UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Proyecto fin de grado

DISEÑO Y MODELADO PARAMÉTRICO DE SUPERFICIES SUSTENTADORAS Y DE CONTROL AERODINÁMICO PARA FABRICACIÓN ADITIVA

Autor

Ricardo García Garre

Tutor

Juan Antonio García Manrique

Grado en Ingeniería Aeroespacial

Valencia, junio de 2021

Curso 2020/21

Diseño y modelado paramétrico de superficies sustentadoras y de control aerodinámico para fabricación aditiva

Ricardo García Garre

Universitat Politècnica de València
Escuela Técnica Superior de Ingeniería del Diseño
Grado en Ingeniería Aeroespacial

Junio de 2021

Dedicado a mi abuelo Pablo.

Agradecimientos

Me gustaría comenzar agradeciendo a mis padres, Juan y Pino, el apoyo que me han dado a lo largo de estos años, por ofrecerme una vida sin obstáculos, por llenarme de oportunidades que no dejarían indiferente a nadie y por haber confiado siempre en mí. Por todo vuestro sacrificio y esfuerzo, gracias.

Agradecer también a mi tutor, Juan Antonio, por ofrecerme este proyecto que tanto he disfrutado hacer, por la dedicación e interés mostrados y por haberme acompañado en mis últimos meses de carrera.

Gracias también a todos mis amigos, en especial, a mis dos compañeros de «batalla», Marcos y Noemi, sin los cuales estos cuatro años no habrían sido lo mismo.

A todos los que me habéis acompañado y me habéis visto alcanzar esta importante meta en mi vida, gracias.

Abstract

Since the beginning of aviation during the first decade of the 20th century, the design and manufacturing processes of wings and aerodynamic control surfaces have been a great technological challenge whose complexity has increased over time.

The wide variety of existing wing geometries, together with the recent boom in 3D printing and the increase in computational power of computers, have aroused great interest in the design and optimization of both airfoils and wing surfaces.

However, this design operation results in a complex task that involves the execution of an iterative process in order to guarantee the best compromise between aerodynamic and structural efficiency, trying to reduce manufacturing costs as much as possible.

In order to speed up this process, an aerodynamic surface design program has been developed, capable of reproducing the wing geometries most used by current and past aircrafts. To achieve this task, the main parameters that define a wing have been analyzed and introduced into a computer program in such a way that their manipulation is easy for an inexperienced user in the matter.

Said program, in addition to allowing an intuitive design and real-time visual analysis of wings, enables the possibility of exporting the geometries created to computer-aided design programs in order to be manufactured by additive manufacturing or to be analyzed aerodynamically or structurally through CFD or FEM programs respectively.

To demonstrate the usefulness of the developed project and the synergy of this type of program with additive manufacturing processes, the half wing of a small aircraft has been designed and manufactured.

The results obtained show that the design and modeling process of a complex wing geometry has been reduced by two orders of magnitude through the use of the developed program, reducing the design and modeling time from 180 minutes (3 hours) to as low as 5 minutes, which represents a substantial decrease of 97 %.

Key words

Parametric design program, Python, design and manufacture of wings.

Resumen

Desde el inicio de la aviación a comienzos del siglo XX, el diseño y fabricación de superficies sustentadoras y de control aerodinámico ha supuesto un gran reto tecnológico cuya complejidad ha ido en aumento con el paso de los años.

La extensa variedad de geometrías alares existentes, junto al reciente auge de la impresión 3D y el aumento de potencia computacional de los ordenadores han despertado un gran interés por el diseño y optimización de tanto perfiles como superficies alares.

No obstante, esta operación de diseño, resulta en una tarea compleja que conlleva la ejecución de un proceso iterativo con el fin de garantizar el mejor compromiso entre eficiencia aerodinámica y estructural, tratando de reducir al máximo los costes de fabricación.

Con el objetivo de agilizar este proceso, se ha desarrollado un programa de diseño de superficies aerodinámicas, capaz de generar las geometrías alares más usadas por las aeronaves actuales y pasadas. Para ello, se han analizado los parámetros principales que definen un ala y se han integrado dentro de un programa informático de tal forma que su manipulación resulte sencilla para un usuario inexperto en la materia.

Dicho programa, además de permitir el diseño intuitivo y análisis visual en tiempo real de alas, faculta la posibilidad de exportar las geometrías creadas a programas de diseño asistido por ordenador con el objetivo de ser fabricadas mediante fabricación aditiva o ser analizadas aerodinámica o estructuralmente mediante programas CFD o FEM respectivamente.

Para demostrar la utilidad del proyecto desarrollado y la sinergia de este tipo de programas con los procesos de fabricación aditiva, se ha diseñado y fabricado el semiala de una pequeña aeronave.

Los resultados obtenidos muestran que, el proceso de diseño y modelado de una geometría compleja de ala se ha visto reducido en dos ordenes de magnitud mediante el uso del programa desarrollado, rebajando el tiempo de diseño y modelado desde los 180 minutos (3 horas) a tan solo 5 minutos, lo que supone un descenso sustancial del 97 %.

Palabras clave

Programa de diseño paramétrico, Python, diseño y fabricación de alas.

Resum

Des de l'inici de l'aviació al començament del segle XX, el disseny i fabricació de superfícies sustentadores i de control aerodinàmic ha suposat un gran repte tecnològic la complexitat ha anat en augment amb el pas dels anys.

L'extensa varietat de geometries existents, al costat del recent auge de la impressió 3D i l'augment de potència computacional dels ordinadors han despertat un gran interès pel disseny i optimització de tant perfils com a superfícies alars.

No obstant això, aquesta operació de disseny, resulta en una tasca complexa que comporta l'execució d'un procés iteratiu per tal de garantir el millor compromís entre eficiència aerodinàmica i estructural tractant de reduir al màxim els costos de fabricació.

Amb l'objectiu d'agilitzar aquest procés, s'ha desenvolupat un programa de disseny de superfícies aerodinàmiques, capaç de generar les geometries alars més usades per les aeronaus actuals i passades. Per a això, s'han analitzat els paràmetres principals que defineixen una ala i s'han integrat dins d'un programa informàtic de tal manera que la seva manipulació resulti senzilla per a un usuari inexpert en la matèria.

Aquest programa, a més de permetre el disseny intuïtiu i anàlisi visual en temps real d'ales, faculta la possibilitat d'exportar les geometries creades a programes de disseny assistit per ordinador amb l'objectiu de ser fabricades mitjançant fabricació additiva o ser analitzades aerodinàmica o estructuralment mitjançant programes CFD o FEM respectivament.

Per demostrar la utilitat del projecte desenvolupat i la sinergia d'aquest tipus de programes amb els processos de fabricació additiva, s'ha dissenyat i fabricat el semi ala d'una petita aeronau.

Els resultats obtinguts mostren que el procés de disseny i modelatge d'una geometria complexa d'ala s'ha vist reduït en dos ordres de magnitud mitjançant l'ús del programa desenvolupat, rebaixant el temps de disseny i modelatge des dels 180 minuts (3 hores) a tan només 5 minuts, fet que suposa un descens substancial del 97 %.

Paraules clau

Programa de disseny paramètric, Python, disseny i fabricació d'ales.

Índice general

Índice de figuras

Índice de tablas

Nomenclatura

Acrónimos

AM	Aditive Manufacturing
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CFD	Computational Fluid Dynamics
CNC	Control Numérico por computadora
CSV	Comma Separated Values
DIMM	Departamento de Ingeniería Mecánica y de Materiales
DSN	Dirección de Sustentación Nula
EBM	Electron Beam Melting
FDM	Fused Deposition Modeling
FEM	Finite Element Method
LE	Leading Edge
LENS	Laser Engineered Net Shaping
LOM	Laminated Object Manufacturing
NACA	National Advisory Commitee for Aeronautics
NASA	National Aviation and Space Administration
NLF	Natural-Laminar-Flow
PLA	Ácido Poliláctico
SL	Stereolithography (Técnica de fabricación aditiva)
SLS	Selective Laser Sintering
STL	Stereolithography (Archivo Informático)
TE	Trailing Edge

Letras romanas

a	Sección genérica / Semieje mayor de la elipse
b	Envergadura del ala / Semieje menor de la elipse
b_a	Envergadura de un sector genérico
c	Cuerda del perfil
c_m	Cuerda media aerodinámica
c_r	Cuerda en el encastre del ala
c_t	Cuerda en la punta del ala
m	Curvatura máxima del perfil alar
m_{NACA}	Primer dígito de la nomenclatura NACA 4 dígitos
p	Posición de la curvatura máxima del perfil alar
p_{NACA}	Segundo dígito de la nomenclatura NACA de 4 dígitos
S	Superficie alar
t	Espesor máximo del perfil alar
p_{NACA}	Tercer y cuarto dígitos de la nomenclatura NACA 4 dígitos
y_c	Ecuación de la linea de curvatura media
y'_c	Derivada de la ecuación de la linea de curvatura media
x	Posición de la cuerda
x_l	Coordenada x de un punto de la parte inferior de un perfil
x_u	Coordenada x de un punto de la parte superior de un perfil
y_l	Coordenada y de un punto de la parte inferior de un perfil
y_t	Ordenada de distribución del espesor
y_u	Coordenada y de un punto de la parte superior de un perfil

Letras griegas

α	Ángulo de rotación de la sección
β	Conjunto de valores entre 0 y π
ε_a	Torsión aerodinámica
ε_g	Torsión geométrica
Λ	Alargamiento del ala
λ	Estrechamiento del ala
Γ	Ángulo de diedro del ala
θ	Ángulo entre la línea tangente a la línea de curvatura media y la horizontal
π	Número Pi (3.1415)
φ	Ángulo de flecha del ala
φ_0	Ángulo de flecha al 0 % de cuerda
$\varphi_{0.25}$	Ángulo de flecha al 25 % de cuerda

Símbolos matemáticos

Δ	Incremento
$\in$	Perteneciente a
$\mathbb{R}$	Conjunto de números reales

Otros símbolos

(x, y, z)	Coordenadas de puntos en sistema de referencia absoluto

1

Introducción

1.1. Motivación

Tradicionalmente, el modelado y fabricación de superficies alares y de control aerodinámico ha supuesto un reto tecnológico tanto para empresas del sector como para investigadores e incluso aficionados al aeromodelismo. Dos factores fundamentales son los que dificultan este proceso de diseño y manufactura.

Por un lado, se ha de disponer de cierto conocimiento técnico sobre diseño de alas así como de modelado en entornos de diseño asistido por ordenador. Este proceso de aprendizaje, en especial el del manejo de *software* CAD, puede llegar a ser muy prolongado e ineficiente si el objetivo final del usuario es crear o diseñar geometrías alares.

Por otro lado, se halla la problemática del proceso de fabricación, el cual ha permanecido firme a lo largo del tiempo sin presentar grandes modificaciones. Este proceso ha consistido principalmente en discretizar un ala en lo que se conoce como una estructura semimonocasco, cuyos elementos son fabricados mediante maquinaria costosa al alcance de muy pocos, como cortadoras láser o fresadoras controladas numéricamente por un ordenador (CNC).

El reciente auge de la fabricación aditiva (comúnmente conocida como «impresión 3D») ha conseguido simplificar de forma considerable la complejidad del proceso de fabricación de geometrías complejas, abriendo un nuevo abanico de posibilidades entre las que se encuentra, por supuesto, la fabricación de elementos aeronáuticos pertenecientes al ala de un avión. El gran éxito conseguido por este innovador y tan versátil proceso de fabri-

cación está generando un gran interés y acercamiento de usuarios al mundo aeronáutico. Este hecho, sumado al abaratamiento de los componentes electrónicos, hacen viable la fabricación y prototipado de nuevas aeronaves de pequeño tamaño mediante fabricación aditiva.

No obstante, en cuanto al proceso de diseño y modelado se refiere, aún sigue siendo costoso y poco eficiente. Actualmente, la mayor parte de las operaciones se realizan de forma manual. El diseñador selecciona los perfiles aerodinámicos que desea utilizar y posteriormente los escala, rota y sitúa en la posición correspondiente dentro del programa de diseño para conseguir la geometría deseada. Cuando se modela un ala sencilla, como puede ser un ala recta, esto se realiza de forma más o menos ágil y rápida, sin embargo, con geometrías más complejas en las que es necesario introducir más de un par de perfiles para definir la superficie, el proceso se complica significativamente hasta resultar en proyectos inviables, donde se requiera el diseño de múltiples geometrías para analizar su comportamiento ya sea en túneles de viento o mediante simulaciones CFD.

Con el objetivo de simplificar y automatizar este procedimiento, se ha diseñado un programa de diseño de superficies aerodinámicas capaz de recrear la mayoría de las geometrías alares y superficies de control tipo más empleadas en aeronaves hasta la fecha. A través de este programa, se pueden diseñar de forma sencilla e intuitiva, a partir de una serie de parámetros básicos, tanto perfiles alares como geometrías completas de ala en tan solo unos minutos, con el objetivo final de importarlas a programas de edición y diseño 3D y generar los archivos necesarios para su manufactura mediante técnicas de fabricación aditiva.

Así, se facilita la accesibilidad al diseño y modelado de alas a cualquier usuario con conocimientos básicos sobre diseño asistido por ordenador o superficies alares, pudiendo replicar geometrías complejas de otras aeronaves o diseñar alas propias con fines experimentales o recreativos.

Además de ser una buena alternativa para el rápido diseño de alas con fines recreativos o de investigación científica, el programa también podrá ser empleado con fines académicos, asistiendo a alumnos de ingeniería aeroespacial u otras ramas de la ingeniería dedicadas al estudio o fabricación de estas superficies a analizar e interiorizar de forma directa y visual la influencia de los distintos parámetros que definen estas geometrías, facilitando así la comprensión de conceptos teóricos.

1.2. Objetivos

El objetivo principal de este proyecto es la creación de un programa versátil e intuitivo capaz de modelar de forma paramétrica geometrías alares definidas mediante secciones rectangulares, trapezoidales o elípticas para su posterior análisis o manufactura mediante técnicas de fabricación aditiva.

Esta definición generalista engloba de forma colateral la persecución de los objetivos descritos a continuación.

- Describir las geometrías alares y superficies aerodinámicas de control a partir de una serie de parámetros que faciliten su modelado. Lo que se conoce como diseño paramétrico.
- Automatizar un proceso complejo como es el modelado de alas para que un usuario promedio pueda realizar la tarea en un tiempo inferior a los 10 minutos.
- Demostrar la validez del programa y del potencial de la fabricación aditiva mediante la impresión de un ala.
- Facilitar las fases de diseño de alas, descartando *software* complejo para acceder directamente a la fase de fabricación o simulación.
- Acercar a aficionados del aeromodelismo o usuarios interesados en la materia a experimentar y diseñar sus propias geometrías.
- Facilitar el aprendizaje a nuevas promociones de ingenieros aeroespaciales.

1.3. Organización de la memoria

Este documento se divide en 8 capítulos principales, que describen el proceso completo de creación de un programa enfocado al diseño de alas y su uso en la construcción del semiala de una aeronave de pequeño tamaño fabricada mediante técnicas de fabricación aditiva. El contenido de los capítulos es el siguiente.

Capítulo 1: En este capítulo se realiza una breve introducción al proyecto realizado, donde se muestra la problemática actual en el diseño de alas y la propuesta de mejora desarrollada.

Capítulo 2: En este capítulo se detallan las distintas arquitecturas que pueden ser o han sido empleadas para la fabricación de aeronaves, así como las técnicas de fabricación más empleadas en la manufactura de alas y los programas principales que son empleados en el diseño de estas superficies. Se concluye el capítulo con una descripción detalla de los procesos de fabricación tradicionales y aditiva y las ventajas de la fabricación aditiva frente a la tradicional.

Capítulo 3: En este capítulo se expone una clasificación de las distintas geometrías de ala según su vista en planta, destacando ventajas e inconvenientes tanto estructurales como aerodinámicos de cada una de ellas. Posteriormente se analizan los parámetros que definen las alas y los perfiles aerodinámico.

Capítulo 4: En este capítulo se define la filosofía principal empleada en el desarrollo del programa y como se han implementado, desde una perspectiva matemática, los parámetros de un ala para poder ser modelado.

Capítulo 5: En este capítulo se detalla la forma en la que el programa almacena y gestiona la información generada a nivel interno y como organiza los datos para ser exportados y utilizados por otros programas. Al final del capítulo se ilustra como los archivos generados pueden ser importados a programas CAD de forma rápida y sencilla.

Capítulo 6: En este capítulo se presenta la interfaz gráfica del programa, bautizado como *Parametric Wing Designer* y se ilustran las diferentes pestañas que componen la interfaz, su funcionamiento y proceso a seguir para ser utilizado de forma adecuada.

Capítulo 7: En este capítulo se expone el proceso de diseño y fabricado, con técnicas de fabricación aditiva, un semiala de una pequeña aeronave, mediante el uso del programa desarrollado.

Capítulo 8: En este capítulo se concluye el documento analizando los resultados obtenidos, objetivos alcanzados y los pasos siguiente a realizar para la continuación del proyecto.

2

Estado del arte

2.1. Introducción

Desde el comienzo de la historia de la aviación, la industria aeronáutica ha sufrido innumerables cambios debido a una rápida mejora tecnológica, potenciada principalmente por la irrupción de la Segunda Guerra Mundial a mediados del siglo XX. El desarrollo de nuevos materiales con mejores propiedades mecánicas produjo cambios significativos en la forma en la que la que se construían las aeronaves, afectando directamente a la arquitectura interna de las mismas y a los procesos de fabricación.

Posteriormente, en la década de 1980, los grandes avances en el ámbito electrónico e informático derivaron en un alto grado de automatización de los procesos industriales, dando lugar a la aparición de nuevas técnicas de fabricación, como la fabricación aditiva [1]. Estas nuevas técnicas permiten reproducir piezas de geometría compleja con bajo coste de producción, lo que ha supuesto una revolución total en la industria.

A lo largo de este capítulo se detalla la evolución de la arquitectura interna de las aeronaves a lo largo del tiempo, contrastando las ventajas e inconvenientes de cada tipo.

Posteriormente se tratan los procesos de diseño y modelado empleados para esbozar las geometrías alares, introduciendo los programas de diseño principales que son empleados para realizar la tarea. Se concluye con una comparativa entre los procesos de fabricación tradicionales y los procesos de fabricación aditiva analizando las ventajas e inconvenientes entre ambos procesos.

2.2. Arquitectura general de superficies aerodinámicas

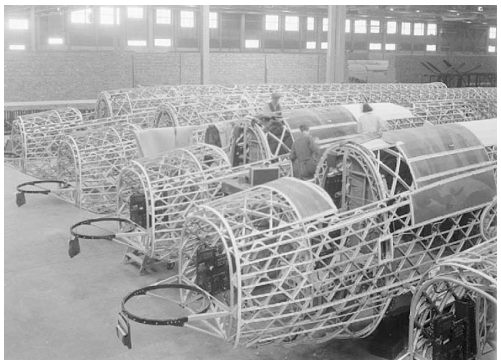
Las estructuras aeronáuticas se caracterizan por ser especialmente ligeras, debido a su gran optimización, enfocada a soportar una gran variedad de esfuerzos empleando el mínimo peso estructural [2]. Estas propiedades se consiguen mediante el uso de vigas o paneles para su construcción.

Las arquitecturas más empleadas en la fabricación de estas estructuras pueden ser divididas en tres tipos: reticuladas, monocasco y semimonocasco.

2.2.1. Estructuras reticuladas

Las estructuras reticuladas están formadas por un conjunto de barras y cables que soportan toda la carga y por un revestimiento exterior encargado de definir la forma. Dentro de las estructuras reticuladas se distinguen tres subtipos, cuya identificación varía según los elementos empleados, su configuración y organización dentro de la estructura. Se encuentra la estructura de *Pratt* (Figura 2.1a), de *Warren* y geodésica (Figura 2.1b).

(a) Aeronave Blériot XI. Primera aeronave en cruzar el Canal de la Mancha [3]



(b) Bombarderos *Vickers Wellington* bajo construcción [4]

Figura 2.1: Ejemplo de aeronaves construidas con estructura de *Pratt* y geodésica

Esta arquitectura fue muy empleada en la fabricación de fuselajes durante la primera mitad del siglo XX, con un uso limitado en la fabricación de alas. En la actualidad, se ha visto sustituida por otras configuraciones más eficientes, tal y como se detallará más adelante.

2.2.2. Estructuras monocasco

Las estructuras monocasco son aquellas formadas únicamente por un revestimiento exterior, encargado de soportar la mayoría de las cargas a las que se somete la estructura.

Este tipo de arquitectura presenta generalmente un mal comportamiento soportando esfuerzos de compresión, siendo susceptible al pandeo. Por ello, las estructuras monocasco se refuerzan mediante una serie de armaduras verticales conocidas como cuadernas o costillas[1] encargadas de aportar la forma y rigidez. Cuando una estructura monocasco ha sido reforzada mediante estos elementos se le denomina «monocasco reforzado» [2].

Las estructuras monocasco no son muy comunes en las construcciones aeronáuticas actuales debido a que su fabricación mediante materiales metálicos resulta en una estructura de gran espesor, muy pesada y poco eficiente frente a otras arquitecturas como la semimonocasco (ver Subsección 2.2.3). No obstante, con el desarrollo de los materiales compuestos y las nuevas técnicas de fabricación aditiva se abre la posibilidad de fabricar componentes monocasco más eficientes y de menor peso que los fabricados actualmente con aleaciones de aluminio [5]. Este es el caso del prototipo *TorresWing* (Figura 2.2) desarrollado por la empresa *MTorres*, consistente en la fabricación de una aeronave monocasco (tanto fuselaje como ala) cuyo peso afirma ser inferior a su equivalente fabricada mediante arquitectura semimonocasco.

Figura 2.2: Interior del fuselaje *TorresWing* donde se aprecia el revestimiento y las cuadernas de fibra de carbono [6]

[1]Si el elemento se encuentra localizado en el fuselaje se denomina cuaderna mientras que si se sitúa en el ala se denomina costilla.

2.2.3. Estructuras semimonocasco

Las estructuras semimonocasco son la evolución de las estructuras monocasco reforzado [2], donde el revestimiento es fabricado con un espesor mucho más fino y está reforzado con múltiples elementos que le dan rigidez y evitan el pandeo[2] de la estructura. En el caso del ala de una aeronave, estos elementos adicionales sirven además para transmitir las enormes cargas y momentos a los que se ve sometido el ala durante el vuelo.

Esta arquitectura domina en las aeronaves actuales, ya que permite reducir el peso manteniendo la utilización de materiales metálicos para su construcción. Además del revestimiento, se distinguen los siguientes elementos dentro de la estructura de un ala (ver Figura 2.3).

- **Largueros**. Recorren el ala longitudinalmente, uniendo el revestimiento superior y el inferior. Existen alas de un larguero, dos largueros o multilarguero, aunque la configuración predominante en las aeronaves comerciales es la de dos largueros [2]. Este elemento soporta los grandes esfuerzos de flexión del ala durante el vuelo, lo que lo en el elemento estructural más importante.
- **Larguerillos**. Son una serie de finas vigas que recorren el revestimiento del ala longitudinalmente por la parte superior e inferior. Previenen el pandeo y soportan cargas de compresión, tracción y flexión.
- **Costillas**. Son los elementos estructurales curvos que dan forma al revestimiento y que sirven de apoyo del mismo y de los larguerillos [2]. Estos elementos se orientan de forma transversal. Cuando forman parte del fuselaje se denominan cuadernas.

Los elementos anteriormente mencionados son los mas relevantes dentro de la estructura semimonocasco ya que soportan los esfuerzos más importantes y definen la forma de la estructura, no obstante, existen otros elementos, como los mamparos de presión empelados en fuselajes, que delimitan la zona presurizada anterior y posterior; y los errajes, que unen el ala al fuselaje y las superficies de control al ala.

[2]El pandeo es un fenómeno de inestabilidad elástica que puede darse en elementos comprimidos esbeltos, y que se manifiesta por la aparición de desplazamientos importantes transversales a la dirección principal de compresión.

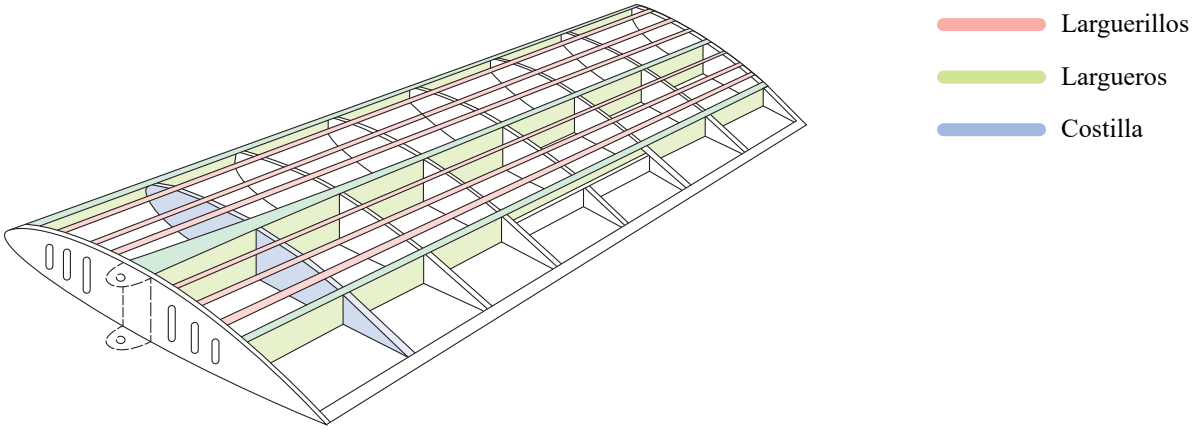


Figura 2.3: Estructura interna de un ala semimonocasco

2.3. Diseño y modelado

El desarrollo de cualquier elemento de una aeronave comienza por una fase de diseño y optimización, la cual puede ser realizada de dos formas distintas.

Por un lado, la forma tradicional de diseñar consiste en la elaboración de una serie de planos que definen la geometría del elemento para posteriormente ser fabricada y analizada para detectar los defectos y las mejoras que pueden ser implementadas. A continuación, se elaboran nuevos planos con el objeto de mejora y se repite el proceso de fabricación y análisis. Como puede parecer evidente, este primer método de diseño es muy primitivo y poco eficiente, ya que requiere de un proceso iterativo en el que se involucra directamente el proceso de fabricación.

En el ámbito aeronáutico resultaría inviable fabricar una aeronave moderna completa mediante este primer método debido a la alta complejidad de los elementos que la componen y el alto coste de fabricación que supondría cada iteración. Es por ello que surge como segundo método de diseño el modelado de superficies y geometrías mediante programas de diseño asistido por ordenador. Este tipo de programas consiguen reducir considerablemente el tiempo de diseño y optimización de los elementos que componen las aeronaves.

El modelado 3D por ordenador permite representar y analizar de forma digital cualquier objeto o superficie tridimensional [7]. Además, mediante el empleo de otros programas especializados en la realización de simulaciones estructurales (FEM) y aerodinámicas (CFD) se consigue descartar completamente el proceso de fabricación hasta las últimas

fases de certificación de un prototipo.

Actualmente existen multitud de programas enfocados al diseño por ordenador, no obstante, los más destacados en el ámbito de la ingeniería son CATIA® y SolidWorks®, desarrollados por la compañía Dassault Systèmes©; NX®, desarrollado por Siemens© o Autodesk Inventor® y Autodesk Fusion 360® desarrollados por Autodesk©. Los programas anteriormente mencionados pertenecen a compañías privadas y por ende son *software* de pago. Existen otras alternativas gratuitas de código abierto como FreeCad®.

La mayoría de estos programas poseen módulos externos que permiten simplificar en cierto modo el modelado de geometrías alares. El funcionamiento explícito y la mejora propuesta a estos módulos se encuentra de forma más detallada en la Sección 5.4. A modo de resumen, lo que hacen estos módulos es unir, dentro de un plano bidimensional, una serie de puntos proporcionados por el usuario a través de un archivo externo. La automatización de esta tarea es especialmente práctica cuando se requiere la unión de multitud de puntos, como ocurre con el modelado de perfiles alares (ver Sección 4.5), sin embargo, no resulta especialmente útil para diseñar superficies completas de alas ya que es solo una herramienta que agiliza un proceso de modelado pero no de diseño. En FreeCad® este módulo o «macro» se puede encontrar con el nombre de *Airfoil Import & Scale* mientras que en Fusion 360® se le denomina *ImportSplineCSV*.

Además de estos programas CAD, existen otros programas específicos aeronáuticos, de diseño y análisis o fabricación de superficies alares. Una gran parte de estos programas están enfocados al análisis del comportamiento aerodinámico de estas superficies sustentadoras, como es el caso de PredimRC©. Por otro lado, existen programas específicos de diseño, siendo devWing© uno de los más destacados.

Los programas de diseño, como ya el mencionado devWing©, pueden resultar un tanto complejos e intimidantes para usuarios inexpertos, al ofrecer multitud de posibilidades de diseño y un gran número de opciones que la mayoría de usuarios no llega a utilizar. Este tipo de programas están optimizados para el diseño y fabricación de alas basadas en una arquitectura semimonocasco, por lo que su objetivo principal es diseñar las costillas que componen la geometría y la locación de los largueros y larguerillos dentro de la misma. Este hecho lo convierte en un programa muy conveniente si el objetivo es fabricar alas semimonocasco, ya que se diseña directamente el elemento más complejo que compone el ala. No obstante, el inconveniente principal de estos programas es la

suscripción de pago y la complejidad en el diseño.

El programa desarrollado en este proyecto está enfocado únicamente al diseño del revestimiento exterior del ala, con el objetivo de que sea fabricado con una arquitectura monocasco o monocasco reforzado mediante fabricación aditiva o sea empleado directamente en el análisis aerodinámico de la superficie. Se caracteriza por ser un programa sencillo, intuitivo, ligero y de suscripción gratuita. En cuanto a sus inconvenientes principales destaca la falta de algunas opciones avanzadas en el diseño de puntas de ala o la integración de espacios para albergar las superficies hipersustentadoras y de control aerodinámico, lo que deriva en la necesidad de utilizar *software* CAD para generar la superficie y realizar un ligero postprocesado. Una discusión más detallada sobre las limitaciones y futuras mejoras del proyecto se encuentra desarrollada en el Capítulo 8.

2.4. Fabricación

El ala de una aeronave, no es considerado necesariamente el componente más difícil de fabricar, sin embargo, su gran tamaño y las altas solicitaciones mecánicas que requieren provocan que para su manufactura actual sean empleadas grandes máquinas de mecanizado y varios días de trabajo [8].

Los procesos de fabricación son un factor importante para la optimización y la construcción de estructuras más eficientes, constituidas por un menor número de elementos y con una menor masa pero manteniendo las mismas propiedades mecánicas y un menor coste [9]. Este hecho justifica la aparición de nuevas técnicas de fabricación a favor de la industria aeroespacial, como es el caso de la fabricación aditiva [10].

En esta sección se tratan los procesos de fabricación tradicionales más relevantes involucrados en la manufactura de las costillas y el revestimiento de un ala. Posteriormente se introduce el concepto de fabricación aditiva y el potencial que tienen estos procesos de sustituir algunos de los métodos de fabricación actuales. Finalmente se realiza una comparativa entre ambos procesos.

2.4.1. Fabricación tradicional

Los materiales y los procesos de fabricación de las alas en sí no ha cambiado mucho a lo largo de los años, a diferencia de los componentes del motor, que han experimentado un gran cambio.

Dependiendo del tipo de aeronave, del tamaño del ala, de las cargas que sea necesario soportar y del presupuesto del proyecto, se ajustarán los procesos de fabricación. En aeronaves de gran tamaño, la mayoría de las costillas son fabricadas mediante un proceso de forjado o mecanizado [11], mientras que en aeronaves más pequeñas se emplean procesos de estampado, ya que resultan más sencillos y con un menor coste [9]. En el caso del revestimiento, en aeronaves de gran tamaño se emplea un proceso de mecanizado de planchas de aluminio, donde la aleación 6061-T6 es la más empleada, mientras que en las aeronaves pequeñas se estampan las láminas para adoptar la forma del perfil [8].

Los procesos de producción anteriormente mencionados se describen a continuación.

Mecanizado

El mecanizado consiste en un proceso para modificar formas, dimensiones y acabado superficial de las piezas, mediante el arrancado de una capa de sobrematerial o creces que es transformada en viruta mediante una herramienta de corte [12]. Este proceso permite la modificación de materiales como madera, plásticos o metales.

Algunos de los procesos incluidos dentro del mecanizado son el taladrado, fresado, rectificado, torneado o mecanizado de agujeros. Estas operaciones requieren de herramientas consumibles como brocas o insertos de mecanizado [9] y son realizadas por lo que se conoce como máquina herramienta, es decir, una máquina no portable que operada por una fuente de energía exterior conforman los materiales por arranque de viruta, abrasión, choque, presión, técnicas eléctricas ó una combinación de ellas [12].

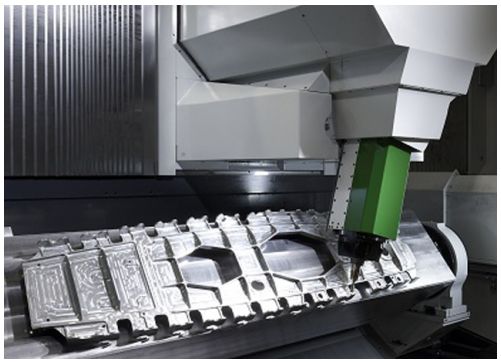
Para la manufactura de elementos de geometría compleja, como el revestimiento del ala o las costillas, son especialmente útiles las máquinas herramienta manejadas mediante control numérico. Este tipo de máquinas, conocidas como CNC, están constituidas por dos elementos principales, un *sofware* cuya misión es leer un código que contiene instrucciones sobre las operaciones que se han de realizar sobre la pieza y una parte de *hardware*,

encargada de realizar y reproducir estos movimientos.

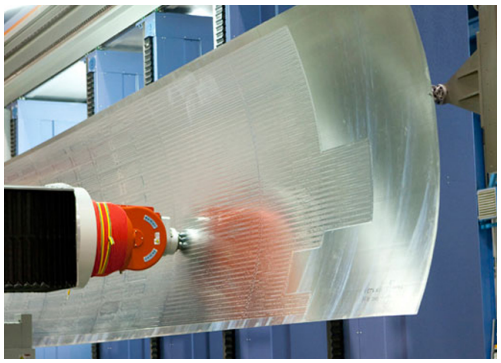
Una de las ventajas principales del mecanizado CNC es que es completamente autónomo, únicamente se requiere de un operario que configure y escriba el programa con las instrucciones para fabricar la pieza. Una vez que la máquina CNC se ha configurado correctamente, necesita un mantenimiento mínimo para su funcionamiento, lo que permite una tasa de producción más rápida, traducida en ciclos de producción muy elevados [13]. Otra ventaja es el bajo coste de fabricación, debido a alta velocidad de producción y los bajos requisitos de personal, no obstante, el mecanizado CNC es solamente rentable para la producción de una tirada de gran volumen [13]. Por último, las máquinas CNC consiguen una producción uniforme, debido a su gran precisión, produciendo un alto nivel de consistencia de diseño entre sus productos [13].

Las principales desventajas de este tipo de operación son la gran cantidad de material desperdiciado y la imposibilidad de fabricar piezas extremadamente complejas con huecos y geometrías que impiden el paso de la herramienta.

En la industria aeroespacial, este proceso de mecanizado CNC se emplea en en la fabricación de las costillas de las alas y en el conformado de las planchas de aluminio que componen el revestimiento, ya que, otra gran ventaja de estas máquinas, es el buen acabado superficial que producen [12], crucial para disminuir la rugosidad de la superficie y disminuir así la resistencia aerodinámica durante el vuelo.



(a) Costilla número 6 del ala de una aeronave A320 siendo mecanizada por una fresadora CNC [14]



(b) Fresadora CNC reduciendo el espesor de una lámina de alumino perteneciente al revestimiento de un ala [15]

Figura 2.4: Operaciones de mecanizado de una costilla y revestimiento de un ala mediante máquinas CNC

Forjado

La forja es un proceso de conformado por deformación plástica mediante compresión en caliente[3], con grandes presiones para realizar piezas acabadas o preformas [16]. Se realiza sobre un material colocado entre matrices, donde éstas, son componentes de prensas de gran tamaño capaces de ejercer una gran presión.

Según si las matrices imprimen una forma determinada sobre el material o no, se puede distinguir entre forja libre y forja con estampa. La forja libre es utilizada para crear piezas únicas, grandes o series pequeñas, mientras que la forja con estampa (Figura 2.5) crea piezas con preforma y cuyas matrices pueden ser empleadas para fabricar un lote entero de mediano o gran volumen [16].

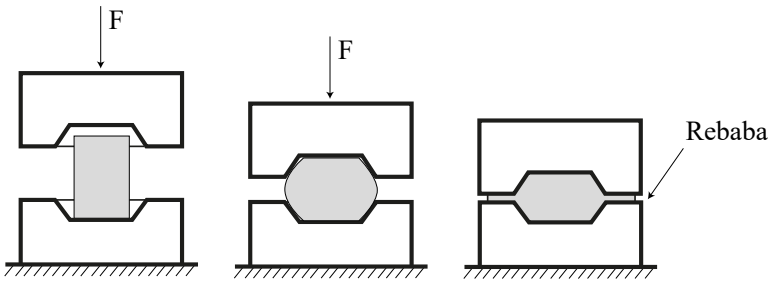


Figura 2.5: Proceso de forjado en matriz cerrada

El forjado puede realizarse por impacto, conocido como «martillete» en uno o varios golpes o de forma gradual y continua, prensado. Un efecto directo de este proceso es la obtención de piezas sin defectos internos, es decir, sin cavidades ni poros. Además, la forja permite obtener piezas un 20 % más resistentes frente a esfuerzos mecánicos y a fatiga que otras piezas equivalentes fabricadas mediante procesos de mecanizado o moldeo [16]. Por este motivo, en un ala, las piezas forjadas son costillas reservadas para zonas sometidas a altas solicitaciones mecánicas [17].

Otra de las principales ventajas del proceso de forjado es que no genera material residual, por lo que el desperdicio del mismo es prácticamente nulo [18].

Sin embargo, la forja presenta inconvenientes como la mala flexibilidad de las matrices. Este hecho deriva en la necesidad de adquirir una serie de herramientas costosas que solo serán amortizadas en caso de fabricar una cantidad de elementos determinados. La

[3]El metal se encuentra en estado sólido pero calentado a una temperatura de trabajo mayor que la temperatura de equicohesión del material.

geometría de las piezas está limitada por la viscosidad del metal, que ha de ser capaz de fluir por todas las cavidades del molde sin que ninguna quede sin rellenar [18]. Otra desventaja es la falta de calidad de acabado, por lo que podría necesitar de más procesos para asegurar dimensiones precisas o una superficie lisa [9].

Conformado de chapa

El conformado de chapa es un proceso de fabricación similar al forjado, consistente en someter a un metal a una carga de compresión entre dos moldes. Este proceso puede realizarse tanto en frío como en caliente y mediante impacto o prensado progresivo.

El conformado de chapa engloba una variedad de procesos que pueden ser empleados en la manipulación de chapas de metal, algunos de estos procesos son el punzonado, estampado, doblado, curvado, rebornado, repujado o embutición [16].

Este proceso se emplea sobre todo en la fabricación de costillas de ala pertenecientes a aeronaves pequeñas o de bajo coste. Principalmente se utilizan las operaciones de punzonado y doblado.

Para piezas que requieren múltiples operaciones de conformado, como la costilla mostrada en la Figura 2.6, se suele emplear lo que se conoce como punzonado con matriz progresiva. Este es un proceso en el cual una banda de chapa entra en la prensa y progresivamente los punzones van punzonando, estampando o doblando la pieza por etapas hasta que en el último paso se separa la pieza del retal [16].

Debido a la posibilidad de automatización que ofrece el conformado de chapa, se considera como uno de los métodos de fabricación mas económicamente rentables, debido a la buena repetibilidad para la fabricación de lotes de gran volumen a gran velocidad con la mínima interacción de operarios [9].

Figura 2.6: Costilla de un ala estampada perteneciente a la avioneta Piper J-3

2.4.2. Fabricación aditiva

La fabricación aditiva o AM (del inglés *Additive Manufacturing*) se ha utilizado en aplicaciones aeroespaciales desde sus inicios a finales de la década de 1980, siendo el fabricante aeroespacial estadounidense *Pratt & Whitney* uno de los primeros en emplearla. Debido a los desarrollos recientes, la fabricación aditiva se ha convertido rápidamente en una tecnología estratégica que generará ingresos en toda la cadena de suministro aeroespacial [19].

La fabricación aditiva hace referencia a un proceso de producción industrial consistente en el depositado gradual y controlado de material capa a capa, de tal forma que cada nueva capa se adhiere a la anterior para formar un cuerpo sólido. Esta técnica es comúnmente conocida como «impresión 3D» y se diferencia de las técnicas de fabricación tradicional ya que está basada en la adición progresiva de material, en contraposición con la eliminación de material expuesto anteriormente en técnicas de fabricación como el mecanizado [1].

Para el fabricado de piezas mediante este sistema de producción es condición fundamental que la pieza haya sido diseñada mediante programas CAD y se encuentre en formato STL, un formato de archivo informático que define geometrías tridimensionales. Este archivo STL se deberá manipular mediante un tipo de programa conocido como «*slicer*» que secciona la pieza en capas y genera el código con las instrucciones que ha de interpretar la máquina encargada de fabricar la pieza.

Actualmente existen multitud de técnicas de fabricación aditiva, cada una adecuada a un tipo de geometría específica y con unos materiales determinados. Según el principio de funcionamiento, los procesos de AM pueden ser clasificados en tres grandes grupos, distinguiéndose así procesos en base líquida, base sólida y base en polvo [20].

A continuación se trata de forma detallada algunos de los procesos más relevantes que son potencialmente útiles en el mundo aeroespacial, ya sea porque son empleados actualmente o serán utilizados en un futuro próximo. Cabe destacar que, los procesos basados en líquidos y polvo poseen mayor potencial que aquellos basados en sólido [20].

Base líquida

Los procesos de fabricación aditiva basados en líquidos emplean como material de fabricación plásticos o resinas en estado líquido o en transición vítrea (en el caso de los plásticos) en el momento del fabricado. Destacan los procesos mostrados a continuación.

- **Estereolitografía** (**SL**). La estereolitografía es la técnica de fabricación aditiva que consiste en el curado o solidificación de un polímero fotosensible, generalmente resina, cuando un láser ultravioleta entra en contacto con el material [20].

  La técnica SL utiliza una plataforma de construcción sumergida en un tanque translúcido lleno de resina fotopolimerizable en estado líquido. Según el tipo de máquina empleada, esta plataforma puede desplazarse hacia dentro del tanque que contiene la resina (Figura 2.7) o hacia fuera.

  El proceso de fabricación consiste en la adición de capas de resina fotopolimerizada cuyo espesor viene definido por la distancia existente entre la superficie del líquido y la plataforma mencionada anteriormente o una capa anterior [1]. El espesor de capa se encuentra al rededor de los 50 μm [9].

  Este proceso se caracteriza por la gran precisión y resolución que puede obtener, de hasta 20 μm, y la fabricación de piezas de gran complejidad. En cuanto a aspectos negativos, la resina necesita un curado con luz ultravioleta para terminar de definir la forma y mejorar las propiedades mecánicas [21].

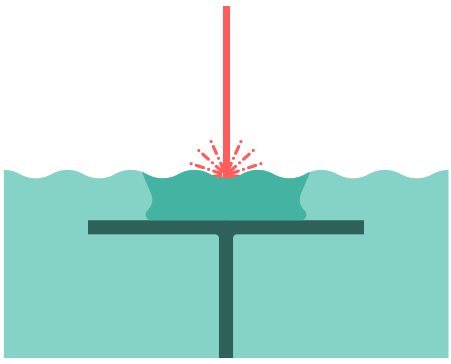


Figura 2.7: Representación esquemática del proceso de estereolitografía con plataforma desplazada hacia el interior del tanque

- **Moldeado por Deposición Fundida** (**FDM**). El moldeado por deposición fundida o *Fused Deposition Modeling* es el método de fabricación aditiva más extendido actualmente. Consiste en la fabricación de piezas mediante hilos de material termoplástico fundido.

  El material se presenta inicialmente en forma de filamento sólido. Para su utilización, el filamento se empuja a través de una boquilla (elemento gris mostrado en la Figura 2.8) calentada a una temperatura ligeramente superior a la temperatura de transición vítrea del material, lo que provoca que abandone la boquilla de la «impresora» con una textura gomosa capaz de fluir. La impresora mueve la boquilla continuamente, depositando el material derretido en las ubicaciones precisas siguiendo una ruta predeterminada [21]. Una vez depositado el material se solidifica de forma prácticamente instantánea a temperatura ambiente. La altura de capa obtenida en este proceso varia entre los 200 μm y 800 μm.

  El moldeado por deposición fundida es una forma rápida y rentable de producir prototipos de plástico. Es una tecnología de bajo coste y con un amplio rango de materiales plásticos disponibles, sin embargo, presenta limitaciones en cuanto a precisión dimensional y las propiedades mecánicas se encuentran muy influenciadas por el grosor y la dirección en la que se aplican o imprimen las capas (propiedades anisotrópicas) [21].

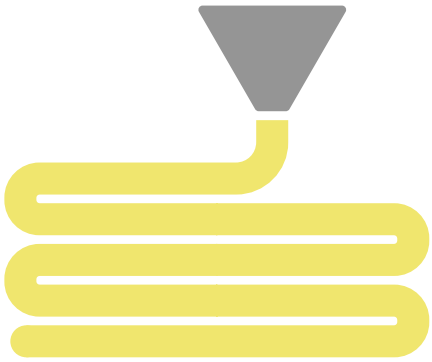


Figura 2.8: Representación esquemática del moldeado por deposición fundida

Base sólida

Entre los procesos de fabricación aditiva basados en sólidos destaca el proceso de Fabricación de Objetos Laminados. Este tipo de fabricación es rara vez empleado en la

manufactura de piezas funcionales, sin embargo, resulta útil en prototipos debido a su bajo coste de fabricación.

- **Fabricación de Objetos Laminados** (**LOM**). La fabricación de objetos laminados o *Laminated Object Manufacturing* es un proceso que combina técnicas aditivas y sustractivas para la fabricación de una pieza capa a capa [20].

  El material se presenta en forma de hoja o láminas. El proceso consiste en cortar estas láminas con la forma de cada capa mediante un láser de dióxido de carbono, para posteriormente ser unidas mediante presión y calor o utilizando un material adhesivo térmico [9]. El hecho de que el material tenga que ser introducido en forma de lámina abre la posibilidad a que este proceso pueda ser realizado con materiales como papel, compuestos, madera o metales.

  Las ventajas principales son su bajo coste, que puede ser aplicado a piezas de gran tamaño y que no se produce un cambio de fase del material. No obstante, presenta las desventajas de los procesos tradicionales de fabricación, desperdicio del material recordado. Además, las piezas poseen baja definición en su superficie y las cavidades internas complejas son difíciles de construir [20].

Base en polvo

La fabricación aditiva de base en polvo permite principalmente la fabricación de piezas metálicas, aunque también se pueden emplear plásticos como el Nylon o TPU [21]. Destacan los siguientes procesos.

- **Sinterización por Láser Selectiva** (**SLS**). La sinterización por láser selectiva (Figura 2.9a) o *Selective Laser Sintering* consiste en un proceso de fabricación en el que un polvo es sintetizado mediante la aplicación de un láser de dióxido de carbono.

  Este proceso de fabricación es muy similar a la estereolitografía con la excepción de que el material ya no es una resina líquida, si no un material pulverizado. El proceso comienza calentando la cámara a una temperatura cercana al punto de fusión del material, posteriormente se esparce una capa inicial de polvo sobre la plataforma de construcción [20]. El láser sintetiza la capa esparcida siguiendo la estructura de

la pieza y una vez se ha solidificado, la plataforma de construcción desciende y se aplica de nuevo otra capa para repetir el proceso [21].

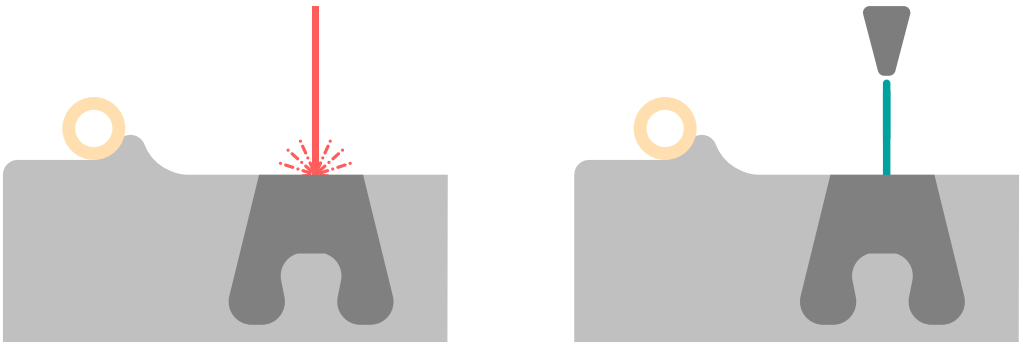
El resultado de la sintetización por láser selectiva es un componente completamente envuelto en polvo sin sinterizar. La pieza se retira de la cama de polvo en la que se encuentra envuelta, se limpia y se deja lista para su uso o postprocesado.

Las principales ventajas de esta tecnología son la amplia variedad de materiales que pueden emplearse, pláticos, metales, composites, cerámicas y combinaciones de estos tipos son algunos ejemplos [20]. Además, el polvo tiene una doble utilidad dentro de este proceso ya que actúa tanto de material de fabricación como de soporte para la estructura. El material no utilizado puede ser reciclado y utilizado en otras ocasiones.

Este proceso se encuentra limitado en cuanto a precisión (la altura de capa es de unos 100 μm) [9]. Además, debe realizarse en una atmósfera de gas inerte para evitar la oxidación en caso de trabajar con materiales metálicos.

- **Fusión por Haz de Electrones** (**EBM**). La fusión por haz de electrones (Figura 2.9b) o *Electron Beam Melting* es un proceso similar al SLS. En este caso, el láser de CO_2 que funde el polvo es sustituido por un haz de electrones.

  El proceso se lleva a cabo en una cámara de vacío para evitar problemas de oxidación ya que está específicamente diseñado para la fabricación de piezas metálicas.



(a) Sinterización por láser selectiva (b) Fusión de haz de electrones

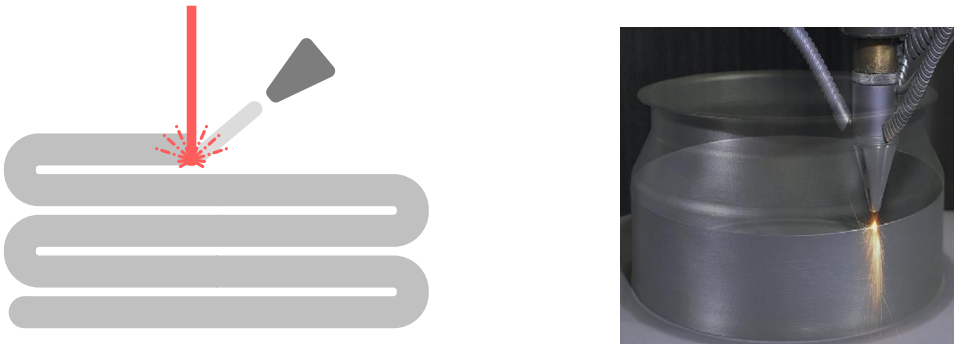
Figura 2.9: Representación esquemática de procesos SLS y EBM

- **Laser Engineered Net Shaping** (**LENS**). En este proceso, representado de forma esquemática en la Figura 2.10a, la pieza se construye fundiendo polvo de metal inyectado por un cabezal de deposición en una ubicación específica [20].

El cabezal de una máquina LENS está constituido por un cabezal láser que calienta la zona sobre la que se ha de depositar el material, creando una región de material fundido y una boquilla inyectora de metal, que pulveriza el material sobre la zona fundida para crear una nueva capa. El proceso ocurre en una cámara cerrada con atmósfera inerte, generalmente de argón [20].

Este proceso permite el uso de una gran variedad de metales y combinaciones de ellos como acero inoxidable, aleaciones de níquel, titanio-6 aluminio-4 vanadio etc. El resultado es una placa plana de metal o una pieza ya existente sobre la que se agrega el material para repararla [21]. Debido a que este proceso genera una placa plana de metal, podría ser empleado para la fabricación del revestimiento de un ala. Un particular uso de LENS en la industria aeroespacial es la fabricación de lanzaderas espaciales (cohetes) [22], toberas de motores (Figura 2.10b) y la reparación de álabes de los rotores de una turbina o compresor.

Uno de los inconvenientes de este proceso son las tensiones residuales generadas por los procesos de calentamiento y enfriado desiguales.



(a) Laser Engineered Net Shaping (b) Aplicación del proceso LENS

Figura 2.10: Representación esquemática y aplicación de la técnica Laser Engineered Net Shaping

2.4.3. Comparación entre fabricación tradicional y aditiva

Son muchas las ventajas que ofrece la fabricación aditiva frente a los métodos de fabricación tradicionales. Algunas de estas ventajas se enumeran a continuación.

- **Ahorro de material**. Aunque en ocasiones algunas piezas requieran de soportes u otros elementos de apoyo para su fabricación, las técnicas de fabricación aditiva

utilizan únicamente el material necesario para formar la geometría. En ocasiones, este material sobrante empleado en los apoyos puede ser reutilizado, tal y como ocurre en SLS o EBM. Además, en elementos que no requieren de altas solicitaciones mecánicas, las piezas se fabrican con interiores huecos o con un bajo relleno que ayuda a reducir la cantidad de material utilizado y el peso.

- **Simplicidad de recursos**. Para fabricar un elemento mediante técnicas de fabricación aditiva únicamente es necesario disponer de un ordenador con el *software* apropiado, el material con el que se desea fabricar la pieza y la máquina o «impresora 3D». Se elimina el uso de máquinas y utillaje de alto coste así como elementos consumibles (herramientas de corte) y lubricantes [9].

- **Flexibilidad de diseño**. Puesto que las máquinas de fabricación aditiva pueden reproducir casi cualquier geometría compleja se elimina la necesidad de fabricar moldes dedicados a la fabricación de una pieza concreta. No obstante, el tamaño de la pieza puede suponer un factor limitante que obligue a descartar un proceso de AM. Además, se pueden fabricar elementos complejos de una sola pieza en los que de haber empleado métodos tradicionales se tendrían que haber fabricado por partes [9]. Al no requerir ningún tipo de molde específico, el diseño puede modificarse en cualquier fase del proyecto sin graves consecuencias económicas.

- **Menor interacción humana**. Las máquinas funcionan de forma autónoma, requiriendo la interacción de un operario únicamente para retirar las piezas o para reponer el material... incluso ambos procesos podrían ser automatizados, descartando por completo la necesidad de un operario. Por este motivo, la calidad del producto o pieza dependerá únicamente de la precisión de la máquina [9] y no de las capacidades de un técnico.

- **Coste del lote**. Las técnicas de fabricación aditiva no requieren de ningún molde, maquinaria o herramienta específica más allá de la impresora 3D, por ello, el coste de fabricación de un número de piezas idénticas será igual al coste de un lote con el mismo numero de piezas customizadas con geometrías distintas. No obstante, en caso de necesitar producir un lote de piezas idénticas de gran volumen, los procesos de fabricación tradicionales son más rápidos y eficientes que los de fabricación aditiva [1].

3

Estudio de geometrías alares

3.1. Introducción

El ala es una estructura fina, plana y hueca que constituye el elemento más crítico de una aeronave [23], su optimización será de vital importancia para maximizar la eficiencia aerodinámica durante el vuelo. La forma en planta de este elemento debe establecerse y dimensionarse para cumplir los requerimientos de vuelo, variando significativamente entre aeronaves.

Este capítulo detalla en primer lugar la evolución de estas geometrías, destacando las tipologías más relevantes con el objetivo de analizar posteriormente los parámetros principales que las definen. A continuación, se hace una breve referencia a las superficies hipersustentadoras y de control aerodinámico, cuyas características paramétricas y principios de funcionamiento vienen heredados de las superficies alares.

Una vez expuestas las geometrías de ala más relevantes, se analizan los parámetros principales que las definen, tanto en su vista en planta, como alzado y perfil.

3.2. Geometrías alares

Una geometría alar o superficie sustentadora es una superficie que, sometida a una corriente de aire con un ángulo de ataque apropiado, produce más sustentación que resistencia [24], son comúnmente conocidas con el nombre de alas. En el análisis de la evolución y tipos de geometrías alares se ha tomado la vista en planta de las topologías

más representativas con el objetivo de facilitar la distinción entre los diferentes tipos.

Previo a este análisis se introduce a continuación la terminología empleada a lo largo de este documento. Los términos aparecen representados en la Figura 3.1.

Borde de ataque, también conocido como *Leading edge* (LE) en inglés. Es el borde delantero de un perfil aerodinámico o pala. Es el borde que primeramente se encuentra con el flujo de aire [25].

Borde de fuga o *Trailing edge* (TE) es el borde posterior de un perfil aerodinámico o pala. Es el borde por el que normalmente pasa el flujo en último lugar [25].

Cuerda, es la linea imaginaria que une el borde de ataque con el borde de fuga.

Encastre alar, es la parte del ala de una aeronave de ala fija más cercana al fuselaje [26]. Es la zona donde ala y fuselaje entran en contacto.

Punta de ala, es la parte del ala más alejada del fuselaje.

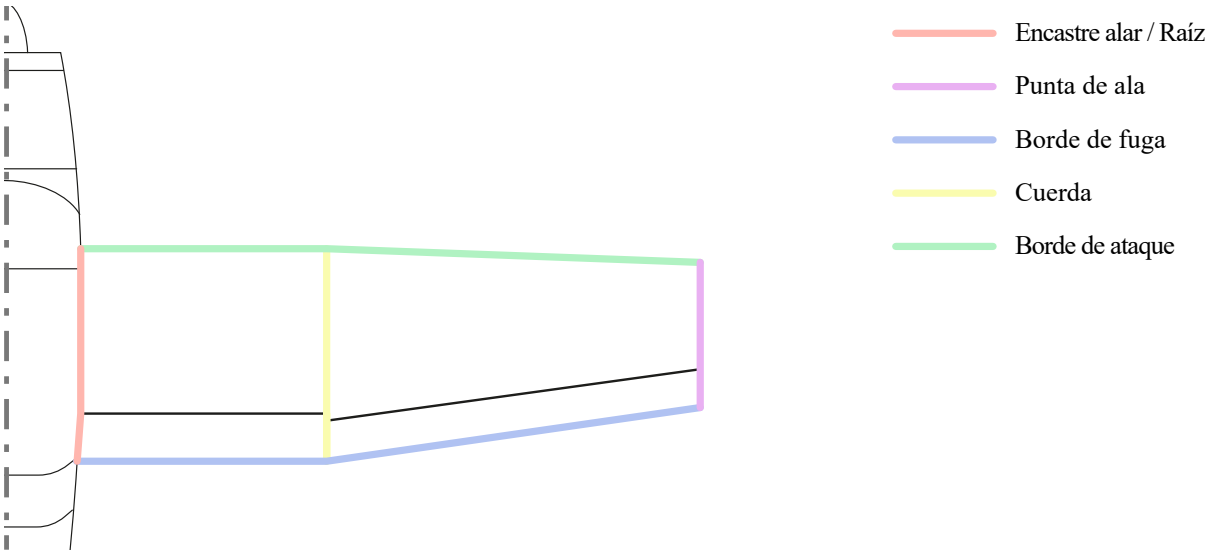


Figura 3.1: Representación visual de la terminología básica común a geometrías alares. Vista en planta del ala perteneciente a la aeronave Cessna 172

3.2.1. Ala recta

Es la forma más primitiva de las aeronaves que se conocen hoy en día. Utilizada por los hermanos *Wright* en 1903 en el desarrollo del *Flyer 1*[1] fue el tipo de geometría dominante durante inicios del siglo XX. El ala recta, también conocida como ala de cuerda constante, destaca por su mínimo coste de fabricación [27] , derivado de su sencillez geométrica. Este hecho se debe a que todas las costillas que dan forma al ala poseen la misma cuerda, siendo únicamente necesario el diseño de una sección para definir la totalidad del ala, además, el larguero que une las costillas es una simple viga de espesor constante. Esta simplicidad afecta no solo a las costillas del ala y al larguero, sino también a las superficies de control [28].

A pesar de esta ventajosa característica, su uso actual se limita a un pequeño número de aeronaves debido al abaratamiento en los costes de fabricación que permiten la fabricación de alas más aerodinámicamente eficientes[2] con el mismo coste. Aún asi, esta configuración es ideal para aviones de entrenamiento o aviones para los que el costo de fabricación es imperativo [28].

Esta forma en planta rectangular se emplea en aeronaves de baja velocidad (es decir, en condiciones de flujo incompresible) donde el Mach es inferior a 0.4 [23]. Dicha geometría aparece reflejada en la Figura 3.2, donde se observa como el borde de ataque y borde de fuga en este tipo de ala son paralelos y por lo tanto la cuerda constante.

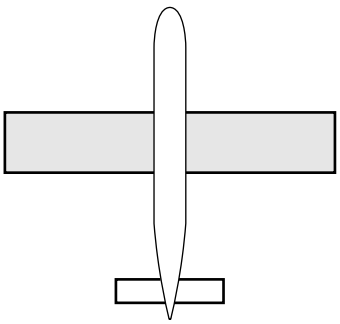


Figura 3.2: Geometría básica de la forma en planta de una aeronave de ala recta

[1]El *Flyer 1* fue la primera aeronave en la historia capaz de realizar un vuelo controlado. Este acontecimiento tuvo lugar el 17 de diciembre de 1903. La aeronave realizó 4 vuelos de forma satisfactoria.

[2]La Real Academia de Ingeniería define la eficiencia aerodinámica como el cociente entre la sustentación y la resistencia aerodinámica.

3.2.2. Ala trapezoidal

Se denomina ala trapezoidal a aquella cuya cuerda en la raíz o encastre es distinta de la cuerda en la punta, de modo que, la cuerda más pequeña reside dentro de las lineas perpendiculares trazadas entre el eje de simetría de la aeronave y los bordes de ataque y de fuga de la sección más grande (ver Figura 3.3). Este tipo de geometrías, a pesar de conllevar un proceso de fabricación más complejo que en el caso anterior, genera una menor resistencia aerodinámica al mismo tiempo que mejora la eficiencia estructural frente al ala recta [29]. Es una de las configuraciones más extendidas hoy en día en aeronaves ligeras [27].

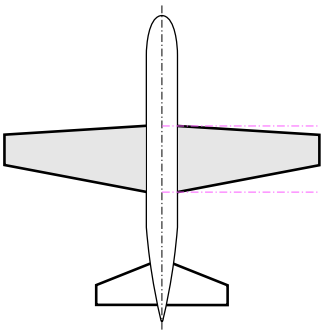
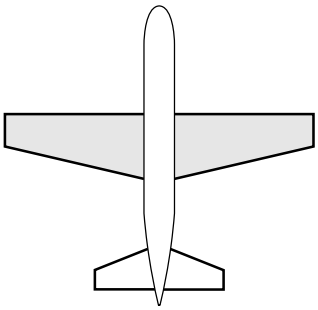
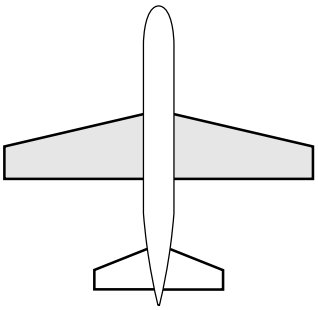


Figura 3.3: Geometría básica de la forma en planta de una aeronave de ala trapezoidal

La posición de la sección de punta de ala relativa a la del encastre alar es un parámetro de diseño que influye directamente sobre la posición del centro de gravedad de la aeronave, sin tener una gran influencia sobre la aerodinámica [28]. De esta forma, un ala trapezoidal con borde de ataque perpendicular al eje del avión (Figura 3.4a) tendrá el centro de gravedad más desplazado hacia el morro que una aeronave de ala trapezoidal con borde de fuga perpendicular al eje de la aeronave (Figura 3.4b).



(a) Ala trapezoidal con borde de ataque perpendicular al eje de simetría de la aeronave

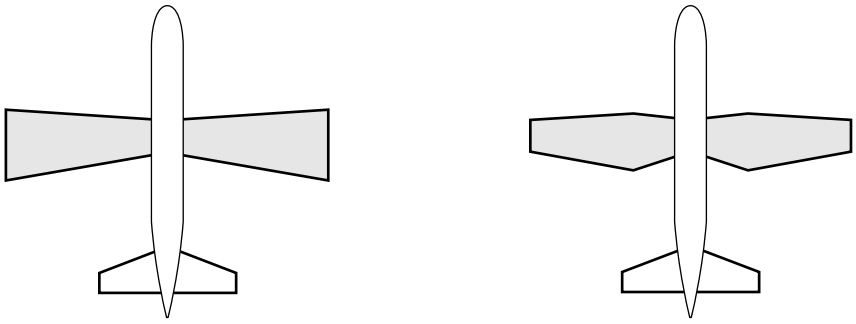


(b) Ala trapezoidal con borde de fuga perpendicular al eje de simetría de la aeronave

Figura 3.4: Variaciones del ala trapezoidal según la posición de la sección de punta

Los costes de producción de un ala trapezoidal son más elevados que los de un ala rectangular, debido a que la cuerda de las diferentes costillas que componen la geometría varía a lo largo de la envergadura. Sin embargo, el ala trapezoidal mantienen unos bordes de ataque y de fuga que son modelados mediante una linea recta, lo que simplifica su proceso de diseño [23].

A partir de este concepto de ala trapezoidal surgen otras configuraciones como el ala trapezoidal inversa (Figura 3.5a) o trapezoidal compuesta (Figura 3.5b) caracterizada por la combinación entre ambos tipos. Cabe destacar que, el ala trapezoidal inversa se caracteriza por una mala eficiencia aerodinámica debido a la elevada resistencia inducida que genera, únicamente ha sido empleada en aeronaves experimentales como el *Republic XF-91 Thunderceptor*. Por otro lado, el ala trapezoidal compuesta tampoco presenta ninguna mejora significativa en términos aerodinámicos y su uso se restringe a la aeronave *Westland Lysander*, desarrollada en 1930 y cuyo único objetivo del ala es mejorar la visibilidad del piloto [28].



(a) Vista en planta de una aeronave con ala trapezoidal inversa

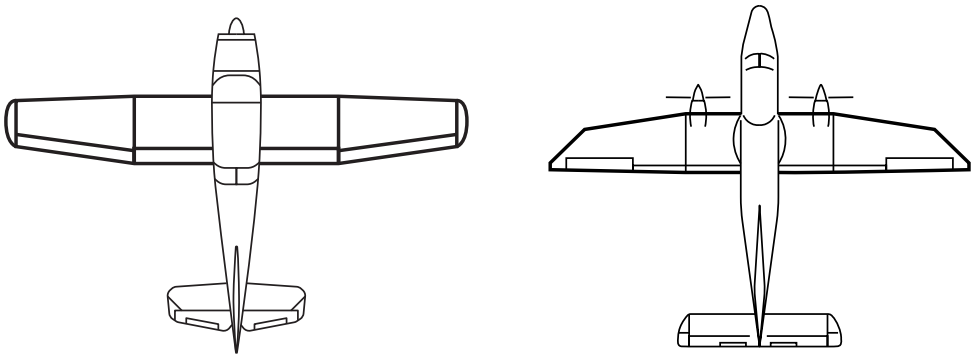
(b) Vista en planta de una aeronave con ala trapezoidal compuesta

Figura 3.5: Variaciones de la geometría de ala trapezoidal

Tanto el ala recta, analizada en la Subsección 3.2.1, como el ala trapezoidal, pueden ser combinadas de manera conjunta para formar nuevas geometrías. Un ejemplo de esto es el ala utilizada en el Cessna 172 (Figura 3.6a). En esta aeronave se aprecia como la sección interior (correspondiente al encastre alar) es recta, mientras que la sección mas alejada del fuselaje es una sección trapezoidal.

De igual forma, varias secciones trapezoidales pueden ser combinadas para formar geometrías más complejas. Esto se reproduce sobre todo en planeadores como el Stemme S-10 y aviones regionales como el Dornier Do-228 (Figura 3.6b). Como se observa en la Figura 3.6b, la combinación de secciones trapezoidales generan una buena aproximación

de la geometría a un ala elíptica, con los beneficios que conlleva de un bajo valor de la resistencia inducida. Por este motivo esta aproximación es muy empleada en planeadores y otras aeronaves en las que prima la eficiencia aerodinámica. No obstante, al introducir varias secciones trapezoidales se vuelve a generar el problema de la complejidad de fabricación.



(a) Vista en planta de la aeronave Cessna 172

(b) Vista en planta de la aeronave Dornier Do-228

Figura 3.6: Aeronaves de ala trapezoidal de distintas secciones

3.2.3. Ala elíptica

Surgida durante los años 1939 y 1945, correspondientes a la Segunda Guerra Mundial, el ala elíptica (Figura 3.7) se presentó como la geometría triunfadora de la época, utilizada en aeronaves tan emblemáticas como el *Supermarine Spitfire*.

Se denomina ala elíptica a aquella que genera una distribución de la sustentación elíptica. Por lo general, su forma en planta se asemejará a la de una elipse [30] (ver Sección 4.7).

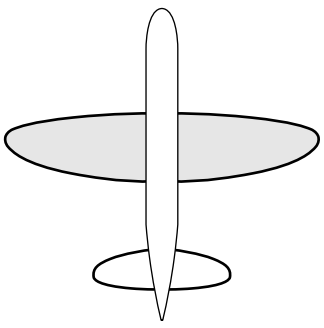
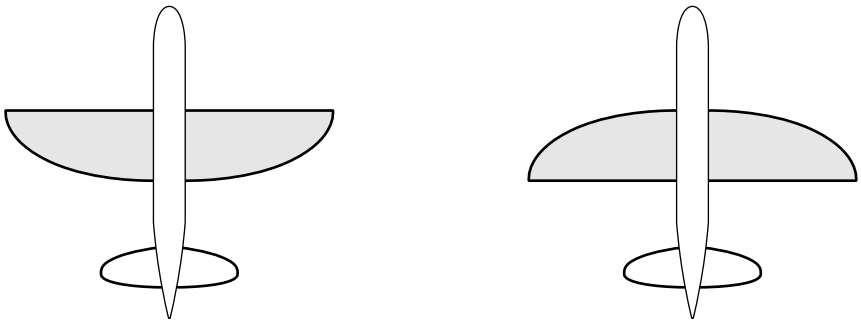


Figura 3.7: Geometría básica de la forma en planta de una aeronave de ala elíptica

Esta geometría fue diseñada en un primer momento para poder almacenar el máximo de munición y combustible dentro del ala de la aeronave. Pronto se evidenció que aquellas aeronaves que volaban con una geometría elíptica alcanzaban mayores velocidades. Estudios posteriores demostraron que el ala elíptica es la forma más aerodinámicamente eficiente para las velocidades de vuelo de una aeronave turbohélice, además es la que consigue un mejor reparto de los esfuerzos mecánicos a lo largo de la estructura [30].

A pesar de estas ventajosas características, las alas elípticas presentan dos inconvenientes que han provocado su práctica desaparición hasta restringir su uso en réplicas a escala de aeronaves de la Segunda Guerra Mundial. Por un lado presentan un inconveniente aerodinámico. Esta geometría entra súbitamente en pérdida, provocando la pérdida de control sobre la aeronave cuando se opera a bajas velocidades y a un elevado ángulo de ataque [30]. Por otro lado, el mayor inconveniente de esta forma en planta es su capacidad de producción, gravemente afectada por la complejidad de la superficie. Es muy difícil de fabricar con aluminio ya que requiere que las láminas sean moldeadas mediante técnicas de hidroconformado[3] o similares [28]. Sin embargo, es mucho más fácil de producir utilizando compuestos modernos como láminas de fibra de vidrio o carbono.

Las formas en planta elípticas con un borde de ataque recto (Figura 3.8a) o un borde de fuga (Figura 3.8b), también conocidos como ala de media luna, son comunes en aviones y planeadores pequeños de radiocontrol [28].



(a) Vista en planta de una aeronave con ala semielíptica con borde de ataque recto

(b) Vista en planta de una aeronave con ala semielíptica con borde de fuga recto

Figura 3.8: Variaciones de la geometría de ala semielíptica

[3]El hidroconformado consiste en moldear un material mediante la acción de un líquido sometido a presión.

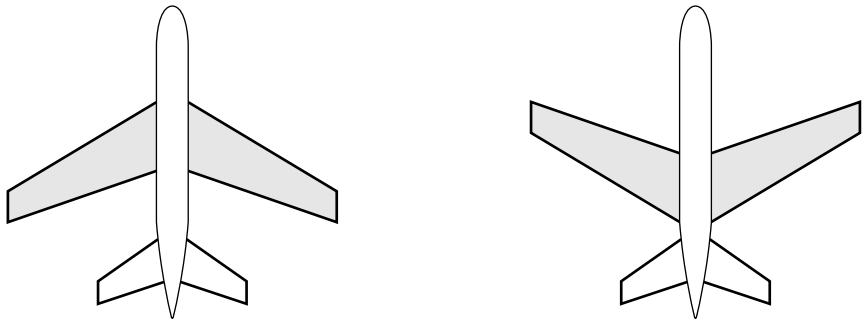
3.2.4. Ala en flecha

El ala en flecha es la geometría más empleada, al igual que el ala trapezoidal, en la aviación actual. Esta geometría domina entre las aeronaves comerciales de régimen transónico.

La principal ventaja de la utilización de esta geometría sobre aeronaves de alta velocidad es el aumento del Mach crítico[4] y la reducción del valor de la onda de choque.

La flecha es el ángulo formado entre el borde de ataque y la linea perpendicular al eje de la aeronave (ver Sección 3.4) y puede ser positiva (Figura 3.9a) o negativa (Figura 3.9b). La flecha positiva es mucho más común, principalmente por razones de diseño y estabilidad. Las alas con flecha negativa son propensas a la divergencia aeroelástica [30].

Como regla general, un ala ha de ser diseñada con el mínimo valor de flecha posible para unas condiciones de vuelo dadas, para evitar problemas estructurales y simplificar su fabricación. Además este tipo de geometrías generan una menor sustentación a bajas velocidades (están optimizadas para velocidades de vuelo cercanas a la velocidad del sonido) por lo que es necesario incorporarles elementos hipersustentadores (ver Sección 3.3).



(a) Vista en planta de una aeronave con ala en flecha positiva

(b) Vista en planta de una aeronave con ala en flecha negativa

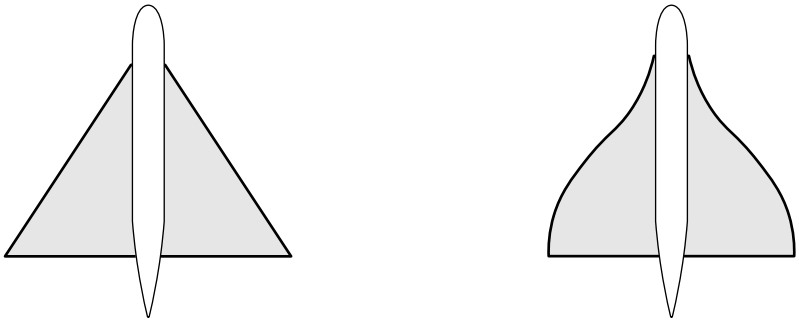
Figura 3.9: Variaciones de la geometría de ala en flecha

[4]El Mach se define como la relación entre la velocidad de un cuerpo que se desplaza a través de un fluido y la velocidad de propagación del sonido en dicho fluido. El Mach crítico ocurre cuando esta relación entre velocidad del cuerpo y del sonido alcanza el valor de la unidad.

3.2.5. Ala delta y ojival

El ala delta (Figura 3.10a) es empleada principalmente en aeronaves de alta velocidad (cazas de combate supersónicos aunque también pueden ser usadas en régimen transónico) y su uso está muy restringido en vuelo subsónico, limitándose únicamente a aeronaves recreativas de aeromodelismo.

Por otro lado, el ala ojival presenta un uso similar al ala delta, con el inconveniente añadido de una mayor complejidad y costes en su manufactura. Este tipo de geometría está especialmente diseñada para un tipo de aeronave concreto o son empleadas con fines experimentales.



(a) Geometría básica de la forma en planta de una aeronave de ala delta

(b) Geometría básica de la forma en planta de una aeronave de ala ojival

Figura 3.10: Vista en planta de las geometrías de la delta y ojival

Como se ha podido ver, son muchas y muy diversas las geometrías alares existentes. Cabe mencionar también, dentro de esta subsección de alas más contemporáneas, las ala volantes, una geometría que cobrará vital importancia en los años venideros.

3.3. Superficies hipersustentadoras y de control

Las superficies hipersustentadoras, conocidas con el nombre de *flaps*, son dispositivos móvibles colocados en el borde de ataque y de fuga del ala para favorecer el aumento de la sustentación o retrasar la entrada en pérdida. Este tipo de dispositivos es especialmente útil en aeronaves con ala de flecha u otras geometrías diseñadas para realizar vuelos transónicos.

Existen varios tipos de *flaps*. En el borde de fuga se distingue el *flap* simple, de

intradós, ranurado, *fowler* etc. Mientras que en el borde de ataque se distinguen el *flap Kruger* y *slat*.

En cuanto a las superficies de control aerodinámico, son empleadas para variar la atitud de la aeronave durante el vuelo. Alerones, timón de cola vertical o *rudder* y timón de profundidad o *elevator* son las superficies de control más empleadas en la aviación actual ya que permiten variar los ángulos de alabeo, cabeceo y guiñada de forma controlada.

Todas estas superficies mencionadas aparecen reflejadas en la Figura 3.11, correspondiente a una aeronave B737.

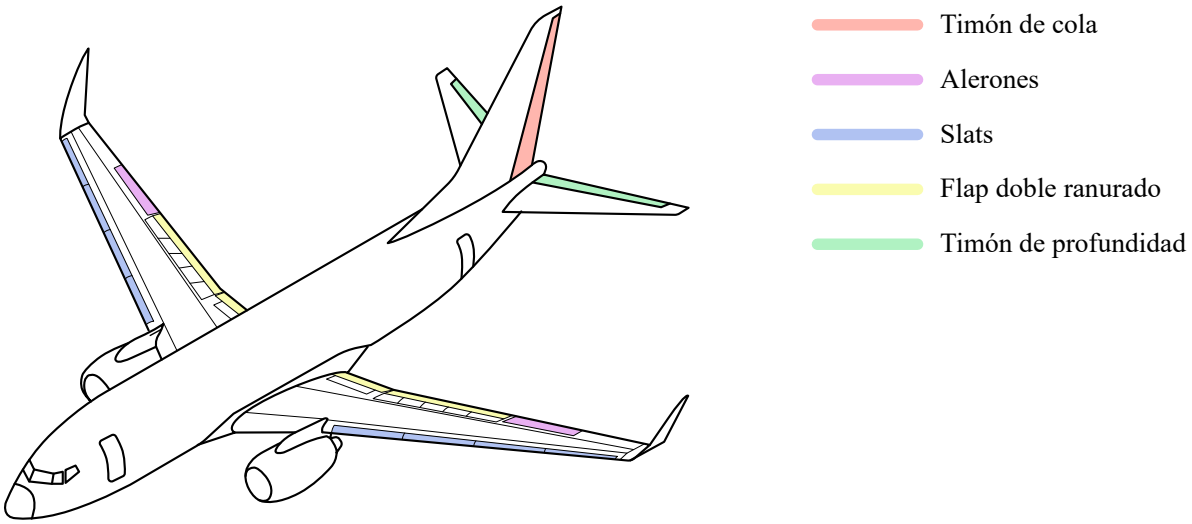


Figura 3.11: Superficies hipersustentadoras y de control aerodinámico de una aeronave B737

La mención a este tipo de superficies viene derivada de que su geometría y procesos de fabricación son muy similares a los de las alas de una aeronave tradicional ya que su forma en planta por lo general presenta una forma similar a la de un ala recta o trapezoidal.

3.4. Parámetros alares

Los parámetros principales de un ala se dividen en tres categorías según la vista sobre la que tienen más influencia (planta, alzado o frente y perfil). Estos parámetros son definidos a continuación y son representados en la Figura 3.12 y Figura 3.13.

Envergadura

Este parámetro es común tanto a la vista en planta como a la frontal. La envergadura o *span* (b) es la distancia comprendida entre las puntas de un ala. Cuando se hace referencia al semiala, es decir, a la mitad izquierda o derecha de un ala este parámetro se conoce como semienvergadura ($b/2$).

Cuerda

Este parámetro es común tanto a la vista en planta como al perfil. El ala tiene dos cuerdas características que son utilizadas para definir otros parámetros como el estrechamiento, desarrollado más adelante. Estas cuerdas son, la cuerda en la sección del encastre alar (c_r) y la cuerda en la punta de ala (c_t). Cuando $c_t \neq c_r$ aparece un nuevo término conocido como Cuerda Media Aerodinámica (c_m), que es la cuerda de un ala rectangular equivalente con el mismo área y mismas propiedades aerodinámicas.

Alargamiento

El alargamiento (Λ), también conocido como relación de aspecto, es la relación entre el cuadrado de la envergadura del ala y el área (S) de la misma (Ecuación 3.1).

$$\Lambda = \frac{b^2}{S} = \frac{b}{c_m} \tag{3.1}$$

El alargamiento tiene una influencia directa sobre la resistencia inducida, el peso y la envergadura del ala [31]. La principal tendencia es hacia una relación de aspecto alta, que reduce la resistencia aerodinámica. No obstante, un gran alargamiento implica una masa estructural alta [32].

Estrechamiento

El estrechamiento o *tapper ratio* (λ) es la relación entre la cuerda en el encastre y la cuerda en la raíz, ambas medidas en la dirección del flujo del aire. Este parámetro pertenece a la vista en planta.

El estrechamiento dota al ala de una distribución de la sustentación semielíptica, generando una menor resistencia inducida. Un bajo valor del estrechamiento también resultará beneficioso a nivel estructural pero puede suponer un problema de control aerodinámico si no se deja la cuerda necesaria para que las superficies de control de alabeo (alerones) cumplan adecuadamente su función [32]. Por ello, este parámetro influye directamente sobre peso de la aeronave, la entrada en pérdida de la punta de ala, el volumen de combustible máximo que puede ser alojado y el coste de producción [31].

Flecha

La flecha (φ) es el ángulo que forma el ala, generalmente definido por la linea del 25 % de cuerda o el borde de ataque, con el eje transversal del fuselaje.

La flecha puede ser negativa (hacia adelante) o positiva (hacia atrás). Este parámetro se implementa en alas diseñadas para volar en regímenes transónicos, ya que aumenta el número de Mach crítico y reduce el valor de la onda de choque y la resistencia por fenómenos de complesibilidad [32].

La flecha positiva es la más común en aeronaves comerciales por razones de diseño y estabilidad. Como regla general, se debe dotar a un ala con el mínimo ángulo de flecha como sea posible para una determinada condición de vuelo ya que la flecha implica penalizaciones estructurales, de control y económicas [32].

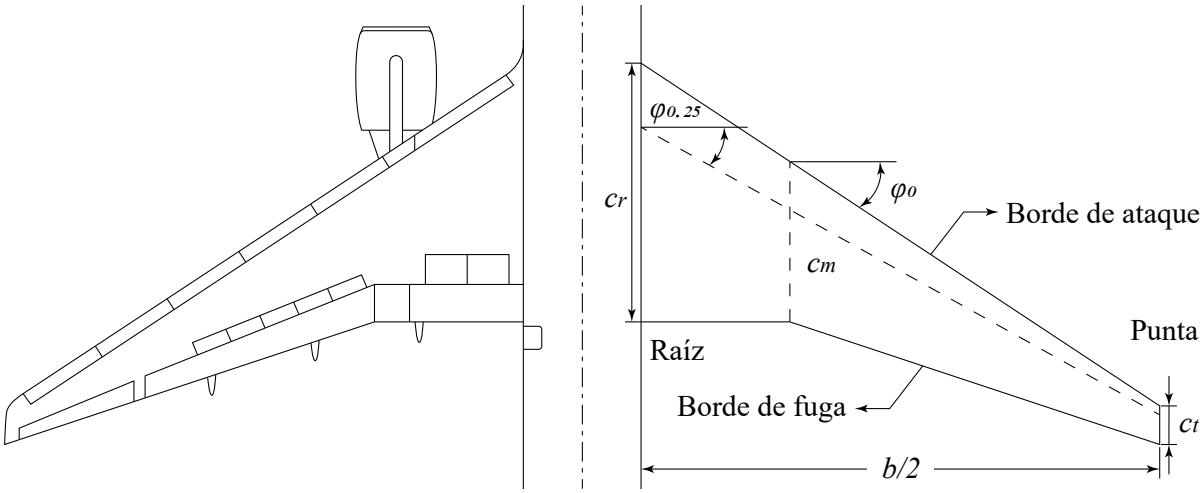


Figura 3.12: Representación esquemática de una aeronave B777-200ER con ángulo de flecha positivo donde se definen los parámetros principales de la vista en planta

Diedro

El diedro (Γ) es el ángulo que forma la línea que une el borde de ataque de la sección en el encastre y la punta con el plano horizontal.

Al igual que la flecha, el diedro puede ser negativo si el ala se inclina hacia abajo o positivo si se inclina hacia arriba. Este parámetro se implementa por dos motivos principales. El primero es para mejorar la estabilidad lateral del avión o aumentar la maniobrabilidad. El segundo es para ajustar la distancia entre el ala y el suelo, ayudando a acomodar los motores en caso de que se encuentren situados bajo el ala. El diedro también puede implementarse para subir o bajar el centro de gravedad de la aeronave [28].

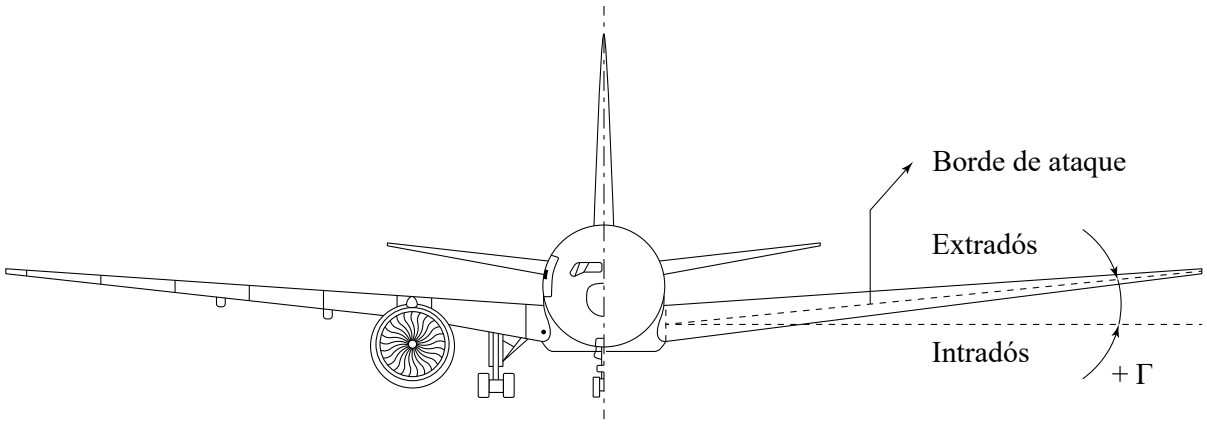


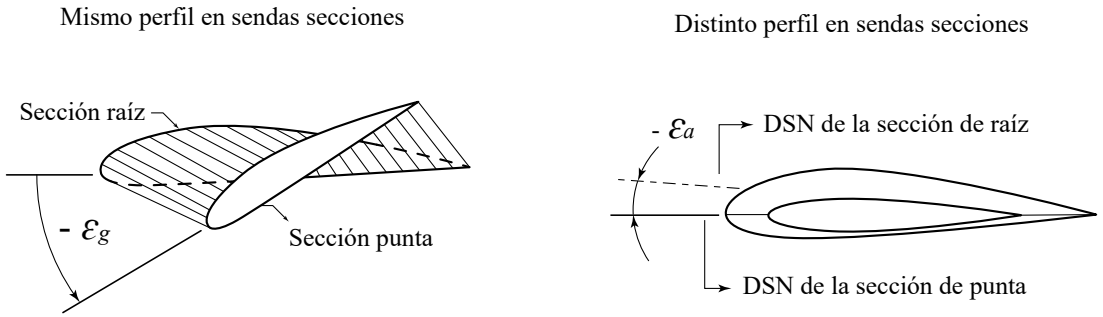
Figura 3.13: Representación esquemática de una aeronave B777-200ER con ángulo de diedro positivo donde se definen los parámetros principales de la vista de frente

Torsión

La torsión es la variación a lo largo de la envergadura del ángulo formado por una línea del perfil con un plano de referencia normal al de simetría [33]. La torsión puede ser negativa, si la diferencia entre el ángulo en la punta y la raíz es negativa o positiva si ocurre el caso contrario.

Cuando la línea de perfil tomada como referencia es la cuerda, se denomina torsión geométrica (ε_g) y se obtiene torsionando físicamente el ala para conseguir un ángulo de ataque en la punta distinto al de la raíz. Para este tipo de torsión se suele emplear el mismo perfil a lo largo de toda la envergadura.

Por otro lado, cuando la línea del perfil es la dirección de sustentación nula (DSN) se denomina torsión aerodinámica (ε_a). Para conseguir esta torsión es necesario utilizar distintos perfiles aerodinámicos en el encastre y en la punta.



(a) Torsión geométrica negativa (b) Torsión aerodinámica

Figura 3.14: Comparación entre torsión geométrica y torsión aerodinámica

3.5. Perfil alar

Se denomina perfil alar o perfil aerodinámico a la forma transversal de un ala [23] que desplazada a través de un fluido crea a su alrededor una distribución de presiones que genera una fuerza de sustentación. Esta fuerza de sustentación y las características de entrada en pérdida del ala dependen en gran medida de la geometría de las secciones de los perfiles aerodinámicos que la componen.

Los perfiles son empleados tanto en el diseño de alas de avión como en las palas de un rotor de helicóptero, turbocompresores, turbinas eólicas, hélices e incluso barcos (hidroalas) [28].

Un perfil se diseña para una una aplicación específica, con el objetivo de que un elemento, como el ala de un avión, pueda operar generando una resistencia aerodinámica mínima en condiciones de crucero. Dentro del programa de diseño y desarrollo de una aeronave, el coste de diseño de un perfil aerodinámico suele suponer un gasto despreciable, en especial si la aeronave es empleada con fines comerciales [28].

3.5.1. Terminología del perfil

Se presenta en la Figura 3.15 los parámetros principales que describen la geometría de un perfil aerodinámico [23]. Estos parámetros son utilizados para determinar las características aerodinámicas del perfil.

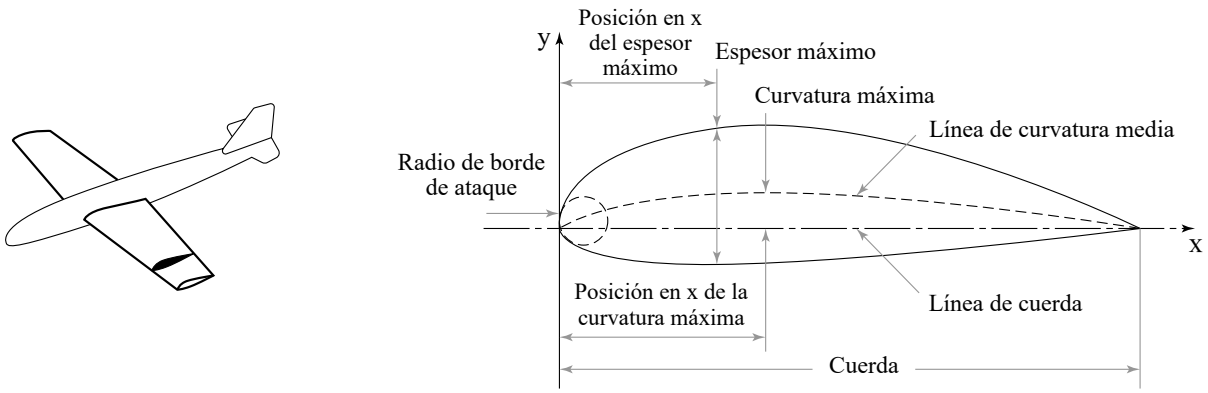


Figura 3.15: Terminología del perfil alar

- La **cuerda** (*Chord*), definida anteriormente en la Sección 3.2, es la línea imaginaria que une el borde de ataque con el borde de fuga.
- La **línea de curvatura media** (*Mean Camber Line*) es la línea equidistante entre el extradós y el intradós [34].
- La **curvatura máxima** (*Maximum camber*) es la máxima distancia entre la línea de curvatura media y la cuerda del perfil [34]. Este valor suele expresarse en tanto por cien (%) del valor de la cuerda.
- El **espesor** es la distancia entre la linea del contorno superior del perfil (extradós) y la linea del contorno inferior (intradós) a lo largo de la cuerda. Esta distancia se mide perpendicular a la linea de curvatura media y se expresa como un porcentaje del valor total de la cuerda [23]. El punto de la cuerda donde este espesor alcanza el máximo valor se denomina **espesor máximo**.
- El **radio de curvatura del borde de ataque** suaviza el contorno del perfil aerodinámico. Este parámetro influye sobre la máxima sustentación que puede generar el perfil así como la resistencia generada en crucero. Cuanto más amplio sea este el radio de curvatura, mayor sustentación generará el perfil a elevados ángulos de ataque [28].

En cuanto al borde de fuga del perfil, suele terminar en forma de pico, sin embargo, en ocasiones se emplea un borde de fuga cuadrado para disminuir el gradiente de presiones adverso generado por el perfil. Este tipo de borde de fuga aparece en aeronaves comerciales con perfiles NLF (*Natural-Laminar-Flow*) [28], donde la máxima curvatura está muy retrasada con respecto a la cuerda.

Los perfiles se clasifican en dos grandes grupos según su simetría. Por un lado, los perfiles simétricos poseen la misma geometría en ambos contornos, superior e interior. Estos perfiles son empleados en rotores de helicóptero o en alas de aviones acrobáticos. Por otro lado, los perfiles asimétricos son capaces de generar una mayor sustentación.

3.5.2. Perfiles NACA

Los perfiles NACA son perfiles aerodinámicos cuya geometría se describe mediante una serie de parámetros que modelan la línea de curvatura media y el espesor.

En 1929, la *National Advisory Commitee for Aeronautics* (NACA), actualmente NASA, generó una nomenclatura que agruparía los perfiles alares en series o familias. Los perfiles NACA se organizan en tres series principales: NACA de cuatro dígitos, cinco dígitos y seis dígitos [23].

Según la serie NACA a la que pertenezca el perfil, la nomenclatura empleada para nombrarlo será el sobrenombre NACA seguido por tantos dígitos comprendidos entre el 0 y el 9 como números indique la serie. De esta forma un perfil NACA serie de cuatro dígitos se nombrará como NACA seguido de cuatro números.

A pesar de la existencia de los perfiles NACA, su implementación dentro de un programa de diseño de alas no es especialmente útil, ya que estos perfiles, aunque siguen siendo utilizados actualmente, pueden ser sustituidos por otro perfiles ajenos a esta nomenclatura. En el siguiente capítulo se expondrá como se ha implementado la nomenclatura NACA de cuatro dígitos dentro del programa y como se ha flexibilizado para poder importar cualquier tipo de perfil.

4

Implementación

4.1. Introducción

Este capítulo trata sobre el desarrollo del programa creado. Aspectos como la filosofía principal empleada, los principios matemáticos usados para modificar y modelar los parámetros vistos en el capítulo anterior y su translación al código fuente del programa serán tratados a lo largo de este capítulo.

4.2. Consideraciones generales

El programa está escrito en *Python*, un lenguaje de alto nivel cuyo aprendizaje resulta especialmente sencillo y es frecuentemente empleado en proyectos que requieren de un corto periodo de desarrollo. Dispone de una amplia variedad de módulos que permiten realizar todo tipo de proyectos, desde sencillos *scripts* matemáticos hasta complejas interfaces gráficas.

El uso de este lenguaje de programación para el desarrollo del proyecto se debe al amplio soporte técnico disponible en *internet* y por ser el lenguaje con el que el Autor de este documento posee más destreza.

El programa consiste en un archivo con extensión *.exe* que puede ser ejecutado en cualquier dispositivo que utilice un sistema operativo *Windows* 10.

4.3. Sistema de referencia

El programa trabaja mediante un sistema de referencia absoluto cartesiano cuyos ejes son nombrados como x, y y z. Los ejes x e y definen el plano vertical (plano xy), mientras que los ejes x y z definen el plano horizontal (plano xz).

Sobre el plano xy se sitúan los perfiles aerodinámicos que describen el ala mientras que el eje z indica la profundidad o envergadura del ala. Los ejes y su sentido positivo aparecen reflejados en la Figura 4.1.

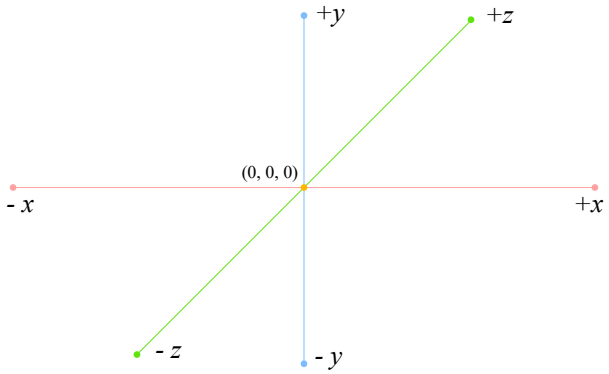


Figura 4.1: Sistema de referencia absoluto

4.4. Discretización del ala

Para abordar el problema de modelado de una superficie alar, la geometría del ala se discretiza en lo que se ha denominado como «secciones».

Las secciones son elementos bidimensionales contenidos dentro de un plano vertical y constituyen el elemento fundamental que emplea el programa para determinar las dimensiones y posición en el espacio de un perfil aerodinámico. Las secciones pueden entenderse como regiones transversales de un ala separadas en el eje z.

Estos elementos son paralelos entre sí y su posición relativa en el espacio determina la geometría del ala. Puesto que una sección se define como un elemento bidimensional, se necesita un mínimo de dos secciones separadas una distancia $\Delta z \neq 0$ en el eje z para definir una geometría tridimensional. De esta forma, se puede decir que todas las alas están constituidas por una sección perteneciente al encastre alar, llamada «Sección 0» y

otra sección final perteneciente a la punta cuyo número de identificación varía en función de las secciones intermedias que formen el ala.

Las secciones son unidas mediante interpolación lineal. Este hecho facilita su implementación y manejo dentro de un programa de diseño, sin embargo, tiene el inconveniente de limitar la unión entre dos secciones a geometrías rectangulares o trapezoidales.

Un ejemplo visual que ayuda a comprender el concepto de las secciones aparece en la Figura 4.2, donde se representa la vista en planta del ala de una aeronave B737. Como se puede observar, el ala está formada por una sección perteneciente al encastre alar (Sección 0), una sección intermedia (Sección 1) y la sección de punta de ala (Sección 2). Las secciones intermedias aparecen como consecuencia de la ruptura de linealidad en alguno de los parámetros del ala. De esta forma, la Sección 1 aparece ya que se produce un cambio en la pendiente del borde de fuga, debido a que el ala lo forman dos «sectores» trapezoidales con mismo ángulo de flecha pero distinto estrechamiento y envergadura.

Los sectores son el espacio comprendido entre dos secciones, donde cada sector va ligado a la sección con su mismo número identificador. En la Figura 4.2 se observan tres sectores, el Sector 0, comprendido entre el eje longitudinal de la aeronave y la Sección 0, el Sector 1, comprendido entre la Sección 0 y la Sección 1 y el Sector 2, comprendido entre la Sección 1 y la Sección 2.

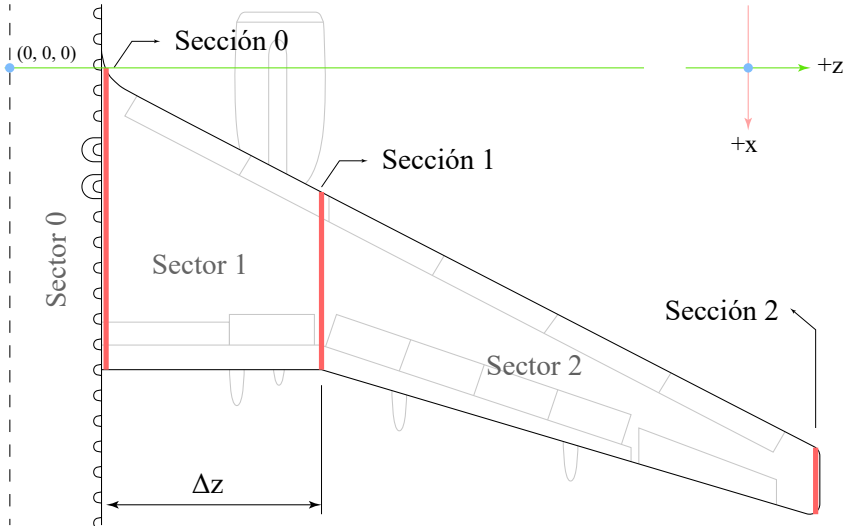


Figura 4.2: Vista en planta del semiala derecho de una aeronave B737 donde aparecen remarcadas en rojo las secciones que componen la geometría

El punto (0, 0, 0) del sistema de referencia global siempre será la intersección entre la

linea perpendicular al eje de simetría de la aeronave que pasa por el borde de ataque de la Sección 0 y el propio eje de simetría del fuselaje (punto azul situado en la intersección entre la línea discontinua y la línea verde de la Figura 4.2).

4.4.1. Definición de la sección

A nivel interno de programa, las secciones son tratadas como objetos que contienen la información mostrada a continuación.

Identificador. Definida como ***name***, es una variable que almacena una cadena de texto con el nombre identificador de la sección.

Nombre del perfil alar. Definida como ***airfoil_name*** es una variable de texto que almacena en nombre del perfil alar empleado para nombrar la sección.

Perfil alar. El perfil es almacenado en dos *arrays*, es decir, en dos variables que contienen las coordenadas de los puntos que definen el perfil. Estas variables son nombradas como $\boldsymbol{x}$ e $\boldsymbol{y}$. Una discusión sobre la discretización del perfil alar se detalla más adelante en la Sección 4.5.

Linea de curvatura media. Al igual que el perfil, este parámetro se almacena en dos variables tipo *array* que contienen las coordenadas en el plano xy de los puntos que definen la linea de curvatura media. Se nombran como ***camber_x*** y ***camber_y***.

Cuerda del perfil. Definida como ***cord***, es una variable que almacena el valor numérico de la longitud de la cuerda del perfil.

Envergadura. Definida como ***span***, es una variable que almacena el valor numérico de la envergadura del sector correspondiente a la sección.

Ángulo de rotación del perfil. Definida como ***alpha***, es una variable que almacena el valor numérico del ángulo de ataque del perfil que compone la sección.

Ángulo de flecha. Definida como ***arrow***, es una variable que almacena el valor numérico del ángulo de flecha del sector correspondiente a la sección.

Ángulo de diedro. Definida como ***diedro***, es una variable que almacena el valor numérico del ángulo de diedro del sector correspondiente a la sección.

Coordenadas absolutas. Bajo el nombre de ***xw***, ***yw*** y ***zw***, estas variables contienen las coordenadas tridimensionales en el sistema de referencia absoluto de todos los puntos que componen la sección. Esta información se almacena en forma de *arrays* y es la que la finalmente se exporta y utiliza para generar la superficie del ala en programas CAD.

Además de estas variables, también se almacenan otras variables auxiliares que especifican el desplazamiento relativo de cada sección respecto a la sección anterior para poder computar su posición absoluta en el espacio. Estas variables reciben el nombre de ***x_offset***, ***y_offset*** o ***z_offset*** según al eje al que corresponda.

Como se ha podido observar, parámetros como la torsión aerodinámica o la torsión geométrica no aparecen nombrados explícitamente. Ambos parámetros son implementados de forma indirecta mediante la variación del ángulo de rotación del perfil y de las coordenadas almacenadas en la variable del perfil alar.

Las variables del sistema no están fijadas dentro de ningún sistema métrico concreto a excepción de los ángulos, expresados en grados. Es por ello que, la dimensionalización de la geometría viene determinada por el sistema métrico empleado por el programa CAD empleado para generar la superficie de ala.

El código desarrollado para la implementación de las secciones se puede consultar en la Subsección D.1.1 del Apéndice D.

4.5. Discretización del perfil

El perfil es el elemento principal que define la forma de una sección, ya que el resto de parámetros solo se encargan de escalar y posicionar el perfil en el espacio.

La forma de un perfil puede llegar a ser muy compleja y definirla mediante ecuaciones supone problemas de compatitibilidad entre programas. La solución comúnmente adoptada para definir la forma de un perfil es discretizarla en una serie de puntos bidimensionales con coordenadas x e y de tal forma que la geometría del perfil esté contenida en la unión lineal de dichos puntos.

En la Figura 4.3 se puede apreciar una comparativa entre un perfil NACA 0012 sin

discretizar y discretizado. Como ya se ha mencionado, la forma de un perfil discretizado viene determinada por la unión lineal entre los puntos que lo componen, de esta forma, cuantos más puntos sean empleados para definir el perfil, mayor resolución tendrá y zonas como el borde de ataque quedarán mejor definidas.

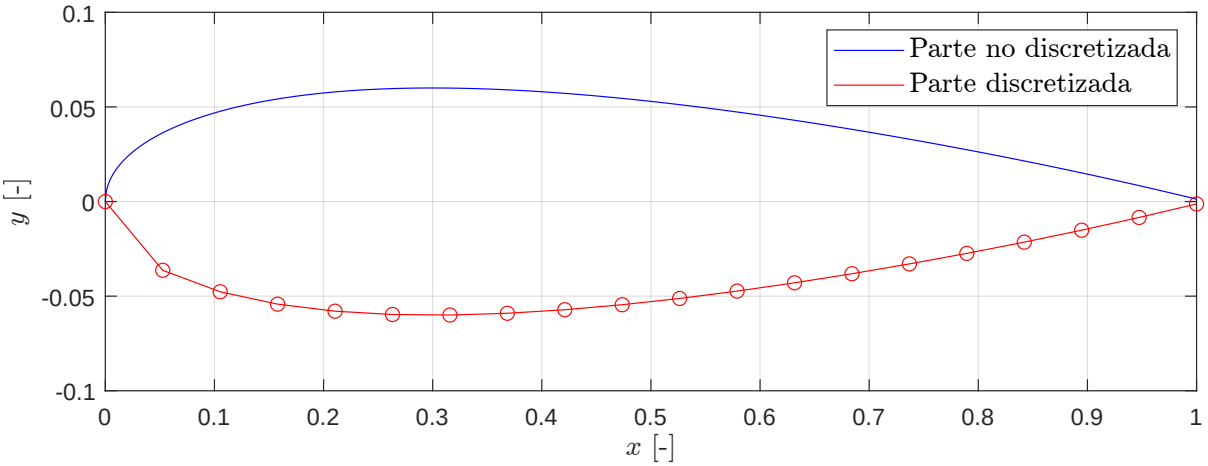


Figura 4.3: Comparación de un perfil alar NACA 0012 con parte superior no discretizada y parte inferior discretizada mediante 20 puntos distribuidos de forma equidistante

De la Figura 4.3 es necesario destacar tres aspectos importantes sobre como se han de representar los puntos de un perfil discretizado para facilitar el traspase de información entre programas o para simplemente facilitar el manejo de los perfiles dentro de un mismo programa.

En primer lugar, la cuerda del perfil debe ser unitaria, donde la diferencia entre las coordenadas x del punto final e inicial es 1. Esto simplifica el proceso de escalado del perfil en caso de que se requiera una cuerda con un valor específico distinto a la unidad. El escalado del perfil se detalla en la Subsección 4.6.1.

En segundo lugar, la línea de cuerda del perfil pasa por la recta definida mediante la ecuación $y = 0$, indicando nulo ángulo de ataque. Esto facilita la rotación del perfil para generar la torsión geométrica deseada.

Finalmente, un perfil alar discretizado posee únicamente dos puntos cuyas coordenadas en x son igual a 1 (borde de fuga) y otros dos cuyas coordenadas en x son 0 (borde de ataque). Los puntos donde $x = 1$ indican el punto inicial y el punto final del perfil, mientras que los puntos donde $x = 0$ forman los puntos intermedios. Un archivo que contenga puntos de un perfil alar debe iniciar con uno de los puntos donde $x = 1$ y deberá

finalizar con el restante. El resto de puntos se ordenan de forma creciente o decreciente según el siguiente criterio.

- Si el punto inicial del archivo es el punto $x = 1$ correspondiente a la parte superior del perfil (extradós), los puntos intermedios comprendidos entre ese punto y el $x = 0$ deben ordenarse según la coordenada x en orden decreciente. A continuación, los puntos comprendidos entre $x = 0$ y $x = 1$ de la parte inferior (intradós) se ordenan en orden creciente según la coordenada x. De esta forma se obtiene un archivo cuyos puntos inicial y final son $x = 1$ y el resto están ordenados en sentido contrario al de las agujas de un reloj.
- En caso de que el punto inicial del archivo sea el $x = 1$ correspondiente al intradós, la organización de los puntos debe ser la opuesta a la descrita anteriormente, quedando los puntos organizados según el movimiento de la agujas del reloj.

Ordenar los puntos mediante este criterio impedirá que se produzcan saltos no deseados cuando se realice la unión entre ellos.

4.5.1. Implementación perfiles NACA de 4 dígitos

Cada uno de los dígitos de la nomenclatura representa una propiedad geométrica. A continuación se define el perfil NACA de cuatro dígitos ya que es la familia de perfiles implementada en el programa desarrollado.

Primer dígito (m_{NACA}). Indica la curvatura máxima del perfil en tanto por cien de la longitud de la cuerda.

Segundo dígito (p_{NACA}). Indica la posición de la curvatura máxima del perfil en tanto por diez de la longitud de la cuerda.

Tercer y cuarto dígito (t_{NACA}). Indica el espesor máximo del perfil en tanto por cien de la longitud de la cuerda.

A continuación, se detalla el proceso que sigue el programa para crear los puntos de un perfil NACA de cuatro dígitos especificado por el usuario.

Función *create_naca*

Esta función se ejecuta para generar los puntos de una perfil NACA de cuatro dígitos cualquiera. Para ello, emplea una cuerda unitaria tal y como se ha justificado anteriormente para simplificar el proceso de escalado posterior.

La función requiere de cuatro parámetros para poder ser ejecutada, m, p y t que definen la geometría del perfil y un parámetro extra, n, que indica el número de puntos usados para definir la forma. Recordar que, a mayor número de puntos mejor será la resolución del perfil.

La función devuelve cuatro *arrays* de valores que contienen las coordenadas tanto del perfil aerodinámico (*x_airfoil*, *y_airfoil*) como de la linea media de curvatura media (*x_camber*, *y_camber*). Esta función se puede consultar en la Subsección D.1.2 del Apéndice D.

Cálculos iniciales

Al ejecutarse, la función define un *array* o conjunto de valores pertenecientes al eje x a partir de los cuales se calculan las coordenadas en el eje y correspondientes para cada valor de x. El número de valores que contiene este *array* se especifica en la variable n proporcionada por el usuario. Los valores se acotan entre 0 y 1 para obtener una cuerda unitaria.

Tal y como ha podido observarse en la Figura 4.3, una separación equidistante de los puntos en la coordenada x no resulta especialmente efectiva para obtener una fiel representación de la geometría real del perfil empleando un bajo número de puntos. Con el objetivo de solventar este problema y disminuir el número de puntos necesarios para describir de forma apropiada el perfil, se emplea una distribución tipo «$1 - \cos$».

La distribución tipo $1-\cos$ acota un conjunto de valores entre 0 y 1 y los distribuye de tal forma que se agrupan entorno a los extremos del conjunto. Así, se consigue aumentar la densidad de puntos en las zonas más críticas del perfil, como es el borde de ataque.

Para visualizar la agrupación de puntos generada por la distribución $1-\cos$ se muestra en la Figura 4.4 un perfil NACA 0012 cuyo extradós ha sido generado mediante una distribución equidistante y el intradós mediante una distribución $1 - \cos$.

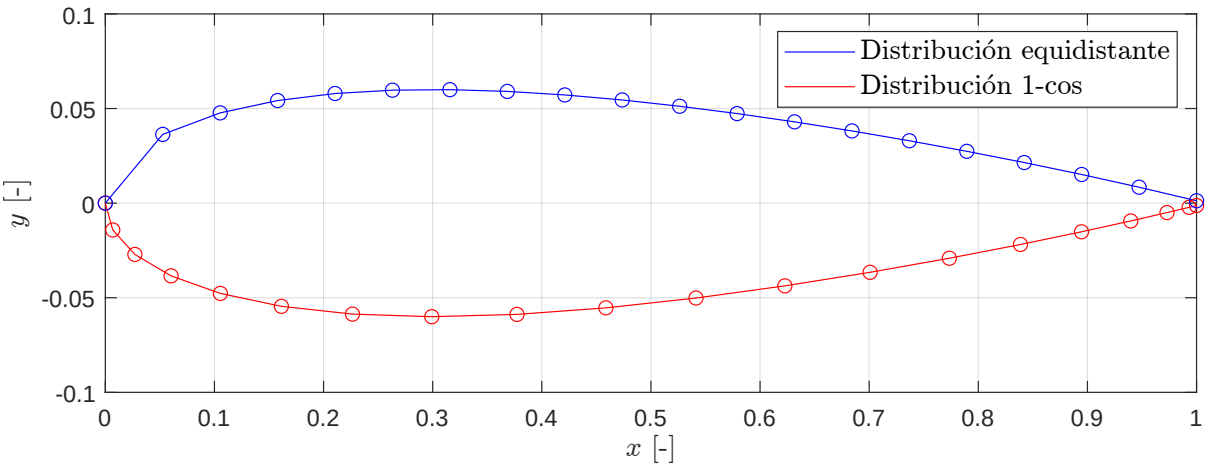


Figura 4.4: Comparación de un perfil alar NACA 0012 con los puntos de la parte superior distribuidos de forma equidistante y parte inferior distribuida mediante $1-\cos$ empleando 20 puntos en cada mitad

Para poder aplicar esta distribución se comienza definiendo un conjunto de valores β tal que $\beta = \{x \in \mathbb{R}/0 \leq x \leq \pi\}$ formado por tantos valores como especifique el usuario. Este conjunto de valores equidistantes se introduce dentro de la Ecuación 4.1 para aplicar la distribución $1-\cos$. El resultado es un conjunto de valores que indican las coordenadas en x de cada punto del perfil.

$$x = \frac{1 - \cos(\beta)}{2} \tag{4.1}$$

A continuación se calcula el valor del espesor en tanto por cien de la cuerda (Ecuación 4.2), y en caso de que el perfil sea asimétrico también se calcula la posición de la curvatura máxima en tanto por diez de la cuerda (Ecuación 4.3) y el espesor máximo del perfil en tanto por cien de la cuerda (Ecuación 4.4).

$$t = t_{\text{NACA}}/(c \cdot 100) \tag{4.2}$$

$$p = p_{\text{NACA}}/(c \cdot 10) \tag{4.3}$$

$$t = t_{\text{NACA}}/(c \cdot 100) \tag{4.4}$$

Cálculo de la linea de curvatura media

El cálculo de la linea de curvatura media es muy sencillo de realizar en perfiles simétricos[1] como el NACA 0012 visto anteriormente en la Figura 4.4 ya que esta línea queda definida por dos puntos cuyas coordenadas son (x_0, y_0)=(0, 0) y (x_1, y_1)=(1, 0), es decir, coincide con la expresión $y = 0$ que es la linea de cuerda.

En perfiles asimétricos, la línea media de curvatura (y_c) se define como dos arcos parabólicos tangentes en la posición de la cuerda donde se produce la máxima curvatura. Estos arcos vienen dados por [35] y se muestran en la Ecuación 4.5.

$$y_c(x) = \begin{cases} \dfrac{m}{p^2} \cdot (2px - x^2) & \text{si} \quad 0 \leq x \leq p \\ \dfrac{m}{(1-p)^2} \cdot [(1-2p) + 2px - x^2] & \text{si} \quad p \leq x \leq c \end{cases} \tag{4.5}$$

Esta ecuación a trozos se implementa para calcular la línea de curvatura media del perfil NACA 8414 (asimétrico) mostrado en la Figura 4.5. A partir del segundo dígito y según la nomenclatura vista anteriormente, se espera que el punto de máxima curvatura (p) del perfil se de al 40 % de su cuerda ($x = 0.4$). Este hecho se puede comprobar en la Figura 4.5.

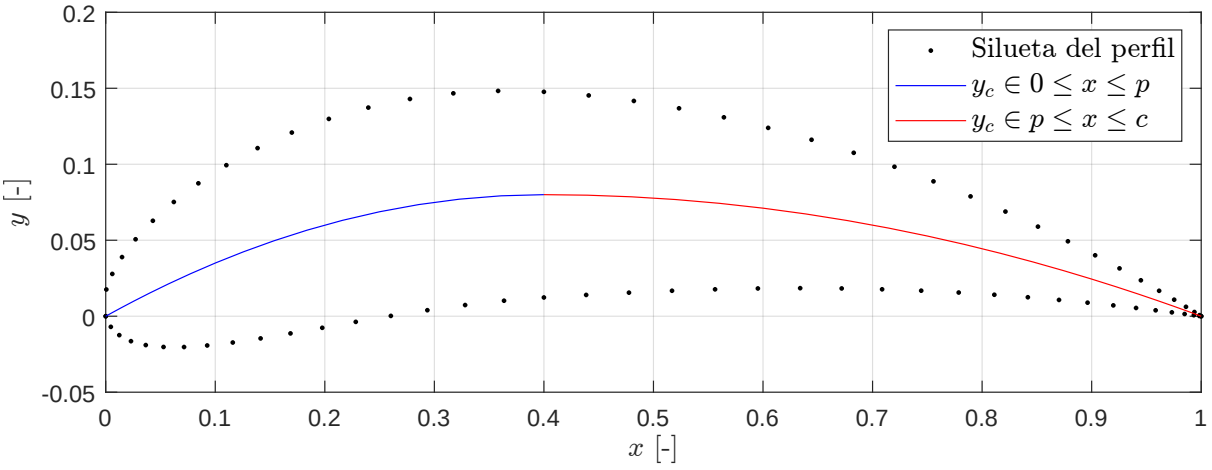


Figura 4.5: Representación de la línea de curvatura media de un perfil NACA 8414

[1]El programa detecta que un perfil es simétrico cuando no exisite espesor, es decir, el parámetro $m = 0$.

Cálculo de la distribución del espesor

El siguiente paso es calcular el espesor del perfil en cada punto de la cuerda. El método unificado para definir la distribución del espesor de un perfil NACA depende únicamente del espesor máximo, t. La curva de mejor ajuste para obtener las coordenadas del espesor es, de acuerdo con [35] la Ecuación 4.6.

$$\pm y_t = \frac{t}{0.2} \cdot (0.2969 \cdot \sqrt{x} - 0.126 \cdot x - 0.3516 \cdot x^2 + 0.2843 \cdot x^3 - 0.1036 \cdot x^4) \tag{4.6}$$

Aplicando esta definición a un perfil simétrico se obtienen directamente las coordenadas de todos puntos que definen el perfil (Figura 4.7).

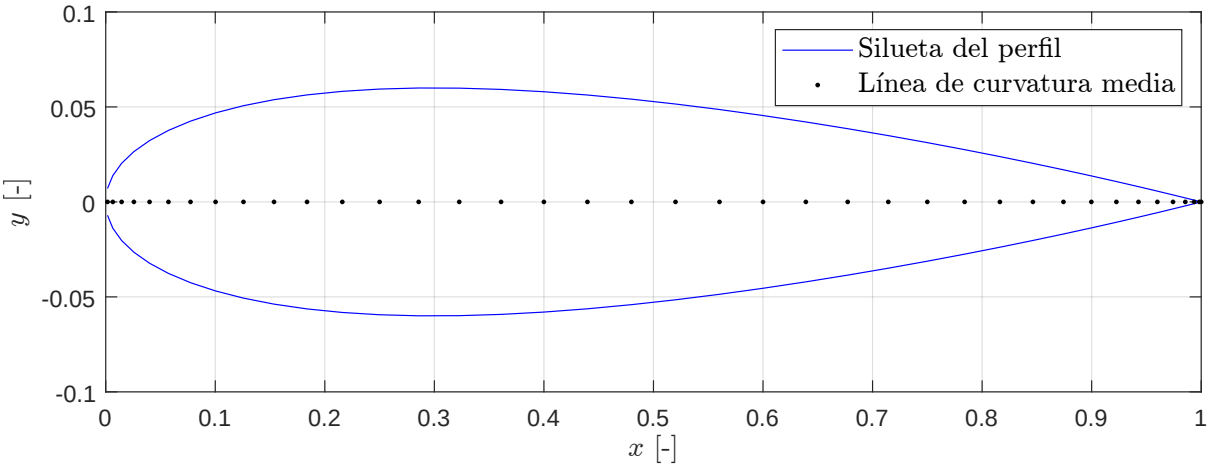


Figura 4.6: Perfil NACA 0012

Sin embargo, en el caso de los perfiles asimétricos, la pendiente de la línea de curvatura media varía en función de la posición en la cuerda, por tanto, y, aunque la expresión que define el espesor (Ecuación 4.6) sea la misma, es necesario calcular la derivada de la línea de curvatura media para determinar el ángulo de la pendiente en cada punto y así poder situar correctamente el espesor, tangente a y_c. La función de la recta tangente a la linea de curvatura media viene determinada por la Ecuación 4.7.

$$\frac{dy_c}{dx} = \begin{cases} \dfrac{2m}{p^2} \cdot (p - x) & \text{si} \quad 0 \leq x \leq p \\ \dfrac{2m}{(1-p)^2} \cdot (p - x) & \text{si} \quad p \leq x \leq c \end{cases} \tag{4.7}$$

Una vez calculados los valores de la pendiente de la recta tangente en cada punto se calcula el ángulo de la pendiente con el eje horizontal (θ) en cada punto mediante la Ecuación 4.8.

$$\theta = \arctan\left(\frac{dy_c}{dx}\right) \tag{4.8}$$

La descripción de este proceso se puede comprender de manera visual a través de la Figura 4.7

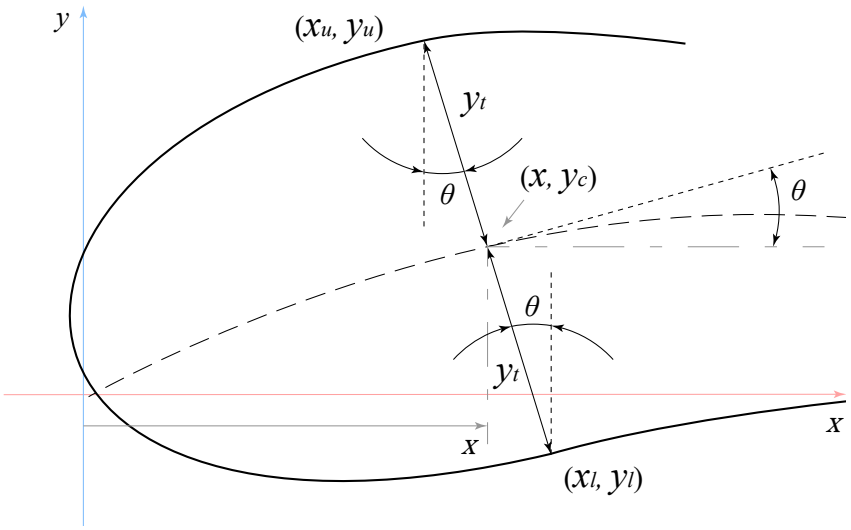


Figura 4.7: Detalle del proceso de modelado de un perfil aerodinámico asimétrico [35]

Finalmente, las distribuciones de espesor se combinan matemáticamente con la línea de curvatura media mediante el siguiente método. Las coordenadas de la superficie superior se calculan mediante la Ecuación 4.9.

$$\begin{aligned} x_u &= x - y_t \cdot \sin(\theta) \\ y_u &= y_c + y_t \cdot \cos(\theta) \end{aligned} \tag{4.9}$$

Y las coordenadas de los puntos de la parte inferior se calculan mediante la Ecuación 4.10.

$$\begin{aligned} x_l &= x + y_t \cdot \sin(\theta) \\ y_l &= y_c - y_t \cdot \cos(\theta) \end{aligned} \tag{4.10}$$

Una vez computados todos los puntos, se ordenan según convenio y se presentan en un gráfico.

Dentro del programa, el usuario puede variar los parámetros de los perfiles NACA de cuatro dígitos a conveniencia y podrá ver en tiempo real el efecto de cada parámetro sobre la forma que adopta el perfil.

Importación de perfiles

El programa posee una opción que permite importar cualquier tipo de perfil, ya sea de nomenclatura NACA o no. Para ello solo hay que suministrar un archivo que contenga organizado en dos columnas las coordenadas x e y de los puntos que conforman el perfil.

Es importante que los puntos estén organizados según el convenio mencionado, de tal forma que el archivo inicie con un punto cuya coordenada x sea 1 y el resto de puntos se encuentren organizados en sentido horario o anti horario hasta finalizar con el otro punto de coordenada $x = 1$.

4.6. Manipulación de las secciones

Las operaciones detalladas a continuación se encuentran incorporadas dentro de la función *compute* perteneciente a la clase *section* (Subsección D.1.1 del Apéndice D) mencionada anteriormente. Esta función toma los parámetros de la sección y calcula sus dimensiones, posición y rotación en el espacio a partir del sistema de referencia absoluto.

4.6.1. Escalado

Escalar una sección consiste en variar el área encerrada dentro de la misma en el plano xy y es utilizado para variar la longitud de la cuerda del perfil.

Para escalar una sección se realiza el producto vectorial de un factor de escalado por las coordenadas x e y de todos los puntos que componen la sección. Puesto que el perfil se define mediante una cuerda unitaria, en caso de querer incrementar el doble el tamaño de la sección solo hay que multiplicar por 2 cada conjunto de coordenadas para obtener una sección de cuerda 2. Un esquema de este proceso se representa en la Figura 4.8a.

4.6.2. Rotado

La rotación de las secciones es utilizada para proporcionar torsión geométrica al ala. El programa realiza la rotación de cada sección al rededor del eje z empleando como punto de rotación el punto del perfil que define el borde de ataque (ver Figura 4.8b).

Para rotar la sección un ángulo α determinado, se emplea la matriz de rotación de *Euler* al rededor del eje z (Ecuación 4.11) y se aplica a cada punto que compone la sección para obtener las nuevas coordenadas del punto rotado.

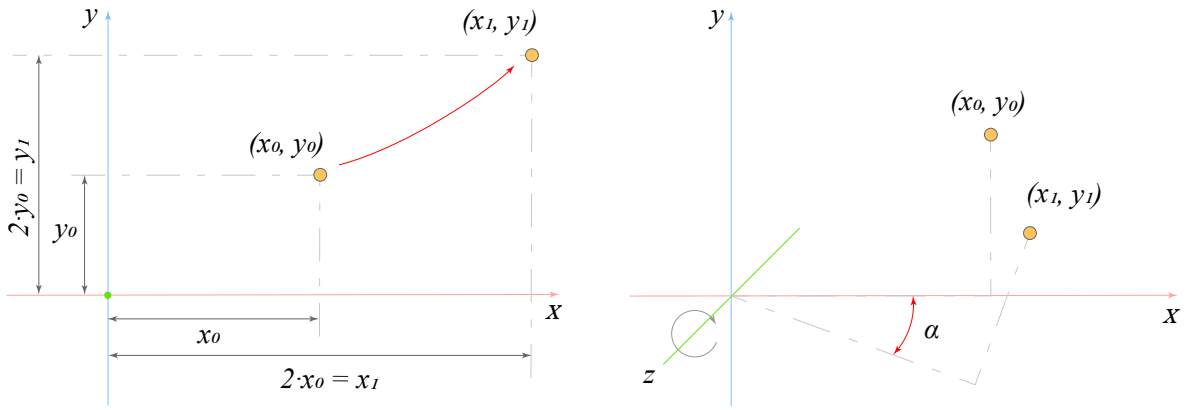
$$\begin{bmatrix} x_{\text{rotado}} \\ y_{\text{rotado}} \\ z_{\text{rotado}} \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \cdot \cos(\alpha) + y \cdot \sin(\alpha) \\ -x \cdot \sin(\alpha) + y \cdot \cos(\alpha) \\ z \end{bmatrix} \tag{4.11}$$

Nótese como ahora aparecen puntos con coordenadas en los ejes x, y y z. Esto se debe a que, a diferencia de las coordenadas bidimensionales que componen un perfil aerodinámico, las secciones están compuestas por puntos tridimensionales.

4.6.3. Envergadura

La envergadura se define a través de la coordenada en z de los puntos de la sección. Las secciones, al estar contenidas dentro del plano xy solo tendrán un único valor de z común a todos los puntos.

El usuario especifica dentro del programa el valor de la envergadura de cada sector, no de la geometría alar completa. De esta forma puede asignar valores concretos de



(a) Operación de escalado de un punto al doble de su posición inicial

(b) Operación de rotación de un punto un ángulo α alrededor del eje z

Figura 4.8: Representación de las operaciones de escalado y rotado de un punto

envergadura a cada sector por separado. La envergadura total del ala es la resultante del sumatorio de las envergaduras individuales de cada sector.

Para calcular la coordenada en z de los puntos de una sección determinada, el programa toma el valor almacenado con el nombre z_offset, que indica el desplazamiento absoluto en z de la sección anterior, y le añade el valor de la envergadura del nuevo sector. El resultado de esta operación es el desplazamiento en z de la nueva sección con respecto al sistema de referencia absoluto.

4.6.4. Ángulo de flecha y diedro

El ángulo de flecha y diedro no son más que el resultado del desplazamiento relativo entre dos secciones en los ejes x o y. Cuando dos secciones separadas una determinada distancia $z \neq 0$ son desplazadas relativamente una respecto a la otra en el eje x se obtiene en la geometría ángulo de flecha, mientras que si este desplazamiento relativo se produce en el eje y se obtiene ángulo de diedro.

El programa calcula las posiciones de las secciones con ángulo de flecha y diedro de los sectores comprendidos entre dos secciones. Es decir, la Sección 0 es la única a la que no se le puede implementar ni ángulo de ataque ni diedro ya que el sector solo se compone por una sección y no tendría sentido implementar estos ángulos en un sector que no forma parte de la propia geometría del ala. En los demás sectores, como el Sector 1, comprendido entre la Sección 0 y la Sección 1, si se le podrán aplicar estos ángulos de

forma individual.

Para realizar el cálculo del desplazamiento en x o y de una sección con ángulo de flecha o diedro es preciso conocer la posición del borde de ataque de la sección más cercana al eje del fuselaje que compone el sector (x_{a-1}, y_{a-1}), la envergadura dada al sector (b_a) y el ángulo de flecha o diedro que se desea implementar.

El proceso para el cálculo del ángulo de flecha o diedro es idéntico en ambos casos. A continuación se describe en detalle únicamente el cálculo del ángulo de flecha (ver Figura 4.9).

El objetivo es calcular el desplazamiento absoluto en x del borde de ataque de la sección más alejada al fuselaje (x_a). Para ello se calcula primeramente el desplazamiento relativo entre secciones a partir de la envergadura del sector b_a y el ángulo φ y posteriormente se añade el desplazamiento absoluto en x del bode de ataque perteneciente a la sección más cercana al fuselaje (x_{a-1}). El valor resultante es la distancia absoluta respecto al eje z que se debe sumar a todas las coordenadas x que componen la sección a. Este cálculo se refleja en la Ecuación 4.12.

$$x_a = x_{a-1} + b_a \cdot \tan(\varphi) \tag{4.12}$$

El proceso para el cálculo del ángulo de diedro es idéntico, pero en este caso la expresión empleada para determinar el desplazamiento en y en la mostrada en la Ecuación 4.13.

$$y_a = y_{a-1} + b_a \cdot \tan(\Gamma) \tag{4.13}$$

4.7. Geometrías elípticas

Los métodos de generación de secciones vistos anteriormente únicamente permiten crear sectores rectangulares o trapezoidales. Mediante la creación de múltiples secciones y colocándolas manualmente de forma cuidadosa en la posición correspondiente se pueden llegar a crear geometrías elípticas, no obstante, este proceso es largo, poco eficiente y costoso. Para automatizarlo, existe dentro del programa una pestaña que permite crear geometrías de alas elípticas de forma automática.

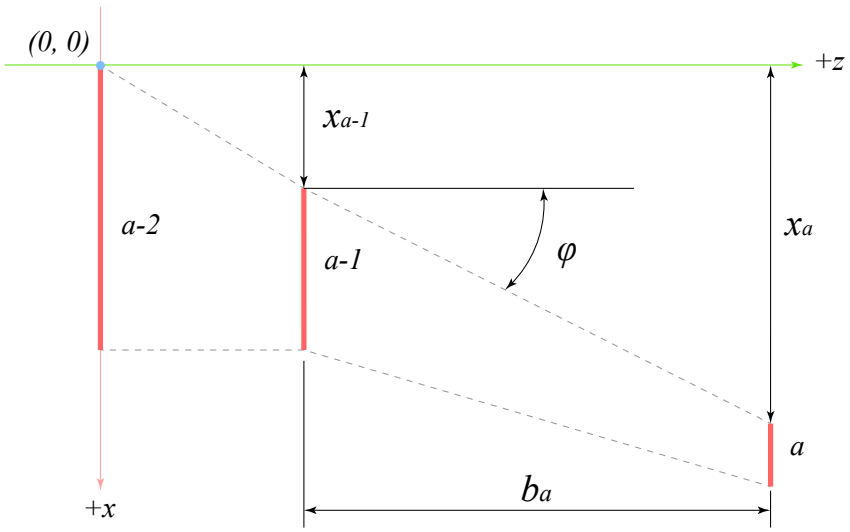


Figura 4.9: Vista en planta del semiala derecho de una aeronave B737 donde se detalla el proceso de implementado del ángulo de flecha

Las geometrías elípticas generadas pueden ser combinadas junto con otros sectores trapezoidales, sin embargo, la elipse siempre constituye la sección de punta de ala.

El procedimiento empleado para crear las geometrías elípticas es generar un conjunto de secciones que son escaladas y posicionadas de tal forma que se asemejan a la forma de la elipse mediante los procesos de unión lineal tratados anteriormente. Las geometrías elípticas no son más que un conjunto de secciones unidas a través de sectores rectangulares.

La función que realiza esta tarea se denomina *create_eliptical_wing* y se puede consultar en la Subsección D.1.3 del Apéndice D. El usuario proporciona la envergadura, el ángulo de diedro, el ángulo de flecha, el número de secciones y la posición de la última sección que forma la punta de ala con respecto a la del inicio de la elipse. La función devuelve un diccionario (ver Sección 5.2) que contiene la información de todas la secciones creadas.

Las coordenadas en y de la elipse se definen a partir de la Ecuación 4.14, donde a es el semieje mayor y b el semieje menor. El sumatorio de la envergadura de los sectores contenidos dentro de la elipse será a, mientra que b se define como la mitad de la cuerda de la sección previa a la elipse.

$$x = \sqrt{b^2 \cdot \left(1 - \frac{z^2}{a^2}\right)} \tag{4.14}$$

En este caso, z es un *array* de puntos que indican la posición en z de las secciones y se encuentran espaciados según la distribución $1 - \cos$.

El resultado de esta operación son las coordenadas x y z donde se sitúa el borde de ataque de las secciones que componen la elipse. Tal y como se observa en la Figura 4.10, donde se presenta una elipse definida por 6 puntos, la coordenada x es la mitad del valor que debe tener la cuerda para cada sección, por tanto, el valor de la cuerda se obtiene multiplicando la ordenada por 2.

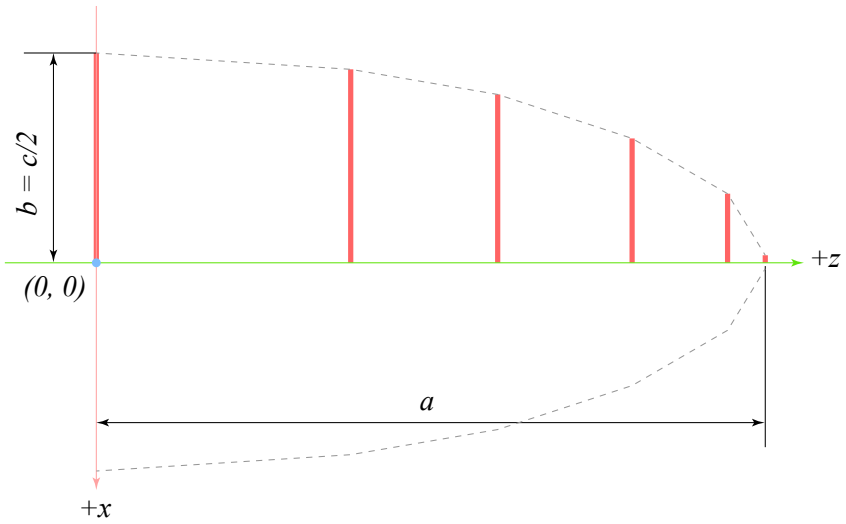


Figura 4.10: Representación de los parámetros de una elipse definida mediante 6 puntos separados mediante la distribución $1 - \cos$

El siguiente paso es situar cada sección en la posición x correspondiente. Este proceso se realiza siguiendo los siguientes pasos.

- 1°. Se calcula la diferencia entre el valor de la cuerda en la sección inicial (la más cercana a la aeronave) y el valor de la cuerda de la sección que se quiere posicionar. Esto resulta en un valor que indica la diferencia entre cuerdas.
- 2°. La diferencia de cuerdas calculada se multiplica por un valor que indica la posición relativa en tanto por ciento de la posición en x del borde de ataque de la sección a posicionar respecto al borde de ataque de la sección inicial. Este valor varía entre 0 % y 100 %. Un valor de 0 % producirá una geometría elíptica con borde de ataque recto. En caso de que el valor sea 100 % se generará un ala elíptica con borde de fuga recto. Si el valor es 50 % resultará en un ala elíptica real.

La forma mostrada en la Figura 4.10 es una elipse pura, lo que significa que una

línea recta se extiende de punta a punta al 50 % de la cuerda. En las construcción de aeronaves elípticas no existe ningún requisito de que la forma en planta deba cumplir con esa geometría. En realidad, resulta estructuralmente más práctico diseñar el ala de tal manera que el cuarto de cuerda fuera una línea recta, tal y como se hizo en el diseño del *Supermarine Spitfire* (Figura 4.11). Esto simplifica el diseño y permite que un larguero principal recto se coloque en el cuarto de cuerda [28]. Por este motivo, es de especial interés incorporar la opción de que el programa pueda variar la geometría del borde de ataque de un ala elíptica.

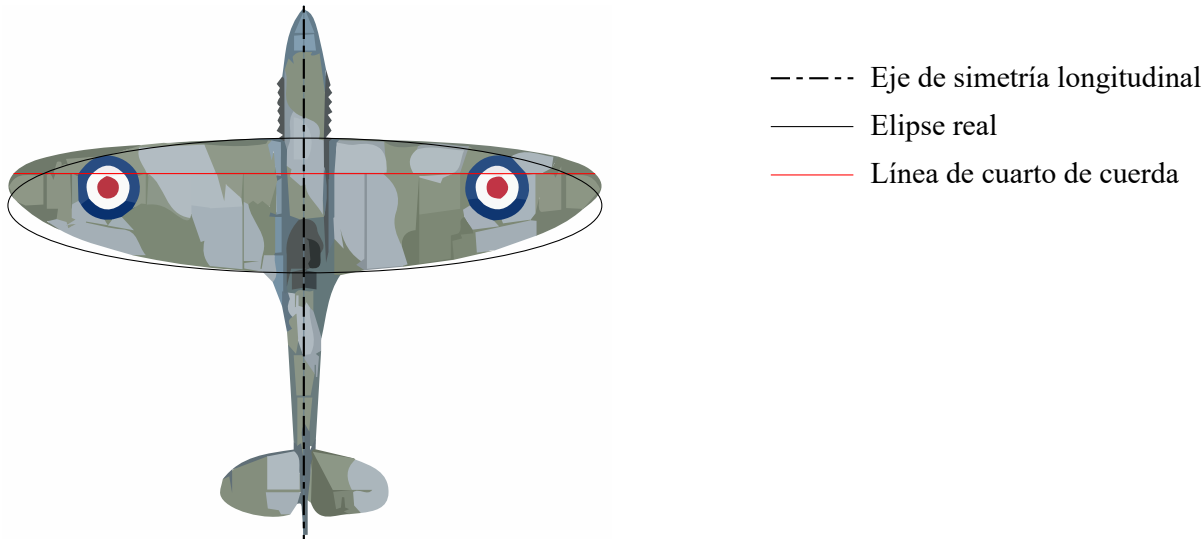


Figura 4.11: Vista en planta de la aeronave *Supermarine Spitfire* donde se compara su ala elíptica con una geometría elíptica real

- 3°. Finalmente, se desplaza la geometría hasta alinearse el borde de ataque de la primera sección con el borde de ataque de la sección previa a la parte elíptica.

5

Gestión de datos

5.1. Introducción

Una vez conocidos los procedimientos empleados para determinar la posición, rotación y tamaño de las secciones que componen el ala se ha de conocer como almacena y organiza el programa toda esta información. Se detalla, a través de este capítulo, las estructuras usadas para guardar las secciones, las operaciones que se pueden realizar con ellas y como se exportan finalmente los datos para poder se utilizados por otros programas. El capítulo concluye con el desarrollo de un *script* para FUSION 360® que automatiza por completo el proceso de importado de las secciones a este programa.

5.2. Gestión de las secciones

Las secciones creadas por el usuario se almacenan dentro de un diccionario denominado «*sections*». Un diccionario consiste en un contenedor de información cuyo funcionamiento se asemeja al uso de un diccionario convencional, donde aparece una palabra seguida de su definición. El diccionario del programa funciona de forma similar, las secciones son nombradas internamente con un nombre identificador y su definición consiste en todos los parámetros que definen el objeto: perfil usado, ángulos de flecha y diedro, longitud de cuerda, etc.

La diferencia principal con respecto a un diccionario convencional es que las palabras, o el nombre identificador de cada sección, no aparece ordenado o estructurado en orden

alfabético ni de ninguna manera concreta. Por ello, para seleccionar una sección determinada almacenada dentro de este contenedor, se ha de especificar el nombre preciso de la sección que se pretende manipular.

Puesto que en ningún momento es conocido el número de secciones que formarán finalmente la geometría, estas se crean, borran o editan de forma dinámica. Para ello, el programa conoce en todo momento el número de secciones que se han creado y el nombre identificador de cada una.

Los nombres que reciben las secciones siguen la nomenclatura siguiente: el antenombre «*mainrib_*» seguido por el número de secciones existentes anterior a la creación de la nueva sección. De esta forma, la sección situada en el encastre alar siempre se llamará «*mainrib_0*» mientras que la de punta de ala irá adoptando un nombre dinámico en función del número de secciones intermedias.

En cuanto a las elipses, el nombre del diccionario que contiene las secciones proporcionadas por la función *create_eliptical_wing* se denomina «*sectionsElip*» y la forma de nombrarlas es idéntica al caso anterior a excepción del antenombre, donde se utiliza el término «*tiprib_*» seguido del número identificador.

La justificación de emplear dos diccionarios distintos para almacenar las secciones trapezoidales y elípticas por separado viene del hecho de que las secciones elípticas solo pueden componer la sección de punta de ala. Según el procedimiento empleado, si todas las secciones se almacenaran dentro del mismo diccionario con el mismo antenombre y un usuario ha creado anteriormente una geometría elíptica, las nuevas secciones adoptarían un número identificador erróneo, ya que se posicionarían a continuación de la estructura elíptica y no antes del comienzo de esta.

Las secciones pueden ser creadas, editadas o eliminadas. A continuación, se profundiza sobre estos procesos.

Crear secciones

Cada vez que el usuario añade una sección, el programa crea una instancia de la clase *section*. Esta clase no es más que una estructura sin contenido que define qué variables componen una sección y las funciones que emplea para calcular su posición en el espacio.

Para completar la información de la nueva instancia, lo primero que se realiza es la asignación del nombre tal y como se ha visto anteriormente y se añade al diccionario *sections*. A continuación, el programa toma algunos de los parámetros de la instancia anterior y se los atribuye a la nueva con el fin de garantizar consistencia y agilizar el proceso de diseño. Estos parámetros son: las coordenadas de los puntos del perfil y la curvatura media, el desplazamiento absoluto del borde de ataque en los ejes x, y y z, la longitud de la cuerda y los ángulos de diedro y flecha. Finalmente, se ejecuta la función *compute* contenida dentro de la clase *section* para crear las coordenadas tridimensionales de los puntos de la sección.

Estas operaciones las realiza la función *addsection* contenida en el archivo de la Subsección D.1.6.

Un caso especial es la creación de la Sección 0, generada automáticamente al inicio del programa. Puesto que esta sección no posee una sección previa de la que pueda heredar valores, estos son proporcionados por defecto al inicio. La Sección 0 puede ser editada, sin embargo, siempre iniciará con los valores de todos los parámetros nulos, a excepción de la cuerda de valor unitario y del perfil alar, NACA 0012.

Editar secciones

Cuando se actualiza algún parámetro de una sección cualquiera, se ejecuta la función *compute* de esa misma sección para actualizar sus dimensiones y posición. Algunos parámetros, como la envergadura del sector o el ángulo de flecha y diedro, afectan indirectamente a la posición de las secciones siguientes. Por este motivo, se recalculan las coordenadas de todas las secciones existentes posteriores a la sección editada.

Este proceso lo realiza la función *updateposition*, contenida en el archivo mostrado en la Subsección D.1.6.

Eliminar secciones

Las secciones se borran de forma secuencial en orden inverso a como fueron añadidas, de tal forma que, cuando el botón de «Eliminar sección» es pulsado, se borra la última sección creada. La última sección se determina tomando aquella con el numero

identificador más alto.

Sólo las secciones generadas por medio del botón «Añadir sección» podrán ser eliminadas. Esto quiere decir que, la Sección 0, creada automáticamente al inicio del programa, no puede ser borrada. De igual forma, las secciones pertenecientes a la geometría elíptica, creadas automáticamente por el programa, tampoco pueden ser eliminadas mediante el botón «Eliminar sección», si no que desaparecen automáticamente al dar un valor nulo a la envergadura de la sección elíptica.

5.3. Exportación de datos

El objetivo final del proyecto es poder exportar las geometrías creadas a otros programas CAD capaces de generar las superficies. Por ello, el exportado de perfiles y geometrías completas es uno de los aspectos más importantes del programa.

La transmisión de información entre *software* distintos se realiza por medio de archivos de texto con extensión *.txt*, *.dat* o *.csv*. Estos archivos están formados por 2 o 3 columnas donde cada una representa un eje diferente. En la exportación de perfiles alares solo se involucran dos columnas puesto que los puntos únicamente poseen coordenadas en x e y mientras que los archivos de secciones contienen 3 columnas, puesto que se introduce la coordenada de profundidad z. De esta forma, en cada fila se almacenan las coordenadas de un punto concreto. El archivo contendrá tantas filas como puntos definan la sección.

Los archivos de extensión *.txt*, *.dat* son idénticos. La principal diferencia entre estos archivos y aquel de extensión *.csv* es que en *.txt* y *.dat* las columnas se separan mediante espacios tabulados (Figura 5.1a), mientras que en *.csv* se separan mediante comas (*CSV: Comma Separated Values*) (Figura 5.1b).

Los procesos de exportado de perfiles y de geometrías se describen a continuación.

5.3.1. Exportación de perfiles

Los perfiles NACA de cuatro dígitos creados por un usuario pueden ser exportados a través de la pestaña de «Editor de perfil», descrita más adelante.

NACA 0012	
1.0	0.0
0.883022	0.015804
0.586824	0.046552
0.25	0.059407
0.116978	0.049459
0.030154	0.028467
0.0	0.0
0.0	0.0
0.030154	-0.028467
0.116978	-0.049459
0.25	-0.059407
0.586824	-0.046552
0.883022	-0.015804
1.0	0.0

(a) Archivo de extensión *.txt*

NACA 0012
1.0 , 0.0
0.883022 , 0.015804
0.586824 , 0.046552
0.25 , 0.059407
0.116978 , 0.049459
0.030154 , 0.028467
0.0 , 0.0
0.0 , 0.0
0.030154 , -0.028467
0.116978 , -0.049459
0.25 , -0.059407
0.586824 , -0.046552
0.883022 , -0.015804
1.0 , 0.0

(b) Archivo de extensión *.csv*

Figura 5.1: Comparación entre archivos de extensión *.txt* y *.csv* en los que se ha exportado las coordenadas de un perfil alar NACA 0012 con una resolución de 14 puntos

La arquitectura del archivo generado es la siguiente. En la primera línea se encuentra el nombre del perfil, en este caso «NACA» seguido de cuatro dígitos. El resto de líneas contienen las coordenadas de los puntos que componen el perfil ordenadas en sentido contrario al movimiento de las agujas del reloj. En la primera columna se escriben las coordenadas en x de los puntos y en la segunda columna las coordenadas en y. Todo esto se observa en la Figura 5.1.

La ruta donde se guarda este archivo es especificada por el usuario.

La función encargada de realizar esta tarea se llama *save_airfoil* y se encuentra dentro del archivo *file_manager* expuesto en la Subsección D.1.4 del Apéndice D.

5.3.2. Exportación de geometrías

Para exportar una geometría se ha de acudir a la pestaña «Editor de geometría» del programa.

Cuando un usuario exporta una geometría debe facilitar una ruta de guardado. A diferencia de la exportación de un perfil, una geometría se compone de varios perfiles o secciones que derivan en la generación de varios ficheros. Con el fin de evitar que todos estos archivos inunden el directorio proporcionado, el programa crea una carpeta

donde los almacena, bajo el nombre de «*wing*» en caso de que solo exista una carpeta con este nombre o «*wing_**id***». El índice *id* es un valor numérico que comienza en 1 y se incrementa en pasos unitarios en caso de existir otra carpeta con el mismo nombre en el mismo directorio. De esta forma, si el usuario guarda una geometría en el escritorio y observa que se ha creado una carpeta con el nombre «wing_2» puede esperar la existencia de otra carpeta «wing_1» en ese mismo directorio.

Los archivos generados, a diferencia de los de perfiles, no inician con el nombre del perfil usado, si no que comienzan directamente con los valores de los puntos. En los demás aspectos, la arquitectura interna del archivo es idéntica a la de los perfiles.

Por otro lado, en el proceso de guardado de un perfil, el usuario puede concretar el nombre con el que el desea guardar el archivo, sin embargo, en el exportado de una geometría, los ficheros se nombran de forma automática según el nombre interno de la sección correspondiente (*mainrib_**id***, *tiprib_**id***). La ventaja principal de este procedimiento es la agilización de la operación y el ordenado automático de todas las secciones dentro de la carpeta.

Además de estos archivos, se genera dentro de la carpeta que contiene las secciones del ala un archivo adicional, llamado «*geometry*», que contiene una recopilación de las coordenadas de todos los puntos que componen las secciones, la cuales se separan mediante el indicador «&».

5.4. Importado de geometría a programas CAD

Como se ha esclarecido al comienzo de este documento, los programas CAD suelen tener un módulo incorporado que permite representar y unir en un plano las coordenada contenidas en un archivo de texto.

Generalmente, este módulo o *script* es muy básico, ya que solo permite importar y unir las coordenadas de un único archivo de forma automática. El funcionamiento de este módulo perteneciente al programa Fusion 360®, llamado *ImportSplineCSV*, se refleja en el diagrama de flujo de la Figura 5.2. El funcionamiento de los módulos análogos a este pertenecientes a otros programas presentan un funcionamiento similar o idéntico.

En la Figura 5.3 se detalla el proceso a seguir para importar varios archivos de puntos.

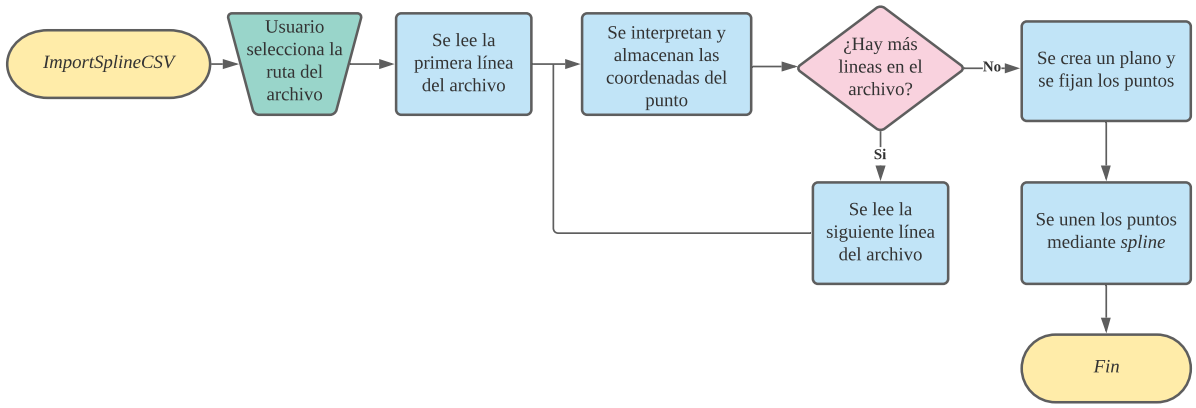


Figura 5.2: Diagrama de flujo del *script ImportSplineCSV* de Fusion 360®

Tal y como se observa en el diagrama de la Figura 5.3, cada vez que se desea importar una sección nueva se requiere la intervención del usuario para ejecutar el módulo y seleccionar la ruta del archivo a importar. Dependiendo de las secciones que compongan la geometría este proceso puede llegar a resultar un tanto tedioso y que acaba consumiendo un tiempo importante para el usuario, un tiempo que podría ser dedicado a realizar otras tareas más relevantes.

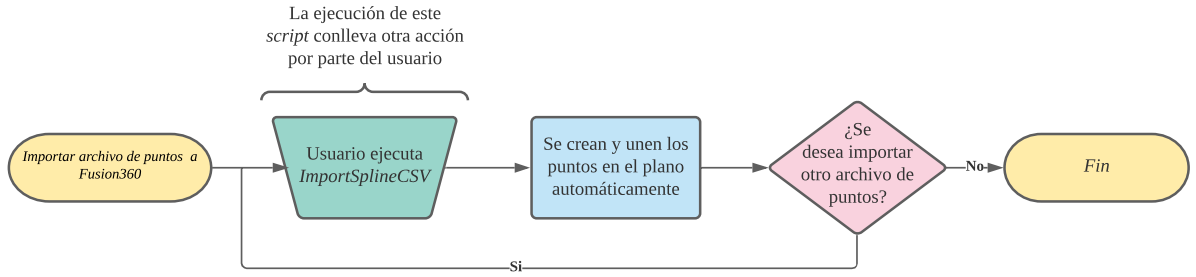


Figura 5.3: Diagrama de flujo del proceso de importado de un archivo de puntos a Fusion 360®

Para automatizar este proceso, se ha diseñado un nuevo módulo que únicamente requiere de una sola acción por parte del usuario para importar la geometría del ala completa. Dicho módulo se ha desarrollado para ser ejecutado dentro del programa Fusion 360®.

5.4.1. Automatización del importado de geometrías completas

El módulo *ImportSplineCSV* posee dos graves inconvenientes que lo descartan para realizar de manera eficiente esta tarea.

Por un lado, el inconveniente ya tratado de que solo permite importar un archivo por cada vez que se ejecuta el módulo. Por otro lado, el método empleado para unir los puntos es mediante *splines*, es decir curvas diferenciables definidas en porciones. Se ha comprobado que este método de unión falla en ocasiones dentro de FUSION 360® al importar cierto tipo de perfiles. Por ello se ha desarrollado un nuevo módulo, llamado *ImportWingGeometryLines* con las siguientes características:

- Proporcionando el archivo *geometry* con extensión *.csv* generado por el programa *Parametric Wing Designer* al módulo *ImportWingGeometryLines* se importan de manera automática todas las secciones que componen el ala sin necesidad de ejecutar múltiples veces el *script*.
- El método empleado para unir los puntos es mediante segmentos rectos, obteniendo una buena resolución y calidad en el trazado de los perfiles sin el riesgo de fallo del programa.

A continuación, en la Figura 5.4 se muestra el diagrama de flujo del funcionamiento del módulo *ImportWingGeometryLines* donde se puede apreciar que la acción del usuario no se encuentra contenida dentro de ningún bucle y por tanto solo se requiere una única vez.

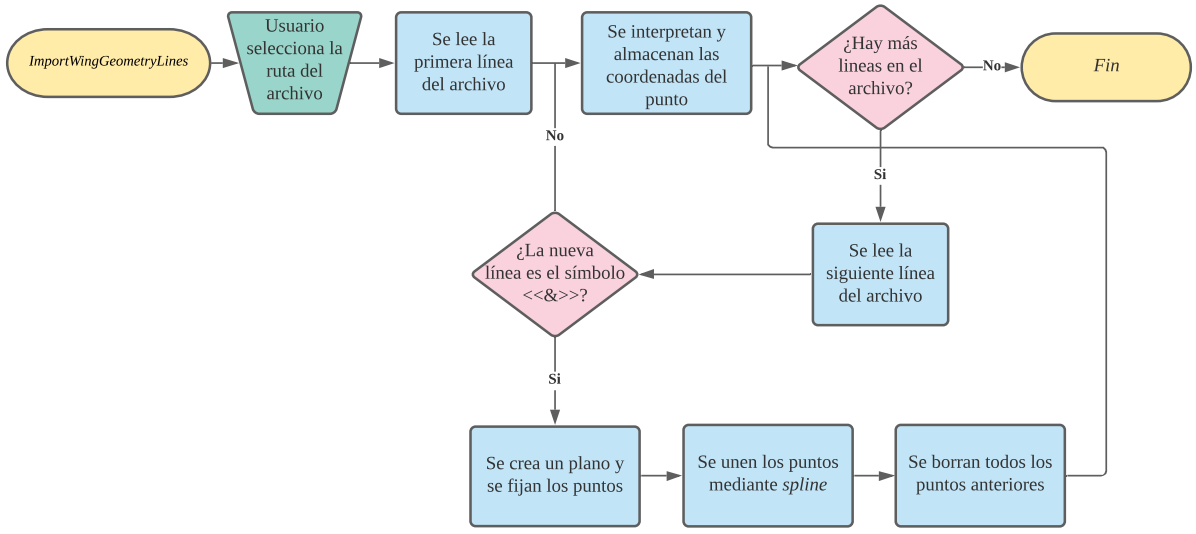


Figura 5.4: Diagrama de flujo del *script ImportWingGeometryLines*

El código de este módulo se ha desarrollado utilizando *Python* y se puede consultar en la Subsección D.2.1 del Apéndice D.

6

Interfaz gráfica

6.1. Introducción

Hasta este punto del proyecto, todo ha sido tratado desde una perspectiva teórica y la implementación paramétrica de las alas ha consistido en un conjunto de funciones difíciles de manipular intuitivamente. Este capítulo trata sobre el desarrollo de la interfaz gráfica y como se ha simplificado el proceso de diseño para hacerlo intuitivo, sencillo y consistente.

6.2. Organización general

6.2.1. Menú principal

Para organizar la información de manera sencilla, el proceso de diseño de un ala se ha dividido en dos etapas principales, por un lado, el diseño de la geometría y por otro, la definición del perfil. Estas dos fases se separan en dos pestañas independientes mediante las que se puede acceder haciendo uso del menú situado en el lado izquierdo del programa.

En la Figura 6.1 se muestra una captura de pantalla de la interfaz gráfica donde se ha enmarcado el menú dentro de un contorno rojo. El menú consiste en la banda gris situada en el lado izquierdo de la imagen. Recorriendo esta franja en sentido descendente se pueden distinguir varias regiones de interés.

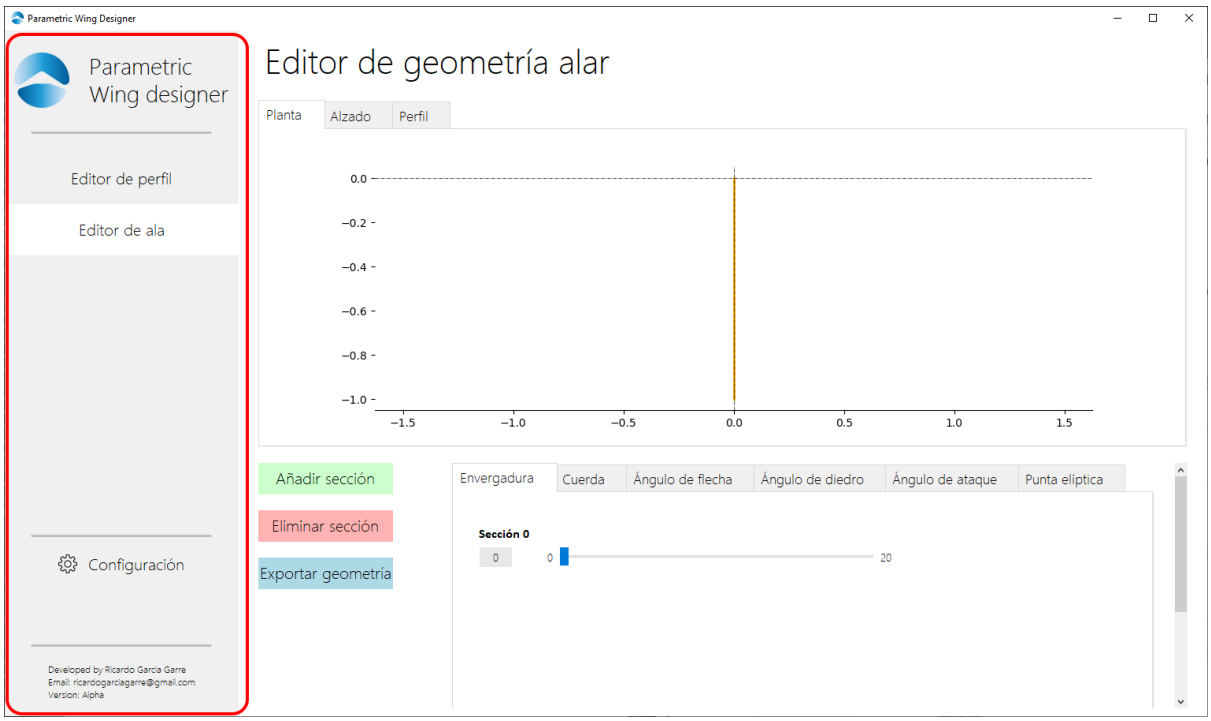


Figura 6.1: Menú principal del programa *Parametric Wing Designer*

En primer lugar, se encuentra el logotipo del programa junto al nombre propuesto para el mismo: *Parametric Wing Designer*. A continuación, se muestran dos botones, pulsando uno u otro se accede a la pestaña del «Editor de ala» o «Editor de perfil» según lo requiera el usuario. Estos dos botones separan el programa en las dos fases de diseño mencionadas. Tras un espacio considerable, situado estratégicamente para permitir la expansión del proyecto en ampliaciones futuras, se encuentra el botón de configuración, tratado más adelante. Finalmente, en el extremo inferior se detalla la información del Autor, contacto y versión en la que se encuentra el programa.

6.2.2. Intercomunicación *software*-usuario

Todos los parámetros configurables se editan o manipulan por medio de los denominados «*sliders*». El *slider* consiste en el elemento principal empleado para introducir información, su función es actuar de intermediario entre usuario y *software*.

Figura 6.2: Arquitectura de un *slider* genérico

En la Figura 6.2 se muestra el aspecto de este elemento. Como se puede observar, va encabezado por un título, indicando el parámetro o la sección afectada. Inmediatamente debajo se encuentra el indicador del valor del parámetro y a su derecha una barra o *slider* que puede ser desplazada horizontalmente para ajustar el valor. Si la barra se desplaza hacia la izquierda, el valor disminuye, mientras que si se desplaza a la derecha aumenta. En los extremos de la barra aparecen dos números que indican los límites inferior y superior que puede alcanzar el valor del parámetro.

Además de desplazando el *slider*, el parámetro puede ajustarse directamente introduciendo el valor deseado dentro del indicador mediante el uso de la parte numérica del teclado.

Los *sliders* son instancias de la clase *slider* cuyo código se exhibe en el Subsección D.1.5 del Apéndice D.

6.3. Editor de perfil

A través de esta pestaña, mostrada en la Figura 6.3, se pueden configurar de forma individual todos los perfiles de las diferentes secciones que componen el ala.

Como se puede observar en la figura, esta pestaña se inicia con un perfil NACA 0012 por defecto perteneciente a la Sección 0, tal y como se especificó en el capítulo anterior. En el gráfico se muestra tanto el contorno del perfil (azul) como la línea de curvatura media (naranja).

6.3.1. Organización del Editor de perfil

Esta pestaña se divide en sentido descendente en tres secciones. En el extremo superior aparece el título de la pestaña, a continuación, un gráfico donde se puede observar la forma del perfil creado o importado y finalmente la sección interactiva que contiene los botones y *sliders*.

Esta última sección se subdivide en tres partes, izquierda, derecha y central.

La parte izquierda está formada por tres botones que permiten importar, usar o

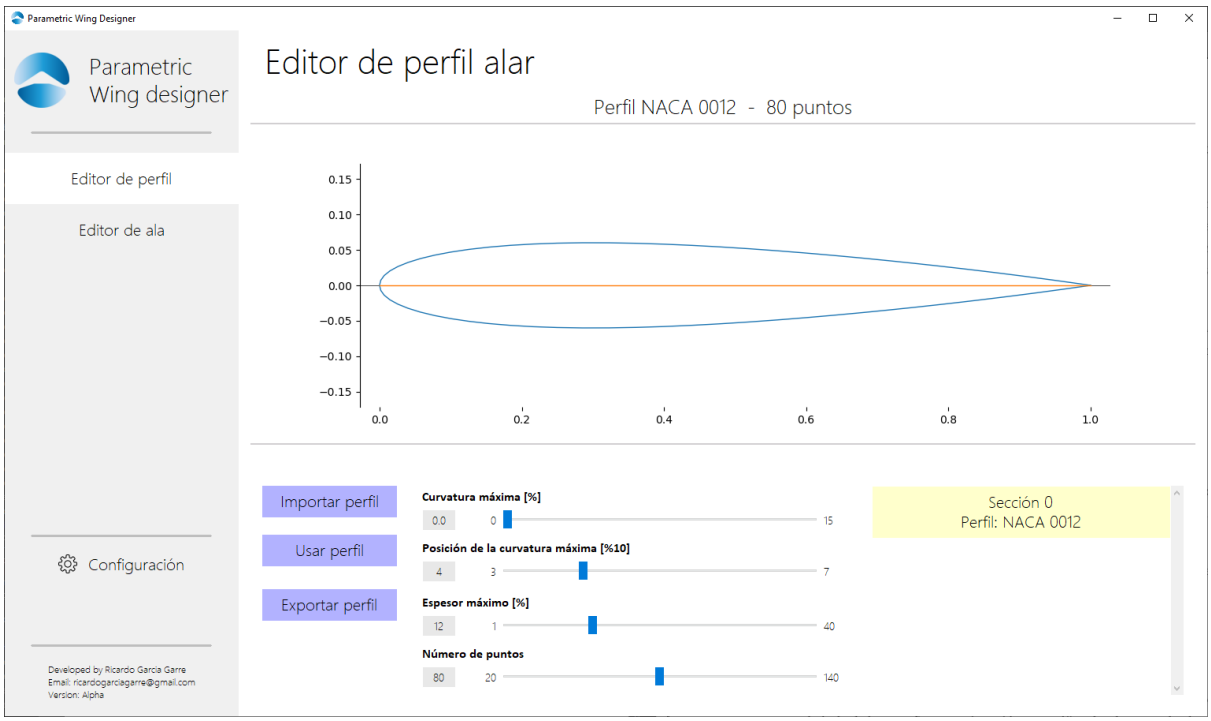


Figura 6.3: Aspecto de la pestaña del Editor de perfil alar al iniciar el programa

exportar un perfil. El botón «Importar perfil» abre el explorador de archivos del sistema operativo para que el usuario pueda seleccionar un documento de extensión *.txt*, *.dat* o *.csv* que contenga el perfil alar que desea utilizar. El botón «Usar perfil» actualiza el perfil de la sección que se encuentre seleccionada en ese momento por el perfil alar mostrado en el gráfico. El botón «Exportar perfil» abre el explorador de archivos para especificar la ruta donde se desea guardar el archivo que contiene los puntos que definen el perfil mostrado en ese momento por la gráfica.

En la parte central de la pestaña se localizan cuatro *sliders* empleados para generar perfiles NACA de cuatro dígitos, cada uno actúa sobre un parámetro distinto. El *slider* superior opera sobre el primer dígito de la nomenclatura NACA, es decir, sobre la curvatura máxima del perfil, el siguiente *slider* manipula la posición de la curvatura máxima (segundo digito NACA), posteriormente, el tercer *slider* varía el espesor (tercer y cuarto dígitos NACA) y finalmente, el último *slider* especifica el número de puntos usados para generar el perfil. La parte derecha de la pestaña contiene una lista con el nombre de todas las secciones que componen la geometría y el perfil alar que usa cada una de ellas. Estas secciones aparecen por defecto en color azulado, sin embargo, pueden ser seleccionadas de una en una para actualizar su perfil. Cuando una sección se selecciona cambia su color a amarillo.

En la Figura 6.4 se observa un ejemplo del uso de esta pestaña. En la imagen se muestra una geometría alar formada por tres secciones, donde la Sección 0 posee un perfil NACA 0012, la Sección 1 un perfil CLARK-Y y la Sección 3 aparece seleccionada y con un perfil NACA 4412, mostrado en el gráfico y cuyos dígitos aparecen en los *sliders*, de donde se puede extraer que el perfil ha sido creado mediante 80 puntos.

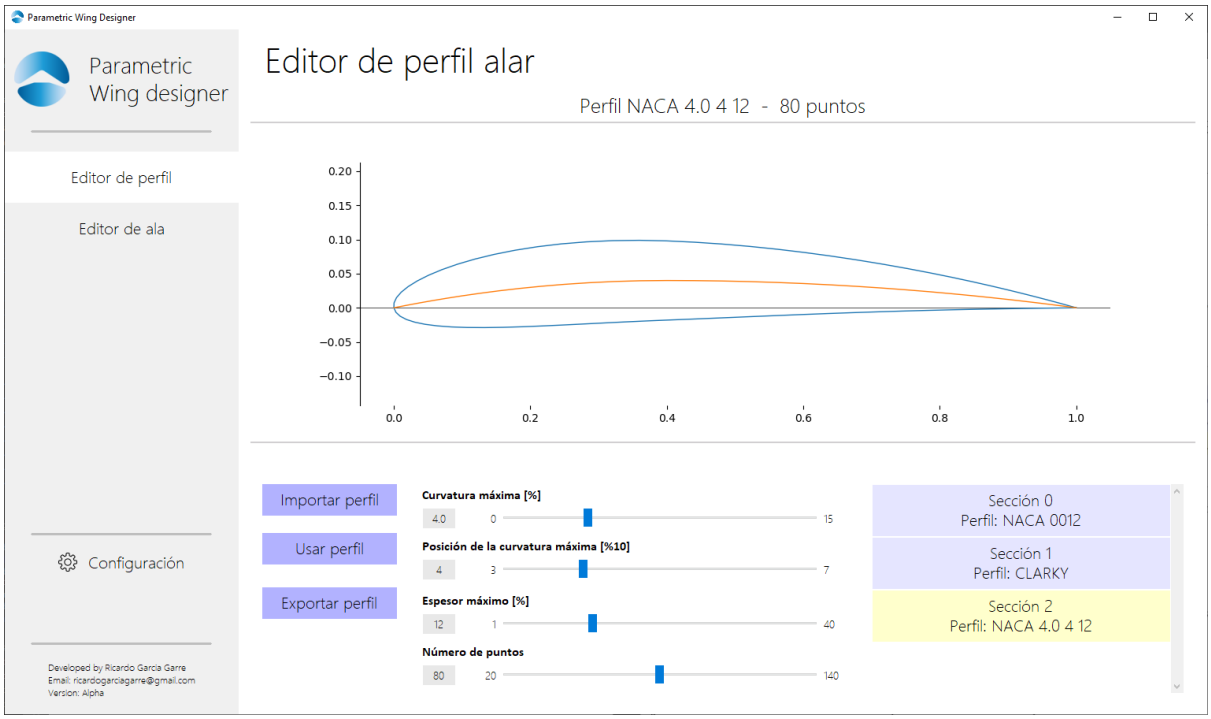


Figura 6.4: Ejemplo de uso de la pestaña del Editor de perfil alar que contiene tres secciones y se muestra la Sección 2 con un perfil NACA 4412

6.3.2. Funcionamiento del Editor de perfil

Esta pestaña es utilizada para implementar perfiles aerodinámicos sobre las diferentes secciones. El proceso se realiza siguiendo los pasos descritos a continuación.

1°. Del listado de secciones de la parte inferior derecha de la pantalla se selecciona la sección sobre la que se desea cambiar el perfil aerodinámico.

2°. Si el perfil a implementar es un NACA de cuatro dígitos, el usuario puede elegir entre crearlo mediante el uso de los *sliders* de la parte central o importarlo vía un archivo externo, en sendos casos el resultado será el mismo. Si el perfil NACA se crea mediante el uso de los *sliders* debido a que no se dispone del archivo de puntos,

este puede exportarse para disponer de él en proyectos futuros o para analizar el perfil en otros programas. En caso de que el perfil a emplear sea distinto de la nomenclatura NACA de cuatro dígitos, se debe necesariamente importar a través de un archivo externo.

3°. Finalmente, una vez se encuentre representado en el gráfico el perfil deseado, se acciona el botón «Usar perfil» para actualizar el perfil de la sección seleccionada al inicio. Este proceso se realiza cada vez que se desee cambiar el perfil de una sección distinta.

En la Figura 6.5 se muestra un diagrama de flujo sobre las operaciones que realiza el programa y el usuario dentro de esta pestaña.

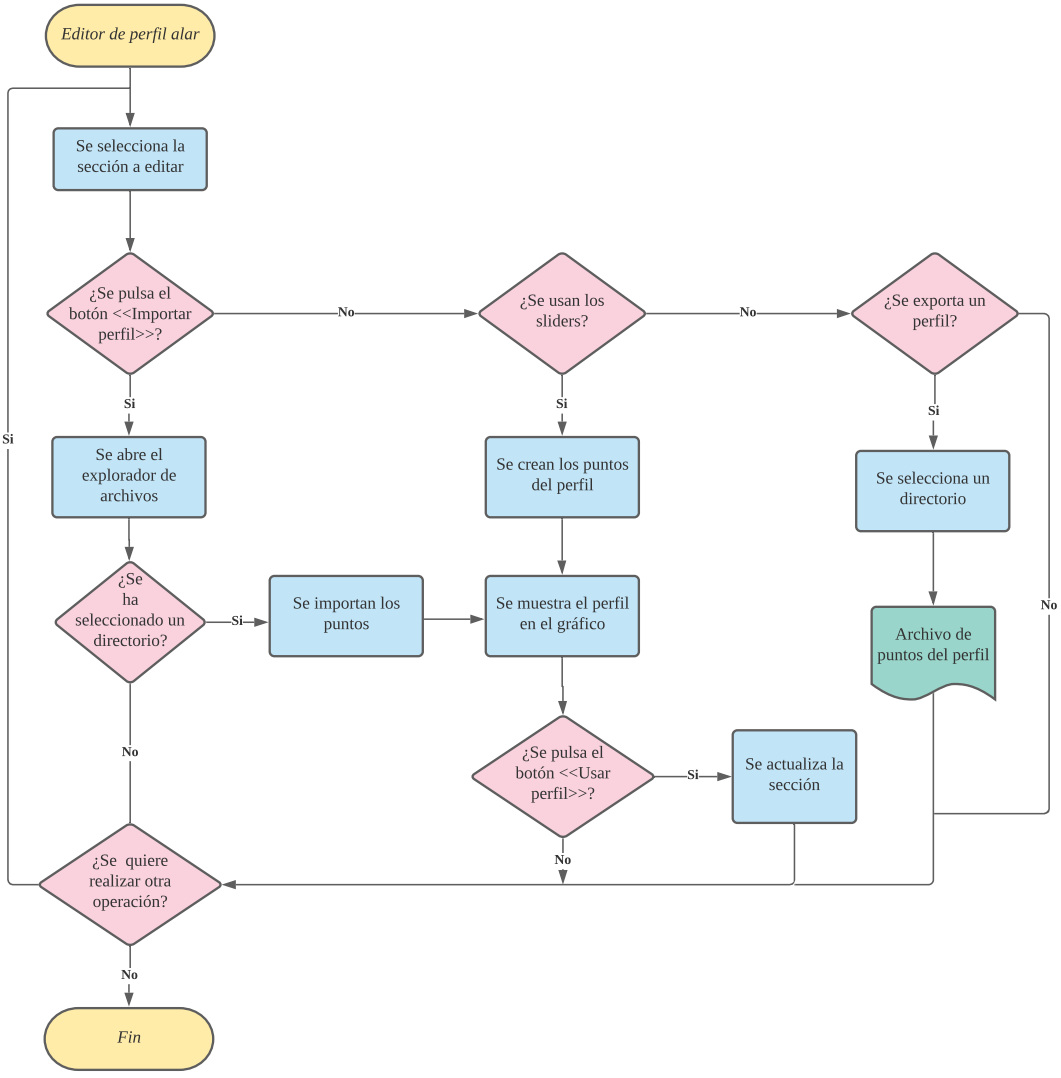


Figura 6.5: Diagrama de flujo de la pestaña del Editor de perfil alar

6.4. Editor de geometría

En esta pestaña se editan todos los parámetros correspondientes a la geometría del ala. En la Figura 6.6 se muestra el aspecto de la página por defecto al iniciar el programa.

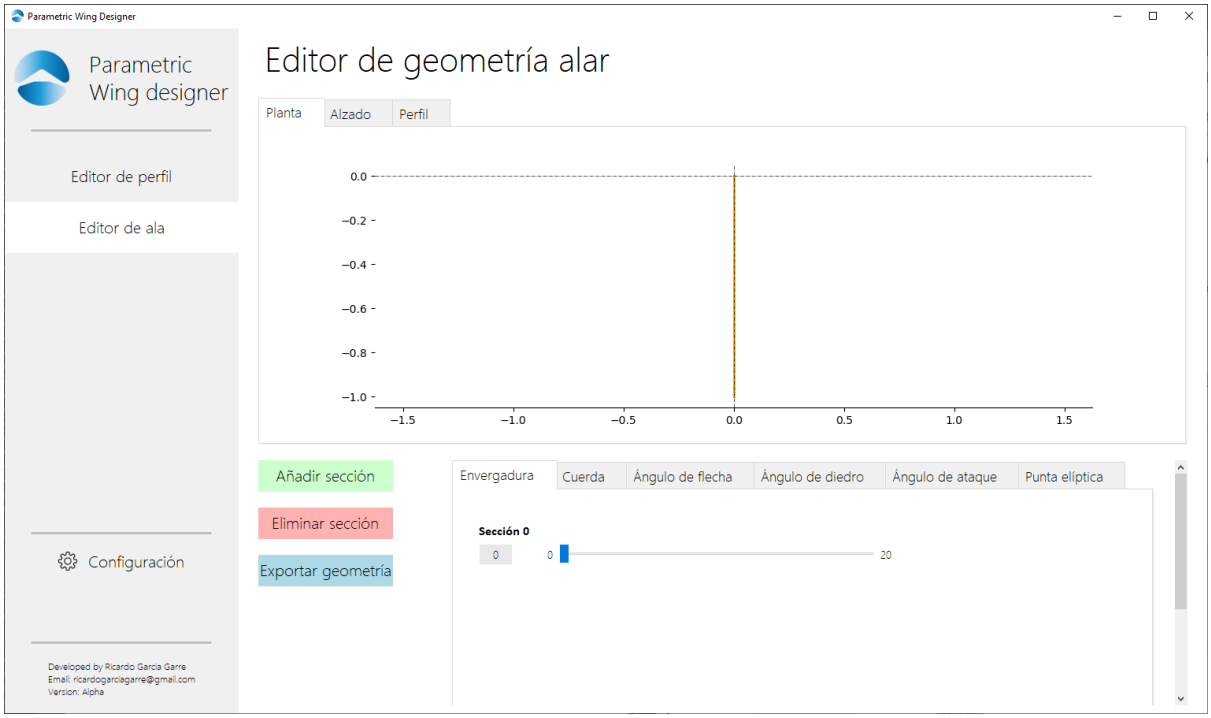


Figura 6.6: Aspecto de la pestaña del Editor de geometría alar al iniciar el programa

6.4.1. Organización del Editor de geometría

Al igual que el «Editor de perfil», la pestaña «Editor de geometría» se divide verticalmente en tres secciones. En la parte superior se aparece el título, a continuación, se muestra un cuaderno que presenta los gráficos de las tres vistas del ala: planta, alzado o frente y perfil. En la parte inferior se sitúan los botones y *sliders* que actualizan los parámetros.

Botones laterales

En el lateral izquierdo existen tres botones organizados verticalmente y distinguidos entre sí por tres colores distintos.

El botón «Añadir sección» crea una nueva sección en la geometría y añade en el cuaderno de parámetros un nuevo *slider* para cada parámetro con el nombre de la sección. Cuando una sección es creada, lo hace con una envergadura de sector nula, por lo que para poder visualizar esta nueva sección en el gráfico es necesario dar un valor positivo al parámetro de envergadura.

El botón «Eliminar sección» elimina la última sección creada, borra los *sliders* correspondientes a esta y actualiza los gráficos con las secciones restantes.

El botón «Exportar geometría» abre el explorador de archivos para seleccionar la ruta donde se desee generar la carpeta que contendrá los archivos de las secciones.

Cuaderno de gráficos y parámetros

El cuaderno de gráficos consiste en una estructura formada por un menú superior y una ventana inferior. El menú permite seleccionar la vista que se desea estudiar y la ventana muestra la vista correspondiente de la geometría.

Por otro lado, el cuaderno de parámetros contiene en el menú el nombre de los parámetros editables para diseñar el ala y la ventana inferior muestra los *sliders* con el valor del parámetro de cada sección.

Para facilitar la comprensión de la información que se facilita en cada cuaderno, se ha reproducido dentro del programa el ala de una aeronave McDonnel Douglas DC-8, donde se ha supuesto un perfil NACA 4419, un perfil grueso que permite exagerar la vista de frente para poder observar mejor algunas características del programa.

En la Figura 6.7 se muestra la ventana «Editor de geometría» con el ala creada. En esta ocasión aparece seleccionada la vista en planta y el parámetro de la envergadura. En el cuaderno de parámetros se aprecia que el ala ha sido reproducida mediante el uso de tres secciones. Además, se ve como estas secciones se separan entre sí una distancia de sector de 0.6, 1.2 y 5.6 respectivamente. En cuanto al gráfico, los distintos segmentos que forman la vista en planta del ala se diferencian entre sí por colores. Las líneas anaranjadas constituyen las secciones que componen la geometría, la línea azul oscuro hace referencia al borde de ataque del ala y por consiguiente la línea verde forma el borde de fuga. Existe un cuarto color, azul claro, que indica la línea de cuarto de cuerda del ala, en ocasiones de especial relevancia para medir el ángulo de flecha al cuarto de cuerda o para ajustar

la posición de punta de ala de las secciones elípticas.

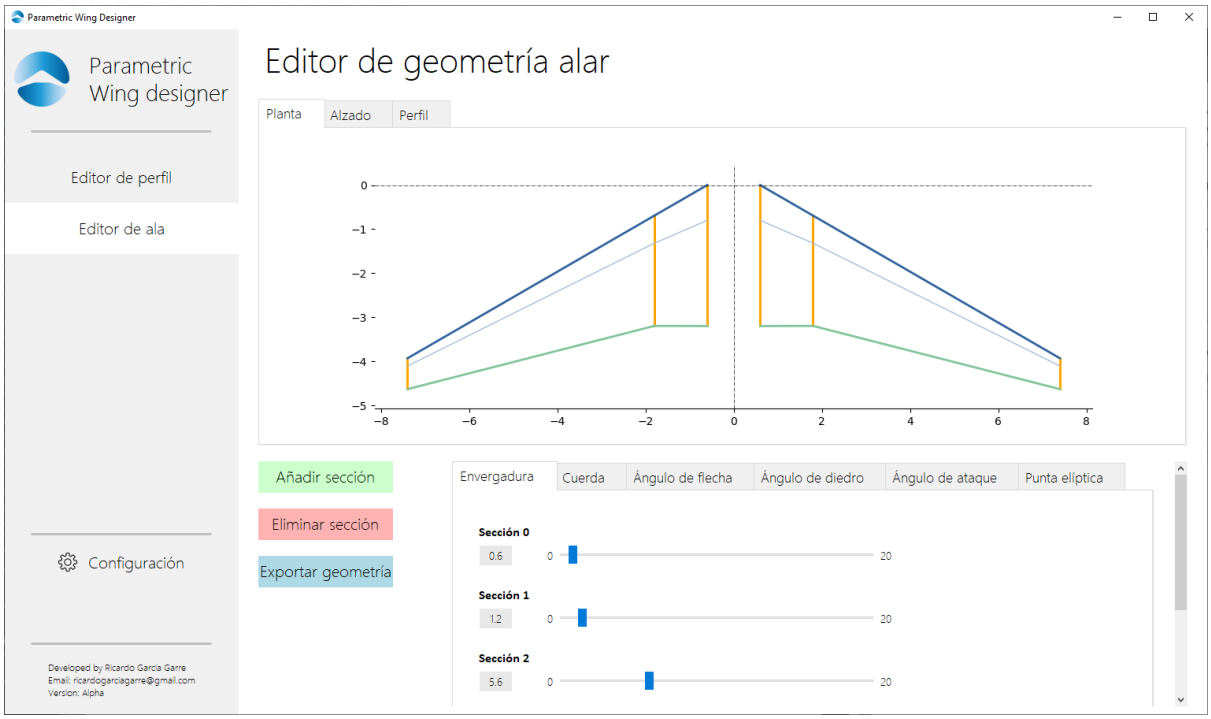


Figura 6.7: Aspecto de la pestaña del Editor de geometría alar en la que se recrea la vista en planta del ala de una aeronave McDonnel Douglas DC-8 Super 70 Series

A continuación, en la Figura 6.8, se muestra la ventana del programa con vista de alzado y el ángulo de diedro seleccionados. La vista de frente es la única que permite la visualización de este ángulo geométrico del ala. Como se puede observar en el cuaderno de parámetros, dentro del ángulo de diedro solo aparecen dos *sliders* a pesar de tener tres secciones. Esto se debe a que, tal como se ha comentado en capítulos anteriores, un ala formada por n secciones poseerá $n-1$ sectores, y, puesto que los ángulos de flecha y diedro manipulan un sector completo, sólo se necesitan dos *sliders* para definir el diedro y flecha de un ala formada por tres secciones. El ala del DC-8 posee un ángulo de diedro de 7°.

En cuanto a la gráfica de la vista de frente, las secciones vuelven a tener el color anaranjado descrito en la vista anterior. Además, al igual que en la Figura 6.7 el borde de ataque y de fuga se trazan de colores azul oscuro y verde respectivamente. Nótese como el borde de fuga aparece representado mediante una línea discontinua, indicando que dicha línea se encuentra oculta por el resto del ala. Las líneas superiores e inferiores que definen el extradós e intradós del ala se representan igualmente de color anaranjado ya que su unión vertical representa infinitas secciones resultantes de la interpolación lineal

de las secciones definidas por el usuario.

Figura 6.8: Aspecto de la pestaña del Editor de geometría alar en la que se recrea la vista de alzado del ala de una aeronave McDonnel Douglas DC-8 Super 70 Series

Por último, en la Figura 6.9, se muestra la ventana del programa donde se ha seleccionado la vista de perfil y el parámetro del ángulo de ataque o rotación de la sección. La influencia del parámetro del ángulo de ataque se aprecia mejor en la vista de perfil, aunque también afecta a la vista en planta y al alzado. La vista en planta de una sección es la proyección en el eje horizontal de la cuerda de la sección, por lo que un aumento en el ángulo de ataque acorta la longitud de esta proyección. En el alzado, dotar a una sección de ángulo de ataque provoca que dejen de coincidir la línea de borde de ataque con la de borde de fuga.

El gráfico del perfil muestra la proyección en un plano vertical de las posiciones relativas entre las secciones que componen el ala. Como se puede apreciar, únicamente aparecen los contornos de color azul de los perfiles dimensionados. El código de color empleado para representar estas secciones es perfilar con un color azul oscuro el perfil de la sección más cercana al encastre del ala y degradar este color progresivamente a medida que la sección se acerca más a la punta del ala.

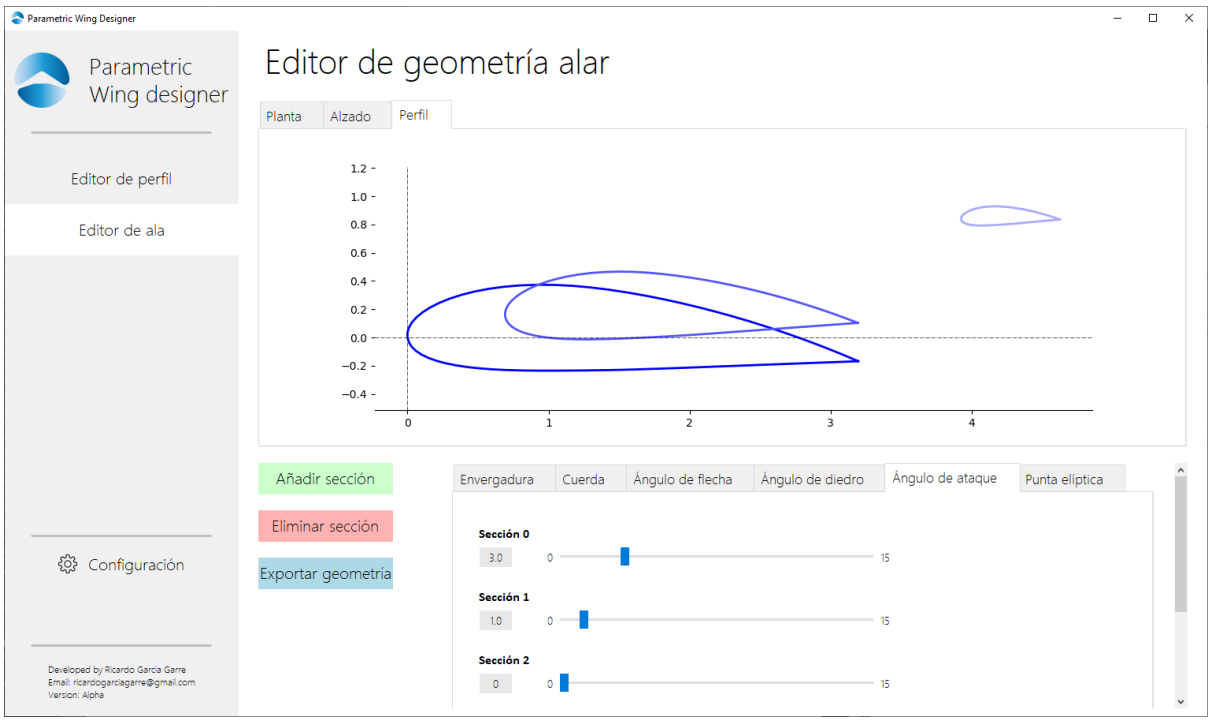


Figura 6.9: Aspecto de la pestaña del Editor de geometría alar en la que se recrea la vista de perfil del ala de una aeronave McDonnel Douglas DC-8 Super 70 Series

6.4.2. Funcionamiento del Editor de geometría

En esta pestaña no existe un procedimiento específico sobre como se deben realizar las operaciones, ya que depende principalmente de la preferencia del usuario. El orden propuesto por el Autor es el siguiente, pero puede variar según las circunstancias.

1°. Se determina el número de secciones que componen el ala y se crean mediante el botón «Añadir sección». En caso de haber creado alguna sección innecesaria por error, siempre se pueden eliminar mediante el botón «Eliminar sección».

2°. Una vez definidas todas las secciones, se avanza a través del cuaderno de parámetros, editando progresivamente cada parámetro de las distintas secciones que componen el ala. Los tres primeros parámetros («Envergadura», «Cuerda» y «Ángulo de flecha») se recomienda que sean editados visualizando la vista en planta de la geometría. A continuación, se cambia la vista al alzado para tener una mejor perspectiva al variar el parámetro de «Ángulo de diedro». Seguido, se avanza al parámetro «Ángulo de ataque» para rotar las secciones, esta operación se debería realizar mediante la vista de perfil. Finalmente, se añade una punta de ala elíptica

en caso de que sea necesario para la geometría y se configuran los parámetros de la misma.

3°. El último paso es exportar las secciones mediante el uso del botón «Exportar geometría». Se abre el explorador de archivos y el usuario selecciona la ruta donde desea guardar el ala.

En la Figura 6.10 se muestra un diagrama de flujo sobre el proceso propuesto para el diseño de una geometría alar cualquiera.

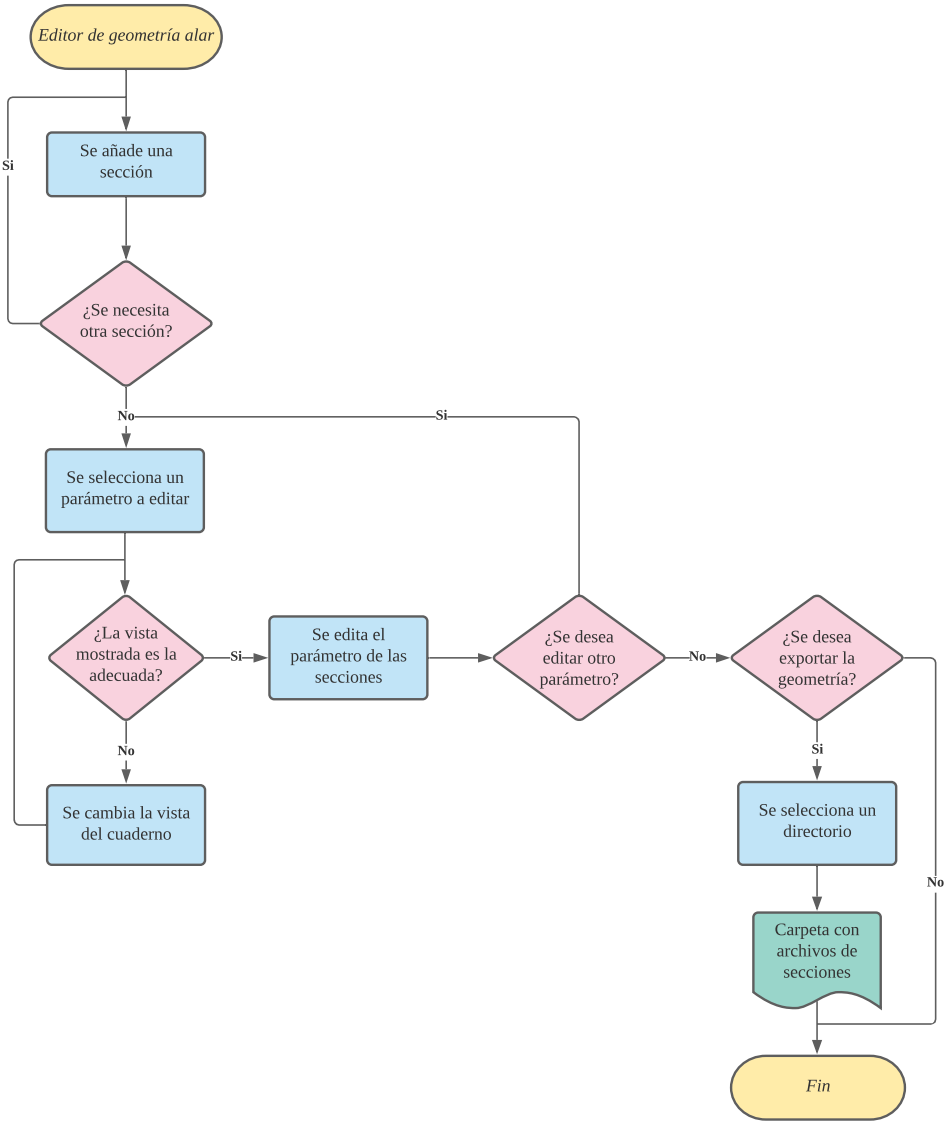


Figura 6.10: Diagrama de flujo para la creación de un ala en la pestaña del Editor de geometría alar

7

Aplicación práctica: Diseño y fabricado de un semiala

7.1. Introducción

Para demostrar uno de los usos del programa y su sinergia con la fabricación aditiva, se ha diseñado y fabricado un semiala mediante moldeado por deposición fundida. A continuación se expone el proceso de diseño y fabricado.

7.2. Propuesta de diseño

El ala diseñada consta de dos sectores principales: un primer sector recto y un segundo semielíptico con borde de fuga recto. Además, el sector semielíptico se ha definido con una resolución de ocho secciones. En la Figura 7.1 se muestra la vista en planta del ala descrita.

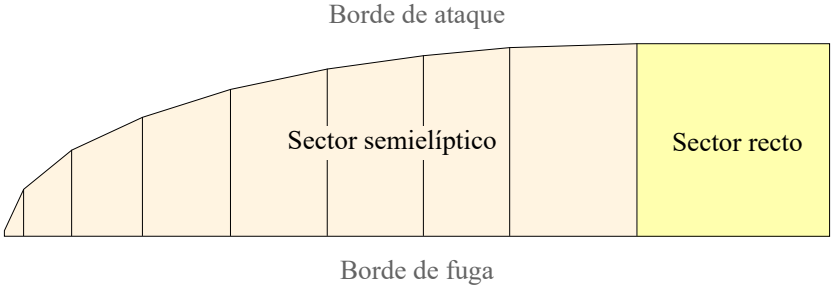


Figura 7.1: Vista en planta del ala diseñada

El proceso de diseño completo del ala dentro de *Parametric Wing Designer* se muestra en la Subsección A.2.2. En cuanto a aspectos relevantes, mencionar que el ala se ha discretizado mediante dos secciones que definen el sector recto y una geometría semielíptica para el otro sector. Se trata de un ala sencillo sin torsión ni ángulo de flecha o diedro, por lo que la vista de perfil o alzado resulta trivial. El ala posee un perfil alar NACA 4412.

Una vez modelado el ala, se exporta. La carpeta creada, mostrada en la Figura 7.2, contiene 11 archivos de extensión *.csv*, esencial para poder exportar la geometría a FUSION 360®. Se puede apreciar como los archivos de las secciones son nombrados mediante la nomenclatura que identifica a cada una de ellas, describiendo su posición en la geometría. El archivo *geometry.csv* contiene todas las secciones unificadas y se usará más adelante para importar las secciones de forma automática en cualquier programa CAD con un módulo compatible.

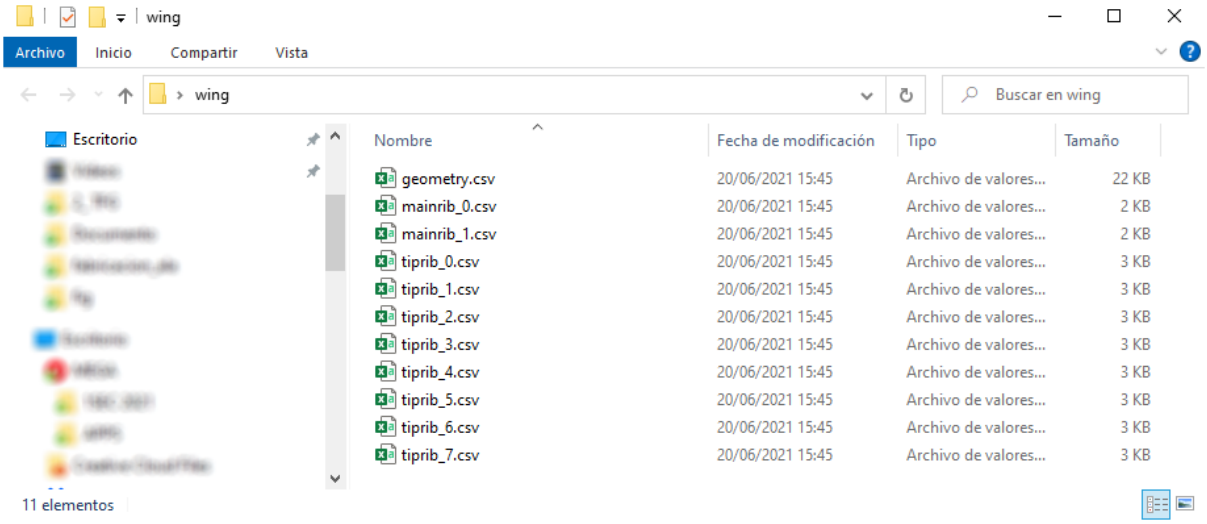


Figura 7.2: Archivos contenidos dentro de la carpeta *wing* creada al exportar el ala

A continuación, se abre FUSION 360® y se ejecuta el módulo *ImportWingGeometryLines*. El resultado de esta operación, mostrado en la Figura 7.3 es un conjunto de planos o *sketches* donde cada uno contiene una sección.

Finalmente, la geometría se modela usando la herramienta *loft* para generar las superficies que unen las secciones (Figura 7.4). Se guarda la pieza en formato *.stl* y se exporta a un programa de laminado para impresión 3D.

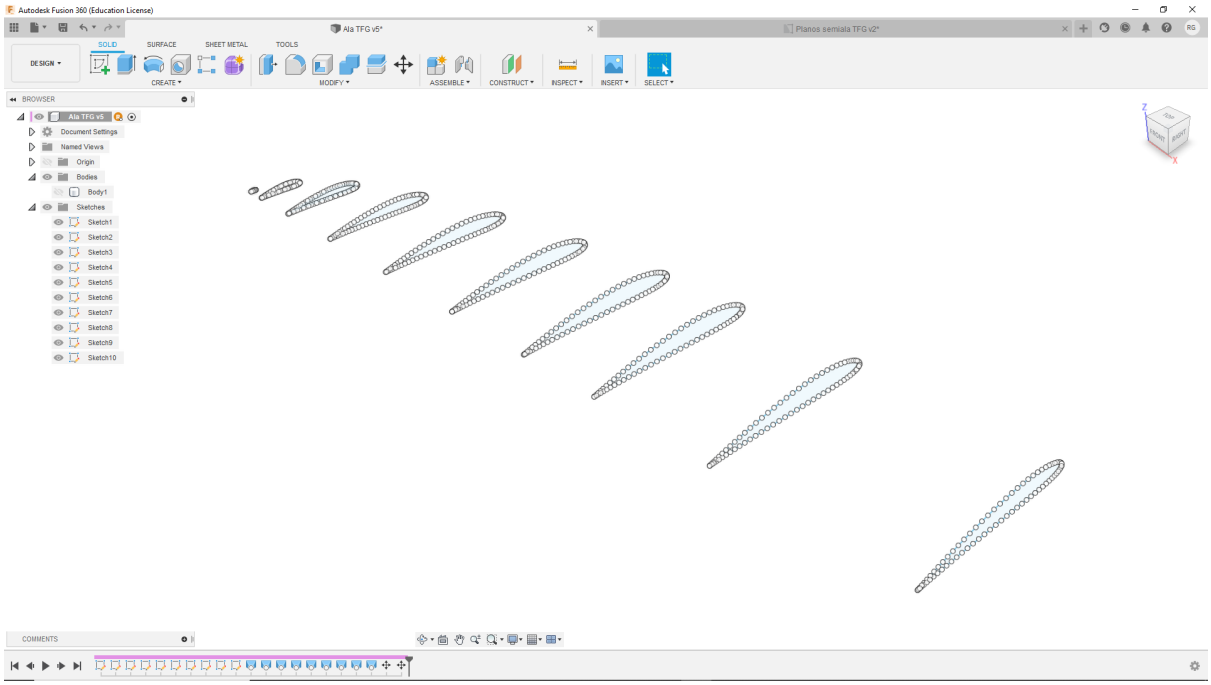


Figura 7.3: Secciones del ala diseñada introducidas en FUSION 360® mediante el módulo *ImportWingGeometryLines*

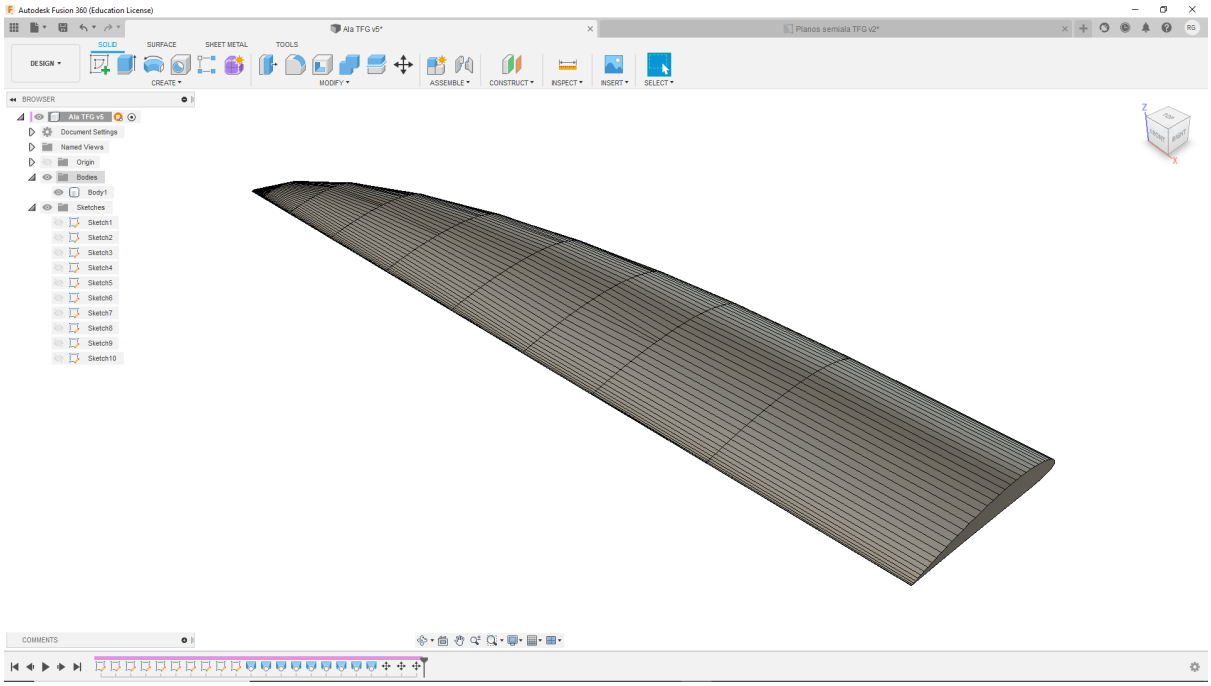


Figura 7.4: Modelo sólido del ala diseñada tras unir las secciones mediante la herramienta *loft*

7.3. Proceso, equipo y material de fabricación

El proceso de fabricación aditiva seleccionado para fabricar la pieza es FDM. La ventaja principal de FDM frente a otros procesos de interés como LENS es su bajo coste tanto en equipo como material.

Tal y como se expuso en la Subsección 2.4.2, en el proceso FDM, las capas que componen una pieza se fabrican mediante la deposición de finos hilos de material termoplástico fundido que son depositados siguiendo un recorrido determinado.

Para realizar este proceso se ha seleccionado la impresora *Blackbelt 3D*, mostrada en la Figura 7.5, disponible en el Departamento de Ingeniería Mecánica y de Materiales (DIMM) de la Universitat Politècnica de València. Las especificaciones técnicas de la impresora se muestran en la Sección A.4.

Figura 7.5: Impresora *Blackbelt 3D* [36]

Esta máquina se diferencia del resto por posibilitar una impresión infinita en uno de sus tres ejes, debido al uso de una cinta móvil rotativa que desplaza la pieza a medida que se crean las capas. Esto permite la impresión de piezas alargadas, como el ala de un prototipo de aeronave, sin necesidad de ser fraccionadas, como ocurre con las impresoras 3D tradicionales.

El material empleado para la fabricación es PLA. El PLA o ácido poliláctico es un poliéster biodegradable obtenido a partir de recursos vegetales como el almidón de maíz. Es frecuentemente empleado para la impresión 3D en los ámbitos de medicina e industria debido a sus propiedades y facilidad de uso. Se puede presentar como PLA puro o como material compuesto, reforzado con aditivos como metales o fibras que mejoran sus propiedades mecánicas y físicas [9]. El material se presenta en forma de filamento de 1.75 mm o 2.85 mm de diámetro. Las propiedades térmicas y mecánicas de este material pueden consultarse en la Sección A.6.

La elección del PLA frente a otros materiales termoplásticos admitidos por la impresora reside en su bajo coste y alta disponibilidad. Puesto que la pieza no será ensayada mecánicamente ni se realizará un estudio de peso, el PLA es la opción más económica para fabricar el ala con fines puramente demostrativos. Además, el PLA no requiere de unas condiciones de impresión exigentes, donde se deban evitar factores externos como corrientes de aire o altos gradientes de temperatura.

Una vez que se dispone del archivo *.stl* de la pieza, se importa en el programa Ultimaker Cura para laminarla y generar el *G-Code*[1] legible por la impresora. A continuación, se definen los parámetros de impresión, los cuales pueden consultarse en la Tabla A.5 del Apéndice A. Entre estos parámetros destaca el uso de una única capa de material para definir la pared externa e interna de la pieza, lo que reduce considerablemente el peso.

Previo a la impresión del semiala diseñado, se comprobó el funcionamiento de la impresora con una pieza de prueba. En la Figura 7.6 se observa el proceso de fabricación de la misma empleando la impresora *Blackbelt 3D*.

El resultado obtenido mediante esta máquina es una pieza donde las uniones entre fibras no se realizan de manera correcta. Este hecho provoca que el elemento fuera especialmente quebradizo y con un acabado de la superficie de muy baja calidad, al no tener unidos los filamentos que conforman las capas exteriores. Este defecto no se debe a la impresora en sí, sino al programa empleado para generar el *G-Code*, ya que no está optimizado para la fabricación de piezas con un ángulo de 45°. En la Figura 7.7 se puede apreciar de forma más detallada la superficie del semiala impreso.

[1] El *G-Code* es un lenguaje de programación usado en control numérico.

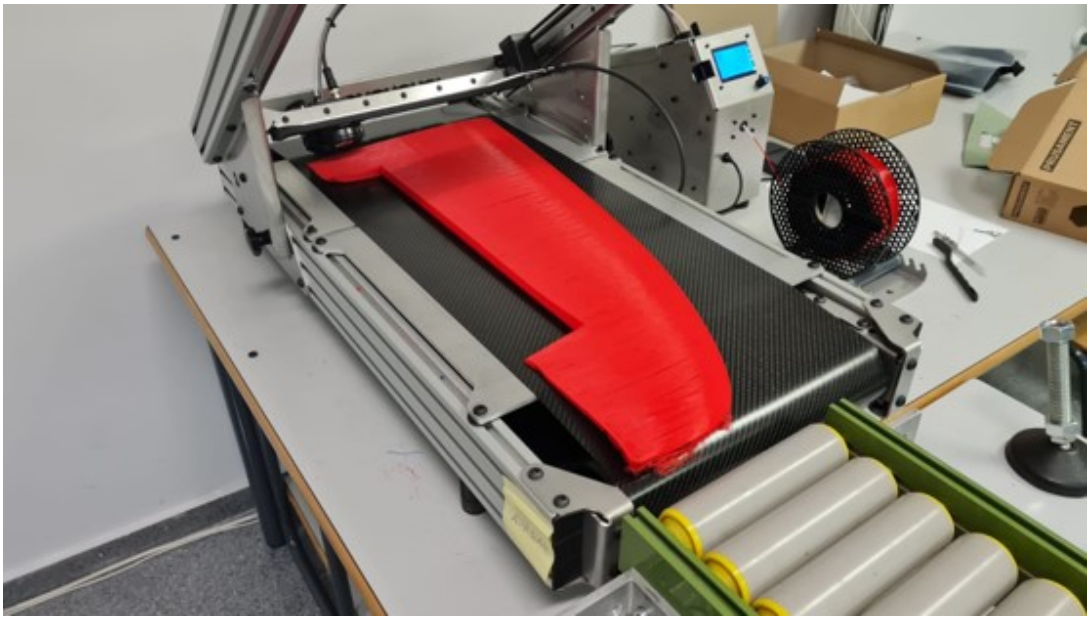


Figura 7.6: Prueba de impresión de un semiala mediante la impresora *Blackbelt 3D*

Figura 7.7: Enfoque de la punta del ala donde se aprecia el defecto detectado sobre la superficie de la pieza

Con el fin de obtener mejores resultados, se hizo uso de la impresora *Ultimaker 3 Extended*. Esta máquina, mostrada en la Figura 7.8, es considerada como una impresora 3D tradicional, al no poseer unas características en el volumen de impresión que la distinga significativamente del resto.

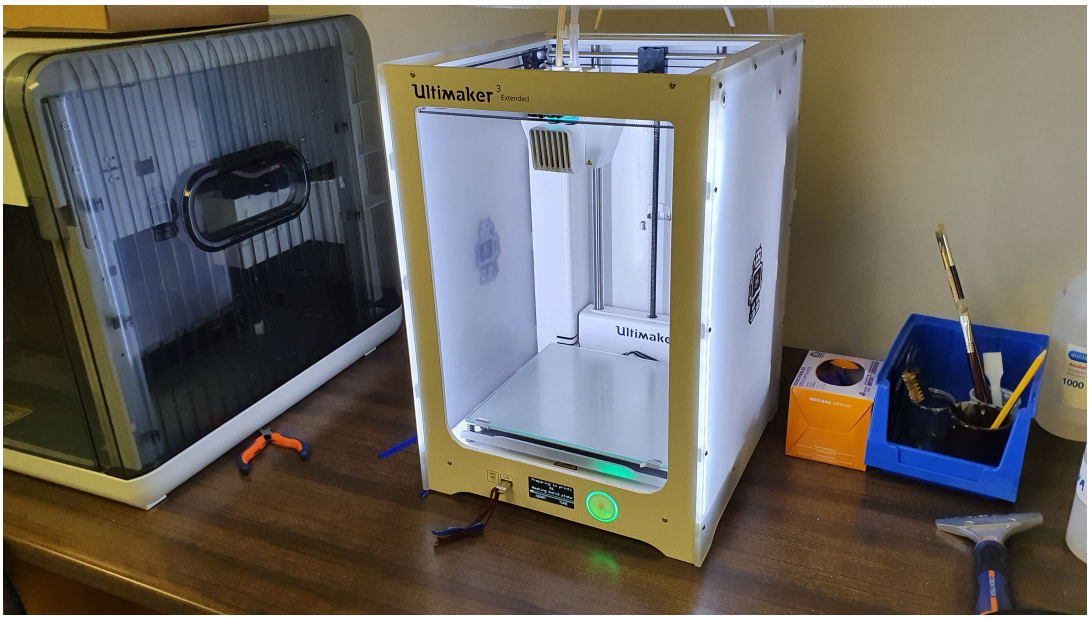


Figura 7.8: Impresora *Ultimaker 3 Extended*

Para imprimir el semiala mediante la *Ultimaker 3 Extended*, se fraccionó en tres secciones para poder ser impresa en una misma sesión. Además, se redujo el tamaño de la pieza en un 50 %, con el fin de que la cuerda en el encastre pudiera ser contenida dentro de los límites de la diagonal de la cama caliente.

Los parámetros de impresión en esta ocasión se mantienen idénticos al caso anterior. En cuanto a la orientación de la pieza, se encuentra en posición vertical, donde el contacto con la cama caliente lo realiza la sección transversal del semiala. En la Figura 7.9 se observa la primera capa apoyada sobre la cama caliente donde se aprecian las tres secciones en las que ha sido dividido el semiala.

El resultado (ver Figura 7.10), es una pieza de alta calidad, ligera y fraccionada en tres segmentos que son unidos posteriormente. El semiala pesa 230 g con lo que el ala completa, de 1.5 m de envergadura, pesaría 460 g. Además, el tiempo de impresión ha resultado de 1 dia y 21 minutos.

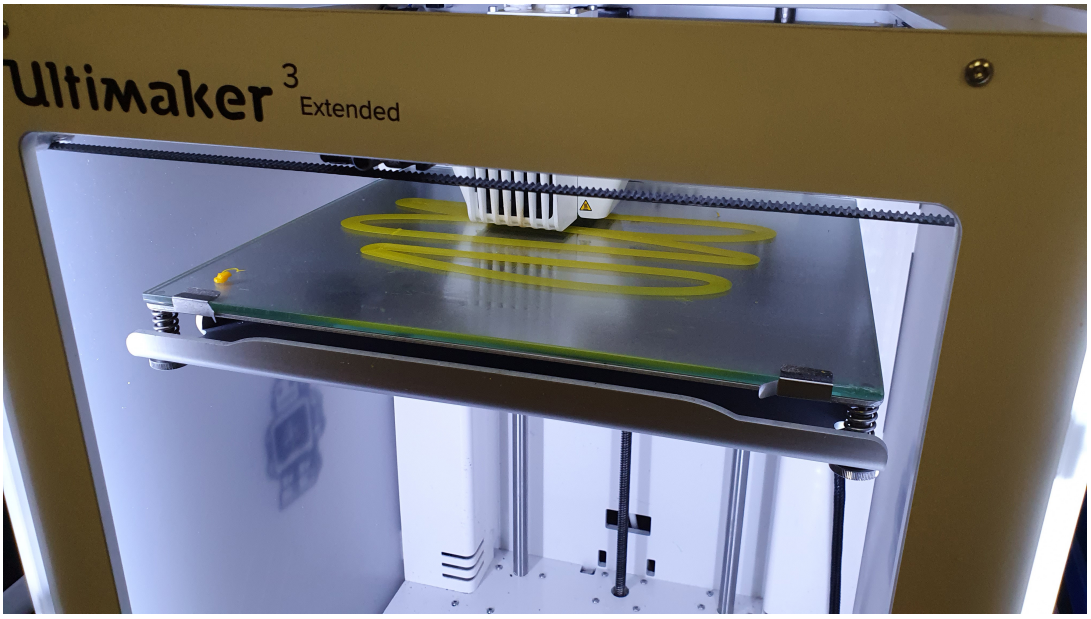


Figura 7.9: Proceso de impresión del semiala diseñada, seccionada en tres partes, siendo impresa mediante la máquina *Ultimaker 3 Extended*

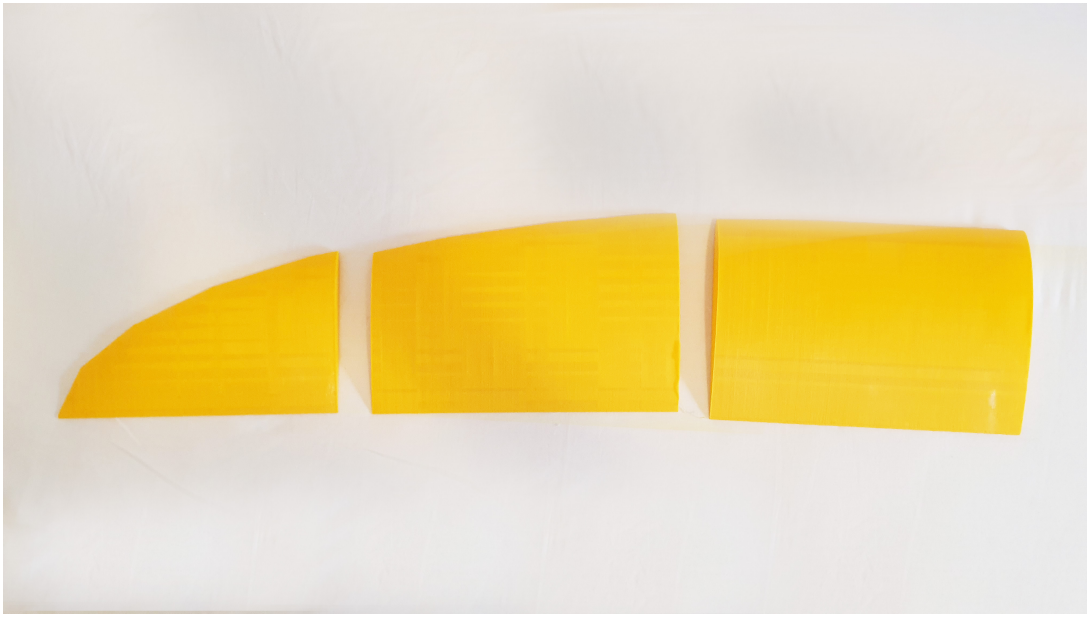


Figura 7.10: Resultado del proceso de impresión del semiala diseñada, seccionada en tres partes

Una vez unidas todas las secciones, se da por finalizada la fabricación de la pieza. Tal y como se observa en la Figura 7.11, mediante fabricación aditiva, se consigue fabricar un semiala donde la región semielíptica destaca como una característica de especial interés.

Como se expuso en la Subsección 3.2.3, uno de los motivos por lo que cesó la fabricación de alas elípticas es su elevado coste, ya que, tanto el proceso de diseño como de manufactura eran especialmente complejos. Mediante el uso de *Parametric Wing Designer* como herramienta principal de diseño, en combinación con las técnicas de fabricación aditiva, el ala mostrada en la Figura 7.11 ha sido diseñada y fabricada en menos de 25 horas.

Esto abre un amplio abanico de posibilidades en el que es posible fabricar ,de manera rápida, multitud de geometrías distintas con múltiples fines experimentales.

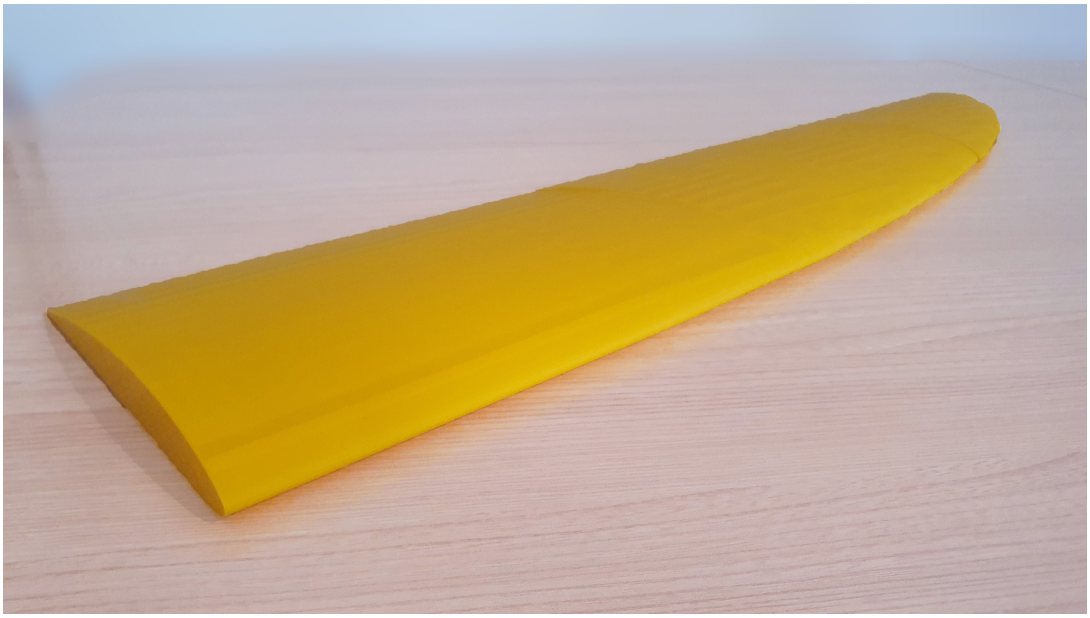


Figura 7.11: Pieza final diseñada mediante *Parametric Wing Designer* y fabricada mediante FDM

Es evidente que, con el nivel tecnológico actual, la impresión 3D de alas está limitada a aeronaves recreativas o a pequeños modelos a escala. Sin embargo, estas técnicas ya están siendo utilizadas actualmente en la fabricación de toberas de motor, álabes de turbinas e incluso lanzaderas espaciales. Por todo ello, y a juicio del Autor, tan solo es cuestión de tiempo que la fabricación aditiva cobre especial importancia en el sector aeronáutico en la manufactura de componentes de aeronaves.

8

Conclusión y pasos futuros

8.1. Resumen del proyecto y conclusiones

En este trabajo se ha presentado el proceso de desarrollo del programa *Parametric Wing Designer*, un programa que permite el diseño paramétrico de geometrías alares a través de una interfaz gráfica de uso sencillo. El objetivo principal con el que se inició el proyecto fue reducir considerablemente el tiempo de diseño y modelado de estas superficies.

Para ello, se ha realizado un proceso de análisis de las distintas geometrías alares existentes y los principales parámetros geométricos que las definen. Posteriormente, se han implementado los parámetros más relevantes mediante el uso de expresiones trigonométricas y algebraicas básicas y otras ecuaciones contrastadas de otros autores, como las utilizadas en la definición geométrica de los perfiles NACA. Además, se implementa un sistema de discretización de alas en secciones que facilita el proceso de diseño y la automatización del trazado de superficies elípticas.

A través de la interfaz gráfica, el proceso de diseño de superficies alares se ha divido en dos partes. Por un lado, el trazado de los perfiles aerodinámicos y por otro, el modelado de la propia geometría del ala. La variación de cualquier parámetro puede visualizarse en tiempo real a través de los gráficos incorporados, facilitando en todo momento al usuario el aspecto final del ala.

Parametric Wing Designer incorpora un menú de configuración que permite modificar el idioma entre castellano e inglés, ampliando exponencialmente el rango de usuarios que

pueden hacer uso de la aplicación. De igual manera, dentro del menú de configuración, el usuario puede cambiar el formato de los archivos exportados para que cualquier otra aplicación pueda hacer uso de ellos, esto incluye aplicaciones CAD, o cualquier otra del tipo hoja de cálculo.

El resultado de este proyecto es una aplicación versátil, ligera, que no requiere instalación y de uso sencillo de la que se puede extraer las siguientes conclusiones.

- El tiempo de diseño y modelado de un ala dentro de un programa CAD se ha conseguido reducir en dos órdenes de magnitud. El proceso de modelado de un ala directamente dentro de un programa CAD suele conllevar un rango de entre 2-3 horas (120 - 180 minutos) en geometrías complejas, dónde no solo se ha de crear la geometría manualmente sino que además se deben calcular previamente las posiciones y ángulos de rotación de cada sección para ser posteriormente colocadas de forma correcta. Mediante el uso de *Parametric Wing Designer* este proceso se realiza en un tiempo no superior a los 10 minutos independientemente de la complejidad de la geometría, ya que el programa se encarga de realizar los cálculos a partir de una serie de parámetros sencillos de determinar y exporta archivos donde la posición y rotación de las secciones se incorpora dentro de las coordenadas de los puntos que definen las mismas.

- *Parametric Wing Designer* ha sido desarrollado íntegramente mediante el lenguaje de programación *Python*. Este lenguaje ha permitido un tiempo de desarrollo del programa muy corto, inferior al mes y medio. *Python* cuenta con múltiples módulos que permiten realizar interfaces gráficas, crear gráficos o manejar datos de forma sencilla y con un amplio soporte en la red. La ventaja principal de este lenguaje frente a otros como MATLAB®, muy empleado por ingenieros, es que *Python* es gratuito y permite crear programas que pueden ser ejecutados directamente por el sistema operativo sin necesidad de soporte de otro programa. Sin embargo, *Python* posee un inconveniente principal, y es que, al ser un lenguaje interpretado es, por construcción, más lento que su homólogo compilado, por lo que el mismo programa desarrollando mediante otro lenguaje sería más rápido realizando las operaciones.

- El proceso de diseño de un ala se ha conseguido dividir en dos partes. Por un lado, la selección del perfil aerodinámico y por otro el modelado de la geometría. Esto se ha incorporado dentro del programa dividiendo los procesos en dos pestañas distintas para no plagar la pantalla de información difícil de interpretar.

- La estandarización de la arquitectura interna de los archivos de transmisión de datos entre programas amplía el uso de *Parametric Wing Designer* para que pueda ser utilizado con múltiples objetivos, entre los que se encuentra: la manipulación unitaria de perfiles aerodinámicos, donde las operaciones de escalado y rotado pueden llegar a resultar de especial interés si se exportan perfiles o geometrías para ser analizadas mediante técnicas CFD; exportar geometrías completas compuestas por múltiples secciones a programas CAD/CAM para ser analizadas y fabricadas mediante técnicas de fabricación aditiva u otros procesos de fabricación o exportar estas secciones a programas de ensayo mediante técnicas FEM para analizar el comportamiento mecánico de la geometría. Como se puede ver, son diversas las aplicaciones en las que este programa puede tener un uso relevante y por ello, los archivos generados para transmitir información al resto de programas se ha generalizado lo máximo posible, evitando ambigüedades en el formato de escritura de los mismos.

Para ejemplificar la utilidad del programa y su uso en la fabricación de alas mediante técnicas de fabricación aditiva, se ha diseñado y construido un ala mediante deposición de material fundido (FDM), comúnmente conocido como «impresión 3D». Las principales conclusiones extraídas del proceso de fabricación y la pieza obtenida son las siguientes.

- No se ha conseguido el resultado esperado imprimiendo el ala mediante la *Blackbelt 3D*, aun así, el ala ha podido ser fabricada en secciones mediante el uso de una impresora 3D convencional. Se ha comprobado que la mejor forma de imprimir este tipo de piezas finas es en dirección vertical. Un ala podría ser impresa en una sola pieza empleando una máquina que disponga de una gran altura de impresión.
- El uso de máquinas de fabricación 3D como la *Blackelt 3D* son especialmente interesantes en la fabricación de alas, ya que, al permitir una impresión «infinita» en uno de sus ejes se pueden fabricar elementos alargados como es el caso. Sin embargo, esta tecnología se encuentra todavía en un estado de desarrollo y muy limitada en la fabricación de objetos caracterizados por bordes finos.
- Mediante el uso de la fabricación aditiva, se ha demostrado que es posible fabricar alas completas de aeronaves con el mínimo soporte técnico. La necesidad de fabricar moldes de una geometría completa y la asistencia de personal para poder fabricar este tipo de piezas se ha eliminado por completo, abaratando constes de producción y personal y eliminando el riesgo o la posibilidad de fallo humano.

8.2. Pasos y proyectos futuros

Parametric Wing Designer es un programa que puede seguir actualizándose y expandiéndose con nuevas funcionalidades. A continuación se enumera una lista de las acciones que pueden realizarse para seguir con el desarrollo del programa.

- Generar un algoritmo que modele la geometría tridimensional en formato *.stl* para que pueda ser exportada directamente a programas de impresión 3D y se evite el paso intermedio de programas CAD.
- Desarrollar un motor gráfico que permita visualizar y manipular la geometría alar dentro del propio programa.
- Diseñar una nueva pestaña que proporcione o amplíe la información sobre el ala diseñada. Aquí se pueden calcular parámetros relevantes como la cuerda media aerodinámica, el ángulo de flecha al cuarto y medio de cuerda, el alargamiento, el estrechamiento, la superficie mojada o el volumen interno de la geometría. El cálculo de estos parámetros posee especial interés dentro del ámbito académico ya que puede ayudar a alumnos de ingeniería aeroespacial a agilizar los procesos de diseño y análisis de aeronaves.
- Siguiendo con la dinámica del punto anterior, se propone añadir una nueva pestaña enfocada a calcular las características aerodinámicas del ala. Estos parámetros se usarían para realizar estimaciones del comportamiento del ala en términos de resistencia parásita e inducida o sustentación generada.
- En cuanto a la punta de ala, *Parametric Wing Designer* solo es capaz de generar extremos rectos, elípticos o circulares. Se propone ampliar las opciones de punta de ala para que también se puedan diseñar *winglets*. Este proceso se pretendía llevar a cabo en este proyecto, no obstante, a pesar de que no ha sido posible, el algoritmo encargado de mostrar las vistas de la geometría se encuentra ya optimizado para mostrar secciones rotadas en cualquier eje.
- La aplicación se ha traducido a lengua castellana e inglesa. La ampliación a más idiomas se recomienda encarecidamente en caso de que sea usada a nivel académico por alumnos extranjeros. Para traducir el texto tan solo se ha de acudir a la carpeta *config*, añadir un archivo de texto con el nombre del nuevo idioma y replicar los contenidos mostrados en los archivos *spanish* o *english* traduciendo únicamente

aquello que se encuentre a la derecha del doble punto («:»). Posteriormente de ha de añadir el nuevo idioma dentro de la linea *self.availableLanguage* del bloque *SettingsPage* del código contenido en la Subsección D.1.6.

- Añadir la opción de recortar los perfiles alares para poder introducir las superficies de control aerodinámico o elementos hipersustentadores y evitar así el exportado de la geometría a programas CAD.
- El rendimiento de la aplicación puede ser mejorado actualizando los parámetros únicamente cuando sea necesario. Actualmente, al modificar un parámetro de una sección concreta, el programa recalcula la posición de todas las secciones cuando solo debería calcular aquellas que se encuentren a partir de la última sección modificada.
- Aunque es perfectamente válido, *Python* no es el mejor lenguaje de programación para desarrollar una interfaz gráfica o para realizar un gran número de operaciones ágilmente. Por ello, se propone transcribir el código a otro lenguaje de programación como pueda ser *C#* o *Java*.

En cuanto a la fabricación de alas mediante técnicas de fabricación aditiva:

- En caso de tener la oportunidad de trabajar con una máquina de fabricado LENS, sería interesante emplearla para fabricar en aluminio el revestimiento externo de un ala y analizar las propiedades mecánicas, peso y comportamiento aerodinámico de la pieza.

Bibliografía

[1] J. P. Kruth, M. C. Leu, and T. Nakagawa. Progress in Additive Manufacturing and Rapid Prototyping. *CIRP Annals - Manufacturing Technology*, 47, 1998.

[2] Luis Miguel García-Cuevas González, Marcos Carreres Talens, and Andrés Omar Tiseira Izaguirre. Arquitectura General de Aeronaves. Technical report, Universitat Politècnica de València.

[3] Stefan Krause. Un Blériot XI, volado por Mikael Carlson en el Oldtimer Fliegertreffen, aeródromo de Hahnweide, Alemania, September 2013. URL: https://commons.wikimedia.org/wiki/File:Bleriot_XI_OTT_2013_D7N9461_004.jpg.

[4] Royal Air Force official photographer. Vickers Wellington bombers under construction, May 2018.

[5] Eduardo Ribelles. MTorres hace el primer fuselaje de una sola pieza del mundo, May 2017. Consultado el 1 de Junio de 2021. URL: https://www.laverdad.es/murcia/201705/27/mtorres-hace-primer-fuselaje-20170527002942-v.html.

[6] Miguel Bidegain. El proyecto 'Torreswing' de MTorres entra en una nueva fase, August 2017. Consultado 1 de Junio de 2021. URL: https://navarracapital.es/el-proyecto-torreswing-de-mtorres-entra-en-una-nueva-fase/.

[7] Josh Petty. What is 3D Modeling and What's It Used For? Consultado 3 de Junio de 2021. URL: https://conceptartempire.com/what-is-3d-modeling/.

[8] Don Sears. The craft of building today's aircraft wings, May 2019. Consultado 3 de Junio de 2021. URL: https://www.mscdirect.com/betterMRO/metalworking/craft-building-today\OT1\textquoterights-aircraft-wings.

[9] David Herruzo Moral. Manufacturing process optimization of an airplane wing rib by using additive manufacturing. resreport, Universidad Carlos III de Madrid, 2019. URL: https://bit.ly/34I5U0v.

[10] P. H. Moreira Magalhães. New materials being used in aircraft industry, December 2014. Consultado 4 de Junio de 2021. URL: https://degradationworld.wordpress.com/2014/12/14/new-materials-being-used-in-aircraft-industry/.

[11] Faculty of Aerospace Engineering. Aircraft and spacecraft wing structures, November 2011. URL: https://ocw.tudelft.nl/wp-content/uploads/AE1102_Structures_Slides_4.pdf.

[12] Department of Mechanical Engineering and Materials. Conformado por Arranque de Viruta. Universitat Politècnica de València.

[13] Thomas Publishing Company. Types of Machining. Consultado 4 de Junio de 2021. URL: https://www.thomasnet.com/articles/custom-manufacturing-fabricating/types-machining/.

[14] Unisign Machine Tools. structural parts; wingrib, spar, stringer. Consultado 4 de Junio de 2021. URL: https://www.unisign.com/applications/aerospace/structural-parts-fuselage-emparage-bulkhead/.

[15] MTorres. Torres Surface Milling. Consultado 4 de Junio de 2021. URL: https://www.mtorres.es/en/aeronautics/products/metallic/torres-surface-milling.

[16] Isabel Ordeig, Carlos Vila, and Maria Dolores Navarro. Procesos de conformado por Deformación plástica. Universitat Politècnica de València.

[17] Garre Parthasarathy and Arjun G. Venkata. Modeling and analysis of a RIBS and Spars of an airplane wing for bending and shear loads. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, 5(2), 2017.

[18] Taylan Altan, Gracious Ngaile, and Gangshu Shen. Cold and Hot Forging - Fundamentals and Application. *ASM International*, 2005. URL: https://bit.ly/34U5RP7.

[19] R. Liu. *Laser Additive Manufacturing Aerospace applications of laser additive manufacturing*. 2017. doi:10.1016/B978-0-08-100433-3.00013-0.

[20] Kaufui V. Wong and Aldo Hernandez. A review of additive manufacturing. *International scholarly research notices*, 2012, 2012.

[21] Ben Redwood. Additive manufacturing technologies: An overview. URL: https://www.hubs.com/knowledge-base/additive-manufacturing-technologies-overview/#introduction.

[22] Relativity Space. EXPANDING THE POSSIBILITIES FOR HUMAN EXPERIENCE. URL: https://www.relativityspace.com/mission.

[23] Ajoy Kundu. *Aircraft Design*. Cambridge Aerospace, 1 edition, 2010.

[24] Organización de Aviación Civil Internacional, Montreal. *Manual de investigación de accidentes e incidentes de aviación. Parte III: Investigación*, 2011. (Doc 9756 AN/965).

[25] Frank Davis Adams. *Aeronautical dictionary*. US Government Printing Office, 1959.

[26] Peppler IL. *From the ground up*. Aviation Publishers Co. Limited, Ottawa Ontario, twenty seventh revised edition, 1996.

[27] Egbert Torenbeek. *Synthesis of Subsonic Airplane Design: An Introduction to the Preliminary Design of Subsonic General Aviation and Transport Aircraft, with Emphasis on Layout, Aerodynamic Design, Propulsion and Performance*. Delft University Press, 1st edition, 1982.

[28] Snorri Gudmundsson. *General Aviation Aircraft Design: Applied Methods and Procedures*. Butterworth-Heinemann, 1 edition, 2013.

[29] Jan Roskam and C Edward Lan. *Airplane aerodynamics and performance*. Airplane design and analysis. DARcorporation, rev. ed edition, 1997.

[30] Noemí de Haro Soria, Ricardo García Garre, Amparo Lucas Cardós, and Ana Lull Piera. ESTUDIO AERODINÁMICO SOBRE LA EVOLUCIÓN DE LA GEOMETRÍA DEL ALA A LO LARGO DEL TIEMPO. Technical report, Universitat Politècnica de Valencia, Valencia, May 2020.

[31] Jan Roskam. *Airplane design: Part III: Layout design of cockpit, fuselage, wing and empennage: Cutaways and inboard profiles*. University of Kansas, 2002.

[32] Denis Howe. *Aircraft Conceptual Design Synthesis - Configuration of the Wing*, volume 10.1002/9781118903094. 2000. doi:10.1002/9781118903094.ch5.

[33] Real Academia de Ingeniería. Torsión de un ala. URL: http://diccionario.raing.es.

[34] Mario Fernández Gómez and Enrique Barbero Pozuelo. Análisis estructural de una pala del rotor de un helicóptero medio. Technical report, UNIVERSIDAD CARLOS

III DE MADRID - Deperartamento de Mecánica de Medios Continuos y Teoría de Estructuras, June 2010.

[35] William T. Scarbrough. NACA FOUR-DIGIT AIRFOIL SECTION GENERATIONUSING CUBIC PARAMETRIC CURVE SEGMENTS ANDTHE GOLDEN SECTION. mathesis, Department of Mechanical Engineering Rochester Institute of Technology, One Lomb Memorial DriveRochester, New York, April 1992.

[36] Blackbelt 3D. Blackbelt 3D: The benchmark in 3D belt printing, 2021. URL: https://blackbelt-3d.com/the-blackbelt-3d-printer/.

[37] Autodesk Support. How to install an add-in or script in Fusion 360, June 2021. URL: https://knowledge.autodesk.com/support/fusion-360/troubleshooting/caas/sfdcarticles/sfdcarticles/How-to-install-an-ADD-IN-and-Script-in-Fusion-360.html.

[38] Ultimaker. Ultimaker 3 Extended - Specification sheet. Consultado 29 de Junio de 2021. URL: https://docs.rs-online.com/b905/A700000006511621.pdf.

[39] Shady Farah, Daniel G. Anderson, and Robert Langer. Physical and mechanical properties of PLA, and their functions in widespread applications—A comprehensive review. *Advanced drug delivery reviews*, 107:367–392, December 2016. URL: https://dspace.mit.edu/handle/1721.1/112940.

Apéndices

A

Pliego de condiciones

A.1. Requisitos del sistema

Para poder ejecutar el programa *Parametric Wing Designer*, se ha de disponer de un sistema operativo *Windows* 10 y un espacio mínimo en la memoria de 51.2 MB.

El rendimiento de la aplicación podría variar de forma significativa en función del *hardware* de la máquina utilizada.

A.2. Manual de usuario

Al iniciar el programa, este dará la bienvenida al usuario a través de la pestaña mostrada en la Figura A.1.

El programa ofrece tres opciones para comenzar. Pulsando el botón «Editor perfil» se accede a la pestaña del editor de perfiles alares, donde se pueden crear perfiles NACA de cuatro dígitos o importar perfiles de una fuente externa. Pulsando el botón «Editor de ala» se accede a la pestaña del editor de la geometría. Por último, pulsando sobre el icono de rueda dentada se accede al menú de configuración de la aplicación.

En caso de pulsar un botón no deseado en esta primera pestaña de bienvenida se puede cambiar en todo momento haciendo uso del nuevo menú que aparecerá a la izquierda de la pantalla.

Figura A.1: Pestaña de inicio de *Parametric Wing Designer*

A.2.1. Menú de configuración

A través de esta pestaña se pueden variar aspectos tales como el idioma de la interfaz de usuario entre catellano e ingles y la extensión empleada para exportar los archivos de los perfiles y secciones de forma individual.

Para actualizar el idioma se ha de seleccionar el idioma deseado, pulsar el botón «Guardar» y **reiniciar la aplicación**.

Para cambiar la extensión empleada en la exportación de archivos se ha de seleccionar la extensión deseada entre las opciones disponibles (*.txt*, *.dat* o *.csv*) y pulsar el botón «Guardar». En este caso **no** es necesario **reiniciar la aplicación**.

A.2.2. Proceso de diseño

Una explicación detallada sobre el funcionamiento y organización de las pestañas del Editor de perfil y Editor de ala se puede consultar respectivamente en las secciones 6.3 y 6.4 de este documento.

Para ejemplificar el proceso de diseño de un ala y uso del programa se realiza como

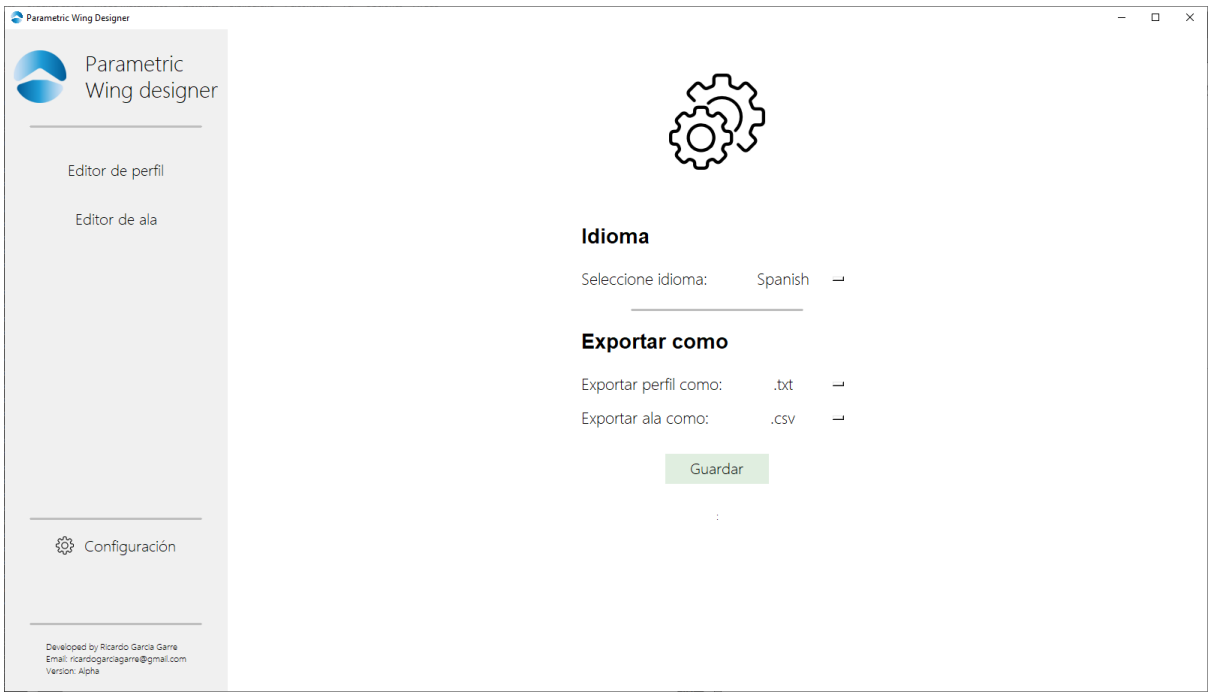


Figura A.2: Pestaña de configuración de *Parametric Wing Designer*

ejemplo el ala fabricada en este proyecto y cuyo plano se muestra en el Apéndice B. Los parámetros más relevantes a tener en cuenta para el diseño dentro del programa se muestran en la Tabla A.1.

	Sección 0	Sección 1	Sección elíptica
Envergadura	0	0.35	1.15
Cuerda	0.35	0.35	–
Ángulo de flecha	0	0	0
Ángulo de diedro	0	0	0
Ángulo de ataque	0	0	0
Posición de punta	–	–	100 %
Número de secciones	–	–	8
Perfil alar	NACA 4412	NACA 4412	NACA 4412

Tabla A.1: Tabla de parámetros de las secciones del ala fabricada

Puesto que el perfil alar es común a las tres secciones, se comienza el proceso de diseño en la pestaña del Editor de perfil. Al crear una sección nueva, esta adopta el perfil alar de la sección anterior, de esta forma, definiendo inicialmente el perfil a utilizar en la Sección 0 se definen indirectamente todos los perfiles de las secciones posteriores.

Para implementar el perfil NACA 4412 sobre la Sección 0 se opta por utilizar los *sliders* del programa. En caso de emplear la opción de importado de perfiles es importante

asegurar que cumplen con las condiciones especificadas en la Sección 4.5. Seguidamente, se pulsa el botón «Usar perfil» para confirmar la operación y actualizar el perfil de la sección seleccionada (Sección 0 en este caso).

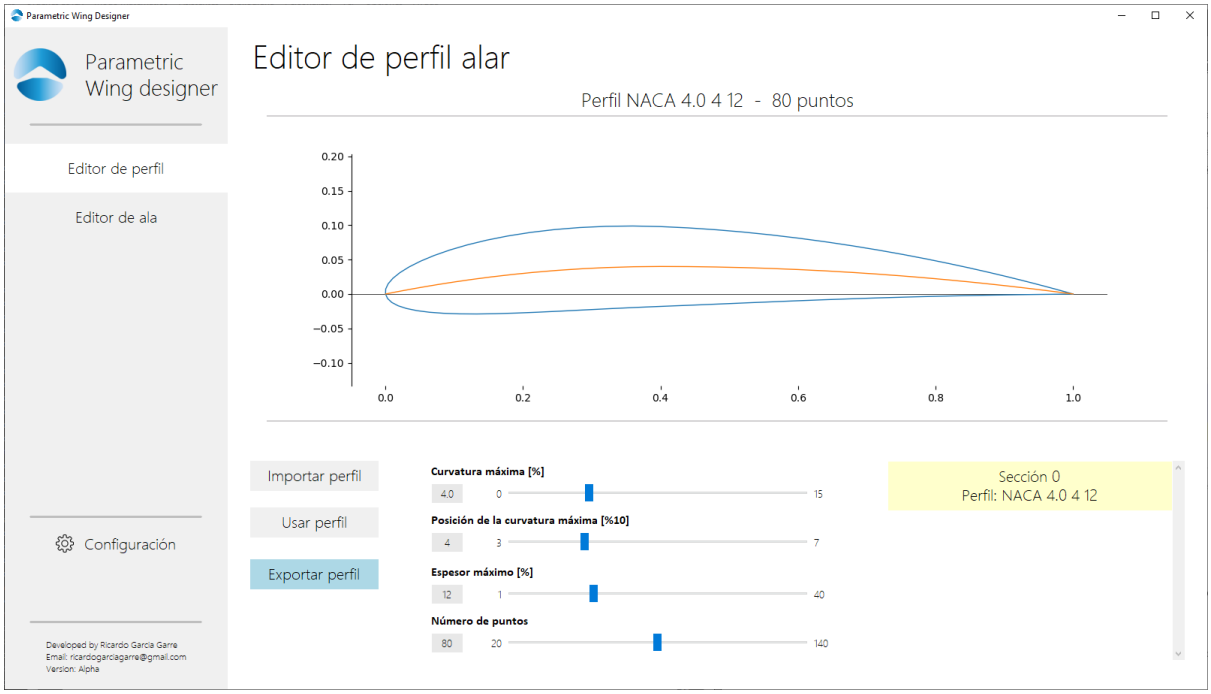


Figura A.3: Creación del perfil NACA 4412 formado por 80 puntos en la pestaña de Editor de perfil

Posteriormente, se accede a la pestaña del Editor de ala y se crea una sección nueva haciendo uso del botón «Añadir sección». Se introducen los parámetros mostrados en la Tabla A.1 para cada sección, avanzando progresivamente por el cuaderno de parámetros y variando la vista mostrada según el interés del usuario. En la Figura A.4 se muestra como se ha introducido la cuerda de valor 0.35 en ambas secciones, obteniendo el sector recto.

A continuación, se avanza hasta la pestaña «Punta elíptica» para generar la sección semielíptica. Introduciendo los parámetros de esta sección mostrados en la Tabla A.1, se obtiene la geometría final mostrada en la Figura A.5.

Finalmente, se pulsa sobre el botón «Exportar geometría» y se selecciona el directorio en el que se desea guardar los archivos. Tan solo se debe pulsar sobre el botón «Seleccionar carpeta» y la geometría se exportará automáticamente dentro de una carpeta.

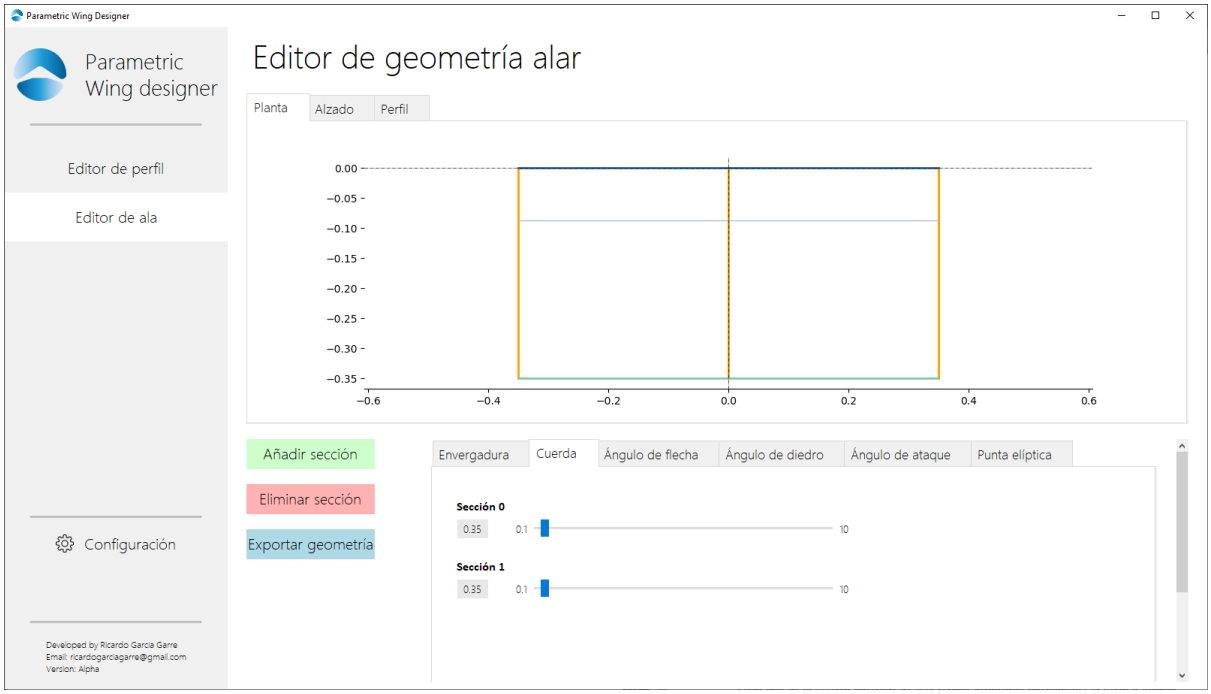


Figura A.4: Pestaña del Editor de ala donde se varía el parámetro de la cuerda de las secciones y se visualiza la vista en planta de la geometría

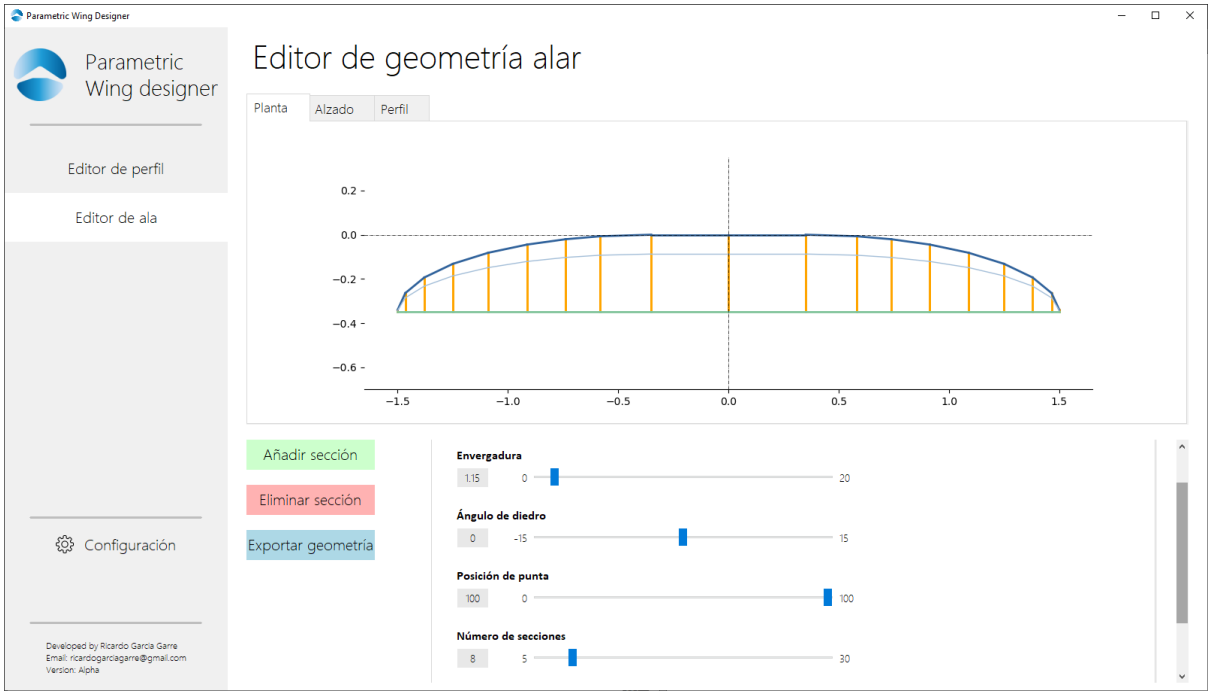


Figura A.5: Pestaña del Editor de ala donde se varía la forma y resolución de la sección semielíptica y se visualiza la vista en planta de la geometría

A.3. Uso de *ImportWingGeometryLines*

Para poder ejecutar el módulo *ImportWingGeometryLines* es necesario disponer del programa FUSION 360®.

Para instalar el módulo de forma correcta se recomienda consultar la guía proporcionada por AUTODESK© [37]. El módulo puede ser instalado de forma alternativa copiando y pegando el código de la Subsección D.2.1 directamente en la opción crear *script* del apartado *ADD-INS*.

Para ejecutar el módulo una vez instalado, tan solo hay que acceder a la pestaña *Tools* y pulsar sobre el botón *ADD-INS* mostrado en la Figura A.6.

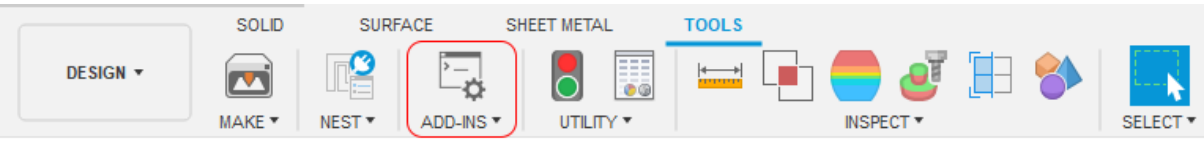


Figura A.6: Opciones de la pestaña *Tools* de FUSION 360®

Se selecciona *ImportWingGeometryLines* y a continuación se abre el explorador de archivos del sistema operativo. Se debe seleccionar el archivo *geometry.csv*, almacenado en la carpeta que contenga la geometría del ala. Tras la ejecución del módulo se obtiene el resultado mostrado en la Figura 7.3.

Cabe mencionar que, *ImportWingGeometryLines* interpreta las coordenadas en el sistema métrico internacional. Por ello, todas las medidas introducidas en *Parametric Wing Designer* se han supuesto en metros.

A.4. Especificaciones de la impresora *Blackbelt 3D*

En la Tabla A.2 se muestra una lista con las especificaciones técnicas de la impresora *Blackbelt 3D* facilitadas por el fabricante [36].

Parámetro	Valor	Unidad
Dimensiones	59 x 105 x 66	cm
Peso	45	kg
Volumen de impresión	340 x 300 x ∞	mm
Máxima velocidad del cabezal	150	mm/s
Precisión	$\pm$0.4	mm
Altura de capa	0.5 – 0.8	mm
Temperatura máxima de cama caliente	140	°C
Temperatura máxima de cabezal	300	°C
Requisitos de energía	120 – 240	V AC
Potencia de la fuente de alimentación	600	W
Potencial de la fuente de alimentación	24	V

Tabla A.2: Especificaciones técnicas de la impresora *Blackbelt 3D*

A.5. Especificaciones de la impresora *Ultimaker 3 Extended*

En la Tabla A.3 se muestra una lista con las especificaciones técnicas de la impresora *Ultimaker 3 Extended* facilitadas por el fabricante [38].

Parámetro	Valor	Unidad
Propiedades de impresión e impresora		
Cabezal de impresión	Doble extrusor	[–]
Volumen de impresión	215 x 215 x 300	mm
Diámetro del filamento	2.85	mm
Resolución de capa	20 – 200	μm
Precisión XYZ	12.5, 12.5, 2.5	μm
Velocidad del cabezal	30 – 300	mm/s

Continuación en la página siguiente

Tabla A.3 – *Continuación de la página anterior*

Parámetro	Valor	Unidad
Temperatura máxima de cama caliente	20 – 100	°C
Nivelación de la cama caliente	Nivelación activa	[–]
Diámetro de la boquilla	0.4	mm
Temperatura de la boquilla	180 – 280	°C
Tiempo de calentamiento de la boquilla	< 2	min
Tiempo de calentamiento de la cama	< 4	min
Sonido de funcionamiento	50	dBA
Dimensiones físicas		
Dimensiones	342 x 380 x 489	mm
Peso neto	11.3	kg
Requerimientos de energía		
Entrada	100 – 240	V
	4	A
	50 – 60	Hz
	221	W máx
Salida	24	V DC
	9.2	A
	220	W

Tabla A.3: Especificaciones técnicas de la impresora *Ultimaker 3 Extended*

En cuanto a materiales compatibles, la máquina *Blackbelt 3D* puede imprimir con PLA, PETG, Ngen, XT, H, ABS y TPU entre otros, mientras que la *Ultimaker 3 Extended* solo puede imprimir Nylon, PLA, ABS, CPE y PVA.

A.6. Propiedades del Ácido poliláctico

Las principales propiedades térmicas y mecánicas del ácido poliláctico o PLA se muestran resumidas en la Tabla A.4 [39].

Parámetro	Símbolo	Valor	Unidad
Densidad del polímero	ρ	1.21 – 1.25	g/cm^3
Fuerza de tensión	σ	21 – 60	MPa
Módulo de Young	E	0.35 – 3.5	GPa
Tensión máxima	ε	2.5 – 6	%
Fuerza de tensión específica	σ*	16.8 – 48.0	Nm/g
Módulo de Young específico	E*	0.28 – 2.80	kNm/g
Temperatura de transición vítrea	T_g	45 – 60	°C
Temperatura de fusión	T_m	150 – 162	°C

Tabla A.4: Propiedades del PLA

A.7. Parámetros de impresión

En la Tabla A.5 se muestran los parámetros de impresión utilizados para fabricar la pieza. Cabe destacar que uno de los parámetros más relevantes es el «Recuento de líneas de pared» fijado en 1.

Parámetro	Valor	Unidad
Calidad		
Altura de capa	0.3	mm
Altura de capa inicial	0.3	mm
Ancho de línea	0.4	mm
Ancho de línea de pared	0.4	mm
Ancho de línea de la pared exterior	0.4	mm
Ancho de línea de pared interna	0.4	mm
Ancho de línea superior/inferior	0.4	mm
Ancho de línea de relleno	0.4	mm
Ancho de línea de falda/borde	0.4	mm

Continuación en la página siguiente

Tabla A.5 – *Continuación de la página anterior*

Parámetro	Valor	Unidad
Ancho de línea de la capa inicial	100 %	%
Perímetro		
Grosor de la pared	0.8	mm
Recuento de líneas de pared	1	[–]
Distancia de pasada de la pared exterior	0.0	mm
Capas de la superficie superior del forro	0	[–]
Grosor superior/inferior	1.2	mm
Grosor superior	1.2	mm
Capas superiores	2	[–]
Grosor inferior	1.2	mm
Capas inferiores	2	[–]
Patrón superior/inferior	Líneas	[–]
Patrón inferior de la capa inicial	Líneas	[–]
Direcciones de línea superior/inferior	[]	[–]
Entrante en la pared exterior	0	mm
Optimizar el orden de impresión de paredes	Si	[–]
Paredes exteriores antes que interiores	No	[–]
Alternar pared adicional	No	[–]
Compensar superposiciones de pared	Si	[–]
Compensar superposiciones de pared exterior	No	[–]
Compensar superposiciones de pared interior	Si	[–]
Rellenar espacios entre paredes	En todas partes	[–]
Filtrar pequeños huecos	No	[–]
Imprimir paredes finas	No	[–]
Expansión horizontal	0	mm
Expansión horizontal de la capa inicial	0	mm
Expansión horizontal de orificios	0	mm
Alineación de costuras en Z	Especificado usuario	[–]
Posición de costura en Z	Posterior	[–]
X de la costura Z	225.0	mm
Y de la costura Z	450	mm
Preferencia de esquina de costura	Costura inteligente	[–]
Costuras relativas en Z	No	[–]

Continuación en la página siguiente

Tabla A.5 – *Continuación de la página anterior*

Parámetro	Valor	Unidad
Sin forro en huecos en Z	No	[–]
Recuento de paredes adicionales de forro	1	[–]
Habilitar alisado	No	[–]
Porcentaje de superposición del forro	10.0	%
Superposición del forro	0.04	mm
Relleno		
Densidad de relleno	15	%
Distancia de línea de relleno	2.6667	mm
Patrón de relleno	Giroide	[–]
Conectar líneas de relleno	No	[–]
Direccions de línea de relleno	[]	[–]
Comienzo de relleno aleatorio	No	[–]
Multiplicador de línea de relleno	1	[–]
Porcentaje de superposición del relleno	30.0	%
Superposición del relleno	0.12	mm
Distancia de pasada de relleno	0.0	mm
Grosor de la capa de relleno	0.3	mm
Pasos de relleno necesarios	0	[–]
Relleno antes que las paredes	No	[–]
Área de relleno mínima	0	mm^2
Soporte de relleno	No	[–]
Anchura de retirada del forro	0.4	mm
Anchura de retirada del forro superior	0.4	mm
Anchura de retirada del forro inferior	0.4	mm
Distancia de expansión del forro	0.4	mm
Distancia de expansión del forro superior	0.4	mm
Distancia de expansión del forro inferior	0.4	mm
Ángulo máximo de expansión del forro	90	°
Anchura de expansión mínima del forro	0.0	mm
Espesor de soporte de los bordes del forro	0	mm
Capas de soporte de los bordes del forro	0	[–]
Material		
Temperatura de impresión	200	°C

Continuación en la página siguiente

Tabla A.5 – *Continuación de la página anterior*

Parámetro	Valor	Unidad
Temperatura de impresión de la capa inicial	200	°C
Temperatura de impresión inicial	200	°C
Temperatura de impresión final	200	°C
Temperatura de la placa de impresión	60	°C
Temperatura de la placa de impresión inicial	60	°C
Flujo	100	%
Flujo de pared	100	%
Flujo de pared exterior	100	%
Flujo de pared interior	100	%
Flujo superior o inferior	100	%
Flujo de relleno	100	%
Flujo de falda/borde	100	%
Flujo de la torre auxiliar	100	%
Flujo de capa inicial	100	%
Adherencia de la placa de impresión		
Tipo de adherencia de la placa de impresión	Falda	[–]
Recuento de líneas de falda	20	[–]
Distancia de falda	0	mm
Longitud mínima de falda/borde	250	mm

Tabla A.5: Parámetros de impresión de la pieza

B

Planos

A continuación se muestra el plano tres vistas a escala 1:5 del semiala diseñada.

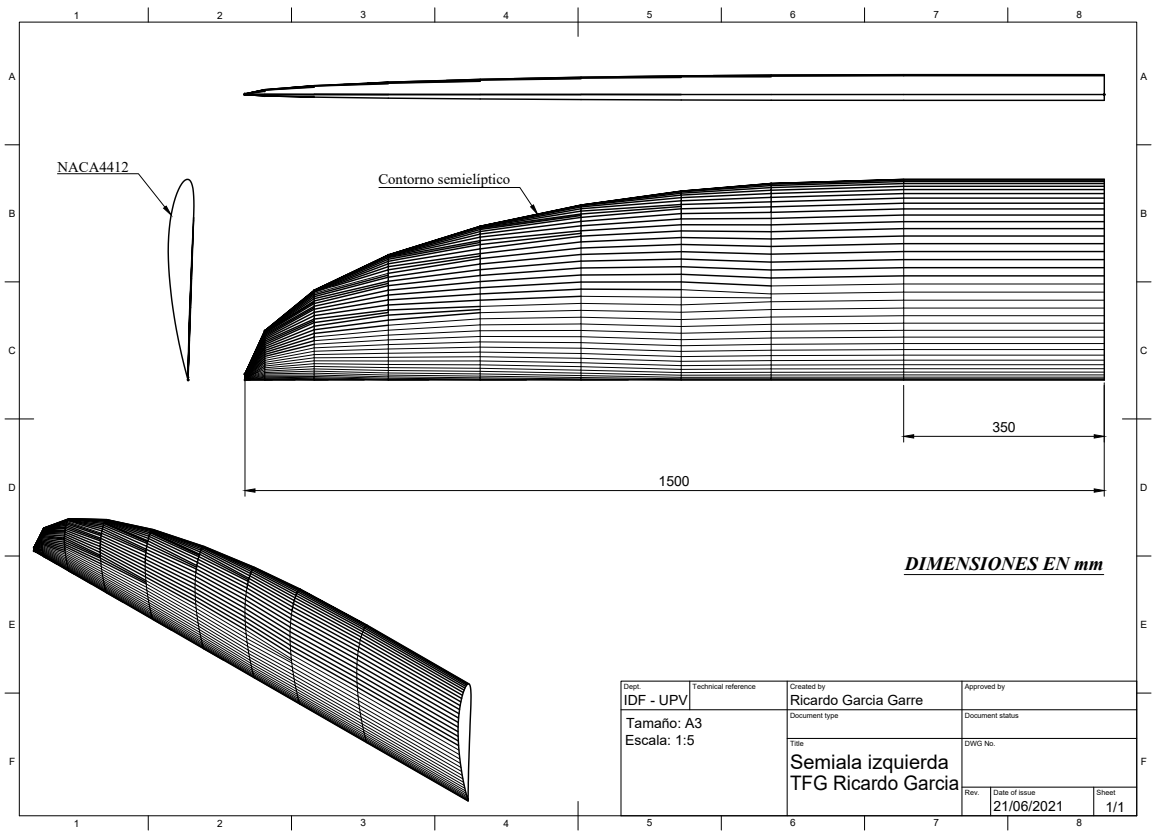


Figura B.1: Plano del ala diseñada

C

Presupuesto

Finalmente, se realiza una estimación del coste económico del proyecto. En primer lugar se expone un desglose de los costes divididos en categorías y finalmente se computa la cantidad total a la que asciende el trabajo.

C.1. Desglose del presupuesto

Los costes se han divido en cinco categorías distintas: mano de obra, licencias de *software*, dispositivos y máquinas, material fungible y consumo eléctrico.

C.1.1. Mano de obra

Este trabajo ha sido elaborado por el Autor del mismo, cuya formación pertenece al grado de ingeniería aeroespacial y ha sido tutorizado por el catedrático de la Universitat Politècnica de València, Juan Antonio García Manrique.

Para computar el coste derivado de la mano de obra se ha tomado el sueldo medio de un ingeniero técnico aeroespacial en España y el de un catedrático de universidad para el año 2021 y se ha estimado el coste de su trabajo por hora. El coste final derivado de la mano de obra se muestra en la Tabla C.1.

Cantidad	Concepto	Coste/ud.	Importe
10	**Horas ingeniero catedrático**	41.50 €	415.00 €
5	Supervisión y guiado del trabajo		
5	Procesado e impresión de la pieza		
402.5	**Horas ingeniero técnico aeronáutico**	20.00 €	8 050.00 €
30	Documentación y formación		
250	Desarrollo de la aplicación		
100	Redacción de la memoria		
15	Elaboración de imágenes en *Ilustrator*®		
5	Validación de las ecuaciones		
2	Desarrollo del módulo de *Fusion360*®		
0.5	Elaboración de pieza y planos		
	Coste total		**8 465.00 €**

Tabla C.1: Desglose del coste derivado de la mano de obra

C.1.2. Licencias de *software*

Para la elaboración del código del programa, la memoria del proyecto y el modelo y procesado de la pieza impresa se ha hecho uso de una serie de programas informáticos cuyo coste aparece reflejado en la Tabla C.2. Para algunos de los programas utilizados ha sido necesaria la obtención de una licencia de uso donde el coste ha sido asumido por el Autor o por la Universitat Politècnica de València, la cual cuenta con licencias de estudiante en programas como Fusion360®. Por otro lado, se han usado programas de suscripción gratuita que no resultan en un incremento del coste final.

Concepto	Coste/ud.	Horas uso	Coste/h	Importe
Licencia mensual de *Ilustrator*®	31.49 €	15	0.066 €	0.98 €
Licencia anual de *Fusion360*®	352.00 €	2.5	0.20 €	0.50 €
Licencia de *Texmaker*®	0.00 €	100	0.00 €	0.00 €
Licencia de *Visual Studio Code*®	0.00 €	250	0.00 €	0.00 €
Licencia de *Ultimaker Cura*®	0.00 €	0.5	0.00 €	0.00 €
Coste total				**1.48 €**

Tabla C.2: Desglose del coste derivado del uso de programas informáticos

El coste de uso de estos programas ha sido dividido en cuotas horarias donde se ha asumido que un año laboral contiene 1760 horas.

C.1.3. Dispositivos y máquinas

El único dispositivo empleado para el desarrollo del código es el ordenador, además, se ha hecho uso de dispositivos periféricos tales como una pantalla y ratón inalámbrico. La pieza, a pesar de que finalmente no ha podido ser fabricada mediante la máquina *Blackbelt 3D*, se ha asumido que ha sido la empleada de fabricar la pieza ya que este era el objetivo principal. El coste de los equipos aparece reflejado en la Tabla C.3.

Para estimar el coste por hora de los dispositivos usados en el desarrollo del programa, se ha estimado un tiempo de vida útil de los mismos de cinco años donde se han supuesto años de 1760 horas. En cuanto a la impresora, de igual forma se ha estimado una vida útil de cinco años y años de 8760 horas, ya que, al contrario que el ordenador, la impresora es completamente autónoma y puede trabajar fuera del horario laboral.

Concepto	**Coste/ud.**	**Horas uso**	**Coste/h**	**Importe**
Desarrollo del programa				
Ordenador Dell Inspiron 5547	416.37 €	402.5	0.047 €	19.04 €
Ratón Logitech M185	9.99 €	402.5	0.001 €	0.46 €
Monitor Asus Vx239h 23”	120.00 €	402.5	0.014 €	5.49 €
Fabricación de la pieza				
Impresora *Blackbelt 3D*	10 700.00 €	24	0.24 €	5.86 €
Coste total				**30.85 €**

Tabla C.3: Desglose del coste derivado de los dispositivos y máquinas

C.1.4. Material fungible

El único material fungible empleado en este proyecto ha sido el material con el que se ha fabricado el ala. Su coste aparece reflejado en la Tabla C.4.

Cantidad	Concepto	Coste/kg	Importe
0.23	Sicnova PLA Yellow	15.00 €	3.45 €
	Coste total		**3.45 €**

Tabla C.4: Desglose del coste derivado del material

C.1.5. Coste energético

La potencia consumida por los diferentes dispositivos y su coste asociado así como la energía consumida por los dispositivos de iluminación que han acondicionado el espacio de trabajo se muestra resumido en la Tabla C.5.

El precio de la luz ha sido calculado a partir de la nueva tarifa con discriminación horaria impuesta en junio de 2021 donde se ha supuesto que la totatilad del trabajo se ha desarrollado en la franja de hora punta donde el coste medio del kWh resulta de 0.2061 €.

Concepto	Potencia [W]	Horas	Consumo [kWh]	Importe
Iluminación				
Flexo LED	10	402.5	4.03	0.83 €
Set de iluminación	15	402.5	6.04	1.24 €
Consumo ordenador	65	402.5	26.16	5.39 €
Consumo Pantalla	18.5	402.5	7.45	1.53 €
Consumo impresora	600	48	28.80	5.94 €
Consumo total		72.47 kWh		
Precio medio kWh		0.2061 €/kWh		
Coste total				**14.94 €**

Tabla C.5: Desglose del coste asociado al consumo energético

C.2. Presupuesto final

Se expone a continuación, en la tabla Tabla C.6, el presupuesto final del presente trabajo.

Concepto	**Importe**
Mano de obra	8 465.00 €
Licencias de *software*	1.48 €€
Equipos	30.85 €
Material fungible	3.45 €
Consumo eléctrico	14.94 €
Coste total (IVA no incluido)	8 515.72 €
IVA (21 %)	1 788.30 €
Presupuesto final TFG	**10 304.03 €**

Tabla C.6: Presupuesto total

Finalmente, el importe total de la elaboración del presente Trabajo Fin de Grado asciende a la cantidad de **DIEZ MIL TRESCIENTOS CUATRO EUROS Y TRES CÉNTIMOS (10 304.03 €)**.

D

Código

D.1. Archivos de *Parametric Wing Designer*

D.1.1. Clase *section*

```
from numpy import sin, cos, tan, pi, zeros, array

class section():
    def __init__(self):

        #Object and airfoil name
        self.name = ""
        self.airfoil_name = ""

        #Two-dimensional coordinates of the unit chord profile
        self.x = []
        self.y = []

        #Two-dimensional coordinates of the airfoil camber line
        self.camber_x = []
        self.camber_y = []

        #Airfoil chord
        self.chord = 1.0

        #Sector span
```

```
22          self.span = 0.0
23          #Offset in z axis
24          self.z_offset = 0
25          self.z_abs = 0
26
27          #Angle of attack
28          self.alpha = 0.0
29          self.alpha_rad = 0
30
31          #Dihedral anglel
32          self.diedro = 0.0
33          self.diedro_rad = 0
34          #Offset in y axis
35          self.y_offset = 0
36          self.y_abs = 0
37
38          #Arrow angle
39          self.arrow = 0
40          self.arrow_rad = 0
41          #Offset in x axis
42          self.x_offset = 0
43          self.x_abs = 0
44
45          #Computed three-dimensional section coordinates
46          self.xw = []
47          self.yw = []
48          self.zw = []
49
50      def compute(self):
51          self.x = array(self.x)
52          self.y = array(self.y)
53
54          #Airfoil scaling
55          if self.chord != 1.0:
56              self.xw = self.x*self.chord
57              self.yw = self.y*self.chord
58          else:
59              self.xw = self.x
```

```
60              self.yw = self.y
61
62          #Airfoil rotation
63          if self.alpha != 0.0:
64              self.alpha_rad = self.alpha*pi/180
65              xprov = self.xw*cos(self.alpha_rad)+self.yw*sin(self.
                    alpha_rad)
66              yprov = -1*self.xw*sin(self.alpha_rad)+self.yw*cos(
                    self.alpha_rad)
67              self.xw=xprov
68              self.yw=yprov
69
70          #Sector arrow angle
71          self.arrow_rad = self.arrow*pi/180
72          xoff = self.span*tan(self.arrow_rad)
73          self.x_abs = self.x_offset + xoff
74          self.xw = self.xw + (self.x_abs)
75
76          #Sector dihedral angle
77          if (self.y_offset + self.diedro)!=0:
78              self.diedro_rad = self.diedro*pi/180
79              yoff = self.span*tan(self.diedro_rad)
80              self.y_abs = self.y_offset + yoff
81              self.yw = self.yw + (self.y_abs)
82
83          #z axis displacement
84          self.zw = zeros(len(self.x))
85          self.z_abs = self.z_offset + self.span
86          self.zw = self.zw + self.z_abs
```

Código D.1: Clase *section*

D.1.2. Función *create_naca*

```
#This function creates any 4 digit naca airfoil
#Inputs (m=first naca digit, p=second naca digit, t=two las naca digits, n=number of points plotted)
#Returns coordinate x points , coordinates y, camber points x and y

from numpy import linspace, pi, sin, cos, tan, sqrt, append, flip, arctan

def create_naca(m, p, t, n):
    beta = linspace(0, pi, n)
    x = (1-cos(beta))/2 #Cosine spacing, groups the points at the start and end of the chord

    t = t/100 #Chord thickness in % (unitary chord)

    if  m==0: #Symmetrical airfoil
        y_upper = 5*t*(0.2969*sqrt(x)-0.126*x-0.3516*x**2+0.2843*x**3-0.1036*x**4) #Thickness function
        y_lower = -1*y_upper #Creates a reflection of the upper thickness

        x_upper = x #Coordinates in x axis for each point and above
        x_lower = x #Coordinates in x axis for each point and lower

        #Coordinates of the mean camber line for symmetrical airfoils
        x_camber=[0,1]
        y_camber=[0,0]

        # Grouping of all x and y coordinates, notice how the points belonging to the lower part are inverted so that when graphing it they are already ordered
        x_airfoil = append(x, flip(x))
```

```
26          y_airfoil = append(y_upper, flip(y_lower))
27
28      else: #Non-symmetrical airfoil
29          m = m/100
30          p = p/10
31
32          y_camber = []
33          dy_camber = []
34
35          for xi in x:
36              if xi <= p:
37                  y_current = (m/p**2)*(2*p*xi-xi**2)
38                  dy_current = (2*m)/p**2*(p-xi)
39
40                  y_camber.append(y_current)
41                  dy_camber.append(dy_current)
42              else:
43                  y_current = (m)/(1-p)**2*((1-2*p)+2*p*xi-xi**2)
44                  dy_current = (2*m)/(1-p)**2*(p-xi)
45
46                  y_camber.append(y_current)
47                  dy_camber.append(dy_current)
48
49          y_t = 5*t*(0.2969*sqrt(x)-0.126*x-0.3516*x**2+0.2843*x
                **3-0.1036*x**4)
50          theta = arctan(dy_camber)
51
52          x_upper = (x-y_t*sin(theta))
53          x_lower = (x+y_t*sin(theta))
54          y_upper = (y_camber+y_t*cos(theta))
55          y_lower = (y_camber-y_t*cos(theta))
56
57          x_camber = x
58
59          x_airfoil = append(flip(x_upper), x_lower)
60          y_airfoil = append(flip(y_upper), y_lower)
61      return x_airfoil, y_airfoil, x_camber, y_camber
```

Código D.2: Función *create_naca*

D.1.3. Función *create_eliptical_wing*

```
from numpy import linspace, sqrt, cos, delete, pi, tan
from section import section


def create_elipsoid(a, b, pointsNumber):
    beta = linspace(0, 3.1, pointsNumber+3) #Equidistant point
       array
    x = ((1-cos(beta))/2)*a #Point distribution and scaling
    x = delete(x, [0,1,2]) #Duplications with the previous rib
       are avoided
    y = sqrt(b**2*(1 - (x**2 / a**2)))

    return x, y

def compute_Xdisplacement(previousSection, y_elip, displacement):
    firstChord = previousSection.chord
    diff = firstChord - (y_elip*2)
    desp = diff * displacement/100
    x_disp = desp + previousSection.x_abs

    return x_disp

def create_eliptical_wing(previousSection, length, pointsNumber,
   displacement, diedro, arrow):
    sections = {} #Container where airfoils will be stored
    b = previousSection.chord/2 #Half of the starting chord
    a = length #Ellipse span

    [x_elip, y_elip] = create_elipsoid(a, b, pointsNumber)

    old = previousSection

    for point in range(len(x_elip)):
        name = "tiprib_" + str(point)
        sections[name] = section()
        sections[name].name = name
```

```
34          sections[name].airfoil_name = previousSection.
               airfoil_name
35          sections[name].x = previousSection.x
36          sections[name].y = previousSection.y
37
38          #Chord
39          sections[name].chord = 2*y_elip[point] #Double that
               calculated on the ellipse
40
41          #Span
42          sections[name].span = x_elip[point] + previousSection.
               z_abs - old.z_abs
43          sections[name].z_offset = old.z_abs
44
45          #Shift in x by ellipse
46          x_disp = compute_Xdisplacement(previousSection, y_elip,
               displacement)
47          sections[name].x_offset = x_disp[point]
48
49          #Arrow
50          arrow_off = ((sections[name].span + old.z_abs) -
               previousSection.z_abs)*tan(arrow*pi/180)
51          sections[name].x_offset =  sections[name].x_offset +
               arrow_off
52
53          #AoA
54          sections[name].alpha = previousSection.alpha
55
56          #Dihedral
57          sections[name].diedro = diedro
58          sections[name].y_offset = old.y_abs
59
60          sections[name].compute()
61
62          old = sections[name]
63
64      return sections
```

Código D.3: Función *create_eliptical_wing*

D.1.4. Archivo *file_manager*

```
from tkinter import filedialog
from tkinter import messagebox
import re, os

def load_airfoil():
    filename = filedialog.askopenfile(initialdir = ".\\dat")

    if filename is None:
        return

    [name, x, y] = process_airfoil(filename)
    filename.close()

    return name, x, y

def process_airfoil(filename):
    # The common airfoil dat format has many flavors
    # This code should work with almost every dialect

    # Regex to identify data rows and throw away unused metadata
    expre = r"^\s*(?P<xval>\-*\d*?\.\d+)\s+(?P<yval>\-*\d*?\.\d+)
       \s*$"
    # read the airfoil name which is always at the first line
    airfoilname = filename.readline().strip()

    x_airfoil=[]
    y_airfoil=[]

    # Collect the data for the upper and the lower side
       seperately if possible
    for lin in filename:
        curdat = re.match(expre,lin)
        if curdat != None:
            x = float(curdat.group("xval"))
            y = float(curdat.group("yval"))

```

```
35              x_airfoil.append(x)
36              y_airfoil.append(y)
37
38      return airfoilname, x_airfoil, y_airfoil
39
40  def save_airfoil(section, fileFormat):
41      f = filedialog.asksaveasfile(initialdir = "C:\\",
            defaultextension=fileFormat)
42
43      if f is None:
44          return
45
46      if fileFormat == ".txt" or fileFormat == ".dat":
47          valueSeparator = "\t"
48      elif fileFormat == ".csv":
49          valueSeparator = ","
50      else:
51          return
52
53      f.write(section.airfoil_name + "\n")
54
55      for point in range(len(section.xw)):
56          x = str(round(section.x[point],6))
57          y = str(round(section.y[point],6))
58
59          f.write(x + valueSeparator + y + "\n")
60      f.close()
61
62  def export_wing(dics, fileFormat):
63      directory = "wing" #Name of the new folder
64      parentDir = filedialog.askdirectory() #Search for directory
65
66      if parentDir == "": #If no directory is given quits
67          return
68
69      path = os.path.join(parentDir, directory) #The new path where
            the files will go is defined
70
```

```
71      index = 1 #Index attached to the folder to avoid duplicates
72      while True:
73          try: #Check that the folder is not yet created
74              os.mkdir(path)
75              break
76
77          except OSError as error:
78              if error.winerror == 183: #Check if the error is
                    because the folder already exists
79                  directory = "wing_" + str(index)
80                  path = os.path.join(parentDir, directory)
81                  index = index + 1
82
83              else:
84                  break
85
86      if fileFormat == ".txt" or fileFormat == ".dat":
87          valueSeparator = "\t"
88      elif fileFormat == ".csv":
89          valueSeparator = ","
90      else:
91          return
92
93      for sectionType in dics: #Dictionaries are passed as an array
            ()
94          for name in sectionType:
95              fileName = sectionType[name].name
96              x = sectionType[name].xw
97              y = sectionType[name].yw
98              z = sectionType[name].zw
99
100             try: #Creates individual file of a section
101                 f = open(path + "/" + fileName + fileFormat, "w")
102             except:
103                 messagebox.showerror("Error","No se ha podido
                        exportar la geometria.")
104                 return
105
```

```
106             for point in range(len(x)):
107                 f.write(str(round(x[point],6)) + valueSeparator)
108                 f.write(str(round(y[point],6)) + valueSeparator)
109                 f.write(str(round(z[point],6)) + "\n")
110 
111             f.close()
112 
113             try: #Creates globbal file containing  all sections
114                 f = open(path + "/" + "geometry" + fileFormat, "a
                        +")
115             except:
116                 messagebox.showerror("Error","No se ha podido
                        exportar la geometria.")
117                 return
118 
119             for point in range(len(x)):
120                 f.write(str(round(x[point],6)) + valueSeparator)
121                 f.write(str(round(y[point],6)) + valueSeparator)
122                 f.write(str(round(z[point],6)) + "\n")
123 
124             f.write("&\n")
125             f.close()
126 
127     messagebox.showinfo("Info","Geometria exportada con exito\
           nDescripcion: " + directory)
```

Código D.4: Archivo *file_manager*

D.1.5. Clase *slider*

```
1  import tkinter as tk
2  import tkinter.ttk as ttk
3
4  class slider(tk.Frame):
5      def __init__(self, parent, title, start, lower, upper, dec,
           updater):
6          tk.Frame.__init__(self, parent) #Frame is started
7          tk.Frame.config(self, bg="white") #Selection of
               background color
8
9          self.toupdate = updater
10         self.roundto = dec
11         self.val = start #Converted value
12         self.value= tk.StringVar() #String final value
13         self.value.set(start) #Initializes with indicated value
14
15         #Title lable
16         self.nameLabel = tk.Label(self, text=title, bg="white",
               font=('Calibri',12,'bold'))
17         self.nameLabel.grid(row=0, column=0, sticky='W',
               columnspan=4)
18
19         #Entry value
20         self.entryNumber = tk.Entry(self, bd=2, width=5, relief="
               flat", bg="#E8E8E8", font=('Microsoft YaHei Light',10)
               ,
21                                                 justify="center",
                                                       textvariable=self.
                                                       value)
22         self.entryNumber.grid(row=1, column=0, padx=4, sticky='W'
               , columnspan=1, pady=5)
23
24         #Slider bar
25         def setEntry(dec):
26             if self.roundto==0:
27                 self.value.set(int(self.slider.get()))
```

```
28              else:
29                  self.value.set(round(self.slider.get(),self.
                        roundto))
30
31          self.style = ttk.Style()
32          self.style.configure('whitebg.Horizontal.TScale',
                background="white")
33          self.slider = ttk.Scale(self, from_=lower, value=start,
                to=upper, style='whitebg.Horizontal.TScale', orient='
                horizontal', length=400, command=setEntry)
34          self.slider.grid(row=1, column=2)
35
36          #Title minimun
37          self.nameLabel = tk.Label(self, text=lower, width=5,
                anchor='e', bg="white", font=('Microsoft YaHei Light'
                ,10))
38          self.nameLabel.grid(row=1, column=1,  padx=5)
39
40          #Title maximum
41          self.nameLabel = tk.Label(self, text=upper, width=3,
                anchor='w', bg="white", font=('Microsoft YaHei Light'
                ,10))
42          self.nameLabel.grid(row=1, column=3, sticky='W', padx=5)
43
44          def Validate(*args): #Checks, filters and updates number
45              try:
46                  '''Checks for allowed characters, numbers, . or -
                        '''
47                  if self.value.get()[-1:].isnumeric()==False and
                        self.value.get()[-1:]!='.' and self.value.get
                        ()[-1:]!='-': #Check there're no letters
48                      self.value.set(self.value.get()[:-1]) #The
                            last character entered is deleted
49
50                  '''Checks there is no minus sign in positive
                        number '''
51                  if int(lower)>=0 and self.value.get()[-1:]=='-':
52                      self.value.set(self.value.get()[:-1]) #The
```

```
                        last character entered is deleted
53
54                  '''Checks there is no dot in an integer '''
55                  if self.roundto==0 and self.value.get()[-1:]=='.'
                        :
56                      self.value.set(self.value.get()[:-1]) #The
                            last character entered is deleted
57
58                  '''Checks there is no doubling in . or - '''
59                  if self.value.get().count('.')>1 or self.value.
                        get().count('-')>1: #Check that there is no
                        more than 1 point
60                      while self.value.get().count('.')>1:
61                          self.value.set(self.value.get()[:-1]) #
                                The last character entered is deleted
62              except:
63                  pass
64              try:
65                  if float(self.value.get()) > upper: #Checks the
                        number is not greater than the upper limit
66                      if self.value.get()[-2:]=='.0':
67                          self.value.set(self.value.get()[:-3])
68                      else:
69                          self.value.set(self.value.get()[:-1])
70
71                  if float(self.value.get()) >= lower:
72                      self.slider.set(round(float(self.value.get())
                            ,self.roundto))
73                      self.val = self.slider.get()
74                      try:
75                          self.toupdate.update()
76                      except:
77                          print('Unable to update')
78              except:
79                  print('Could not validate numer')
80          self.value.trace("w", Validate)
```

Código D.5: Clase *slider*

D.1.6. Archivo *main.py*

```
import tkinter as tk
import tkinter.ttk as ttk

import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure

from numpy import amax, array, cos, where
from create_naca import create_naca
from slider import slider
from section import section
from create_eliptical_wing import create_eliptical_wing
from file_manager import save_airfoil, load_airfoil, export_wing

graph_resolution = 100

sections = {}
sectionsElip = {}

dictionary = {}
airfoilExportFormat = ""
wingExportFormat = ""

def load_config():
    with open("./config/config.txt","r",encoding='utf-8') as f:
        lines = f.read().splitlines() #Eliminates the \n at the
           end. Avoids malfunction of function load language

        language = lines[0]
        airfoilfileformat = lines[1]
        wingfileformat= lines[2]

    dic = load_language(language)
    return dic, airfoilfileformat, wingfileformat

```

```
36 def load_language(language):
37     dic = {}
38     dic["language"] = language
39 
40     language = language.lower()
41     with open("./config/" + language +".txt","r",encoding='utf-8'
       ) as f:
42         lines = f.read().splitlines()
43 
44         for line in lines:
45             dots = line.find(":")
46             definition = line[:dots]
47             text = line[dots+1:]
48 
49             dic[definition] = text
50     return dic
51 
52 def edit_config(parameters):
53     with open("./config/config.txt","w", encoding="utf-8") as f:
54         for line in parameters:
55             f.write(line + "\n")
56 
57 def addsection():
58     '''Adds a new section into the dictionary'''
59     number = len(sections)
60     name = 'mainrib_' + str(number) #Name of section is
       autogenerated
61     sections[name] = section() #The section is entered in the
       dictionary
62     sections[name].name = name #Section name is set into de
       object
63 
64     if len(sections)-1 > 0:
65         previous = sections['mainrib_' + str(len(sections)-2)]
66 
67         sections[name].airfoil_name = previous.airfoil_name
68 
69         sections[name].x = previous.x #x coordinates of previous
```

```
            airfoil
70          sections[name].y = previous.y #y coordinates of previous
               airfoil
71          sections[name].camber_x = previous.camber_x #x
               coordinates of previous camber line
72          sections[name].camber_y = previous.camber_y #y
               coordinates of previous camber line
73
74          #Previous section parameters are implemented into the new
                section
75          sections[name].x_offset = previous.x_abs
76          sections[name].y_offset = previous.y_abs
77          sections[name].z_offset = previous.z_abs
78          sections[name].chord = previous.chord
79          sections[name].arrow = previous.arrow
80          sections[name].diedro = previous.diedro
81
82      sections[name].compute()
83
84  def updateposition():
85      for name in sections:
86          if name != 'mainrib_0':
87              previndex = int(name[name.find('_')+1:])-1
88              previous = sections['mainrib_' + str(previndex)]
89
90              sections[name].x_offset = previous.x_abs
91              sections[name].y_offset = previous.y_abs
92              sections[name].z_offset = previous.z_abs
93
94          sections[name].compute()
95
96  def deletesection():
97      name = 'mainrib_' + str(len(sections)-1)
98      del(sections[name])
99
100 #-----ROOT
101 class root(tk.Tk):
102
```

```
103     def __init__(self, *args, **kwargs):
104         tk.Tk.__init__(self, *args, **kwargs)   #Main frame is
               created
105         tk.Tk.iconbitmap(self, default='icons/app_icon_70.ico') #
               App icon
106         tk.Tk.title(self, "Parametric Wing Designer")   #App name
107 
108         self.ajustePantalla()
109         tk.Tk.resizable(self , True,True)   #Screen readjustment
               allowed
110 
111         global dictionary, airfoilExportFormat, wingExportFormat
112         dictionary, airfoilExportFormat, wingExportFormat =
               load_config()
113 
114         #Frame configuration
115         tk.Tk.rowconfigure(self, 0, weight=1)   #The root is
               configured to be 1x1 cell and frames are expanded
116         #tk.Tk.columnconfigure(self, 0, weight=2)
117         tk.Tk.columnconfigure(self, 1, weight=1)
118 
119         #----- Frames creation
120         self.menu = menuFrame(self)
121         self.menu.grid(row=0, column=0, sticky='nwes', padx=0)
122 
123         self.frames = {} #Dictionary containing frames
124 
125         self.welcomeFrame = WelcomePage(self)
126         self.welcomeFrame.grid(row=0, column=0, sticky="nsew",
               columnspan=2)
127 
128         for F in (WingPage, AirfoilPage, SettingsPage):
129             frame = F(self)             #Frame instance is
                   created
130             self.frames[F] = frame      #Instance is added to the
                    dictionary
131             frame.grid(row=0, column=1, sticky="nsew")
132             self.welcomeFrame.tkraise()
```

```
133
134     def show_frame(self, cont):
135         frame =self.frames[cont] #The frame to bring to the front
                is selected from the dictionary
136         frame.tkraise()
137
138     def ajustePantalla(self):
139         global graph_resolution
140         self.state('zoomed')
141         if int(self.winfo_screenheight()) <= 800:
142             graph_resolution = 70
143         elif int(self.winfo_screenheight()) > 800:
144             graph_resolution = 100
145
146 class menuFrame(tk.Frame):
147     def __init__(self, parent):
148
149         tk.Frame.__init__(self, parent)
150         tk.Frame.columnconfigure(self, 0, weight=1)
151         tk.Frame.rowconfigure(self, 0, weight=5)
152         tk.Frame.rowconfigure(self, 1, weight=80)
153         tk.Frame.rowconfigure(self, 2, weight=15)
154
155         ############################################################
156         self.headerFrame = tk.Frame(self)
157         self.headerFrame.grid(row=0, column=0, sticky="nsew")
158
159         self.iconImage = tk.PhotoImage(file='icons/app_icon_70.
              png')
160         self.iconLabel = tk.Label(self.headerFrame, image=self.
              iconImage , activebackground="white")
161         self.iconLabel.grid(row=0, column=0, sticky="ns", padx
              =10, pady=25)
162
163         self.title = tk.Label(self.headerFrame, text='Parametric\
              nWing designer', justify='left', fg="black", font=('
              Microsoft YaHei Light',20))
164         self.title.grid(row=0, column=1, sticky="nsw", padx=10,
```

```
            pady=25)
165
166         self.sepaImage = tk.PhotoImage(file='icons/separator.png'
                )
167         self.sepaLabel = tk.Label(self.headerFrame, image=self.
                sepaImage , activebackground="white")
168         self.sepaLabel.grid(row=2, column=0, sticky="ew",
                columnspan=2)
169
170         ############################################################
171         self.buttonFrame = tk.Frame(self)
172         self.buttonFrame.columnconfigure(0, weight=1)
173         self.buttonFrame.grid(row=1, column=0, sticky="nsew")
174
175         self.wingButton = tk.Button(self.buttonFrame, text=
                dictionary["menu_wing"], border=0, height="2",font=('
                Microsoft YaHei Light',14), command =lambda: self.
                changeFrame(parent, "wingButton"))
176         self.wingButton.grid(row=2, column=0,  sticky='WE')
177
178         self.airfoilButton = tk.Button(self.buttonFrame, text=
                dictionary["menu_airfoil"], border=0,  bg='white',
                height="2",font=('Microsoft YaHei Light',14), command
                =lambda: self.changeFrame(parent, "airfoilButton"))
179         self.airfoilButton.grid(row=1, column=0, sticky='WE')


180
181
182         ############################################################
183         self.configFrame = tk.Frame(self)
184         self.configFrame.columnconfigure(0, weight=1)
185         self.configFrame.columnconfigure(1, weight=1)
186         self.configFrame.grid(row=2, column=0, sticky="new")
187
188         self.sepaLabel1 = tk.Label(self.configFrame, image=self.
                sepaImage , activebackground="white")
189         self.sepaLabel1.grid(row=0, column=0, columnspan=2,
                sticky="ew")
```

```
190
191         self.gearImage = tk.PhotoImage(file='icons/settings.png')
192         self.gearButton = tk.Button(self.configFrame, image=self.
                gearImage , border=0, command =lambda: self.
                changeFrame(parent, "configButton"))
193         self.gearButton.grid(row=1, column=0, sticky="nse", padx
                =10)
194
195         self.configButton = tk.Button(self.configFrame, text=
                dictionary["name_config"], border=0, height="2",font=(
                'Microsoft YaHei Light',14), anchor="w", command =
                lambda: self.changeFrame(parent, "configButton"))
196         self.configButton.grid(row=1, column=1, sticky='WE')
197
198         ############################################################
199         self.creditsFrame = tk.Frame(self)
200         self.creditsFrame.columnconfigure(0, weight=1)
201         self.creditsFrame.grid(row=3, column=0, sticky="nsew")
202
203         self.sepaLabel2 = tk.Label(self.creditsFrame, image=self.
                sepaImage , activebackground="white")
204         self.sepaLabel2.grid(row=0, column=0, sticky="ew")
205
206         self.creditsLabel = tk.Label(self.creditsFrame, text="\
                nDeveloped by Ricardo Garcia Garre\nEmail:
                ricardogarciagarre@gmail.com\nVersion: Alpha\n",
                justify='left', fg="black", font=('Microsoft YaHei
                Light',8), activebackground="white")
207         self.creditsLabel.grid(row=1, column=0, sticky="ew",
                columnspan=2)
208
209     def changeFrame(self,parent,id):
210
211         if id == "wingButton":
212             parent.show_frame(WingPage)
213             self.wingButton.config(bg="white")
214             self.airfoilButton.config(bg="#F0F0F0")
215         elif id == "airfoilButton":
```

```
216             parent.show_frame(AirfoilPage)
217             self.wingButton.config(bg="#F0F0F0")
218             self.airfoilButton.config(bg="white")
219         else:
220             parent.show_frame(SettingsPage)
221             self.wingButton.config(bg="#F0F0F0")
222             self.airfoilButton.config(bg="#F0F0F0")
223
224 class WelcomePage(tk.Frame):
225
226     def __init__(self, parent):
227         tk.Frame.__init__(self, parent)
228         tk.Frame.config(self, bg="white")
229
230         #Column configure
231         tk.Frame.columnconfigure(self, 0, weight=1)
232
233         #Rowconfigurations
234         tk.Frame.rowconfigure(self,0, weight=2)
235         tk.Frame.rowconfigure(self,1, weight=4)
236         tk.Frame.rowconfigure(self,2, weight=1)
237
238         ########################################
239         self.headerFrame = tk.Frame(self, bg="white")
240         self.headerFrame.grid(row=0, column=0, sticky="nsew")
241         self.headerFrame.columnconfigure( 0, weight=1)
242         self.headerFrame.columnconfigure( 1, weight=1)
243         self.headerFrame.rowconfigure( 0, weight=1)
244
245         self.iconImage = tk.PhotoImage(file='icons/app_icon_110.
               png')
246         self.iconLabel = tk.Label(self.headerFrame, image=self.
               iconImage , bg="white")
247         self.iconLabel.grid(row=0, column=0, sticky="nse", padx
               =15)
248
249         self.title = tk.Label(self.headerFrame, text='Parametric\
               nWing designer', bg="white", justify='left', fg="black
```

```
            ", font=('Microsoft YaHei Light',30))
250         self.title.grid(row=0, column=1, sticky="nsw", padx=15)
251
252         self.barImage = tk.PhotoImage(file='icons/long_separator.
               png')
253         self.barLabel = tk.Label(self.headerFrame, image=self.
               barImage , bg="white")
254         self.barLabel.grid(row=2, column=0, sticky="ew", padx=15,
                columnspan=2)
255
256         ########################################
257         self.buttonFrame = tk.Frame(self, bg="white")
258         self.buttonFrame.grid(row=1, column=0, sticky="nsew",
               pady=15)
259         self.buttonFrame.columnconfigure( 0, weight=1)
260         self.buttonFrame.columnconfigure( 1, weight=1)
261
262         self.message = tk.Label(self.buttonFrame, text=dictionary
               ["message_welcome"], bg="white", justify='left', fg="
               black", font=('Microsoft YaHei Light',15))
263         self.message.grid(row=0, column=0, sticky="ew", padx=15,
               columnspan=2, pady=20)
264
265         airfoilButton = tk.Button(self.buttonFrame, text=
               dictionary["menu_airfoil"], bg='#CCDDDD' ,border=0,
               width="15",font=('Microsoft YaHei Light',14), command=
                lambda: self.changeFrame(parent, "airfoilButton"))
266         airfoilButton.grid(row=1, column=0, sticky='ne', padx=10,
                pady=20)
267
268         useButton = tk.Button(self.buttonFrame, text=dictionary["
               menu_wing"], bg='#CCDDDD', border=0, width="15", font
               =('Microsoft YaHei Light',14), command=lambda: self.
               changeFrame(parent, "wingButton"))
269         useButton.grid(row=1, column=1, sticky='nw', padx=10,
               pady=20) # bg='#b2b2ff'
270
271         self.gearImage = tk.PhotoImage(file='icons/settings.png')
```

```
272         self.gearButton = tk.Button(self.buttonFrame, image=self.
                gearImage, border=0, width=80, height=38, command=
                lambda: self.changeFrame(parent, "configButton")) #,
                bg="lightgrey"
273         self.gearButton.grid(row=2, column=0, sticky="n", padx
                =10, columnspan=2)
274
275         ######################################
276         self.spamFrame = tk.Frame(self, bg="white")
277         self.spamFrame.grid(row=2, column=0, sticky="nsew")
278         self.spamFrame.columnconfigure( 0, weight=1)
279         #self.spamFrame.columnconfigure( 1, weight=1)
280         self.spamFrame.columnconfigure( 2, weight=1)
281
282         self.upvImage = tk.PhotoImage(file='icons/UPV.png')
283         self.upvLabel = tk.Label(self.spamFrame, image=self.
                upvImage , bg="white")
284         self.upvLabel.grid(row=0, column=0, sticky="ne", padx=20)
285
286         self.etsidImage = tk.PhotoImage(file='icons/ETSID.png')
287         self.etsidLabel = tk.Label(self.spamFrame, image=self.
                etsidImage , bg="white")
288         self.etsidLabel.grid(row=0, column=1, sticky="n", padx
                =20)
289
290         self.creditImage = tk.PhotoImage(file='icons/credit.png')
291         self.creditLabel = tk.Label(self.spamFrame, image=self.
                creditImage , bg="white")
292         self.creditLabel.grid(row=0, column=2, sticky="nw", padx
                =20)
293
294     def changeFrame(self,parent,id):
295         parent.menu.changeFrame(parent, id)
296         self.destroy()
297
298 class AirfoilPage(tk.Frame):
299
300     def __init__(self, parent):
```

```
301         tk.Frame.__init__(self, parent)
302         tk.Frame.config(self, bg="white")
303 
304         tk.Frame.columnconfigure(self, 0, weight=1)
305         #tk.Frame.rowconfigure(self,5,weight=1)
306         self.parent = parent
307         self.airfoil_name = 'NACA 0012'
308         self.name = 'mainrib_0'
309 
310         self.pageTitle = tk.Label(self, text=dictionary["
               title_airfoil"], anchor="w", bg='white',font=('
               Microsoft YaHei Light',30))
311         self.pageTitle.grid(row=0, column=0, padx=30, pady=10,
               sticky='ew' )
312 
313         self.title = tk.Label(self, text=dictionary["name_airfoil
               "]+ " " + self.airfoil_name + "  -  " +  "80" +
               dictionary["name_points"] + " ", bg='white',font=('
               Microsoft YaHei Light',18))
314         self.title.grid(row=1, column=0, padx=30, sticky='ew' )
315 
316         self.separatorImage = tk.PhotoImage(file='icons/
               long_separator.png')
317         self.separatorLabel1 = tk.Label(self, image=self.
               separatorImage , bg="white", activebackground="white")
318         self.separatorLabel1.grid(row=2, column=0, sticky="ns",
               padx=10)
319 
320         self.widgets = tk.Frame(self, bg="white")
321         self.widgets.grid(row=5, column=0, pady=50,  padx=30,
               sticky='ew' )
322         self.widgets.columnconfigure(1, weight=1)
323 
324         self.separatorLabel2 = tk.Label(self, image=self.
               separatorImage , bg="white", activebackground="white")
325         self.separatorLabel2.grid(row=4, column=0, sticky="ns",
               padx=10)
326 
```

```
327         self.slider1 = slider(self.widgets, dictionary["
               parameter_max_camber"]+' [%]', 0.0, 0, 15, dec=1,
               updater=self)
328         self.slider1.grid(row=0,column=1,   )
329 
330         self.slider2 = slider(self.widgets, dictionary["
               parameter_max_camber_position"]+' [%10]', 4,3, 7, dec
               =1, updater=self)
331         self.slider2.grid(row=1,column=1, pady=3,  )
332 
333         self.slider3 = slider(self.widgets, dictionary["
               parameter_max_thickness"]+' [%]', 12, 1, 40, dec=0,
               updater=self)
334         self.slider3.grid(row=2,column=1, pady=3,  )
335 
336         self.slider4 = slider(self.widgets, dictionary["
               parameter_number_points"], 80, 20, 140, dec=0, updater
               =self) # 80 20
337         self.slider4.grid(row=3,column=1,  )
338 
339         self.importButton = tk.Button(self.widgets, text=
               dictionary["button_import_airfoil"], border=0, width="
               15",font=('Microsoft YaHei Light',14), command=lambda:
                self.import_airfoil())
340         self.importButton.grid(row=0, column=0,   sticky='ne')
341 
342         self.useButton = tk.Button(self.widgets, text=dictionary[
               "button_use_airfoil"], border=0, width="15", font=('
               Microsoft YaHei Light',14), command=lambda: self.
               update_airfoil())
343         self.useButton.grid(row=1, column=0,   sticky='ne') # bg
               ='#b2b2ff'
344 
345         self.exportButton = tk.Button(self.widgets, text=
               dictionary["button_export_airfoil"], border=0, width="
               15", bg='lightblue',font=('Microsoft YaHei Light',14),
                command=lambda: self.export_airfoil())
346         self.exportButton.grid(row=2, column=0,   sticky='ne')
```

```
347
348         self.m = self.slider1.val
349         self.p = self.slider2.val
350         self.t = int(self.slider3.val)
351         self.n = int(self.slider4.val/2)
352         self.x_airfoil, self.y_airfoil, self.x_camber, self.
                y_camber =create_naca(self.m, self.p, self.t, self.n)
353
354         #Init plot
355         self.f =Figure(figsize=(13,4), dpi=graph_resolution)
356         self.a = self.f.add_subplot(111)
357         self.a.axis('equal')
358         self.a.spines['right'].set_visible(False)
359         self.a.spines['top'].set_visible(False)
360         self.a.spines['bottom'].set_visible(False)
361         self.a.axhline(y=0, color='red', linewidth=.5)
362         self.a.plot(self.x_airfoil, self.y_airfoil, linewidth=1)
363         self.a.plot(self.x_camber, self.y_camber, linewidth=1)
364
365         self.canvas= FigureCanvasTkAgg(self.f, self)
366         self.canvas.draw()
367         self.canvas.get_tk_widget().grid(row=3,column=0,
                columnspan=3)
368
369         self.scrollframe = ScrollFrame(self.widgets)
370         if graph_resolution < 100:
371             self.scrollframe.canvas.config(width=250)
372         self.scrollframe.viewPort.columnconfigure(0, weight=1)
373         self.scrollframe.grid(row=0, column=2,  sticky='ne',
                rowspan=5)
374
375         self.button={}
376     #FUNCTIONS-----------------------
377     def updateNew(self):
378         self.a.clear()
379         self.a.axhline(y=0, color='k', linewidth=.5)
380         self.a.plot(self.x_airfoil, self.y_airfoil, linewidth=1)
381         self.a.plot(self.x_camber, self.y_camber, linewidth=1)
```

```
382         self.canvas.draw()
383 
384 
385     def update(self,*args):
386         self.m = self.slider1.val
387         self.p = self.slider2.val
388         self.t = int(self.slider3.val)
389         self.n = int(self.slider4.val/2)
390         self.x_airfoil, self.y_airfoil, self.x_camber, self.
               y_camber =create_naca(self.m, self.p, self.t, self.n)
391         self.a.clear()
392         self.a.axhline(y=0, color='k', linewidth=.5)
393         self.a.plot(self.x_airfoil, self.y_airfoil, linewidth=1)
394         self.a.plot(self.x_camber, self.y_camber, linewidth=1)
395 
396         self.canvas.draw()
397 
398         if self.m == 0:
399             self.airfoil_name = 'NACA' + ' ' +str(0) + ' ' + str
                   (0) + ' '
400         else:
401             self.airfoil_name = 'NACA' + ' ' +str(self.m) + ' ' +
                    str(self.p) + ' '
402 
403         if self.t < 10:
404             self.airfoil_name = self.airfoil_name + '0' + str(
                   self.t)
405         else:
406             self.airfoil_name = self.airfoil_name +  str(self.t)
407 
408         self.title.config(text=dictionary["name_airfoil"] + " " +
                self.airfoil_name + "  -  " + str(2*self.n) + " " +
               dictionary["name_points"])
409 
410     def addButton(self):
411         dic_id = str(len(sections)-1) #Last dictionary entry is
               selected
412         name = 'mainrib_' + dic_id
```

```
413         title = dictionary["name_section"] + " " + dic_id + '\n'
                + dictionary["name_airfoil"] + ': ' + sections[name].
                airfoil_name#self.airfoil_name
414
415         self.button[name] = tk.Button(self.scrollframe.viewPort,
                text=title, border=0, bg='#e5e5ff',font=('Microsoft
                YaHei Light',14), command = lambda: self.
                change_section(name))
416         self.button[name].grid(row=len(sections)-1, column=0,
                pady=1, sticky='ew')
417
418     def deleteButton(self):
419         name = 'mainrib_' + str(len(sections))
420         self.button[name].destroy()
421         del(self.button[name])
422
423     def change_section(self, name):
424         self.name = name
425         self.x_airfoil = sections[name].x
426         self.y_airfoil = sections[name].y
427         self.x_camber = sections[name].camber_x
428         self.y_camber = sections[name].camber_y
429
430         self.button[name].config(bg='#ffffcc')
431         self.updateNew()
432         self.title.config(text=dictionary["name_airfoil"] + " " +
                sections[self.name].airfoil_name + "  -  " + str(len(
                sections[self.name].x)) + " " + dictionary["
                name_points"])
433
434         for button in self.button:
435             if button != name:
436                 self.button[button].config(bg='#e5e5ff')
437
438     def update_airfoil(self):
439         try:
440             sections[self.name].x = self.x_airfoil
441             sections[self.name].y = self.y_airfoil
```

```
442             sections[self.name].camber_x = self.x_camber
443             sections[self.name].camber_y = self.y_camber
444             sections[self.name].airfoil_name = self.airfoil_name
445             sections[self.name].compute()
446 
447             index = self.name[self.name.find('_')+1:]
448             self.button[self.name].config(text= dictionary["
                    name_section"] + " " + index + '\n' + dictionary["
                    name_airfoil"] + ": " + self.airfoil_name)
449 
450             if self.name == "mainrib_"+str(len(sections)-1):
451                 self.parent.frames[WingPage].notepar.
                        generate_elipWing()
452 
453 
454             self.parent.frames[WingPage].noteplot.updateplots()
455         except:
456             print('No se ha podido actualizar el perfil')
457 
458     def import_airfoil(self):
459         try:
460             [airfoilname, x, y] = load_airfoil()
461             self.x_airfoil = x
462             self.y_airfoil = y
463             self.x_camber = []
464             self.y_camber = []
465             self.airfoil_name = airfoilname
466             self.title.config(text=dictionary["name_airfoil"] + "
                     " + airfoilname + "  -  " + str(len(x)) + " " +
                    dictionary["name_points"])
467 
468             self.updateNew()
469         except:
470             pass
471 
472     def export_airfoil(self):
473         save_airfoil(sections[self.name], airfoilExportFormat)
474 
```

```
475 class WingPage(tk.Frame):
476
477     def __init__(self, parent):
478         tk.Frame.__init__(self, parent)
479         tk.Frame.config(self, bg="white")
480
481         self.parent = parent
482
483         tk.Frame.columnconfigure(self, 1, weight=1)
484         tk.Frame.rowconfigure(self, 4, weight=1)
485
486         self.pageTitle = tk.Label(self, text=dictionary["
               title_wing"], anchor="w", bg='white',font=('Microsoft
               YaHei Light',30))
487         self.pageTitle.grid(row=0, column=0, padx=30, pady=10,
               sticky='ew', columnspan=3 )
488
489         #Notebook containig plots
490         self.noteplot=PlotNotebook(self)
491         self.noteplot.grid(row=1, column=0, padx=25, pady=10,
               sticky='nsew', columnspan=2)
492
493         self.scrollframe = ScrollFrame(self)
494         self.scrollframe.grid(row=2, column=1, padx=25, pady=10,
               sticky='nsew', rowspan=3)
495         self.scrollframe.viewPort.columnconfigure( 0, weight=1)
496         self.scrollframe.viewPort.rowconfigure(0, weight=1)
497
498         #Notebook containing parametros
499         self.notepar=ParameterNotebook(self.scrollframe.viewPort,
                self)
500         self.notepar.grid(row=1, column=0, padx=25, pady=0,
               sticky='nsew')
501
502         #Add button
503         addButton = tk.Button(self, text=dictionary["
               button_add_section"], border=0, width="15", bg='#
               ccffcc',font=('Microsoft YaHei Light',14), command=
```

```
            lambda: self.add())
504         addButton.grid(row=2, column=0, pady=10, padx=25, sticky=
            'nw')
505
506         #Delete button
507         delButton = tk.Button(self, text=dictionary["
            button_delete_section"], border=0, width="15", bg='#
            ffb2b2',font=('Microsoft YaHei Light',14), command=
            lambda: self.delete())
508         delButton.grid(row=3, column=0, pady=10, padx=25, sticky=
            'nw')
509
510         #Export button
511         expButton = tk.Button(self, text=dictionary["
            button_export_wing"], border=0, width="15", bg='
            lightblue',font=('Microsoft YaHei Light',14), command=
            lambda: self.export())
512         expButton.grid(row=4, column=0, pady=10, padx=25, sticky=
            'nw')
513
514
515     def add(self, *args):
516         addsection()
517         self.notepar.addslider()
518         self.noteplot.updateplots()
519         self.parent.frames[AirfoilPage].addButton()
520
521     def delete(self):
522         if len(sections) > 1: #Rib 0 cannot be deleted
523             deletesection()
524             self.notepar.deleteslider()
525             self.noteplot.updateplots()
526             self.parent.frames[AirfoilPage].deleteButton()
527
528     def export(self):
529         export_wing((sections, sectionsElip), wingExportFormat)
530
531 #Notebooks
```

```
532 class ScrollFrame(tk.Frame):
533     def __init__(self, parent):
534         super().__init__(parent) # create a frame (self)
535 
536         self.canvas = tk.Canvas(self, borderwidth=0, background="
               white", highlightthickness=0)  #place canvas on self #
               ffffff
537         self.viewPort = tk.Frame(self.canvas, background="white",
                borderwidth=0)    #place a frame on the canvas, this
               frame will hold the child widgets
538         self.vsb = tk.Scrollbar(self, orient="vertical", command=
               self.canvas.yview) #place a scrollbar on self
539         self.canvas.configure(yscrollcommand=self.vsb.set)  #
               attach scrollbar action to scroll of canvas
540 
541         self.vsb.pack(side="right", fill="y")   #pack scrollbar
               to right of self
542         self.canvas.pack(side="left", fill="both", expand=True) #
               pack canvas to left of self and expand to fil
543         self.canvas_window = self.canvas.create_window((4,4),
               window=self.viewPort, anchor="nw",    #add view port
               frame to canvas
544                                   tags="self.viewPort")
545 
546         self.viewPort.bind("<Configure>", self.onFrameConfigure)
                   #bind an event whenever the size of the viewPort
               frame changes.
547         self.canvas.bind("<Configure>", self.onCanvasConfigure) #
               bind an event whenever the size of the viewPort frame
               changes.
548 
549         self.onFrameConfigure(None) #perform an initial stretch
               on render, otherwise the scroll region has a tiny
               border until the first resize
550 
551     def onFrameConfigure(self, event):
552         '''Reset the scroll region to encompass the inner frame
               '''
```

```
553         self.canvas.configure(scrollregion=self.canvas.bbox("all"
                ")) #whenever the size of the frame changes, alter the
                scroll region respectively.
554
555     def onCanvasConfigure(self, event):
556         '''Reset the canvas window to encompass inner frame when
                required'''
557         canvas_width = event.width
558         self.canvas.itemconfig(self.canvas_window, width =
                canvas_width)    #whenever the size of the canvas
                changes alter the window region respectively.
559
560 class ParameterNotebook(ttk.Notebook):
561
562     def __init__(self, parent, superparent):
563         self.superparent = superparent
564         #Notebook configuration
                ------------------------------------------#
565         ttk.Notebook.__init__(self, parent)
566         self.style = ttk.Style()
567         self.style.configure('TNotebook.Tab', font=("Microsoft
                YaHei Light",'12'), size=25, padding=5)
568         self.style.configure('TNotebook', background='white')
569         self.style.layout("Tab",[('Notebook.tab', {'sticky': '
                nswe', 'children': [('Notebook.padding', {'side': 'top
                ', 'sticky': 'nswe', 'children':[('Notebook.label', {'
                side': 'top', 'sticky': ''})],})],})])
570
571         #Paramteter (span)
                ----------------------------------------------------#
572         self.tab1 = tk.Frame(self) #new frame for tab 1
573         self.tab1.config(bg='white', padx=30, pady=30)
574
575         #Paramteter (chord)
                ----------------------------------------------------#
576         self.tab2 = tk.Frame(self) #new frame for tab 2
577         self.tab2.config(bg='white', padx=30, pady=30)
578
```

```
579         #Paramteter (arrow)
               ----------------------------------------------------#
580         self.tab3 = tk.Frame(self) #new frame for tab 3
581         self.tab3.config(bg='white', padx=30, pady=30)
582
583         #Paramteter (dihedral)
               ----------------------------------------------------#
584         self.tab4 = tk.Frame(self) #new frame for tab 4
585         self.tab4.config(bg='white', padx=30, pady=30)
586
587         #Paramteter (AoA)
               ----------------------------------------------------#
588         self.tab5 = tk.Frame(self) #new frame for tab 5
589         self.tab5.config(bg='white', padx=30, pady=30)
590
591         #Paramteter (ellipsoid)
               ----------------------------------------------------#
592         self.tab6 = tk.Frame(self) #new frame for tab 6
593         self.tab6.config(bg='white', padx=30, pady=30)
594         self.spamElip = slider(self.tab6, title=dictionary["
               parameter_span"], start=0, lower=0, upper=20, dec=2,
               updater=self) #Ellipsoid span
595         self.spamElip.grid(row=0, column=0, pady=10)
596         self.diedroElip = slider(self.tab6, title=dictionary["
               parameter_dihedral"], start=0, lower=-15, upper=15,
               dec=2, updater=self) #Dihedral angle
597         self.diedroElip.grid(row=1, column=0, pady=10)
598         self.tipElip = slider(self.tab6, title=dictionary["
               parameter_elip_tip_position"], start=50, lower=0,
               upper=100, dec=0, updater=self) #Tip position
599         self.tipElip.grid(row=2, column=0, pady=10)
600         self.numberElip = slider(self.tab6, title=dictionary["
               parameter_elip_section_number"], start=7, lower=5,
               upper=30, dec=0, updater=self) #Sections number
601         self.numberElip.grid(row=3, column=0, pady=10)
602         self.arrowElip = slider(self.tab6, title=dictionary["
               parameter_arrow"], start=0, lower=-85, upper=85, dec
               =2, updater=self) #arrow angle
```

```
603         self.arrowElip.grid(row=4, column=0, pady=10)
604 
605 
606         #Add tabs
607         self.add(self.tab1, text=dictionary["parameter_span"] + "
                "")
608         self.add(self.tab2, text=dictionary["parameter_chord"] +
               "    ")
609         self.add(self.tab3, text=dictionary["parameter_arrow"] +
               "    ")
610         self.add(self.tab4, text=dictionary["parameter_dihedral"]
                + "    ")
611         self.add(self.tab5, text=dictionary["parameter_aoa"] + "
                "")
612         self.add(self.tab6, text=dictionary["parameter_elip"] + "
                "")
613 
614         #Dictionaries containing parameters
615         self.span = {} #Section span
616         self.chord = {} #Section chord
617         self.arrow = {} #Arrow angle
618         self.dihedral = {} #Dihedral angle
619         self.aoa = {} #Angle of attack
620 
621     def addslider(self):
622         dic_id = str(len(sections)-1) #Last dictionary entry is
               selected
623         name = 'mainrib_' + dic_id
624         title = dictionary["name_section"] + " " + dic_id
625 
626         self.span[name] = slider(self.tab1, title=title, start=0,
                lower=0, upper=20, dec=2, updater=self) #Span
627         self.span[name].grid(row=len(sections)-1, column=0, pady
               =10)
628 
629         self.chord[name] = slider(self.tab2, title=title, start=
               sections[name].chord, lower=0.1, upper=10, dec=2,
               updater=self) #Chord
```

```
630         self.chord[name].grid(row=len(sections)-1, column=0, pady
                =10)
631 
632         if dic_id != '0':
633             self.arrow[name] = slider(self.tab3, title=title,
                    start=sections[name].arrow, lower=-85, upper=85,
                    dec=2, updater=self) #Arrow angle
634             self.arrow[name].grid(row=len(sections)-1, column=0,
                    pady=10)
635 
636             self.dihedral[name] = slider(self.tab4, title=title,
                    start=sections[name].diedro, lower=-15, upper=15,
                    dec=2, updater=self) #Dihedral angle
637             self.dihedral[name].grid(row=len(sections)-1, column
                    =0, pady=10)
638 
639         self.aoa[name] = slider(self.tab5, title=title, start=0,
                lower=0, upper=15, dec=1, updater=self)
640         self.aoa[name].grid(row=len(sections)-1, column=0, pady
                =10)
641 
642     def deleteslider(self):
643 
644         name = 'mainrib_' + str(len(sections))
645         self.span[name].destroy()
646         self.chord[name].destroy()
647         self.arrow[name].destroy()
648         self.dihedral[name].destroy()
649         self.aoa[name].destroy()
650 
651         self.generate_elipWing()
652 
653     def update(self):
654         '''This function is called by the sliders everytime they
                get updated'''
655         currenttab = self.index("current")
656 
657         if currenttab == 0: #Affects the wingspan values of all
```

```
            sections
658             for name in sections:
659                 sections[name].span = self.span[name].val
660 
661         elif currenttab == 1: #Affects the chord values of all
                sections
662             for name in sections:
663                 sections[name].chord = self.chord[name].val
664 
665         elif currenttab == 2: #Affects the arrow values of all
                sections
666             for name in sections:
667                 if name != 'mainrib_0':
668                     sections[name].arrow = self.arrow[name].val
669 
670         elif currenttab == 3: #Affects the dihedral values of all
                sections
671             for name in sections:
672                 if name != 'mainrib_0':
673                     sections[name].diedro = self.dihedral[name].
                        val
674 
675         elif currenttab == 4: #Affects the aoa values of all
                sections
676             for name in sections:
677                 sections[name].alpha = self.aoa[name].val
678 
679         #All points are recalculated
680         updateposition()
681 
682         self.generate_elipWing()
683         self.superparent.noteplot.updateplots() #Plots are
                updated
684 
685 
686     def generate_elipWing(self):
687         global sectionsElip
688 
```

```
689         if float(self.spamElip.val) != 0 and int(self.numberElip.
               val) != 0:
690             previousSection = sections["mainrib_" + str(len(
                   sections)-1)]
691             span = float(self.spamElip.val)
692             pointNumber = int(self.numberElip.val)
693             displacement = int(self.tipElip.val)
694             diedro = float(self.diedroElip.val)
695             arrow = float(self.arrowElip.val)
696             sectionsElip = create_eliptical_wing(previousSection,
                    span, pointNumber, displacement, diedro, arrow)
697 
698         else:
699             sectionsElip = {}
700 
701 class PlotNotebook(ttk.Notebook):
702 
703     def __init__(self, parent):
704         ttk.Notebook.__init__(self, parent)
705         self.style = ttk.Style()
706         self.style.configure('TNotebook.Tab', font=("Microsoft
               YaHei Light",'12'), size=25, padding=5)
707         self.style.configure('TNotebook', background='white')
708         self.style.layout("Tab",[('Notebook.tab', {'sticky': '
               nswe', 'children': [('Notebook.padding', {'side': 'top
               ', 'sticky': 'nswe', 'children':[('Notebook.label', {'
               side': 'top', 'sticky': ''})],})],})])
709 
710         self.bind("<<NotebookTabChanged>>", self.on_visibility) #
               Detects tab change
711 
712         self.color_perfil = 'orange'
713         self.color_leadingedge = '#31639c'
714         self.color_trailingedge = '#82c59c'
715         self.color_cuartocuerda = '#a4bfd9'
716         #Frame 1 - Plot planform---#
717         self.tab1 = tk.Frame(self) #Frame for Tab1
718         self.tab1.config(bg='white')
```

```
719
720         self.f1 =Figure(figsize=(13,4), dpi=graph_resolution)
721         self.planta = self.f1.add_subplot(111)
722         self.planta.axis('equal')
723         self.planta.spines['right'].set_visible(False)
724         self.planta.spines['top'].set_visible(False)
725         self.planta.spines['left'].set_visible(False)
726
727         self.canvasP= FigureCanvasTkAgg(self.f1, self.tab1)
728         self.canvasP.draw()
729         self.canvasP.get_tk_widget().pack()
730
731         #Frame 2 - Plot front---#
732         self.tab2 = tk.Frame(self) #Frame for Tab2
733         self.tab2.config(bg='white')
734
735         self.f2 =Figure(figsize=(13,4), dpi=graph_resolution)
736         self.alzado = self.f2.add_subplot(111)
737         self.alzado.axis('equal')
738         self.alzado.spines['right'].set_visible(False)
739         self.alzado.spines['top'].set_visible(False)
740         self.alzado.spines['left'].set_visible(False)
741
742         self.canvasA= FigureCanvasTkAgg(self.f2, self.tab2)
743         self.canvasA.draw()
744         self.canvasA.get_tk_widget().pack()
745
746         #Frame 3 - Plot section---#
747         self.tab3 = tk.Frame(self) #Frame for Tab3
748         self.tab3.config(bg='white')
749
750         self.f3 =Figure(figsize=(13,4), dpi=graph_resolution)
751         self.perfil = self.f3.add_subplot(111)
752         self.perfil.axis('equal')
753
754         self.perfil.spines['right'].set_visible(False)
755         self.perfil.spines['top'].set_visible(False)
756         self.perfil.spines['left'].set_visible(False)
```

```
757
758         self.canvasPe= FigureCanvasTkAgg(self.f3, self.tab3)
759         self.canvasPe.draw()
760         self.canvasPe.get_tk_widget().pack()
761
762         #Add tabs to notebook
763         self.add(self.tab1, text=dictionary["plot_top"]+"    ")
764         self.add(self.tab2,text=dictionary["plot_front"]+"    ")
765         self.add(self.tab3,text=dictionary["plot_side"]+"    ")
766
767     def on_visibility(self, event): #Update the graph every time
        the tab is changed to optimize resources
768         self.updateplots()
769
770     def updateplots(self):
771         currenttab = self.index("current")
772
773         if currenttab == 0:
774             self.update_planta()
775         elif currenttab == 1:
776             self.update_alzado()
777         elif currenttab == 2:
778             self.update_perfil()
779
780     def update_planta(self):
781         self.planta.clear()
782
783         old = None
784         oldup = []
785         olddown = []
786
787         #Plotting points straight wing
788         for sectionType in (sections, sectionsElip):
789             for name in sectionType:
790                 new = sectionType[name]
791
792                 newup =   [ new.z_abs , -1*new.x_abs ]
793                 newdown = [ new.z_abs , -1*(new.x_abs+new.chord*
```

```
                    cos(new.alpha_rad)) ]
794
795                 self.planta.plot(sectionType[name].zw, -1*array(
                        sectionType[name].xw), linewidth=2, color=self
                        .color_perfil) #Right sections
796                 self.planta.plot(-1*array(sectionType[name].zw),
                        -1*array(sectionType[name].xw), linewidth=2,
                        color=self.color_perfil) #Left sections
797
798                 if oldup: #Join the sections at the top
799
800                     #Leading and trailing edges with mirror
801                     self.planta.plot( [oldup[0],newup[0]] , [
                            oldup[1],newup[1]] , linewidth=2, color=
                            self.color_leadingedge) #Right leading
                            edge (blue)
802                     self.planta.plot( [olddown[0],newdown[0]] , [
                            olddown[1],newdown[1]] , linewidth=2,
                            color=self.color_trailingedge) #Right
                            trailing edge
803
804                     self.planta.plot( [-1*oldup[0],-1*newup[0]] ,
                             [oldup[1],newup[1]] , linewidth=2, color=
                            self.color_leadingedge)
805                     self.planta.plot( [-1*olddown[0],-1*newdown
                            [0]] , [olddown[1],newdown[1]] , linewidth
                            =2, color=self.color_trailingedge)
806
807                     #1/4 chord line with mirror
808                     self.planta.plot( [oldup[0],newup[0]] , [-1*(
                            old.x_abs+(0.25*old.chord*cos(old.
                            alpha_rad))),-1*(new.x_abs+(0.25*new.chord
                            *cos(new.alpha_rad)))] , linewidth=1,
                            color=self.color_cuartocuerda)
809                     self.planta.plot( [-1*oldup[0],-1*newup[0]] ,
                             [-1*(old.x_abs+(0.25*old.chord*cos(old.
                            alpha_rad))),-1*(new.x_abs+(0.25*new.chord
                            *cos(new.alpha_rad)))] , linewidth=1,
```

```
                        color=self.color_cuartocuerda)
810
811                 old = new
812                 oldup = [ new.z_abs , -1*new.x_abs ]
813                 olddown = [ new.z_abs , -1*(new.x_abs+new.chord*
                        cos(new.alpha_rad))]
814
815         #Axis adjustment
816         self.planta.autoscale(enable=True, axis='both', tight=
                False)
817         #Bars configuration
818         self.planta.axhline(y=0, color='k', linewidth=.5,
                linestyle='-.')
819         self.planta.axvline(x=0, color='k', linewidth=.5,
                linestyle='-.')
820
821         self.canvasP.draw()
822
823     def update_alzado(self):
824         self.alzado.clear()
825
826         old = None
827         oldup = []
828         olddown = []
829
830         #Points plotting
831         for sectionType in (sections, sectionsElip):
832             for name in sectionType:
833                 new = sectionType[name]
834
835                 newup =   [ new.z_abs , max(new.yw) ]
836                 newdown = [ new.z_abs , min(new.yw) ]
837
838                 self.alzado.plot(sectionType[name].zw, array(
                        sectionType[name].yw), linewidth=2, color=self
                        .color_perfil) #Right sections
839                 self.alzado.plot(-1*array(sectionType[name].zw),
                        array(sectionType[name].yw), linewidth=2,
```

```
                    color=self.color_perfil) #left sections
840
841                 if oldup: #Join the profiles at the top
842
843                     #Upper and lower limits
844                     self.alzado.plot( [oldup[0],newup[0]] , [
                            oldup[1],newup[1]] , linewidth=1, color=
                            self.color_perfil) #Upper limit
845                     self.alzado.plot( [olddown[0],newdown[0]] , [
                            olddown[1],newdown[1]] , linewidth=1,
                            color=self.color_perfil) #Lower limit
846                     self.alzado.plot( [-1*oldup[0],-1*newup[0]] ,
                             [oldup[1],newup[1]] , linewidth=1, color=
                            self.color_perfil) #Upper limit mirrow
847                     self.alzado.plot( [-1*olddown[0],-1*newdown
                            [0]] , [olddown[1],newdown[1]] , linewidth
                            =1, color=self.color_perfil) #Lower limit
                            mirror
848
849                     #Tralíling edge
850                     old_te = old.yw[where(old.xw==max(old.xw))
                            [0][0]]
851                     new_te = new.yw[where(new.xw==max(new.xw))
                            [0][0]] #Get the y coordinate at trailing
                            edge
852
853                     self.alzado.plot( [oldup[0],newup[0]] , [
                            old_te, new_te] , linewidth=1, color=self.
                            color_trailingedge, linestyle='-.') #
                            Leading edge (blue)
854                     self.alzado.plot( [-1*oldup[0],-1*newup[0]] ,
                             [old_te, new_te] , linewidth=1, color=
                            self.color_trailingedge, linestyle='-.') #
                            Mirror
855
856                     #Leading edge
857                     old_le = old.yw[where(old.xw==min(old.xw))
                            [0][0]]
```

```
858                     new_le = new.yw[where(new.xw==min(new.xw))
                            [0][0]] #Get the y coordinate at trailing
                            edge
859
860                     self.alzado.plot( [oldup[0],newup[0]] , [
                            old_le , new_le] , linewidth=1, color=self
                            .color_leadingedge)
861                     self.alzado.plot( [-1*oldup[0],-1*newup[0]] ,
                             [old_le , new_le] , linewidth=1, color=
                            self.color_leadingedge)
862
863                 old = new
864                 oldup = [ old.z_abs , max(old.yw) ]
865                 olddown = [ old.z_abs , min(old.yw) ]
866
867         #Axis adjustemt
868         self.alzado.autoscale(enable=True, axis='both', tight=
                False)
869         #Bars configuration
870         self.alzado.axhline(y=0, color='k', linewidth=.5,
                linestyle='-.')
871         self.alzado.axvline(x=0, color='k', linewidth=.5,
                linestyle='-.')
872         self.canvasA.draw()
873
874     def update_perfil(self):
875         self.perfil.clear()
876
877         diference = 0
878         sec = 0
879         partes = len(sections)+len(sectionsElip)
880         try:
881             diference = int(255/partes)
882         except:
883             pass
884
885         #Points plotting
886         for sectionType in (sections, sectionsElip):
```

```
887             for name in sectionType:
888                 rg = int(sec*diference)
889                 self.color=str('#{:02x}{:02x}{:02x}'.format( rg,
                        rg , 255))
890                 self.perfil.plot(sectionType[name].xw,
                        sectionType[name].yw, linewidth=2, color=self.
                        color) #Airfoils
891                 sec+=1
892
893         #Axis adjustement
894         self.perfil.autoscale(enable=True, axis='both', tight=
                False)
895         #Bars configuration
896         self.perfil.axhline(y=0, color='k', linewidth=.5,
                linestyle='-.')
897         self.perfil.axvline(x=0, color='k', linewidth=.5,
                linestyle='-.')
898
899         self.canvasPe.draw()
900
901 class SettingsPage(tk.Frame):
902     def __init__(self, parent):
903
904         tk.Frame.__init__(self, parent)
905         tk.Frame.config(self, bg="white")
906
907         tk.Frame.columnconfigure(self, 0, weight=1)
908         tk.Frame.rowconfigure(self,0, weight=20)
909         tk.Frame.rowconfigure(self,1, weight=80)
910
911         self.gearImage = tk.PhotoImage(file='icons/settings2.png'
                )
912         self.gearLabel = tk.Label(self, image=self.gearImage, bg=
                "white")
913         self.gearLabel.grid(row=0, column=0, sticky="nsew", pady
                =30)
914
915         self.optionFrame= tk.Frame(self, bg="white")
```

```
916         self.optionFrame.grid(row=1, column=0, sticky="n")
917 
918         #Languaje block
919         self.language = tk.Label(self.optionFrame, text=
                dictionary["name_language"], justify='left', anchor="w
                ", fg="black", bg="white", font=('Arial',20, "bold"))
920         self.language.grid(row=0, column=0, sticky="nswe", pady
                =10)
921 
922         self.languageLabel = tk.Label(self.optionFrame, text=
                dictionary["config_select_language"]+ ":", justify='
                left', anchor="w", fg="black", bg="white", font=('
                Microsoft YaHei Light',15))
923         self.languageLabel.grid(row=1, column=0, sticky="nsew",
                pady=10)
924 
925         self.availableLanguage = ["English", "Spanish"]
926         self.selectedLanguage = tk.StringVar()
927         self.selectedLanguage.set(dictionary["language"])
928         self.languageMenu = tk.OptionMenu(self.optionFrame, self.
                selectedLanguage, *self.availableLanguage)
929         self.languageMenu.config(font=('Microsoft YaHei Light'
                ,15), width=10, borderwidth=0, bg="white",
                highlightthickness=0)
930         self.languageMenu.grid(row=1, column=1, padx=10, sticky="
                nswe", pady=10)
931 
932         self.sepaImage = tk.PhotoImage(file='icons/separator.png'
                )
933         self.sepaLabel = tk.Label(self.optionFrame, image=self.
                sepaImage ,bg="white", activebackground="white")
934         self.sepaLabel.grid(row=2, column=0, sticky="ew",
                columnspan=2, pady=10)
935 
936         #Export block
937         self.exportLabel = tk.Label(self.optionFrame, text=
                dictionary["name_export"], justify='left', anchor="w",
                 fg="black", bg="white", font=('Arial',20, "bold"))
```

```
938         self.exportLabel.grid(row=3, column=0, sticky="nswe",
                pady=10)
939
940             #Airfoil exporting
941         self.exportAirfoilLabel = tk.Label(self.optionFrame, text
                =dictionary["config_select_airfoil"]+ ":", justify='
                left', anchor="w", fg="black", bg="white", font=('
                Microsoft YaHei Light',15))
942         self.exportAirfoilLabel.grid(row=4, column=0, sticky="
                nsew", pady=10)
943
944         self.availableFormats = [".txt", ".dat", ".csv"]
945         self.selectedAirfoilFormat = tk.StringVar()
946         self.selectedAirfoilFormat.set(airfoilExportFormat)
947         self.airfoilFormatMenu = tk.OptionMenu(self.optionFrame,
                self.selectedAirfoilFormat, *self.availableFormats)
948         self.airfoilFormatMenu.config(font=('Microsoft YaHei
                Light',15), width=10, borderwidth=0, bg="white",
                highlightthickness=0)
949         self.airfoilFormatMenu.grid(row=4, column=1, padx=10,
                sticky="nswe", pady=10)
950
951             #Wing exporting
952         self.exportWingLabel = tk.Label(self.optionFrame, text=
                dictionary["config_select_wing"]+ ":", justify='left',
                 anchor="w", fg="black", bg="white", font=('Microsoft
                YaHei Light',15))
953         self.exportWingLabel.grid(row=5, column=0, sticky="nsew")
954
955         self.selectedWingFormat = tk.StringVar()
956         self.selectedWingFormat.set(wingExportFormat)
957         self.wingFormatMenu = tk.OptionMenu(self.optionFrame,
                self.selectedWingFormat, *self.availableFormats)
958         self.wingFormatMenu.config(font=('Microsoft YaHei Light'
                ,15), width=10, borderwidth=0, bg="white",
                highlightthickness=0)
959         self.wingFormatMenu.grid(row=5, column=1, padx=10, sticky
                ="nswe")
```

```
960
961         #self.sepaLabel2 = tk.Label(self.optionFrame, image=self.
                sepaImage ,bg="white", activebackground="white")
962         #self.sepaLabel2.grid(row=6, column=0, sticky="ew",
                columnspan=2, pady=10)
963
964         #Buton
965         self.saveButton = tk.Button(self.optionFrame, text=
                dictionary["name_save"], border=0, bg='#E0EEE0', width
                ="12", height="1",font=('Microsoft YaHei Light',14),
                command =lambda: self.save_config())
966         self.saveButton.grid(row=7, column=0,  columnspan=2, pady
                =30)
967
968         self.statusLabel = tk.Label(self.optionFrame, text=""+ ":
                ", justify='left',  anchor="w", bg="white", font=('
                Microsoft YaHei Light',10))
969         self.statusLabel.grid(row=8, column=0, columnspan=2)
970
971     def save_config(self):
972         global dictionary, airfoilExportFormat, wingExportFormat
973
974         language = self.selectedLanguage.get()
975         airfoilExportFormat = self.selectedAirfoilFormat.get()
976         wingExportFormat =  self.selectedWingFormat.get()
977
978         try:
979             edit_config([language, airfoilExportFormat,
                    wingExportFormat])
980             self.statusLabel.config(text=dictionary["
                    message_saved"], fg="#007600")
981         except:
982             self.statusLabel.config(text=dictionary["
                    message_notsaved"], fg="#ee3a1f")
983
984 app= root()
985 app.frames[WingPage].add() #The app is started by a default
       profile
```

```
986 app.frames[AirfoilPage].update_airfoil() #Compute airfoil
       coordinates for mainrib_0
987 app.frames[AirfoilPage].change_section('mainrib_0') #The button
       of the mainrib_0 pressed is set by default
988 updateposition()
989 app.mainloop()
```

Código D.6: Archivo *main.py*

D.1.7. Archivo *configuration_manager.py*

```
def load_config():
    with open("./config/config.txt","r",encoding='utf-8') as f:
        lines = f.read().splitlines() #Eliminates the \n at the
           end. Avoids malfunction of function load language

        language = lines[0]
        airfoilfileformat = lines[1]
        wingfileformat= lines[2]

    dic = load_language(language)
    return dic, airfoilfileformat, wingfileformat

def load_language(language):
    dic = {}
    dic["language"] = language

    language = language.lower()
    with open("./config/" + language +".txt","r",encoding='utf-8'
       ) as f:
        lines = f.read().splitlines()

        for line in lines:
            dots = line.find(":")
            definition = line[:dots]
            text = line[dots+1:]

            dic[definition] = text
    return dic

def edit_config(parameters):
    with open("./config/config.txt","w", encoding="utf-8") as f:
        for line in parameters:
            f.write(line + "\n")
```

Código D.7: Archivo *configuration_manager.py*

D.2. Archivos de importado a programas CAD

D.2.1. *Script ImportWingGeometryLines.py*

```
#Author-Ricardo G. Garre
#Description-Imports a wing geometry which airfoils are separated by &

import adsk.core, adsk.fusion, traceback
import io

def run(context):
    ui = None
    n=0
    p=0
    try:
        app = adsk.core.Application.get()
        ui  = app.userInterface
        # Get all components in the active design.
        product = app.activeProduct
        design = adsk.fusion.Design.cast(product)
        title = 'Import Spline csv'
        if not design:
            ui.messageBox('No active Fusion design', title)
            return

        dlg = ui.createFileDialog()
        dlg.title = 'Open CSV File'
        dlg.filter = 'Comma Separated Values (*.csv);;All Files (*.*)'
        if dlg.showOpen() != adsk.core.DialogResults.DialogOK :
            return

        filename = dlg.filename
        with io.open(filename, 'r', encoding='utf-8-sig') as f:
            points = adsk.core.ObjectCollection.create()
            line = f.readline()
```

```
32              data = []
33              while line:
34                  if str(line) != '&\n':
35                      pntStrArr = line.split(',')
36                      for pntStr in pntStrArr:
37                          try:
38                              data.append(float(pntStr))
39                          except:
40                              break
41
42                      if len(data) >= 3 :
43                          point = adsk.core.Point3D.create(data
                                [0]*100, data[1]*100, data[2]*100)
44                          points.add(point)
45                          p+=1
46                      line = f.readline()
47                      data.clear()
48                  else:
49                      if points.count:
50                          root = design.rootComponent
51                          sketch = root.sketches.add(root.
                                xZConstructionPlane)
52                          lines = sketch.sketchCurves.sketchLines
53
54                          for i in range(p-1):
55                              point0 = points.item(i)
56                              point1 = points.item(i+1)
57                              lines.addByTwoPoints(point0, point1)
58
59                          if points.item(0) != points.item(p-1):
60                              lines.addByTwoPoints(points.item(p-1)
                                    , points.item(0))
61
62                          points.clear()
63                          line = f.readline()
64                          p=0
65                          n+=1
66                      else:
```

```
67                          ui.messageBox('No valid points', title)
68
69
70          if n == 0:
71              if points.count:
72                  root = design.rootComponent
73                  sketch = root.sketches.add(root.
                        xYConstructionPlane)
74                  lines = sketch.sketchCurves.sketchLines
75
76                  for i in range(p-1):
77                      point0 = points.item(i)
78                      point1 = points.item(i+1)
79                      lines.addByTwoPoints(point0, point1)
80
81              else:
82                  ui.messageBox('No valid points', title)
83
84      except:
85          if ui:
86              ui.messageBox('Failed:\n{}'.format(traceback.
                    format_exc()))
```

Código D.8: *Script ImportWingGeometryLines.py*