA CELDA EN EL SEMIALA DERECHA (y > 0)
    for i=1:(size(X,1)-1)
        for j=1:(size(X,2)-1)
          Xp = [X(i,j) X(i,j+1); X(i+1,j), X(i+1,j+1)];
          Yp = [Y(i,j) Y(i,j+1); Y(i+1,j), Y(i+1,j+1)] ;
          %mesh(Xp,Yp,0*zeros(size(Xp)));
        [NODOS,CONTROL,CA,VERTICES,CG,AREA]= CoordenadasPanel(Xp,Yp);
          PANELES.CoordNodos    = [PANELES.CoordNodos; NODOS];
          PANELES.CoordControl  = [PANELES.CoordControl; CONTROL];
          PANELES.CoordCA       = [PANELES.CoordCA; CA];
          PANELES.CoordVertices = [PANELES.CoordVertices; VERTICES];
          PANELES.IdSuperficie  = [PANELES.IdSuperficie; id];
          PANELES.dX            = [PANELES.dX; AREA(1)];
          PANELES.dY            = [PANELES.dY; AREA(2)];
          PANELES.Area          = [PANELES.Area; AREA(3)];
          PANELES.CoordCG       = [PANELES.CoordCG; CG];
        end
    end

    % PANELES DE LA CELDA EN EL SEMIALA IZQUIERDA (y < 0)
    Y = -Y; ... por simetria
    for i=1:(size(X,1)-1)
        for j=1:(size(X,2)-1)
          Xp = [X(i,j) X(i,j+1); X(i+1,j), X(i+1,j+1)];
          Yp = [Y(i,j) Y(i,j+1); Y(i+1,j), Y(i+1,j+1)] ;
          %mesh(Xp,Yp,0*zeros(size(Xp)));
        [NODOS,CONTROL,CA,VERTICES,CG,AREA]= CoordenadasPanel(Xp,Yp);
          PANELES.CoordNodos    = [PANELES.CoordNodos; NODOS];
          PANELES.CoordControl  = [PANELES.CoordControl; CONTROL];
          PANELES.CoordCA       = [PANELES.CoordCA; CA];
          PANELES.CoordVertices = [PANELES.CoordVertices; VERTICES];
```

```
141             PANELES.IdSuperficie  = [PANELES.IdSuperficie; id];
                PANELES.dX            = [PANELES.dX; AREA(1)];
143             PANELES.dY            = [PANELES.dY; AREA(2)];
                PANELES.Area          = [PANELES.Area; AREA(3)];
145             PANELES.CoordCG       = [PANELES.CoordCG; CG];
            end
147     end
end
149
%% LOCALIZACION DE PANELES EN CADA COLUMNA PARALELA AL EJE "y"=======
151 % Esto tiene sentido cuando analizamos una sola superficie de
% sustentacion. No tiene sentido si tuvieramos ala+estabilizador
153 % pues en tal caso habria dos vectores de coordenadas y_nu
% ------------------------------------------------------------------
155 % ny: numero de columnas paralelas al eje "y"
% N : numero total de paneles. En general, se tiene N = nx * ny.
157 % Matriz R: (tamano ny x N) En cada fila "nu", 1<=nu<=ny hay "1" en
% el indice de los paneles que estan en dicha fila y "0" en los que
159 % estan fuera.
% Matriz T: (tamano N x N) En el elemento (j,j) hay "3/4" y el
161 % elemento (j,k) es "1" si x_Ck <= x_Cj AND y_Ck == y_Cj. En caso
% contrario es "0".
163
% Numero total de paneles: N = numero de filas en la matriz de PC
165 N = size(PANELES.CoordControl,1);
tol = 1E-3;
167 % Inicializacion matrices
T = zeros(N,N);
169


171 for j=1:N
    % Identificamos los paneles que estan en la misma columna
173     xVj = PANELES.CoordCA(j,1);
    yVj = PANELES.CoordCA(j,2);
175     % indices de paneles en la columna y=yVj
    id_yVj = (abs(PANELES.CoordCA(:,2) - yVj) < tol);
177     p = 1:N;
    % Identificamos los paneles que verifican x<xVj
179     id_xVj = (PANELES.CoordCA(:,1) < xVj);
    [id_xVj, id_yVj];
181     % Identificamos los paneles que verifican ambas condiciones
    id_xVj_yVj = (id_xVj & id_yVj);
183     T(j,:) = (id_xVj_yVj .* 1)';
    T(j,j) = 3/4;
185     if j==1
        y_nu = [yVj];
187     elseif (j>1) && (min(abs(y_nu - yVj))<tol)
        y_nu = y_nu;
189     else
```

```
        y_nu = [y_nu, yVj];
    end

end

% Inicializacion matrices
% Ordenamos los elementos de yj

y_nu = sort(y_nu);
ny = numel(y_nu);
R = zeros(ny,N);
c = zeros(ny,1);... vector con la cuerda en cada seccion 1<=nu<=ny
dXj = PANELES.dX;... vector con las cuerdas de cada panel
for i=1:ny
    % indices de paneles en la columna y=yVj
    id_ynu = (abs(PANELES.CoordCA(:,2) - y_nu(i)) < tol);
    R(i,:) = (id_ynu .* 1)';
    c(i) = sum(dXj(id_ynu));... cuerda seccion "i" = suma cuerdas de
    los paneles en esa seccion
end

PANELES.MatrizR = R;
PANELES.MatrizT = T;
PANELES.Vector_y = y_nu;
PANELES.Cuerdas = c;

PANELES.NumPaneles = N;
PANELES.nytotal = length(y_nu);
PANELES.nxtotal = N / PANELES.nytotal;

%% FUNCION AUXILIAR GENERACION COORDENADAS NODOS CELDA
% Esta funcion genera la malla de paneles en una celda en particular
% Para ello transforma el cuadrado unidad 1x1 dividido en
% num_panels_x * num_paneles_y en una celda con coordeandas dadas
% por COORD_ESQUINAS

function [Xnodos,Ynodos] = GeneracionCelda(COORD_ESQUINAS,
    num_paneles_x,num_paneles_y)

% COORD: matriz con coordenadas nodos entrada de los limites de la
% malla unidad: COORD = [xNW, yNW; xSW, ySW; xSE, ySE; xNE, yNE]
% y
% |          NE ..........  SE
% |            \             \
% |              \             \
% |                \             \
% |                  \             \
% |                  NW.............SW
% |
```

```
237 % |
    % ------------------------------------> x
239 %
    % condiciones: yNW = ySW; yNE = ySE; 0<yNW < yNE; 0<ySW < ySE
241 % condiciones: xNW < xSW; xNE < xSE;
    % nx, ny: numero de paneles en direccion x y en direccion y en malla
243 % unidad

245 % Extraemos coordenadas
    NW = COORD_ESQUINAS(1,:);
247 SW = COORD_ESQUINAS(2,:);
    SE = COORD_ESQUINAS(3,:);
249 NE = COORD_ESQUINAS(4,:);
    % Variables auxiliares
251 dxW = SW(1) - NW(1);
    dxE = SE(1) - NE(1);
253 dy  = NE(2) - NW(2);

255 % GEOMETRIA DE LA CELDA ==========================================

257 % MALLADO INICIAL CUADRADO UNIDAD
    [Xnodos,Ynodos] = MallaCuadradoInicial(num_paneles_x,num_paneles_y);
259
    % TRANSFORMACIoN 1
261 Xnodos = dxW * Xnodos;
    Ynodos = dy * Ynodos;
263
    % TRANSFORMACIoN 2
265 Xnodos = Xnodos + (1/dy)*(dxE/dxW - 1) * (Xnodos .* Ynodos);

267 % TRANSFORMACIoN 3
    Xnodos = NW(1) + Xnodos + (1/dy) * (NE(1) - NW(1)) * Ynodos;
269 Ynodos = NW(2) + Ynodos;
    % mesh(X,Y,0 * zeros(size(X)))
271
    function [X0,Y0] = MallaCuadradoInicial(nx,ny)
273 % Obtencion coordenadas del cuadrado inicial
    dx_unidad = 1/nx;
275 dy_unidad = 1/ny;

277 % Nodos de los paneles
    vx_0 = 0:dx_unidad:1;
279 vy_0 = 0:dy_unidad:1;

281 [X0,Y0] = meshgrid(vx_0,vy_0);

283 end

285 end
```

```
287 %% COORDENADAS PRINCIPALES DE UN PANEL ==========================
    % Esta funcion coge las cuatro coordenadas de un panel y determina:
289 %       (1) Las coordenadas de los nodos ordenadas: NW,SW,SE,NE
    %       (2) Coordenadas del punto de control del panel
291 %       (3) Coordenadas del c.a. del panel
    %       (4) Coordenadas del vertices del torbellino
293 %       (5) Coordenadas del centro de gravedad del panel
    %       (6) Caracteristicas geometricas: ancho en X
295 %               (dX:cuerda); ancho en X (en el centro)
    %               (dY:enverg); ancho en Y
297 %               (dA:area); area del panel = dX * dY

299 function [CoordNodos, CoordPuntoControl,CoordCentroAerodinamico,
        CoordVerticesTorbellino,CoordCentroGravedad,DatosGeometricos] =
        CoordenadasPanel(X,Y)

301 %% DETERMINACION DE LAS COORDENADAS EN ORDEN (ANTIHOR) NW-SW-SE-NE
    % mesh(X,Y,zeros(size(X)))
303
    % Filtramos los valores minimo y maximo de y
305 ymin = min(min(Y));
    ymax = max(max(Y));
307
    % Detectamos las posiciones de los valores minimo y maximo
309 id_ymin = (ymin - eps <= Y) & (Y <= ymin + eps);
    id_ymax = (ymax - eps <= Y) & (Y <= ymax + eps);
311
    % Los correspondientes a ymin son los NW y SW
313 % Coordenadas X
    X_W = X(id_ymin);
315 Y_W = Y(id_ymin);
    X_NW = min(X_W);
317 X_SW = max(X_W);
    % Coordenadas Y
319 id_NW = (X_W == X_NW);
    Y_NW = Y_W(id_NW);
321 id_SW = (X_W == X_SW);
    Y_SW = Y_W(id_SW);
323
    % Los correspondientes a ymax son los NE y SE
325 % Coordenadas X
    X_E = X(id_ymax);
327 Y_E = Y(id_ymax);
    X_NE = min(X_E);
329 X_SE = max(X_E);
    % Coordenadas Y
331 id_NE = (X_E == X_NE);
    Y_NE = Y_E(id_NE);
```

```
id_SE = (X_E == X_SE);
Y_SE = Y_E(id_SE);

CoordNodos = [X_NW, Y_NW, X_SW, Y_SW, X_SE, Y_SE, X_NE, Y_NE];

X_area = [X_NW, X_SW, X_SE, X_NE, X_NW];
Y_area = [Y_NW, Y_SW, Y_SE, Y_NE, Y_NW];
dX = (1/2)*(X_SW + X_SE) - (1/2)*(X_NW + X_NE);
dY = (1/2)*(Y_NE + Y_SE) - (1/2)*(Y_NW + Y_SW);
AreaPanel_polyarea = polyarea(X_area,Y_area);
AreaPanel_producto = dX * dY;
DatosGeometricos = [dX, dY, AreaPanel_producto];

%% DETERMINACION PUNTOS DE CONTROL ========================================
X_C= (3/8) * X_SW + ...
     (3/8) * X_SE + ...
     (1/8) * X_NW + ...
     (1/8) * X_NE;
Y_C= (3/8) * Y_SW + ...
     (3/8) * Y_SE + ...
     (1/8) * Y_NW+ ...
     (1/8) * Y_NE;

 CoordPuntoControl = [X_C, Y_C];

 %% DETERMINACION CENTRO AERODINAMICO ======================================
X_V= (1/8) * X_SW + ...
     (1/8) * X_SE + ...
     (3/8) * X_NW + ...
     (3/8) * X_NE;
Y_V= (1/8) * Y_SW + ...
     (1/8) * Y_SE + ...
     (3/8) * Y_NW + ...
     (3/8) * Y_NE;

 CoordCentroAerodinamico = [X_V, Y_V];

 %% DETERMINACION VERTICES DEL TORBELLINO (A,B) ============================
X_A = (1/4) * X_SW + ...
      (3/4) * X_NW;
Y_A = (1/4) * Y_SW + ...
      (3/4) * Y_NW;
X_B = (1/4) * X_SE + ...
      (3/4) * X_NE;
Y_B = (1/4) * Y_SE + ...
      (3/4) * Y_NE ;

  CoordVerticesTorbellino = [X_A, Y_A, X_B, Y_B];

```

```
%% DETERMINACION CENTRO DE GRAVEDAD ========================================
X_G= (1/4) * X_SW + ...
     (1/4) * X_SE + ...
     (1/4) * X_NW + ...
     (1/4) * X_NE;
Y_G= (1/4) * Y_SW + ...
     (1/4) * Y_SE + ...
     (1/4) * Y_NW + ...
     (1/4) * Y_NE;

 CoordCentroGravedad = [X_G, Y_G];

end

end
```

Código 2: Generación y discretización en paneles de la geometría de las superficies de sustentación

B.2.2. Cálculo de la matriz de coeficientes de influencia aerodinámicos

```
function [H] = MatrizH(PANELES)

%% DESCRIPCION DE LA FUNCION ================================================
%
% MatrizH calcula la matriz de coeficientes de influencia aerod.
% del metodo malla de torbellinos ("Vortex Lattice Method")
%
% INPUTS ---------------------------------------------------------------------
% PANELES : estructura de datos con la geometria y datos relevantes
% de lavmalla de paneles que modelizan las superficies alares

% OUTPUTS --------------------------------------------------------------------
% H : matriz de coeficientes aerodinamicos

%% CALCULO DE LA MATRIZ H ===================================================

% Datos geometria
N = PANELES.NumPaneles;... numero de paneles totales
gradF = [0 0 1];... vector normal a la superficie
coordZ_sup = zeros(N,1);
RC = PANELES.CoordControl;... coordenadas de los puntos de control
RC = [RC coordZ_sup];
RA = PANELES.CoordVertices(:,1:2);... coordenadas de los vertices A
RA = [RA coordZ_sup];
RB = PANELES.CoordVertices(:,3:4);... coordenadas de los vertices B
RB = [RB coordZ_sup];
H = zeros(N,N);
```

```
28
        for k = 1:N
30          for j = 1:N
                W = TorbellinoHerradura(1,RA(k,:),RB(k,:),RC(j,:));
32              H(j,k) = dot(W,gradF);
            end
34      end

36  end
```

Código 3: Cálculo de la matriz **H**

B.2.3. Velocidad vertical inducida por un torbellino en herradura

```
1   function W = TorbellinoHerradura(G,RA,RB,RP)

3   %% DESCRIPCION DE LA FUNCION ==========================================
    %
5   % TorbellinoHerradura calcula la velocidad en un punto RP(x,y,z)
    % debida a un torbellino en herradura de intensidad Gamma = G.
7   %
    %                                [B]
9   %                          .      |
    %                      .          |
11  %                  [F]            |
    %                .                |
13  %             .                   |
    %        [A]                      |
15  %         |                       |
    %         |                       |
17  %         /\                      \/
    %         |                       |
19  %         |                       |
    %         |                       |
21  %         /\                      \/
    %         /\                      \/
23  %
    %
25  % INPUTS ------------------------------------------------------------
    % G : intensidad del torbellino "Gamma" (m2/s)
27  % RA : coordenadas RA=(xA,yA,zA) del 1er vertice del torbellino
    % RB : coordenadas RB=(xB,yB,zB) del 2o vertice del torbellino
29  % RC : coordenadas RP=(x,y,z) del punto de calculo (metros)
    %
31  % OUTPUTS -----------------------------------------------------------
    % W : componente vertical de la velocidad debida a un torbellino
33  % en herradura
```

```
%% CALCULO DE W ==========================================

% Vector unitario en la direccion de los torbellinos de la estela
% (direccion hacia x=-infinito, vienen desde x=+infinito)
eG = [-1 0 0];

% Vector unitario en la direccion del torbellino de A -> B
eAB = (RB - RA) / norm(RB - RA);

% SE RESULEVE LA VELOCIDAD SUMANDO LAS CONTRIBUCIONES DE LOS
% DIFERENTES FILAMENTOS DE TORBELLINO
% Velocidad debida a la estela desde inf -> A
VA = TorbellinoSemiInfinito(G,RP,RA,eG);

% Velocidad debida al segmento de torbellino desde A -> B
VAB = TorbellinoSemiInfinito(G,RP,RB,eAB) - TorbellinoSemiInfinito(G,
    RP,RA,eAB) ;

% Velocidad debida a la estela desde inf -> A
VB = TorbellinoSemiInfinito(-G,RP,RB,eG);

% Velocidad total en el punto P
V = VA + VAB + VB;

% Todas las velocidades
W = V;
% Componente vertical de la velocidad
%W = V(3);

end
```

Código 4: Torbellino en herradura

B.2.4. Velocidad vertical inducida por un filamento de torbellino semi-infinito

```
function [W] = TorbellinoSemiInfinito(G,RP,RA,eG)

%% DESCRIPCION DE LA FUNCION ==============================
%
% TorbellinoSemiInfinito calcula la velocidad en un punto RP(x,y,z)
% debida a un torbellino semi-infinito de intensidad Gamma = G.
%
% INPUTS ---------------------------------------------------
% G : intensidad del torbellino "Gamma" (m2/s)
% RA : coordenadas RA=(xA,yA,zA) del extremo del filamento
% RP : coordenadas RP=(x,y,z) del punto de calculo (metros)
% eG : vector unitario en direccion del filamento, sentido inf
```

```
%
% OUTPUTS ----------------------------------------------------------------
%W : componente vertical de la velocidad debida a un torb. semi-inf

%% CALCULO DE W ==========================================================
% Vector A -> P
rAP = RP - RA;

% Vector rAP unitario, eAP = rAP/norm(rAP)
eAP = rAP/norm(rAP);

% Producto vectorial p = eG x eAP (de R3)
p = cross(eG,eAP);

% Producto escalar k = eG * eAP (de R)
k = dot(eG,eAP);

% Distancia de A -> P
dAP = norm(rAP);

% Velocidad inducida por el torbellino

W = (G/(4*pi*dAP)) * p / (1 + k);

end
```

Código 5: Torbellino semi-infinito

B.3. Modelo estructural

B.3.1. Generación del emparrillado estructural de alas y estabilizador

```
function [ESTRUCTURA]=GeometriaEstructura(COORD, nx_vector, ny_vector)

%% DESCRIPCION DE LA FUNCION ==============================================

% COORD: array con las coordenadas de los nodos de la celda que forma
    elvemparrillado estructural

% INPUTS =================================================================
% y
% |          NE ............ SE
% |            \              \
% |              \              \
% |                \             \
% |                  \            \
% |                   NW..............SW
% |
% |
% ------------------------------------> x

% INPUT ==================================================================
% condiciones: yNW = ySW; yNE = ySE; 0<yNW < yNE; 0<ySW < ySE
% condiciones: xNW < xSW; xNE < xSE;
% nx_vector: Division de la estructura en direccion x (Nlargueros-1)
% ny_vector: Division de la estructura en direccion y (Ncostillas-1)
% Extraemos coordenadas
%NW = COORD(1,:); Coordenadas [X,Y]
% SW = COORD(2,:); Coordenadas [X,Y]
% SE = COORD(3,:); Coordenadas [X,Y]
% NE = COORD(4,:); Coordenadas [X,Y]

% OUTPUT =================================================================
% ESTRUCTURA.NumNodos = numero total de nodos (N)
% ESTRUCTURA.NumGDL   = numero total de grados de libertad (Nx3)
% ESTRUCTURA.CoordNodos  = coordenadas de los GDL (Nx3). Las filas
    hacen referencia al numero de nodo y las columnas a cada
    coordenada {x,y,z}
% ESTRUCTURA.GDL  = matriz de grados de libertad (Nx3). Las filas
    hacen referencia al numero de nodo y las columnas a cada GDL {
    theta_x, theta_y, w}
% ESTRUCTURA.NumElem = numero total de elementos de union (barras)
% ESTRUCTURA.Conect = matriz de conectividad de la estructura (
    NumElemx2)
% ESTRUCTURA.ConecElem = matriz (Nx1) que contiene la informacion de
    cuantos elementos conecta cada nodo. Fila = N Nodo, columna = N
    Elem
```

```
% ESTRUCTURA.ElemNn = matriz (3x2) que contiene la informacion de
    cuantos nodos hay que unen 2, 3 y 4 elementos. Fila = N Elementos
    unidos (2, 3 o 4), columna = N nodos totales que unen esos
    elementos
%
%
% CROQUIS EMPARRILLADO
%
% ------------------------------------------------------------------> Y
% |        (NW,1)-----------(2)-----------(3)-----------(4)-----------(NE,5)
% |           |              |             |             |              |
% |           |              |             |             |              |
% |           |              |             |             |              |
% |           |              |             |             |              |
% |           |              |             |             |              |
% |           |              |             |             |              |
% |        (SW,6)-----------(7)-----------(8)-----------(9)-----------(SE,10)
% |
% X

ESTRUCTURA.largueros = [];
ESTRUCTURA.costillas = [];
ESTRUCTURA.NumNodos = [];
ESTRUCTURA.NumGDL = [];
ESTRUCTURA.CoordNodos = [];
ESTRUCTURA.GDL = [];
ESTRUCTURA.NumElem = [];
ESTRUCTURA.Conect = [];
ESTRUCTURA.ConecElem = [];
ESTRUCTURA.ElemNn = [];

%% COORDENADAS DE LA CELDA EXTERNA ========================================

NW = COORD(1,:);
SW = COORD(2,:);
SE = COORD(3,:);
NE = COORD(4,:);

%% DIVISION DE LA CELDA EN FUNCION DE LOS LARGUEROS Y COSTILLAS ===========

nx = nx_vector;
ny = ny_vector;
n_largueros = (nx+1);
n_costillas = 2*ny+1;
ESTRUCTURA.largueros = n_largueros;
ESTRUCTURA.costillas = n_costillas;

%% NUMERO DE NODOS, GDL Y MATRIZ DE GDL ===================================

```

```
ESTRUCTURA.NumNodos = n_largueros*n_costillas;
ESTRUCTURA.NumGDL = ESTRUCTURA.NumNodos*3;
ESTRUCTURA.GDL = zeros(ESTRUCTURA.NumNodos,3);

cont = 0;
for i = 1:ESTRUCTURA.NumNodos
    ESTRUCTURA.GDL(i,1) = cont+1;
    ESTRUCTURA.GDL(i,2) = cont+2;
    ESTRUCTURA.GDL(i,3) = cont+3;
    cont = cont+3;
end

%% MATRIZ DE COORDENADAS DE LOS NODOS ==========================

ly = NE(2) - NW(2);
lxW = SW(1) - NW(1);
lxE = SE(1) - NE(1);

% Obtencion coordenadas del cuadrado inicial
dx_unidad = 1/nx;
dy_unidad = 1/ny;

% Nodos de los paneles
vx_0 = 0:dx_unidad:1;
vy_0 = 0:dy_unidad:1;

[Xnodos,Ynodos] = meshgrid(vx_0,vy_0);

% TRANSFORMACION 1
Xnodos = lxW * Xnodos;
Ynodos = ly * Ynodos;

% TRANSFORMACION 2
Xnodos = Xnodos + (1/ly)*(lxE/lxW - 1) * (Xnodos .* Ynodos);

% TRANSFORMACION 3
Xnodos = NW(1) + Xnodos + (1/ly) * (NE(1) - NW(1)) * Ynodos;
Ynodos = NW(2) + Ynodos;

X = Xnodos;
Y = Ynodos;
CoordNodos = [];

for i = 1:size(X,2)
    for j = 1:size(X,1)
        if (Y(j,i) == 0)
            CoordNodos = [CoordNodos ; X(j,i) Y(j,i)];
        else
            CoordNodos = [CoordNodos ; X(j,i) Y(j,i)];
```

```
133                 CoordNodos = [CoordNodos ; X(j,i) -Y(j,i)];
            end
135         end
    end
137
    ESTRUCTURA.CoordNodos= [CoordNodos zeros(size(CoordNodos,1),1)];
139
    %% MATRIZ DE CONECTIVIDAD ==========================================
141
    Conectivity = zeros(ESTRUCTURA.NumNodos,ESTRUCTURA.NumNodos);
143 dy = ly/ny;

145 % Creamos la variable larguero (Nnx1) donde cada nodo lleva asociado
        el numero de larguero al que pertenece

147 larguero = zeros(ESTRUCTURA.NumNodos,1);
    nodos_encastre = 1:n_costillas:ESTRUCTURA.NumNodos;
149
    for i = 1:(n_largueros-1)
151     for j = 1:ESTRUCTURA.NumNodos
            if (j >= nodos_encastre(i)) && (j < nodos_encastre(i+1))
153             larguero(j) = i;
            elseif (j >= nodos_encastre(i+1))
155             larguero(j) = i+1;
            end
157     end
    end
159
    for i = 1:ESTRUCTURA.NumNodos
161     for j = 1:ESTRUCTURA.NumNodos
            if (ESTRUCTURA.CoordNodos(i,2) == ESTRUCTURA.CoordNodos(j,2))
         &&...
163                 (i+n_costillas == j)
                Conectivity(i,j) = 1;
165         elseif (abs(ESTRUCTURA.CoordNodos(i,2)-ESTRUCTURA.CoordNodos(
        j,2))>0) &&...
                    (abs(ESTRUCTURA.CoordNodos(i,2)-ESTRUCTURA.CoordNodos
        (j,2))<=dy+1e-4) &&...
167                 (larguero(i) == larguero(j)) && (i < j)
                Conectivity(i,j) = 1;
169         else
                Conectivity(i,j) = 0;
171         end
        end
173 end

175 [elem_conex1,elem_conex2] = find(Conectivity==1);
    ESTRUCTURA.Conect = [elem_conex1,elem_conex2];
177 ESTRUCTURA.NumElem = size(ESTRUCTURA.Conect,1);
```

```
%% CONTADOR DE ELEMENTOS QUE UNE CADA NODO ========================

[C,~,ic] = unique(ESTRUCTURA.Conect);
a_counts = accumarray(ic,1);
value_counts = [C, a_counts];
ESTRUCTURA.ConecElem = value_counts;
[C,~,ic] = unique(ESTRUCTURA.ConecElem(:,2));
a_counts = accumarray(ic,1);
value_counts = [C, a_counts];
ESTRUCTURA.ElemNn = value_counts;

end
```

Código 6: Generación del emparrillado estructural de superficies aerodinámicas

B.3.2. Acoplamiento estructuras ala, estabilizador y fuselaje

```
function [ESTRUCTURA] = GeometriaAvion(ALA,ESTAB,FUSELAJE,df)

%% DESCRIPCION DE LA FUNCION

% INPUTS ---------------------------------------------------------------
% ALA : estructura de datos que contiene los datos del modelo
    estructural del ala principal
% ESTAB : estructura de datos que contiene los datos del modelo
    estructural del estabilizador horizontal
% FUSELAJE : dimensiones de las 3 partes del fuselaje, nariz, hiato y
    cola. Es un vector 3x1: [nariz hiato cola] con las longitudes de
    cada parte [m]
% df : vector 3x1 con las subdivisiones de cada parte del fuselaje en
    elementos estructurales

% OUTPUTS --------------------------------------------------------------
% ESTRUCTURA.NumNodos = numero total de nodos (N)
% ESTRUCTURA.NumGDL   = numero total de grados de libertad (Nx3)
% ESTRUCTURA.CoordNodos  = coordenadas de los GDL (Nx3). Las filas
    hacen referencia al numero de nodo y las columnas a cada
    coordenada {x,y,z}
% ESTRUCTURA.GDL  = matriz de grados de libertad (Nx3). Las filas
    hacen referencia al numero de nodo y las columnas a cada GDL {
    theta_x, theta_y, w}
% ESTRUCTURA.NumElem = numero total de elementos de union (barras)
% ESTRUCTURA.Conect  = matriz de conectividad de la estructura (
    NumElemx2)
% ESTRUCTURA.ConecElem = matriz (Nx1) que contiene la informacion de
    cuantos elementos conecta cada nodo. Fila = N Nodo, columna = N
    Elem
```

```
% ESTRUCTURA.ElemNn = matriz (3x2) que contiene la informacion de
    cuantos nodos hay que unen 2, 3 y 4 elementos. Fila = N Elementos
    unidos (2, 3 o 4), columna = N nodos totales que unen esos
    elementos
% ESTRUCTURA.ID = identificador de pertenencia de cada elemento: 1 si
     pertenece al ala, 2 al estabilizador y 3 al fuselaje. Sera un
    vector de NumElem x 1

%% UNION DE ALA Y ESTABILIZADOR EN UNA UNICA ESTRUCTURA DE DATOS

% Ala
Nn_w = ALA.NumNodos;
Ngdl_w = ALA.NumGDL;
CN_w = ALA.CoordNodos;
GDL_w = ALA.GDL;
Ne_w = ALA.NumElem;
C_w = ALA.Conect;
CeNn_w = ALA.ElemNn;

% Estabilizador
Nn_e = ESTAB.NumNodos;
Ngdl_e = ESTAB.NumGDL;
CN_e = ESTAB.CoordNodos;
GDL_e = ESTAB.GDL;
Ne_e = ESTAB.NumElem;
C_e = ESTAB.Conect;
CeNn_e = ESTAB.ElemNn;

% Totales ala + estabilizador
Nn = Nn_w + Nn_e;
Ngdl = Ngdl_w + Ngdl_e;
Ne = Ne_w + Ne_e;
CoordNodos = [CN_w; CN_e];
GDL_e = GDL_e + Ngdl_w*ones(size(GDL_e));
GDL = [GDL_w; GDL_e];
C_e = C_e + Nn_w*ones(size(C_e));
Conectivity = [C_w; C_e];

%% INTRODUCCION DEL FUSELAJE

% Nodos y elementos
nariz = FUSELAJE(1,1);... longitud de la nariz [m]
Ne_n = df(1,1);... numero de elementos viga de la nariz
Nn_n = Ne_n;... numero de nodos de la nariz (es igual al de elementos
     porque el nodo de union con el ala ya lo consideramos en el ala
hiato = FUSELAJE(1,2);... longitud del hiato [m]
Ne_h = df(1,2);... numero de elementos viga del hiato
Nn_h = Ne_h - 1;... numero de nodos del hiato
cola = FUSELAJE(1,3);... longitud de la cola [m]
```

```
Ne_c = df(1,3);... numero de elementos viga de la cola
Nn_c = Ne_c;... numero de nodos de la cola (es igual al de elementos
    porque el nodo de union con el estabilizador ya lo consideramos en
     el estabilizador
Nn_f = Nn_n + Nn_h + Nn_c;... nodos totales del fuselaje
Ne_f = Ne_n + Ne_h + Ne_c;... elementos totales del fuselaje

% Matriz de grados de libertad
GDL_f = zeros(Nn_f,3);... matriz de GDL del fuselage
cont = 0;
for i = 1:Nn_f
    GDL_f(i,1) = cont+1;
    GDL_f(i,2) = cont+2;
    GDL_f(i,3) = cont+3;
    cont = cont+3;
end

%% AVION COMPLETO

% Nodos, elementos y grados de libertad
Nn = Nn + Nn_f;
Ne = Ne + Ne_f;
Ngdl = Ngdl + Nn_f*3;
GDL_f = GDL_f + (Ngdl_w+Ngdl_e)*ones(size(GDL_f));
GDL = [GDL; GDL_f];

% Coordenadas y conectividad de los nuevos nodos
CN_f = zeros(Nn_f,3);
CoordNodos = [CoordNodos; CN_f];
C_f = zeros(Nn_f,2);
Conectivity = [Conectivity; C_f];
n_costillas_w = ALA.costillas;
n_costillas_e = ESTAB.costillas;
NE_w = 1:n_costillas_w:Nn_w;... nodos en el empotramiento del ala
NE_e = 1:n_costillas_e:Nn_e;... nodos en el emportramiento del
    estabilizador
NE_e = NE_e + Nn_w*ones(size(NE_e));

% Nariz
for i = 1:Nn_n
    CoordNodos(i+Nn_w+Nn_e,1) = CoordNodos(1,1) - i * nariz/Ne_n;
    if i == 1
        Conectivity(i+Ne_w+Ne_e,1) = i+Nn_w+Nn_e;
        Conectivity(i+Ne_w+Ne_e,2) = NE_w(1);
    else
        Conectivity(i+Ne_w+Ne_e,1) = i+Nn_w+Nn_e;
        Conectivity(i+Ne_w+Ne_e,2) = i+Nn_w+Nn_e-1;
    end
end
```

```
108
% Hiato
110 for i = 1:Nn_h
    CoordNodos(i+Nn_w+Nn_e+Nn_n,1) = CoordNodos(NE_w(end),1) + i *
    hiato/Ne_h;
112     if i == 1
            Conectivity(i+Ne_w+Ne_e+Ne_n,1) = NE_w(end);
114         Conectivity(i+Ne_w+Ne_e+Ne_n,2) = i+Nn_w+Nn_e+Nn_n;
        elseif i == Nn_h
116         Conectivity(i+Ne_w+Ne_e+Ne_n,1) = i+Nn_w+Nn_e+Nn_n-1;
            Conectivity(i+Ne_w+Ne_e+Ne_n,2) = i+Nn_w+Nn_e+Nn_n;
118         Conectivity(i+Ne_w+Ne_e+Ne_n+1,1) = i+Nn_w+Nn_e+Nn_n;
            Conectivity(i+Ne_w+Ne_e+Ne_n+1,2) = NE_e(1);
120     else
            Conectivity(i+Ne_w+Ne_e+Ne_n,1) = i+Nn_w+Nn_e+Nn_n-1;
122         Conectivity(i+Ne_w+Ne_e+Ne_n,2) = i+Nn_w+Nn_e+Nn_n;
        end
124 end

126 % Cola
    for i = 1:Nn_c
128     CoordNodos(i+Nn_w+Nn_e+Nn_n+Nn_h) = CoordNodos(NE_e(end),1) + i *
        cola/Ne_c;
        if i == 1
130         Conectivity(i+Ne_w+Ne_e+Ne_n+Ne_h,1) = NE_e(end);
            Conectivity(i+Ne_w+Ne_e+Ne_n+Ne_h,2) = i+Nn_w+Nn_e+Nn_n+Nn_h;
132     else
            Conectivity(i+Ne_w+Ne_e+Ne_n+Ne_h,1) = i+Nn_w+Nn_e+Nn_n+Nn_h
        -1;
134         Conectivity(i+Ne_w+Ne_e+Ne_n+Ne_h,2) = i+Nn_w+Nn_e+Nn_n+Nn_h;
        end
136 end

138 % Contador de elementos que une cada nodo

140 [C,~,ic] = unique(Conectivity);
    a_counts = accumarray(ic,1);
142 value_counts = [C, a_counts];
    ConecElem = value_counts;
144 [C,~,ic] = unique(ConecElem(:,2));
    a_counts = accumarray(ic,1);
146 value_counts = [C, a_counts];
    ElemNn = value_counts;
148
    % Identificador de elementos y nodos
150 IdElementos = zeros(Ne,1);
    IdElementos(1:Ne_w,1) = 1;... elementos del ala
152 IdElementos(Ne_w:Ne_w+Ne_e,1) = 2;... elementos del estabilizador
    IdElementos(Ne_w+Ne_e:end,1) = 3;... elementos del fuselaje
```

```
154 IdNodos = zeros(Nn,1);
    IdNodos(1:Nn_w,1) = 1;... elementos del ala
156 IdNodos(Nn_w:Nn_w+Nn_e,1) = 2;... elementos del estabilizador
    IdNodos(Nn_w+Nn_e:end,1) = 3;... elementos del fuselaje
158
    %% ESTRUCTURA DE DATOS DEL AVION COMPLETO
160 ESTRUCTURA.NumNodos = Nn;
    ESTRUCTURA.NumGDL = Ngdl;
162 ESTRUCTURA.CoordNodos = CoordNodos;
    ESTRUCTURA.GDL = GDL;
164 ESTRUCTURA.NumElem = Ne;
    ESTRUCTURA.Conect = Conectivity;
166 ESTRUCTURA.ConecElem = ConecElem;
    ESTRUCTURA.ElemNn = ElemNn;
168 ESTRUCTURA.ElemNn_w = CeNn_w;
    ESTRUCTURA.ElemNn_e = CeNn_e;
170 ESTRUCTURA.ID_elem = IdElementos;
    ESTRUCTURA.ID_nodos = IdNodos;
172
    end
```

Código 7: Generación del emparrillado estructural global

B.3.3. Cálculo de la matriz de rigidez

```
    function [K] = RigidezEstructura(ESTRUCTURA)
2
    %% DESCRIPCION DE LA FUNCION
4   % RigidezEstructura calcula la matriz de rigidez de la estructura
    %
6   % INPUTS ----------------------------------------------------------
    % ESTRUCTURA : estructura de datos que contiene los datos del modelo
        de emparrillado estructural de elementos viga Euler-Bernoulli
8   %
    % OUTPUTS ---------------------------------------------------------
10  % K : matriz de rigidez de la estructura (Ngdl x Ngdl)

12  %% CALCULO DE LA MATRIZ DE RIGIDEZ

14  ElemConexion = ESTRUCTURA.Conect;
    NodosGdl = ESTRUCTURA.GDL;
16  NodosCoord = ESTRUCTURA.CoordNodos;
    vecE = ESTRUCTURA.E;
18  vecG = ESTRUCTURA.G;
    CondContorno = ESTRUCTURA.CondContorno;
20  ID = ESTRUCTURA.ID_elem;
    crw = ESTRUCTURA.crw;
22  crt = ESTRUCTURA.crt;
```

```
X = NodosCoord(:,1);
Y = NodosCoord(:,2);
Xmin = min(X);
Xmax = max(X);

% Numero de grados de libertad totales
NumGdl    = numel(NodosGdl);

% Numero de elementos
NumElem = size(ElemConexion,1);


% La rigidez de la estructura es la suma de las rigideces de cada
    elemento sumando cada contribucion por cada gdl (ensamblaje)


%% MATRIZ DE RIGIDEZ CON TODOS LOS GDL
% Inicializacion
Kb = zeros(NumGdl);


for e=1:NumElem
    % Nodos del elemento
    n1 = ElemConexion(e,1);
    n2 = ElemConexion(e,2);

    % Longitud del elemento
    dX = X(n2) - X(n1);
    dY = Y(n2) - Y(n1);
    L = sqrt( dX^2 + dY^2 );

    % Angulo del elemento
    a = atan(dY / dX);

    % Rigidez del elemento
    if (NodosCoord(n1,2) == 0) && (NodosCoord(n2,2) == 0) % fuselaje
    %if ID(e) == 3 % fuselaje
        E = vecE(1,3);
        G = vecG(1,3);
        % Momentos de inercia (consideramos fuselaje cilindrico + 4
    largueros en Z)
        r1 = dim_fuselaje(Xmax,Xmin,crw,X(n1));
        r2 = dim_fuselaje(Xmax,Xmin,crw,X(n2));
        r = (r1 + r2)/2;... radio interno del fuselaje [m]
        t = 0.1*r;... espesor del fuselaje [m]
        R = r+t;... radio externo del fuselaje [m]
        N_l = 4;... numero de largueros a lo largo de la longitud del
     fuselaje
```

```
        b = 0.1*R;... alas de los largueros [m]
        h = b;... alma de los largueros [m]
        t_l = 0.1*h;... espesor de los largueros [m]
        A_l = 2*b*t_l + h*t_l;... areas de los largueros [m]
        I0 = 2*pi*t*R^3 + N_l*A_l*R^2;... momento de inercia respecto del centro del cilindro
        I = I0/2;... Iy = Iz = I
        J = 4*(pi*(R^2))^2/(2*pi*R/t) + N_l/3*(2*b*t_l^3 + h*t_l^3);
    elseif ID(e) == 1 % ala
        E = vecE(1,1);
        G = vecG(1,1);
        % Inercia y rigidez del elemento
        h = 0.12*crw; % suponemos perfil NACA 2412 (espesor max 0.12*crw)
        tc = 0.08*h; % espesor de costillas
        t = 0.08*h; % espesor de largueros
        if a == 0 % costilla
%             I = (1/12)*h*L^3 - (1/12)*((h-2*tc)*(L-2*tc)^3); % [m4]
%             J = 4*(L*h - (L-2*tc)*(h-2*tc))^2 / (2*(L/tc) + 2*(h/tc)); % [m4]
            I = (1/12)*tc^3*h;
            J = (1/3)*tc^3*h; % [m4]
        else % larguero
            b = 0.8*h; % alas de los largueros
            I = (1/12)*t^3*h + 2*((1/12)*b^3*t + b*t*(0)^2); % [m4]
            J = 2*(b*t^3)/3 + (h*t^3)/3; % [m4]
        end

    elseif ID(e) == 2 % estabilizador
        E = vecE(1,2);
        G = vecG(1,2);
        % Inercia y rigidez del elemento
        h = 0.14*crt; % suponemos perfil NACA 0014 (espesor max 0.14*crt)
        tc = 0.12*h; % espesor de costillas
        t = 0.12*h; % espesor de largueros
        if a == 0 % costilla
%             I = (1/12)*h*L^3 - (1/12)*((h-2*tc)*(L-2*tc)^3); % [m4]
%             J = 4*(L*h - (L-2*tc)*(h-2*tc))^2 / (2*(L/tc) + 2*(h/tc)); % [m4]
            I = (1/12)*tc^3*h;
            J = (1/3)*tc^3*h; % [m4]
        else % larguero
            b = 0.8*h; % alas de los largueros
            I = (1/12)*t^3*h + 2*((1/12)*b^3*t + b*t*(0)); % [m4]
            J = 2*(b*t^3)/3 + (h*t^3)/3; % [m4]
        end
    end

```

```
%       E = vecE(1,3);
%       G = vecG(1,3);
%       I = 1.22e-6;
%       J = 4.88e-6;
        EI = E*I;
        GJ = G*J;
        Ke = RigidezElemento(GJ,EI,L,a);

        % Identificacion de los gdl de los nodos
        id = IdentificacionGdlNodos(e, ElemConexion, NodosGdl);

        % Sumamos la rigidez del elemento a la global, unicamente en los
        nodos
        % elegidos por el array id (en fila)
        Kb(id,id) = Kb(id,id) + Ke;
    end

    %% MATRIZ DE RIGIDEZ CON GDL ACTIVOS
    % ELIMINAMOS LOS GDL NULOS DE LA MATRIZ DE RIGIDEZ
    % Nuevo vector con los grados de libertad activos
    % Eliminamos los nulos con la funcion de MATLAB setdiff
    Gdl_Activos = setdiff(1:NumGdl,CondContorno);


    % La matriz de rigidez que recoge unicamente los activos es
    K = Kb(Gdl_Activos,Gdl_Activos);

    end
```

Código 8: Matriz $\mathbf{K}$

```
function [r] = dim_fuselaje(Xmax,Xmin,crw,X)

%% DESCRIPCION DE LA FUNCION

% dim_fuselaje calcula el radio del fuselaje en la seccion X de
    calculo
%
% INPUTS ----------------------------------------------------------
% X : posicion X de la seccion de calculo [m]
% crw : cuerda en la raiz del ala [m]
% Xmin : posicion inicial del fuselaje [m]
% Xmax : posicion final del fuselaje [m]
%
% OUTPUTS ---------------------------------------------------------
% r : radio interno del fuselaje en la seccion de calculo

%% FUNCION DE DIMENSIONAMIENTO DEL FUSELAJE

```

```
Di = 0.5;... diametro inicial [m]
Dc = 1.2;... diametro en la cabina [m]
Df = 0.3;... diametro final [m]
mi = (Dc - Di) / (0 - Xmin);... pendiente del tramo inicial del
    fuselaje
mf = (Df - Dc) / (Xmax - crw);... pendiente del tramo final del
    fuselaje

if X < 0 % x = 0 -> borde de ataque del ala
    D = Di + (X - Xmin) * mi;
elseif (0 <= X) && (X <= crw)
    D = Dc;
elseif (X > crw)
    D = Dc + (X-crw) * mf;
end

r = D/2;

end
```

Código 9: Función del radio del fuselaje para cálculo de la rigidez de la sección

```
function [K] = RigidezElemento(GJ,EI,L,a)

%% DESCRIPCION DE LA FUNCION
% RigidezElemento calcula la matriz de local de cada elemento
%
% INPUTS ------------------------------------------------------------
% GJ : rigidez a torsion del elemento (Nm2)
% EI : rigidez a flexion del elemento (Nm2)
% L : longitud del elemento (m)
% a : angulo del elemento (rad) (a = 0  (i)--------(j) --> Eje X)
%
% OUTPUTS -----------------------------------------------------------
% K : matriz de rigidez de elemento (Ngdl x Ngdl)

%% MATRIZ DE RIGIDEZ BARRA HORIZONAL (a=0)
%
%     K0 = [K11   K12
%           K21   K22]
%     K12 = K21'

    K11= [GJ/L            0             0   ;...
            0        4*EI/L      6*EI/L^2   ;...
            0      6*EI/L^2     12*EI/L^3] ;

    K12= [-GJ/L           0             0  ;...
            0        2*EI/L     -6*EI/L^2  ;...
            0      6*EI/L^2    -12*EI/L^3];
```

```
    %K21= K12';

    K22= [GJ/L              0              0  ;...
             0          4*EI/L     -6*EI/L^2  ;...
             0      -6*EI/L^2      12*EI/L^3];
%end



%% MATRIZ DE CAMBIO DE BASE
% Cosenos y senos del angulo de giro
ca = cos(a);
sa = sin(a);

% Matriz de cambio
T = [ca  -sa   0 ;...
     sa   ca   0 ;...
      0    0   1 ];


%% MATRIZ DE RIGIDEZ BARRA INCLINADA (a <> 0)
%
% K11a = T * K11 * T'
% K12a = T * K12 * T'
% K21a = T * K21 * T'
% K22a = T * K22 * T'

K11a = T' * K11 * T;
K12a = T' * K12 * T;
K21a = K12a';
K22a = T' * K22 * T;

K = [K11a  K12a ; K21a  K22a]; % Globales
```

Código 10: Rigidez de elemento (local)

```
function [N1g,N2g] = FuncionesFormaGlobales(t,L,a)

%% DESCRIPCION DE LA FUNCION
% FuncionesFormaGlobales obtiene las funciones de forma globales de
    la estructura
%
% INPUTS ------------------------------------------------------------
% a : angulo del elemento [-pi,pi],(rad)
% L : longitud del elemento
% t: variable adimensional -1<t<1
%
```

```
% OUTPUTS ------------------------------------------------------------
% N1g : funcion de forma en nudo 1
% N2g : funcion de forma en nudo 2
%
% D = {Theta(t),dw/dt(t),w(t)}' Vector columna con desplazamientos y
    giros locales en barra
% a1 = {Theta1,dw1/dt,w1}' Vector columna con gdl globales en nudo 1
% a2 = {Theta2,dw2/dt,w2}' Vector columna con gdl globales en nudo 2
%
% Funcion de interpolacion: D(t) = N1g(t) * A1  +  N2g(t) * A2
%
% Con:
%
% N1g = T * N1 * T'
% N2g = T * N2 * T'

%% CALCULO DE LAS FUNCIONES DE FORMA GLOBALES
% Funciones de forma locales
[N1,N2] = FuncionesFormaLocales(t,L);

% Matriz de giro. Importante: "a" en [rad]
T = [cos(a)  -sin(a) 0 ; ...
     sin(a)   cos(a) 0 ; ...
          0        0 1 ];


% Funciones de forma globales
N1g = T' * N1 * T;
N2g = T'* N2 * T;
```

Código 11: Matriz de funciones de forma globales

```
function [N1,N2] = FuncionesFormaLocales(t,L)

%% DESCRIPCION DE LA FUNCION
% FuncionesFormaLocales obtiene las funciones de forma locales de la
    estructura
%
% INPUTS ------------------------------------------------------------
% L : longitud del elemento
% t: variable adimensional -1<t<1
%
% OUTPUTS -----------------------------------------------------------
% N1 : funcion de forma en nudo 1
% N2 : funcion de forma en nudo 2
%
% d = {Theta(t),dw/dt(t),w(t)}' Vector columna con desplazamientos y
    giros locales en barra
% a1 = {Theta1,dw1/dt,w1}' Vector columna con gdl locales en nudo 1
```

```
% a2 = {Theta2,dw2/dt,w2}' Vector columna con gdl locales en nudo 2
%
% Funcion de interpolacion:     d(t) = N1(t) * a1  +  N2(t) * a2
%
% Con:
%
% N1 = [H1(t)         0           0
%         0        A1(t)   (L/2)B1(t)
%         0   (2/L)A1'(t)      B1'(t)]
%
% N2 = [H2(t)         0           0
%         0        A2(t)   (L/2)B2(t)
%         0   (2/L)A2'(t)      B2'(t)]

%% CALCULO DE LAS FUNCIONES DE FORMA LOCALES
% Funciones de forma interpolacion movimientos TORSIONALES
H1 = (1/2)*(1 - t);
H2 = (1/2)*(1 + t);

% Funciones de forma interp. movimientos y giros transversales
A1 = (1/4)*(2 - 3*t + t^3);
dA1 = (1/4)*(- 3 + 3*t^2);
B1 = (1/4)*(1 - t - t^2 + t^3);
dB1 = (1/4)*(0 - 1 - 2*t + 3*t^2);
A2 = (1/4)*(2 + 3*t - t^3);
dA2 = (1/4)*(0 + 3 - 3*t^2);
B2 = (1/4)*(-1 - t + t^2 + t^3);
dB2 = (1/4)*(0 - 1 + 2*t + 3*t^2);


% Montaje de las funciones de forma
N1 = [H1          0           0    ;...
        0        dB1      (2/L)*dA1 ;...
        0   (L/2)*B1       A1]    ;
N2 = [H2          0           0    ;...
        0        dB2      (2/L)*dA2 ;...
        0   (L/2)*B2       A2]    ;
```

Código 12: Matriz de funciones de forma locales

```
function [id] = IdentificacionGdlNodos(e,MatrizConexion,
    MatrizGdlNodos)

%% DESCRIPCION DE LA FUNCION
% IdentificacionGdlNodos identifica los gdl de cada nodo
%
% Dado un determinado elemento se obtiene el vector id con la forma
%
% id = [id(1),...,id(n)]
```

```
%
% donde
% n: numero de grados de libertad que suman todos los nodos de
%    el elemento "e".
% id(j): Identificacion del grado de libertad j-esimo del elemento
% 1 <= id(j) <= N , donde N: numero de gdl de toda la estructura
% 1 <= j <= Ne, Ne: numero de grados de libertad del elemento.
% Si el elemento tiene 2 nodos y 3gdl por nodo, Ne = 2*3 = 6
% Si el elemento es plano con 3 nodos y 3 gdl por nodo, Ne = 3*3 = 9.
%
% INPUTS ----------------------------------------------------------
% e : numero de elemento
% MatrizConexion : matriz de conectividad de la estructura (Ne x 2)
% MatrizGdlNodos : matriz de gdl de la estructura (Nn x 3)
%
% OUTPUTS ---------------------------------------------------------
% id : vector con la identificacion de cada gdl de los elementos

%% IDENTIFICACION DE GDL
% Identificacion de nodos asociados al elemento "e" (uso de la matriz
    de
% conexion de nodos y elementos
% La fila "e" es la que contiene los nodos del elemento "e"
% Nodo 1:
  n1 = MatrizConexion(e,1);
% Nodo 2:
  n2 = MatrizConexion(e,2);
% Si hubiera mas nodos por elemento se repetiria el proceso (bucle)

% Obtenemos grados de libertad de cada nodo

% Grados de libertad nodo n1
  id1 = MatrizGdlNodos(n1,:);
% Grados de libertad nodo n1
  id2 = MatrizGdlNodos(n2,:) ;
% Grados de libertad
  id = [id1 , id2];

end
```

Código 13: Identificación de los grados de libertad de cada nodo

B.4. Acoplamiento aeroelástico

B.4.1. Modelo de acoplamiento fluido-estructura por *splines*

```
function [Rs,Nodos_sup,gdl_nodos_sup] = MatrizRs(id_sup,id_nodos,
    Matriz_GDL,Matriz_CoordNodos)

%% DESCRIPCION DE LA FUNCION

% MatrizRs calcula la matriz Rs que relaciona el desplazamiento
% vertical de los gdl estructurales contenidos en una superfice
% con todos los gdl de la estructura. Es una matriz de 0s y 1s,
% de dimensiones (Ngdl_sup + 3)*Ngdl, donde los 1s indican que
% dicho gdl se encuentra contenido en la superficie de calculo
%
% INPUTS ----------------------------------------------------------
% id_sup : identificador de la superficie (1=ala, 2=estabilizador)
% Matriz_GDL : matriz de gdl de la estructura (Nnodos x 3)
% Matriz_CoordNodos : matriz con las coordenadas de los nodos
%
% OUTPUTS ---------------------------------------------------------
% Nodos_sup : nodos contenidos en la superficie
% gdl_nodos_sup : gdl de desplazamiento vertical de los nodos
% contenidos en la superficie
% Rs : matriz con los gdl de desplazamiento vertical estructurales
% contenidos en la superficie de calculo

%% CALCULO DE LA MATRIZ Rs

% Datos de la estructura
Nn = size(Matriz_GDL,1);... numero de nodos totales de la estructura
Ngdl = Nn*3;... numero de gdl totales de la estructura

% Calculo de la matriz de relacion
Nodos_sup = [];... vector con los nodos contenidos en cada superfice

for i = 1:Nn
    if id_nodos(i) == id_sup
        Nodos_sup = [Nodos_sup i];
    end
end

gdl_nodos_sup = Matriz_GDL(Nodos_sup,3);
Rs = zeros(length(Nodos_sup),Ngdl);

for i = 1:length(Nodos_sup)
    for j = 1:Ngdl
        if (gdl_nodos_sup(i) == j)
            Rs(i,j) = 1;
```

```
        end
    end
end

ceros = zeros(3,Ngdl);
Rs = [ceros; Rs];

end
```

Código 14: Matriz $\mathbf{R}_\sigma$

```
function [Bs] = Spline(CoordNodos_sup)

%% DESCRIPCION DE LA FUNCION
% Spline calcula la matriz Bs, spline que define la deformada de una
    superficie definida por los desplazamientos de los nodos que se
    encuentran en ella, es decir, la spline pasa los puntos
    CoordNodos_panel
% INPUTS ----------------------------------------------------------
% CoordNodos_sup : matriz de coordenadas (x,y) de los nodos
% contenidos en la superficie (Nnodos_superficie x 2)
%
% OUTPUT ----------------------------------------------------------
% Bs : spline que define la deformada de la superficie (Nnodos_sup+3)
    x(Nnodos_sup+3)

%% CALCULO DE LAS MATRICES QUE DEFINEN LA SPLINE DE INTERPOLACION
Nns = size(CoordNodos_sup,1);
Bs = zeros(Nns,Nns+3);
Xnodos = CoordNodos_sup(:,1)';
Ynodos = CoordNodos_sup(:,2)';
unos = ones(1,Nns);

for i = 1:Nns
    Xn = Xnodos(i);
    Yn = Ynodos(i);
    Ys = SplineFunction(Xn,Yn,CoordNodos_sup);
    Ys = Ys';
    Bs(i,:) = Ys;
end

fila1 = [0 0 0 unos];
fila2 = [0 0 0 Xnodos];
fila3 = [0 0 0 Ynodos];

Bs = [fila1; fila2; fila3; Bs];

end
```

Código 15: Matriz $\mathbf{B}_\sigma$

```
function [Ys] = SplineFunction(X,Y,CoordNodos_sup)

%% DESCRIPCION DE LA FUNCION

% SplineFunction calcula la matriz Ys, vector de interpolacion de una
    superficie
%
% INPUTS ----------------------------------------------------------------
% X : coordenada X cualquiera de la superficie de calculo
% Y : coordenada Y cualquiera de la superficie de calculo
% CoordNodos_sup : coordenadas de los nodos contenidos en la
% superficie de calculo
%
% OUTPUT ----------------------------------------------------------------
% Ys : vector de interpolacion de la superficie (Nnodos_sup+3 x 1)

%% CALCULO DE LAS MATRICES QUE DEFINEN LA SPLINE DE INTERPOLACION

Nns = size(CoordNodos_sup,1);
phi = zeros(Nns,1);

for i = 1:Nns
    Xn = CoordNodos_sup(i,1);
    Yn = CoordNodos_sup(i,2);
    Ri = sqrt((X-Xn)^2 + (Y-Yn)^2);
    if Ri == 0
        phi(i,1) = 0;
    else
        phi(i,1) = Ri^2 * log(Ri^2);
    end
end

Ys =[1 ; X ; Y ; phi];

end
```

Código 16: Matriz ψ_σ

```
function [DYs] = DSplineFunction(X,Y,CoordNodos_sup)

%% DESCRIPCION DE LA FUNCION

% DSplineFunction calcula la derivada de la matriz Ys repecto de X
%
% INPUTS ----------------------------------------------------------------
% X : coordenada X cualquiera de la superficie de calculo
% Y : coordenada Y cualquiera de la superficie de calculo
% CoordNodos_sup : coordenadas de los nodos contenidos en la
    superficie de calculo
```

```
%
% OUTPUT ----------------------------------------------------------------
% DYs : derivada del vector de interpolacion de la superficie
    respecto de la coordenada X (Nnodos_sup+3 x 1)

%% CALCULO DE LAS MATRICES QUE DEFINEN LA SPLINE DE INTERPOLACION

Nns = size(CoordNodos_sup,1);... numero de nodos contenidos en la
    superficie
Dphi = zeros(Nns,1);

for i = 1:Nns
    Xn = CoordNodos_sup(i,1);
    Yn = CoordNodos_sup(i,2);
    Ri = sqrt((X-Xn)^2 + (Y-Yn)^2);
    if Ri == 0
        Dphi(i,1) = 0;
    else
        Dphi(i,1) = 2 * (X-Xn) * (1 + log(Ri^2));
    end
end

DYs =[0 ; 1 ; 0 ; Dphi];

end
```

Código 17: Matriz $\frac{\partial \psi_\sigma}{\partial x}$

B.4.2. Cálculo de la matriz de masa

```
function [M] = MatrizMasa(PANELES,ESTRUCTURA)

%% DESCRIPCION DE LA FUNCION
% MatrizMasa calcula la matriz de masa del conjunto estructura-
    superficie aerodinamica de un ala
%
% INPUTS ----------------------------------------------------------------
% ESTRUCTURA : estructura de datos que contiene los datos del modelo
% de emparrillado estructural de elementos viga Euler-Bernoulli
% PANELES : estructura de datos que contiene la informacion del
% modelo aerodinamico de paneles (VLM)
%
% OUTPUTS ---------------------------------------------------------------
% M : matriz de masa ala + estabilizador (Ngdl x Ngdl)

%% MASA TOTAL DE ALA, ESTABILIZADOR Y FUSELAJE

% Ala
```

```
masa_bw = PANELES.m_w;... masa distribuida en el ala [kg/m]
bw = PANELES.bw;... envergadura del ala [m]
masa_tot_w = masa_bw * bw;... masa total del ala [kg]
% Estabilizador
masa_be = PANELES.m_e;... masa distribuida en el estabilizador [kg/m]
be = PANELES.be;... envergadura del estabilizador [m]
masa_tot_e = masa_be * be;... masa total del estabilizador [kg]
% Fuselaje
mf = PANELES.m_f;... masa total del fuselaje [kg]

%% DATOS NECESARIOS DEL MODELO AERODINAMICO

% XG = PANELES.CoordCG(:,1);... coordenadas X del CDG de cada panel
% YG = PANELES.CoordCG(:,2);... coordenadas Y del CDG de cada panel
N_sup = PANELES.N_sup;... numero de superficies aerodinamicas del
    modelo
% CoordEsquinas = PANELES.CoordNodos;... coordenadas de las esquinas
    de cada panel [xNW, yNW, xSW, ySW, xSE, ySE, xNE, yNE]

%% DATOS NECESARIOS DEL MODELO ESTRUCTURAL

NumGdl = ESTRUCTURA.NumGDL;... numero total de grados de libertad
Matriz_GDL = ESTRUCTURA.GDL;... matriz de gdl de la estructura
Matriz_CoordNodos = ESTRUCTURA.CoordNodos;... matriz de coordenadas
    de los nodos de la estructura (Nn x 3)
Conectividad = ESTRUCTURA.Conect;... matriz de conectividad
CondContorno = ESTRUCTURA.CondContorno;... matriz de CC
% IdElementos = ESTRUCTURA.ID_elem;... vector de identificacion de
    elementos
IdNodos = ESTRUCTURA.ID_nodos;... vector de identificacion de nodos
ConecElem = ESTRUCTURA.ConecElem;... matriz de cuantos elementos une
    cada nodo
ElemNn_w = ESTRUCTURA.ElemNn_w;... matriz de cuantos nodos hay de
    union de (2, 3 o 4) elementos en el ala
ElemNn_e = ESTRUCTURA.ElemNn_e;... matriz de cuantos nodos hay de
    union de (2, 3 o 4) elementos en el estabilizador

%% DISTRIBUCION DE LA MASA EN LOS NODOS DE LAS SUPERFICIES
    AERODINAMICAS

% Ala
Nn2_w = ElemNn_w(1,2);... numero total de nodos que unen 2 elementos
Nn3_w = ElemNn_w(2,2);... numero total de nodos que unen 3 elementos
if size(ElemNn_w,1) < 3
    Nn4_w = 0;
else
    Nn4_w = ElemNn_w(3,2);... numero total de nodos que unen 4
    elementos
end
```

```
mn2_w = masa_tot_w / (Nn2_w + (3/2)*Nn3_w + (4/2)*Nn4_w);
mn3_w = (3/2) * mn2_w;
mn4_w = (4/2) * mn2_w;
% Estabilizador
Nn2_e = ElemNn_e(1,2);... numero total de nodos que unen 2 elementos
Nn3_e = ElemNn_e(2,2);... numero total de nodos que unen 3 elementos
if size(ElemNn_e,1) < 3
    Nn4_e = 0;
else
    Nn4_e = ElemNn_e(3,2);... numero total de nodos que unen 4
    elementos
end
mn2_e = masa_tot_e / (Nn2_e + (3/2)*Nn3_e + (4/2)*Nn4_e);
mn3_e = (3/2) * mn2_e;
mn4_e = (4/2) * mn2_e;
% Agrupamos las masas en una matriz (la primera columna es el ala y
% la segunda el estabilizador
masas_puntuales = [mn2_w mn2_e;mn3_w mn3_e;mn4_w mn4_e];

%% CALCULO DE LA MATRIZ DE MASA (ALA Y ESTABILIZADOR)

% Inicializamos la variable M
Mb = 0;

for i = 1:N_sup
    id_sup = i;... id_sup = 1 ala, = 2 estabilizador
    [Rs,Nodos_sup,~] = MatrizRs(id_sup,IdNodos,Matriz_GDL,
    Matriz_CoordNodos);
    CoordNodos_sup = Matriz_CoordNodos(Nodos_sup,1:2);
    ConecElem_sup = ConecElem(Nodos_sup,:);
    Bs = Spline(CoordNodos_sup);
    Nn_sup = size(CoordNodos_sup,1);
    for j = 1:Nn_sup
        Ys = SplineFunction(CoordNodos_sup(j,1),CoordNodos_sup(j,2),
    CoordNodos_sup);
        Nz = (Ys' * inv(Bs) * Rs)';
        if ConecElem_sup(j,2) == 2
            mj = masas_puntuales(1,i);
        elseif ConecElem_sup(j,2) == 3
            mj = masas_puntuales(2,i);
        else
            mj = masas_puntuales(3,i);
        end
        Mb = Mb + mj * (Nz * Nz');
    end
end

%% MATRIZ DE MASA DEL FUSELAJE
```

```
Nn = size(Matriz_CoordNodos,1);
Nodos_fuselaje = [];

for i = 1:Nn
    if (Matriz_CoordNodos(i,2) == 0) % Si Y = 0, el nodo pertenece al
    fuselaje
        Nodos_fuselaje = [Nodos_fuselaje i];
    end
end

Xf = Matriz_CoordNodos(Nodos_fuselaje,1);
mj = dist_fuselaje(Xf,mf,1);
gdl_nodos_fuselaje = Matriz_GDL(Nodos_fuselaje,3);

Mf = zeros(NumGdl,NumGdl);
Mf(gdl_nodos_fuselaje,gdl_nodos_fuselaje) = diag(mj);

%% MATRIZ DE MASA GLOBAL
Mg = Mb + Mf;

%% MATRIZ DE MASA CON GDL ACTIVOS
% ELIMINAMOS LOS GDL NULOS DE LA MATRIZ DE MASA
% Nuevo vector con los grados de libertad activos
% Eliminamos los nulos con la funcion de MATLAB setdiff
Gdl_Activos = setdiff(1:NumGdl,CondContorno);

% La matriz de rigidez que recoge unicamente los activos es
M = Mg(Gdl_Activos,Gdl_Activos);


end
```

Código 18: Matriz **M**

```
function [mj] = dist_fuselaje(Xf,mf,tipo)

%% DESCRIPCION DE LA FUNCION
% Dist_Fuselaje calcula la masa de cada nodo del fuselaje en funcion
% de la distribucion de masa real, para la cual el CG del avion se
% encuentra a 2.56m desde la nariz. Para las masas del avion
% consideradas, el CG del fuselaje debe estar a 1.60m de la nariz.
% Tambien lo calculo para una distribucion uniforme de masa a lo
% largo del fuselaje
%
% INPUTS ----------------------------------------------------------
% Xf : vector con la posicion X de cada nodo del fuselaje (Nn_fx1)
% mf : masa total del fuselaje
% tipo : tipo de distribucion de masa (masa variable=1, masa uniforme
    =2)
```

```
%
% OUTPUTS ------------------------------------------------------------------
% mj : vector con la masa nodal correspondiente a cada nodo del
    fuselaje

%% CALCULO DE LA DISTRIBUCION DE MASA NODAL DEL FUSELAJE

if tipo == 1 % masa variable
    Xf = sort(Xf);
    CGf = 1.75 + Xf(1);... posicion del CG del fuselaje respecto de
    la nariz
    PosAbs = abs(Xf - CGf);
    PosPercent = PosAbs/sum(PosAbs);
    MassPercent = (1./PosPercent) ./ sum(1./PosPercent);
    mj = MassPercent*mf;
elseif tipo == 2 % masa distribuida uniformemente entre nodos
    Nnf = length(Xf);
    mj = mf/Nnf * ones(Nnf,1);
end

end
```

Código 19: Distribución de masa del fuselaje

B.4.3. Cálculo de las matrices de acoplamiento dinámicas

```
function [Du,Dv,Dz] = MatricesAcoplamiento_Dinamicas(ESTRUCTURA,
    PANELES)

%% DESCRIPCION DE LA FUNCION
% MatricesAcoplamiento_Dinamicas calculan las matrices de
% acoplamiento fluido-estructura de una superficie alar
%
% INPUTS -------------------------------------------------------------------
% ESTRUCTURA : estructura de datos que contiene los datos del modelo
% de emparrillado estructural de elementos viga Euler-Bernoulli
% PANELES : estructura de datos que contiene la informacion del
    modelo
% aerodinamico de paneles (VLM)
%
% OUTPUTS
    ------------------------------------------------------------------
% Du : Matriz acomplamiento dws/dx = Du * u (Np x Ngdl)
% Dv : Matriz acomplamiento (1/U)*dws/dt = (1/U)*Dv*du/dt (Np x Ngdl)
% Dz : Matriz de acoplamiento zA = Dz.u (Np x Ngdl)

%% DATOS NECESARIOS
```

```
% MODELO AERODINAMICO
Np = PANELES.NumPaneles;... numero de paneles totales
XCA = PANELES.CoordCA(:,1);... coordenadas X de los CA
YCA = PANELES.CoordCA(:,2);... coordenadas Y de los CA
XC = PANELES.CoordControl(:,1);... coordenadas X de los p.c.
YC = PANELES.CoordControl(:,2);... coordenadas Y de los p.c.
N_sup = PANELES.N_sup;... numero de superficies aerodinamicas del
    modelo
CoordEsquinas = PANELES.CoordNodos;... coordenadas de las esquinas de
     cada panel [xNW, yNW, xSW, ySW, xSE, ySE, xNE, yNE]

% ESTRUCTURAL
NumGdl = ESTRUCTURA.NumGDL;... numero total de grados de libertad
Matriz_GDL = ESTRUCTURA.GDL;... matriz de gdl de la estructura (Nnx3)
Matriz_CoordNodos = ESTRUCTURA.CoordNodos;... matriz de coordenadas
    de los nodos de la estructura (Nn x 3)
IdNodos = ESTRUCTURA.ID_nodos;... vector de identificacion de nodos
CondContorno = ESTRUCTURA.CondContorno;... vector de gdl nulos por CC

%% CALCULO DE LAS MATRICES DE ACOPLAMIENTO DINAMICAS

% Inicializamos las matrices
Du = [];
Dv = [];
Dz = [];

for i = 1:N_sup
    id_sup = i;... id_sup = 1 ala, = 2 estabilizador
    [Rs,Nodos_sup,~] = MatrizRs(id_sup,IdNodos,Matriz_GDL,
    Matriz_CoordNodos);
    CoordNodos_sup = Matriz_CoordNodos(Nodos_sup,1:2);
    Bs = Spline(CoordNodos_sup);
    paneles_sup = (PANELES.IdSuperficie == i);
    Np_sup = size(CoordEsquinas(paneles_sup,:),1);
    XCA_sup = XCA(paneles_sup);
    YCA_sup = YCA(paneles_sup);
    XC_sup = XC(paneles_sup);
    YC_sup = YC(paneles_sup);
    Dz_sup = zeros(Np_sup,NumGdl);
    Du_sup = zeros(Np_sup,NumGdl);
    Dv_sup = zeros(Np_sup,NumGdl);
    for j = 1:Np_sup
        YsCA = SplineFunction(XCA_sup(j),YCA_sup(j),CoordNodos_sup);
        YsC = SplineFunction(XC_sup(j),YC_sup(j),CoordNodos_sup);
        DYsC = DSplineFunction(XC_sup(j),YC_sup(j),CoordNodos_sup);
        NzA = (YsCA' * inv(Bs) * Rs)';
        NzC = (YsC' * inv(Bs) * Rs)';
        DNzC = (DYsC' * inv(Bs) * Rs)';
        Dz_sup(j,:) = NzA';
```

```
            Du_sup(j,:) = DNzC';
            Dv_sup(j,:) = NzC';
        end
        Dz = [Dz; Dz_sup];
        Du = [Du; Du_sup];
        Dv = [Dv; Dv_sup];
end

%% MATRICES CON GDL ACTIVOS
% ELIMINAMOS LOS GDL NULOS DE LA MATRIZ DE MASA
% Nuevo vector con los grados de libertad activos
% Eliminamos los nulos con la funcion de MATLAB setdiff
Gdl_Activos = setdiff(1:NumGdl, CondContorno);

% Las matrices que recogen unicamente los activos es
Du = Du(:, Gdl_Activos);
Dv = Dv(:, Gdl_Activos);
Dz = Dz(:, Gdl_Activos);

end
```

Código 20: Cálculo de las matrices $\mathbf{D_u}$, $\mathbf{D_v}$, $\mathbf{D_z}$

B.5. Cálculo vibraciones libres

B.5.1. Resolución del problema de vibraciones libres

```
function [wm,V] = ModosVibracionLibre(h,m,ESTRUCTURA,M,K)

%% DESCRIPCION DE LA FUNCION
% ModosVibracionLibre calcula los autovalores y autovectores del
% problema de vibraciones libres de una estructura. Tambien
% representa la deformada modal de la estructura.
%
% INPUTS ------------------------------------------------------------
% h : factor de escalado para la representacion de la deformada
% m : numero de modos a obtener
% ESTRUCTURA : estructura de datos que contiene los datos del modelo
% de emparrillado estructural de elementos viga Euler-Bernoulli
% M : matriz de masa de la estructura
% K : matriz de rigidez de la estructura
%
% OUTPUTS -----------------------------------------------------------
% wm : Frecuencias naturales de vibracion de la estructura [Hz]
% V : vector que define la deformada de cada modo de vibracion

%% DATOS NECESARIOS DE LA ESTRUCTURA

ElemConexion = ESTRUCTURA.Conect;
NodosGdl = ESTRUCTURA.GDL;
NodosCoord = ESTRUCTURA.CoordNodos;
CondContorno = ESTRUCTURA.CondContorno;

%% RESOLUCION DEL PROBLEMA DE AUTOVALORES ============================
%
% Se resuelve el problema de autovalores generalizados tomando hasta
% los m primeros modos

% Se resuelven el problema de autovalores generalizados
[V,J] = eig(K,M);

% n = size(J,1);
%
% for j = 1:n
%     xj = V(:,j);
%     a = 1 / sqrt(xj'* M * xj);
%     vj = a * xj;
%     V(:,j) = vj;
% end

% V' * M * V; % Comprobacion: debe dar la identidad (normalizamos)
```

```
% Paso de rad/s a Hz y ordenacion de menor a mayor
w = sqrt(diag(J))/(2*pi);... paso de rad/s a Hz
[w,id] = sort(w);
V = V(:,id);

% Discretizacion de cada elemento
% (para la deformada)
 t = [-1:0.1:1]';
 if length(w) < m
     m = length(w);
 end
 wm = zeros(m,1);
for k=1:m
% Modo solicitado
 dm = V(:,k);
 wm(k) = w(k);
 DeformadaEstructura(k,... Numero de modo
                     dm,... Vector de grados de libertad activos
                     wm(k),... Frecuencia natural del modo
                     t,... Discretizacion del elemento
                     h,... Factor de escala
                     ElemConexion,... Matriz de conexion de elementos
                     NodosCoord,... Matriz de coordenadas de nodos
                     NodosGdl,... Matriz con gdls de cada nodo
                     CondContorno);... Vector fila con gdl fijos
 end
```

Código 21: Cálculo de frecuencias naturales y representación de modos de vibración

B.6. Cálculo aeroelasticidad estática

B.6.1. Resolución del problema de autovalores y autovectores asociado al problema aeroelástico estático

```
function [uD,V]=ModosDivergencia(h,m,K,As,ESTRUCTURA,PANELES,rho_inf)

%% DESCRIPCION DE LA FUNCION
% ModosDivergencia resuelve el problema de autovalores y autovectores
% asociado a la inestabilidad por divergencia
%
% INPUTS ----------------------------------------------------------------
% h : factor de escalado para la representacion de la deformada
% m : numero de modos a obtener
% ESTRUCTURA : estructura de datos que contiene los datos del modelo
% de emparrillado estructural de elementos viga Euler-Bernoulli
% PANELES : estructura de datos que contiene la informacion del
% modelo de paneles aerodinamico
% As : matriz que relaciona estructura con paneles = 2 * Dz' * DY *
    inv(H) * Du
% K : matriz de rigidez de la estructura
% rho_inf : densidad del aire en el infinito [kg/m3]
%
% OUTPUTS ---------------------------------------------------------------
% uD : Velocidad de inestabilidad [m/s]
% V : vector que define la deformada de cada modo de vibracion

%% RESOLUCION DEL PROBLEMA DE AUTOVALORES ==============================
%
% Se resuelve el problema de autovalores generalizados tomando hasta
% los m primeros modos

% Se resuelven el problema de autovalores generalizados
[V,J] = eig(K,As);

% V' * M * V; % Comprobacion: debe dar la identidad (normalizamos)

% Ordenacion de menor a mayor presion de divergencia y conversion a
% velocidad de vuelo
[qD,id] = sort(diag(J));
V = V(:,id);
qD = qD(imag(qD)==0);
V = V(:,imag(qD)==0);
qD = qD(qD>0);
V = V(:,qD>0);
uD = sqrt(2 * qD / rho_inf);

%% REPRESENTACION DE LOS MODOS SOLICITADOS
```

```
43 vec_div = V(:,1:m);
   u_div = uD(1:m);
45 %DibujoModosPaneles(PANELES,ESTRUCTURA, u_div , vec_div ,h,2);
   DibujoModosEstPaneles(ESTRUCTURA,PANELES, u_div , vec_div ,h,2);
47

 end
```

Código 22: Cálculo de la velocidad de divergencia y representación del modo de divergencia

B.7. Cálculo aeroelasticidad dinámica

B.7.1. Resolución del sistema matricial espacio-estado del problema aeroelástico dinámico

```
function [w,g,k] =...
SolucionAutovalores_Metodo_EspacioEstado(A,B,C,M,K,zeta,V,rho,m)

%% DESCRIPCION DE LA FUNCION
% SolucionAutovalores_Metodo_espacioEstado es una funcion que aplica
% el metodo epacio-estado al problema de aeroelasticidad dinamica
% para resolverlo en el dominio de la frecuencia para cada velocidad
% de calculo
%
% INPUTS ---------------------------------------------------------------
% A, B, C : matrices de las fuerzas generalizadas
% M, K : matriz de masa y rigidez respectivamente
% zeta : amortiguamiento artificial
% V : velocidad de calculo (m/s)
% rho : densidad del flujo libre (kg/m3)
% m : numero de modos de la base modal
%
% OUTPUTS --------------------------------------------------------------
% w : matriz de frecuencias (2N x numel(V))
% g : matriz de amortiguamiento (2N x numel(V))
% k : matriz de frecuencias reducidas (2N x numel(V))


%% MATRICES DEL SISTEMA ================================================
% Matrices equivalentes sistema
% Mequiv * u'' + Cequiv * u' + Kequiv * u = 0
 Mequiv = M - rho * C;
 Dequiv = - rho * V * B;
 Kequiv = K - rho * V^2 * A;
 Identity = eye(size(Mequiv,1));
 Ceros = zeros(size(Mequiv,1));


%% SISTEMA ESPACIO-ESTADO ==============================================
% SOLUCION del problema
% A_EspEstado * z' + B_EspEstado * z = 0
A_EspacioEstado = [Identity Ceros;Ceros Mequiv];
B_EspacioEstado = [Ceros -Identity;Kequiv Dequiv];

[autovalores] = eig(B_EspacioEstado,A_EspacioEstado);

%% TRANSFORMACION AUTOVALORES s --> [w,g,k]
```

```
w = imag(autovalores);
g = real(autovalores);
k = w / V;

end
```

Código 23: Cálculo de los autovalores (frecuencias y amortiguamientos) dada una velocidad de vuelo

B.7.2. Obtención de las curvas de flameo

```
function [w_NonSteady, g_NonSteady, k_NonSteady] = CurvasFlameo(A,B,C,K
    ,M,V, zeta, rho,m)

%% DESCRIPCION DE LA FUNCION
% CurvasFlameo calcula y representa las curvas de flameo asociadas al
% problema de aeroelasticidad dinamica en aeronaves de ala fija a
% partir de un modelo cuasi-estacionario
%
% INPUTS ----------------------------------------------------------------
% A, B, C : matrices de las fuerzas generalizadas
% M, K : matrices de masa y rigidez
% zeta : amortiguamiento artificial
% rho : densidad del aire en el infinito
% m : numero de modos a calcular
%
% OUTPUTS ---------------------------------------------------------------
% w_NonSteady : vector columna con frecuencias propias (complejas)
% g_NonSteady : vector columna con ratios de amortiguamiento
    complejos
% k_NonSteady : frecuencia reducida respecto a la envergadura k = w *
     b/U

%% CURVAS DE FLAMEO. ANALISIS NO-ESTACIONARIO CON FREC. REDUCIDA DE
    REFERENCIA

w = [];
g = [];
k = [];

% Barremos la lista de velocidades
numel_V = numel(V);

for j=1:numel_V

 [w,g,k] =...
SolucionAutovalores_Metodo_EspacioEstado(A,B,C,M,K, zeta,V(j),rho,m);
```

```
 n_freq = size(w,1);

% Anadimos el nuevo vector de frecuencias a la matriz
if j==1
    w_lista = w;
    g_lista = g;
    kappa_lista = k;
    % cuerda de referencia cr = cm (cuerda media)
    % el valor obtenido k es respecto a la envergadura.
    % un valor mas realista es kappa = w * cr / U = k * (cr/b);
    %kappa_lista = k * (cm / b);
else
    w_lista = [w_lista , w];
    g_lista = [g_lista , g];
    kappa_lista = [kappa_lista , k];
end


end

w_NonSteady = w_lista;
g_NonSteady = g_lista;
k_NonSteady = kappa_lista;

end
```

Código 24: Cálculo de los autovalores (frecuencias y amortiguamientos) para cada velocidad de vuelo

```
function [w_plot,g_plot] = DibujaCurvasFlameo(w,g,V)

%% DESCRIPCION DE LA FUNCION
% DibujaCurvasFlameo es una funcion que calcula las frecuencias y
% amortiguamientos para representar las curvas de flameo
%
% INPUTS ------------------------------------------------------------
% w : matriz de frecuencias (2N x numel(V))
% g : matriz de amortiguamiento (2N x numel(V))
% V : vector de velocidades
%
% OUTPUTS -----------------------------------------------------------
% Curvas de flameo : Frecuencias (w)vsV y Amrotiguamientos (g)vsV
% w_plot : frecuencias a representar en las curvas
% g_plot : amortiguamiento a representar en las curvas

%% CALCULO W Y G PARA REPRESENTACION DE LAS CURVAS DE FLAMEO

w_plot = [];
g_plot = [];
```

```
21
for k = 1:numel(V)
23      s = - g(:,k) + 1i*w(:,k);
        [~,id] = sort(s);
25      a = w(id,k);
        b = g(id,k);
27
        w_plot = [w_plot a];
29      g_plot = [g_plot b];
end
31
end
```

Código 25: Representación gráfica de las curvas de flameo

B.7.3. Cálculo de la velocidad de divergencia y flameo a partir de la representación gráfica de las curvas de flameo

```
1  function [Ud,Uf,wf] = Calcula_Divergencia_y_Flameo(w,g,V)

3  %% DESCRIPCION DE LA FUNCION
   % Calcula_Divergencia_y_Flameo es una funcion que calcula el punto de
5  % divergencia y flameo a partir de las curvas de flameo dadas por las
   % frecuencias y amortiguamientos para cada velocidad de vuelo
7  %
   % INPUTS ------------------------------------------------------------
9  % w : matriz de frecuencias (2N x numel(V))
   % g : matriz de amortiguamientos (2N x numel(V))
11 % V : vector de velocidades
   %
13 % OUTPUTS -----------------------------------------------------------
   % Uf : velocidad de flameo
15 % wf : frecuencia de flameo
   % Ud : velocidad de divergencia
17
   %% CALCULO VELOCIDADES DE DIVERGENCIA Y FLAMEO
19
   vec_flameo = [];
21 vec_div = [];

23 n_filas = size(w,1);
   n_columnas = size(w,2);
25
   for i = 1:n_filas
27     for j = 1:n_columnas
           if (w(i,j) < 1e-2) && (w(i,j) > -1e-2) &&...
29                 (g(i,j) < 1e-2) && (g(i,j) > -1e-2) && (V(j) ~= 0)
               vec_div = [vec_div; V(j)];
```

```
31          elseif (w(i,j) > 0) && (g(i,j) < 1e-4) &&...
                   (g(i,j) > -1e-4) && (V(j) > 1)
33              vec_flameo = [vec_flameo; V(j) w(i,j)];
            end
35      end
    end
37
    Ud = min(vec_div);
39  [Uf,id] = min(vec_flameo(:,1));
    wf = vec_flameo(id,2);
41
    end
```

Código 26: Cálculo de los puntos de inestabilidad estático y dinámico a partir de los resultados de las curvas de flameo

B.8. Funciones de representación de resultados

B.8.1. Representación de la malla de paneles

```
function DibujoMalla(PANELES)

%% DESCRIPCION DE LA FUNCION
% DibujoMalla Representacion de la geometria de la malla de paneles
    del modelo aerodinamico
%
% INPUTS ------------------------------------------------------------
% PANELES : estructura de datos con la geometria y datos relevantes
    de la malla de paneles que modelizan las superficies de
    sustentacion
%
% OUTPUTS -----------------------------------------------------------
% Dibujo 2D de la malla de paneles

%% DATOS NECESARIOS Y REPRESENTACION
nx = PANELES.nxtotal;
ny = PANELES.nytotal;
b = max(PANELES.CoordNodos(:,8)) - min(PANELES.CoordNodos(:,2));...
    envergadura (diferencia maximas cotas extremos

% Matrices de geometria
CoordNODOS = PANELES.CoordNodos;

nx_text = num2str(nx);
ny_text= num2str(ny);
b_text = num2str(b);
%LF_text = num2str(LF);
%LR_text = num2str(LR);
%xE_text = num2str(xE);

N = size(CoordNODOS,1);

ymax = max(CoordNODOS(:,8));
b = 2*ymax;

xmin = -0.5;
xmax = ceil(max(CoordNODOS(:,5)));

figura_malla = figure;
xlabel('Eje long. $x$ [m]',...
       'Interpreter','LaTex',...
       'FontSize',14)
ylabel('Envergadura, $y$ [m]',...
       'Interpreter','LaTex',...
       'FontSize',14)
```

```
% xlim([-1.1*ymax 1.1*ymax])
% ylim([-xmax 0])
set(gca,'YTick',[ -b/2 , -b/4 , 0 , b/4 , b/2]);
set(gca,'XTick',[-xmax -3*xmax/4 -xmax/2 -xmax/4 0]);
set(gca,'YTickLabel',{'-b/2' , '-b/4' , '0' , 'b/4' , 'b/2'});
set(gca,'XTickLabel',[xmax 3*xmax/4 xmax/2 xmax/4 0]);
set(gca,...
    'YLim',[-1.1*ymax 1.1*ymax],...
    'YTick',[ -b/2 , -b/4 , 0 , b/4 , b/2],...
    'YTickLabel',{'$-b/2$' , '$-b/4$' , '0' , '$b/4$' , '$b/2$'},...
    'XLim',[xmin, xmax],...
    'XTick',[0 xmax/4 xmax/2 3*xmax/4 xmax],...
    'XTickLabel',sprintf('%2.2f\n',[0 xmax/4 xmax/2 3*xmax/4 xmax])
    ,...
    'TickLabelInterpreter','latex',...
    'FontSize',12);
daspect([1 1 1])
ejes = gca;
ejes.View = [90,90];
box on
hold on
grid minor

% PANELES.CoordNodos = [xNW, yNW, xSW, ySW, xSE, ySE, xNE, yNE]
% CROQUIS
%
% ------------------------------------> Y
% |          (NW)-----------------(NE)
% |            |                   |
% |           (A) . . . . . (V) . . . . . (B)
% |            |                   |
% |            |                   |
% |            |    panel j        |
% |            |                   |
% |            |        C          |
% |            |                   |
% |          (SW)-----------------(SE)
% |
% V X
% REPRESENTADOS LA MALLA (COLUMNAS)
for j = 1:N
    % Calculamos las coordenadas
    x = [CoordNODOS(j,7),CoordNODOS(j,1),CoordNODOS(j,3),CoordNODOS(j
    ,5),CoordNODOS(j,7)] ;
    y = [CoordNODOS(j,8),CoordNODOS(j,2),CoordNODOS(j,4),CoordNODOS(j
    ,6),CoordNODOS(j,8)] ;
    relleno = fill(x,y,'red');
    relleno.FaceColor = [0.95,0.95,0.95];
    plot(x,y,'-bs',...
```

```
                    'LineWidth',0.5,...
89                  'MarkerEdgeColor','b',...
                    'MarkerFaceColor','b',...
91                  'MarkerSize',2);
end
93

95 % PUNTOS DE CONTROL

97 CoordCONTROL = PANELES.CoordControl;

99 xcontrol = CoordCONTROL(:,1);
   ycontrol = CoordCONTROL(:,2);
101 plot(xcontrol,ycontrol,'d',...
                          'Marker','x',...
103                       'MarkerEdgeColor','r',...
                          'MarkerSize',2)
105
   % PUNTOS DE FUERZA
107
   CoordCA = PANELES.CoordCA;
109
   xfuerza = CoordCA(:,1);
111 yfuerza = CoordCA(:,2);
   plot(xfuerza,yfuerza,'d',...
113                       'Marker','^',...
                          'MarkerEdgeColor','k',...
115                       'MarkerFaceColor','k',...
                          'MarkerSize',2)
117
   % NUMERACION PANELES
119
   % x_m = (1/2) * (xcontrol + xfuerza);
121 % y_m = (1/2) * (ycontrol + yfuerza);
   % x_m = CoordNODOS(:,3);
123 % y_m = CoordNODOS(:,4);
   %
125 % for j=1:N
   %     texto = text(x_m(j),y_m(j),0,num2str(j));
127 %     texto.Interpreter = 'latex';
   %     texto.FontSize = 12;
129 %     texto.HorizontalAlignment = 'left';
   %     texto.VerticalAlignment = 'bottom';
131 % end


133
   % TITULO GRAFICO ==========================================
135 TextoTitulo = ['Malla de paneles. ', ...
                  '$n_x=$' nx_text ', $n_y=$' ny_text, ...
```

```
                ', $b=$'    b_text  ' m'   ];
title(TextoTitulo,...
'Interpreter','LaTex',...
'FontSize',12);
hold off

end
```

Código 27: Representación de la malla de paneles

B.8.2. Representación del emparrillado estructural

```
function DibujoEstructura(ESTRUCTURA,t)

%% DESCRIPCION DE LA FUNCION
% DibujaEstructura representa la geometria del modelo etsructural
%
% INPUTS ------------------------------------------------------------
% t : vector columna con los puntos intermedios t=-1 (nodo 1), t=+1 (
    nodo 2)
% ESTRUCTURA : estructura de datos con la informacion del
    emparrillado estructural del ala
%
% OUTPUTS -----------------------------------------------------------
% Dibujo 2D de la estructura, nodos y elementos

%% OBTENCION DE DATOS E INICIALIZACION DE LA FIGURA

% Datos de la estructura
% n_largueros = ESTRUCTURA.largueros;
% larg_txt = num2str(n_largueros);
% n_costillas = ESTRUCTURA.costillas;
% cuad_txt = num2str(n_costillas);
N = ESTRUCTURA.NumNodos;
N_txt = num2str(N);
Ne = ESTRUCTURA.NumElem;
Ne_txt = num2str(Ne);
ElemConexion = ESTRUCTURA.Conect;
NodosCoord = ESTRUCTURA.CoordNodos;
NumElem = size(ElemConexion,1);

% Inicializamos la figura
figure

for i = 1:NumElem

    % Identificacion del elemento
    e = i;
```

```
35
    % Numero de puntos a representar a lo largo del elemento
37  p = numel(t);

39  % El vector columna "t" debe tener la siguiente estructura
    % t = [-1 t_2 t_3 .... t_p-2 t_p-1 +1]' : total "p" elementos
41  % con -1 < t_j < 1 , para j=2,...,p-1

43  % Nudos inicial (n1) y final (np) del elemento "e"
    n1 = ElemConexion(e,1);
45  np = ElemConexion(e,2);

47  % Inicializacion de la matriz
    Coord_XY = zeros(p,3);
49

51  %% OBTENCION DE LA GEOMETRiA
    % Asignamos coordenadas nodos inicial y final
53  X1 = NodosCoord(n1,1);
    Y1 = NodosCoord(n1,2);
55  Z1 = NodosCoord(n1,3);
    Xp = NodosCoord(np,1);
57  Yp = NodosCoord(np,2);
    Zp = NodosCoord(n1,3);
59
    Coord_XY(1,1) = X1;
61  Coord_XY(1,2) = Y1;
    Coord_XY(1,3) = Z1;
63  Coord_XY(p,1) = Xp;
    Coord_XY(p,2) = Yp;
65  Coord_XY(p,3) = Zp;

67  for j = 2:(p-1)
        % Funciones de forma lineales
69      N1 = (1/2)*(1 - t(j));
        Np = (1/2)*(1 + t(j));
71      Coord_XY(j,1) = X1 * N1 + Xp * Np;
        Coord_XY(j,2) = Y1 * N1 + Yp * Np;
73      Coord_XY(j,3) = Z1 * N1 + Zp * Np;
    end
75

77  %% REPRESENTACION

79  plot(Coord_XY(:,1),Coord_XY(:,2),... Puntos
                                     '-o',...
81                                   'Color','k',...
                                     'LineWidth',1,...
83                                   'MarkerEdgeColor','k',...
```

```
                                        'MarkerFaceColor','w',...
85                                      'MarkerSize',5);

87      xlabel('$x$ [m]',...
            'Interpreter','LaTex',...
89          'FontSize',14)
        ylabel('$y$ [m]',...
91          'Interpreter','LaTex',...
            'FontSize',14)
93      ax = gca;
        ax.TickLabelInterpreter = 'latex';
95      daspect([1 1 1])
        ax.View = [90,90];
97      hold on
        grid on
99      box on

101     TextoTitulo = ['Emparrillado de la estructura: ' N_txt ' nodos y
        ' Ne_txt ' elementos viga.' ];
        title(TextoTitulo,...
103     'Interpreter','LaTex',...
        'FontSize',12);
105 end
    end
```

Código 28: Representación del emparrillado estructural

B.8.3. Representación de las 2 mallas, aerodinámica y estructural

```
1   function DibujoGlobal(ESTRUCTURA,t,PANELES)

3   %% DESCRIPCION DE LA FUNCION
    % DibujoGlobal Representacion de la geometria de la estructura del
        ala y de la malla de paneles del modelo aerodinámico
5   %
    % INPUTS ------------------------------------------------------------
7   % t : vector columna con los puntos intermedios t=-1 (nodo 1), t=+1 (
        nodo 2)
    % ESTRUCTURA : estructura de datos con la informacion del
        emparrillado estructural del ala
9   % PANELES : estructura de datos con la geometria y datos relevantes
        de la malla de paneles que modelizan las superficies de
        sustentacion
    %
11  % OUTPUTS -----------------------------------------------------------
    % Dibujo 2D de la estructura, nodos y elementos
13
    %% DATOS NECESARIOS
```

```
15
% PANELES
17
% Matrices de geometria
19 CoordNODOS = PANELES.CoordNodos;
Np = size(CoordNODOS,1);
21 Np_txt = num2str(Np);

23 % ESTRUCTURA
Nn = ESTRUCTURA.NumNodos;
25 Nn_txt = num2str(Nn);
ElemConexion = ESTRUCTURA.Conect;
27 NodosCoord = ESTRUCTURA.CoordNodos;
NumElem = size(ElemConexion,1);
29 xmin = min(NodosCoord(:,1))-1;
xmax = max(NodosCoord(:,1))+1;
31
%% REPRESENTACION
33
% MALLA DE PANELES (COLUMNAS)
35 figure
xlabel('Eje long. $x$ [m]',...
37         'Interpreter','LaTex',...
         'FontSize',14)
39 ylabel('Envergadura, $y$ [m]',...
         'Interpreter','LaTex',...
41         'FontSize',14)
daspect([1 1 1])
43 ejes = gca;
ejes.View = [90,90];
45 ejes.TickLabelInterpreter = 'latex';
box on
47 hold on
xlim([xmin xmax])
49 grid minor
title(['Malla de paneles de ala y estabilizador: ' Np_txt ' paneles'
    ],...
51     'Interpreter','LaTex','FontSize',12);

53 for j = 1:Np
    % Calculamos las coordenadas
55     x = [CoordNODOS(j,7),CoordNODOS(j,1),CoordNODOS(j,3),CoordNODOS(j
    ,5),CoordNODOS(j,7)] ;
    y = [CoordNODOS(j,8),CoordNODOS(j,2),CoordNODOS(j,4),CoordNODOS(j
    ,6),CoordNODOS(j,8)] ;
57     relleno = fill(x,y,'red');
    relleno.FaceColor = [0.95,0.95,0.95];
59     plot(x,y,'-bs',...
                'LineWidth',0.5,...
```

```
                    'MarkerEdgeColor','b',...
                    'MarkerFaceColor','b',...
                    'MarkerSize',2);
end


% EMPARRILLADO ESTRUCTURAL
figure
xlabel('Eje long. $x$ [m]',...
       'Interpreter','LaTex',...
       'FontSize',14)
ylabel('Envergadura, $y$ [m]',...
       'Interpreter','LaTex',...
       'FontSize',14)
daspect([1 1 1])
ejes = gca;
ejes.View = [90,90];
ejes.TickLabelInterpreter = 'latex';
box on
hold on
xlim([xmin xmax])
grid minor
title(['Emparrillado estructural: ' Nn_txt ' nodos'],'Interpreter','
   LaTex','FontSize',12);

for i = 1:NumElem

    % Identificacion del elemento
    e = i;

    % Numero de puntos a representar a lo largo del elemento
    p = numel(t);

    % El vector columna "t" debe tener la siguiente estructura
    % t = [-1 t_2 t_3 .... t_p-2 t_p-1 +1]' : total "p" elementos
    % con -1 < t_j < 1 , para j=2,...,p-1

    % Nudos inicial (n1) y final (np) del elemento "e"
    n1 = ElemConexion(e,1);
    np = ElemConexion(e,2);

    % Inicializacion de la matriz
    Coord_XY = zeros(p,3);


    % OBTENCION DE LA GEOMETRiA
    % Asignamos coordenadas nodos inicial y final
    X1 = NodosCoord(n1,1);
    Y1 = NodosCoord(n1,2);
```

```
109     Z1 = NodosCoord(n1,3);
        Xp = NodosCoord(np,1);
111     Yp = NodosCoord(np,2);
        Zp = NodosCoord(n1,3);
113
        Coord_XY(1,1) = X1;
115     Coord_XY(1,2) = Y1;
        Coord_XY(1,3) = Z1;
117     Coord_XY(p,1) = Xp;
        Coord_XY(p,2) = Yp;
119     Coord_XY(p,3) = Zp;

121     for j = 2:(p-1)
            % Funciones de forma lineales
123         N1 = (1/2)*(1 - t(j));
            Np = (1/2)*(1 + t(j));
125         Coord_XY(j,1) = X1 * N1 + Xp * Np;
            Coord_XY(j,2) = Y1 * N1 + Yp * Np;
127         Coord_XY(j,3) = Z1 * N1 + Zp * Np;
        end
129
        plot(Coord_XY(:,1),Coord_XY(:,2),... Puntos
131                                         '-o',...
                                            'Color','k',...
133                                         'LineWidth',1,...
                                            'MarkerEdgeColor','k',...
135                                         'MarkerFaceColor','w',...
                                            'MarkerSize',5);
137

139 end
```

Código 29: Representación global del modelo

B.8.4. Obtención de la deformada de la estructura

```
function DeformadaEstructura(n,da,w,t,h,ElemConexion,NodosCoord,
    NodosGdl,CondContorno)
2
%% DESCRIPCION DE LA FUNCION
4 % DeformadaEstructura representa la deformada global de la estructura
      a partir del calculo hecho con DeformadaElemento
  %
6 % INPUTS ------------------------------------------------------------
  % n : numero de modo a representar
8 % w : frecuencia natural del modo de vibracion a representar
  % ElemConexion : matriz de conectividad de la estructura
10 % NodosCoord : matriz con las coordenadas de cada nodo (Nn x 3)
```

```
% NodosGdl : matriz de grados de libertad de la estructura (Nn x 3)
% CondContorno : vector con las condiciones de contorno (gdl nulos)
% da : vector de gdl activos (no nulos)
% t : discretizacion del elemento
% h : factor de escala
%
% OUTPUTS ---------------------------------------------------------------
% ElementoInicial : coordenadas originales elemento
% ElementoDeformado : coordenadas del elemento deformado

%% SOLUCION TOTAL

% Numero de grados de libertad de la estructura (antes de aplicar las
% condiciones de contorno)
    NumGdl    = numel(NodosGdl);
    NumElem = size(ElemConexion,1);

% d: Vector con grados de libertad activos+nulos

d = zeros(NumGdl,1);

% Nuevo vector con los grados de libertad activos
% Eliminamos los nulos con la funcion de MATLAB setdiff
Gdl_Activos = setdiff(1:NumGdl, CondContorno);


d(Gdl_Activos) = da;

%% CALCULO DEFORMADA

fig = figure;

for e=1:NumElem

    % Configuracion NO deformada ----------------------------------------
    % Calculamos solo los nudos
    t1 = [-1,1]';
    [Coord_XY,~] = DeformadaElemento(e,ElemConexion,NodosCoord,
    NodosGdl,d,t1,h);


    plot3(Coord_XY(:,1),Coord_XY(:,2),Coord_XY(:,3),'--o','Color','
    blue','LineWidth',1,'MarkerEdgeColor','k','MarkerFaceColor','w','
    MarkerSize',5);

    hold on;

    % Configuracion Deformada -------------------------------------------
    % Calculamos puntos intermedios para tener mas resolucion
```

```
56      [~,Coord_xy] = DeformadaElemento(e,ElemConexion,NodosCoord,
        NodosGdl,d,t,h);

58      plot3(Coord_xy(:,1),Coord_xy(:,2),Coord_xy(:,3),'-','Color','k','
        LineWidth',2)

60      xlabel('$x$ [m]', 'Interpreter','LaTex','FontSize',14)
        ylabel('$y$ [m]','Interpreter','LaTex','FontSize',14)
62      zlabel('Def. modal $\omega$ [-]','Interpreter','LaTex','FontSize'
        ,14)
        ax = gca;
64      ax.TickLabelInterpreter = 'latex';
        daspect([1 1 1])
66      ax.View = [-60,15];
        ax.ZLim = [-2,2];
68      hold on
        grid on
70      box on

72      TextoTitulo = ['Modo ' num2str(n-2) ', $\omega_n$ = ' num2str(w,'
        %.2f') ' Hz'];
        title(TextoTitulo,'Interpreter','LaTex','FontSize',12);
74
    end
76


78  end
```

Código 30: Representación de la deformada de la estructura

B.8.5. Obtención de la deformada de cada elemento

```
1   function [ElementoInicial,ElementoDeformado] = DeformadaElemento(e,
        ElemConexion,NodosCoord,NodosGdl,d,t,h)

3   %% DESCRIPCION DE LA FUNCION
    % DeformadaElemento calcula la deformada de cada elemento del
        emparrillado estructural a partir de la solucion de los gdl
        obtenida
5   %
    % El vector d = [U1,V1,g1,....U(nd),V(nd),g(nd)] tiene la solucion de
         movimientos y giros de la estructura.
7   %
    % Las coordenadas iniciales se representan por (X,Y)
9   % Las coordenadas deformadas se representan por (x,y)
    % y se calculan como
11  %                   x(j) = X(j) +  U(j)
    %                   y(j) = Y(j) +  V(j)
```

```
% donde j = 1...nd (nd=Numero de nodos)
% Mediante esta funcion no solo obtendremos la deformada de los nodos
    inicial y final del elemetno sino de una serie de puntos
   intermedios.
%
% INPUTS ----------------------------------------------------------
% e : identificacion de elemento
% ElemConexion : matriz de conectividad de la estructura
% NodosCoord : matriz con las coordenadas de cada nodo (Nn x 3)
% NodosGdl : matriz de grados de libertad de la estructura (Nn x 3)
% d : solucion de los gdl (movimientos y giros)
% t : vector columna con los puntos intermedios t=-1 (nodo 1), t=+1 (
   nodo 2)
% h : factor de escala
%
% OUTPUTS ---------------------------------------------------------
% ElementoInicial : coordenadas originales elemento
% ElementoDeformado : coordenadas del elemento deformado

%% DATOS NECESARIOS
% Numero de puntos a representar a lo largo del elemento
p = numel(t);

% El vector columna "t" debe tener la siguiente estructura
% t = [-1 t_2 t_3 .... t_p-2 t_p-1 +1]' : total "p" elementos
% con -1 < t_j < 1 , para j=2,...,p-1

% Nudos inicial (n1) y final (np) del elemento "e"
n1 = ElemConexion(e,1);
np = ElemConexion(e,2);

% Inicializacion de la matriz "ElementoInicial"
Coord_XY = zeros(p,3);
% Inicializacion de la matriz "ElementoDeformado"
Coord_xy = zeros(p,3);

%% OBTENCION DE LA GEOMETRIA INICIAL
% Asignamos coordenadas nodos inicial y final
X1 = NodosCoord(n1,1);
Y1 = NodosCoord(n1,2);
Z1 = 0;
Xp = NodosCoord(np,1);
Yp = NodosCoord(np,2);
Zp = 0;

Coord_XY(1,1) = X1;
Coord_XY(1,2) = Y1;
Coord_XY(1,3) = Z1;
Coord_XY(p,1) = Xp;
```

```
59 Coord_XY(p,2) = Yp;
   Coord_XY(p,3) = Zp;
61
   for j=2:(p-1)
63     % Funciones de forma lineales
       N1 = (1/2)*(1 - t(j));
65     Np = (1/2)*(1 + t(j));
       Coord_XY(j,1) = X1 * N1 + Xp * Np;
67     Coord_XY(j,2) = Y1 * N1 + Yp * Np;
       Coord_XY(j,3) = Z1 * N1 + Zp * Np;
69 end

71 ElementoInicial = Coord_XY;

73 %% OBTENCION DE LA GEOMETRIA DEFORMADA
   % Obtenemos los indices de los gdl asociados al nudo 1
75 gdl_1 = NodosGdl(n1,:);
   gdl_p = NodosGdl(np,:);
77
   % Obtenemos los resultados de cada nudo (globales)
79 A1 = d(gdl_1);
   Ap = d(gdl_p);
81
   % Longitud del elemento
83 dX = Xp - X1;
   dY = Yp - Y1;
85 L = sqrt( dX^2 + dY^2 );

87 % Angulo del elemento
   a = atan(dY / dX);
89
   for j=1:p
91     % Funciones de forma globales
       [N1g,N2g] = FuncionesFormaGlobales(t(j),L,a);
93     % Desplazamientos (W)=D(3)
       D = N1g * A1 + N2g * Ap; %Globales
95
       % Coordenadas de los puntos deformados
97     Coord_xy(j,1) = Coord_XY(j,1);
       Coord_xy(j,2) = Coord_XY(j,2);
99     Coord_xy(j,3) = Coord_XY(j,3) + h*D(3);

101 end

103 ElementoDeformado = Coord_xy;
```

Código 31: Representación de la deformada de cada elemento

B.8.6. Animación de los modos de vibracion

```
function DibujaModoVibracion(modo,ESTRUCTURA,w_m,vec_m,h)

%% DESCRIPCION DE LA FUNCION
% DibujaModoVibracion representa la animacion del modo de vibracion
    del ala
%
% INPUTS ------------------------------------------------------------
% modo : modo a representar
% ESTRUCTURA : estructura de datos con la informacion del
    emparrillado
% estructural del ala
% w_m : frecuencia natural del modo a representar
% vec_m : autovector del modo a representar
% h : factor de escala
%
% OUTPUTS -----------------------------------------------------------
% Animacion del modo de vibracion

%% DATOS NECESARIOS DE LA ESTRUCTURA

ElemConexion = ESTRUCTURA.Conect;
NodosGdl = ESTRUCTURA.GDL;
NodosCoord = ESTRUCTURA.CoordNodos;
CondContorno = ESTRUCTURA.CondContorno;

%% REPRESENTACION

% Numero de grados de libertad de la estructura (antes de aplicar las
     condiciones de contorno)
     NumGdl   = numel(NodosGdl);
     NumElem = size(ElemConexion,1);

% d: Vector con grados de libertad activos+nulos

d = zeros(NumGdl,1);

% Nuevo vector con los grados de libertad activos
% Eliminamos los nulos con la funcion de MATLAB setdiff
Gdl_Activos = setdiff(1:NumGdl,CondContorno);
d(Gdl_Activos) = vec_m;
vec_m0 = d;

dt = 0.01;
tiempos = 0:dt:1;
w = w_m;

X = [];
```

```
Y = [];
Z = [];

for e=1:NumElem
    % Configuracion Deformada ---------------------------------
    % Calculamos puntos intermedios para tener mas resolucion
    t = [-1:0.1:1]';
    [~,... Coordenadas originales elemento
     Coord_xy] = DeformadaElemento(e,ElemConexion,NodosCoord,NodosGdl
    ,vec_m0,t,h);

    X = [X Coord_xy(:,1)];
    Y = [Y Coord_xy(:,2)];
    Z = [Z Coord_xy(:,3)];
end

figure
graf = plot3(X,Y,Z,'-','Color','k','LineWidth',1.5);
xlabel('$x$ [m]', 'Interpreter','LaTex','FontSize',14)
ylabel('$y$ [m]','Interpreter','LaTex','FontSize',14)
zlabel('Def. modal $\omega$ [-]','Interpreter','LaTex','FontSize',14)
ax = gca;
ax.TickLabelInterpreter = 'latex';
ax.XLim = [-2,7];
ax.YLim = [-6,6];
ax.ZLim = [-1,1];
daspect([1 1 1])
ax.View = [-60,15];
grid on
box on
TextoTitulo = ['Modo ' num2str(modo) ', $\omega_n$ = ' num2str(w_m,'
    %.2f') ' Hz'];
title(TextoTitulo,'Interpreter','LaTex','FontSize',12);
hold on

for j = 1:length(tiempos)
    Zt = Z .* sin(w * tiempos(j));
    for e = 1:numel(graf)
        graf(e).ZData = Zt(:, e);
    end
    pause(0.2);
end

end
```

Código 32: Representación del movimiento oscilatorio de un modo de vibración

B.8.7. Representación conjunta de la deformada de la estructura y de los paneles del modelo aerodinámico

```
function DibujoModosEstPaneles(ESTRUCTURA,PANELES,wm,V,h,tipo)

%% DESCRIPCION DE LA FUNCION
% DibujoModosEstPaneles Representacion de los modos de vibracion,
    modo de divergencia o de flameo de la estructura superpuesta a la
    deformada de los paneles obtenida por interpolacion de la
    deformada de los nodos estructurales mediante splines

% INPUTS ----------------------------------------------------------------
% PANELES : estructura de datos con la geometria y datos relevantes
    de la malla de paneles que modelizan las superficies de
    sustentacion
% ESTRUCTURA : estructura de datos con la informacion del
    emparrillado estructural del ala
% tipo : modo a representar (1 -> vibraciones libres,
%                            2 -> divergencia, 3-> flameo)
% wm : autovalor del modo a representar (1 -> frecuencia natural,
%                                        2 -> vel. de divergencia
%                                        3 -> frecuencia de flameo)
% V : autovector del modo a representar
% h : factor de escala
%
% OUTPUTS ---------------------------------------------------------------
% Dibujo 2D de la malla de paneles

%% DATOS NECESARIOS

% PANELES
N_sup = PANELES.N_sup;... numero de superficies aerodinamicas del
    modelo
CoordEsquinas = PANELES.CoordNodos;... coordenadas de las esquinas de
     cada panel [xNW, yNW, xSW, ySW, xSE, ySE, xNE, yNE]

% ESTRUCTURA
ElemConexion = ESTRUCTURA.Conect;
NodosCoord = ESTRUCTURA.CoordNodos;
NodosGdl = ESTRUCTURA.GDL;
NumElem = size(ElemConexion,1);
NumGdl = ESTRUCTURA.NumGDL;
CondContorno = ESTRUCTURA.CondContorno;
id_nodos = ESTRUCTURA.ID_nodos;... identificador de los nodos
Gdl_Activos = setdiff(1:NumGdl,CondContorno);
d = zeros(NumGdl,1);


%% REPRESENTACION
```

```
39
m = length(wm);
41
for p = 1:m
43      dm = V(:,p);
        d(Gdl_Activos) = dm;
45
        figure
47
        % DEFORMADA PANELES
49      Def_superficies = [];
        for k = 1:N_sup
51          id_sup = k;
            paneles_sup = (PANELES.IdSuperficie == k);
53          CoordEsquinas_sup = CoordEsquinas(paneles_sup,:);
            NumPaneles_sup = size(CoordEsquinas_sup,1);
55          [Rs,Nodos_sup,~] = MatrizRs(id_sup,id_nodos,NodosGdl);
            CoordNodos_sup = NodosCoord(Nodos_sup,1:2);
57          Bs = Spline(CoordNodos_sup);
            Def_paneles = [];
59          for j = 1:NumPaneles_sup
                xNW = CoordEsquinas_sup(j,1);
61              yNW = CoordEsquinas_sup(j,2);
                xSW = CoordEsquinas_sup(j,3);
63              ySW = CoordEsquinas_sup(j,4);
                xSE = CoordEsquinas_sup(j,5);
65              ySE = CoordEsquinas_sup(j,6);
                xNE = CoordEsquinas_sup(j,7);
67              yNE = CoordEsquinas_sup(j,8);
                YsNW = SplineFunction(xNW,yNW,CoordNodos_sup);
69              NzNW = (YsNW' * inv(Bs) * Rs)';
                NzNW = NzNW(Gdl_Activos);
71              wNW = NzNW' * dm;
                YsSW = SplineFunction(xSW,ySW,CoordNodos_sup);
73              NzSW = (YsSW' * inv(Bs) * Rs)';
                NzSW = NzSW(Gdl_Activos);
75              wSW = NzSW' * dm;
                YsSE = SplineFunction(xSE,ySE,CoordNodos_sup);
77              NzSE = (YsSE' * inv(Bs) * Rs)';
                NzSE = NzSE(Gdl_Activos);
79              wSE = NzSE' * dm;
                YsNE = SplineFunction(xNE,yNE,CoordNodos_sup);
81              NzNE = (YsNE' * inv(Bs) * Rs)';
                NzNE = NzNE(Gdl_Activos);
83              wNE = NzNE' * dm;
                Def_paneles = [Def_paneles; xNW yNW wNW; xSW ySW wSW;...
85                  xSE ySE wSE; xNE yNE wNE; xNW yNW wNW];
            end
87
```

```
        Def_superficies = [Def_superficies; Def_paneles];

    end

    X = Def_superficies(:,1);
    Y = Def_superficies(:,2);
    Z = h * Def_superficies(:,3);
    Z_c = zeros(size(Z));

    for l = 1:5:length(X)
        plot3(X(l:l+4),Y(l:l+4),Z_c(l:l+4),'—','Color',"#29B6F6",'LineWidth',1.5)
        hold on
        relleno = fill3(X(l:l+4),Y(l:l+4),Z(l:l+4),'r');
        relleno.FaceColor = [0.61 0.61 0.61];
        plot3(X(l:l+4),Y(l:l+4),Z(l:l+4),'–','Color',"#616161",'LineWidth',1.5)
        hold on
    end

    hold on

    % DEFORMADA ESTRUCTURA
    for e = 1:NumElem
        % Configuracion NO deformada ————————————————————
        % Calculamos solo los nudos
        t1 = [-1,1]';
        [Coord_XY,... Coordenadas originales elemento
         ~] = ... Coordenadas elemento deformado
                  DeformadaElemento(e,ElemConexion,NodosCoord,NodosGdl,d,t1,h);


        plot3(Coord_XY(:,1),Coord_XY(:,2),Coord_XY(:,3),... Puntos
                                        '—',...
                                        'Color','blue',...
                                        'LineWidth',1);

        hold on

        % Configuracion Deformada ————————————————————
        % Calculamos puntos intermedios para tener mas resolucion
        t = [-1:0.1:1]';
        [~,... Coordenadas originales elemento
         Coord_xy] = ... Coordenadas elemento deformado
                  DeformadaElemento(e,ElemConexion,NodosCoord,NodosGdl,d,t,h);

        plot3(Coord_xy(:,1),Coord_xy(:,2),Coord_xy(:,3),... Puntos
```

```
133                                         '_' ,...
                                            'Color','k',...
135                                         'LineWidth',1.5);
        xlabel('$x$ [m]', 'Interpreter','LaTex','FontSize',14)
137     ylabel('$y$ [m]','Interpreter','LaTex','FontSize',14)
        zlabel('Def. modal $\omega$ [-]','Interpreter','LaTex','
    FontSize',14)
139     ax = gca;
        ax.TickLabelInterpreter = 'latex';
141     daspect([1 1 1])
        ax.View = [-60,15];
143     grid on
        box on
145     if (tipo == 1)
            TextoTitulo = ['Modo ' num2str(p) ', $\omega_n$ = '
    num2str(wm(p),'%.2f') ' Hz'];
147     elseif (tipo == 2)
            TextoTitulo = ['Modo ' num2str(p) ', $U_D$ = ' num2str(wm
    (p),'%.2f') ' m/s'];
149         elseif (tipo == 3)
            TextoTitulo = ['Modo ' num2str(p) ', $\omega_f$ = '
    num2str(wm(p),'%.2f') ' m/s'];
151     else
            disp('La variable tipo debe ser 1 (vibraciones libres), 2
     (divergencia) o 3 (flameo)')
153     end
        title(TextoTitulo,'Interpreter','LaTex','FontSize',12);
155

    end
157 end

159 end
```

Código 33: Representación conjunta de la deformada de la estructura y de los paneles del modelo aerodinámico