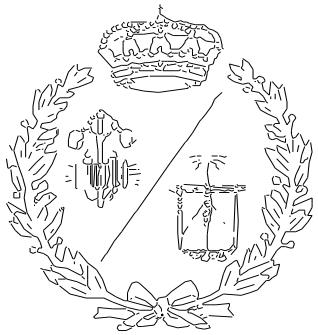
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

Desarrollo de un piloto automático no lineal, adaptativo y tolerante a fallos para un UAV de competición

Development of a non-linear, adaptive and fault-tolerant autopilot for a competition UAV

Para acceder al Título de

GRADUADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Juan Villar Fernández
Director: Carlos Torre Ferrero

Junio - 2025

Agradecimientos

En estos 4 años de carrera he estado rodeado de muchas personas que han influido de forma positiva en mi vida y en este trabajo.

En primer lugar, quiero agradecer a varios profesores su contribución al proyecto: a mi tutor, Carlos Torre, por su ayuda y comentarios al enfocar, elaborar y revisar la memoria; a Elías Revestido, por su guía y orientación para resolver problemas sobre el UKF; y a José Joaquín Sainz, por sus sugerencias e indicaciones con el ajuste del MPC. También debo un agradecimiento muy especial a Luis García por su consejo hace ya 4 años, que me llevó a entrar a este grado y, en consecuencia, a acabar descubriendo mi propio interés por el mundo de los UAV y la automática.

Por otro lado, quiero presentar mi más sincero agradecimiento a todos los miembros del equipo Xtra2. En especial a Alberto, por creer en mí y dejarme formar parte de esta gran aventura.

No puedo olvidarme de mis compañeros, tanto de aquí como de Valencia, con los que he pasado muchos momentos en estos años de carrera. Gracias a Alejandro, Helena, Nacho y Rober, con los que he compartido el agobio y las alegrías de buena parte de tres años. Por supuesto, gracias a Irene y Jaime, por aguantar un año de convivencia conmigo y por tantos buenos momentos juntos. Y, como no podría ser de otra manera, gracias a Carlota, Choticas, Joan, Lorenzo y todos los demás por hacer que mi estancia en Valencia fuera inolvidable.

Por último, doy las gracias a mis padres, por la educación, el amor y el apoyo que me han dado durante toda mi vida y que me han llevado a ser como soy.

A todos, muchas gracias.

RESUMEN

Las aeronaves no tripuladas (UAV) son un mercado emergente en el que la automática juega un papel fundamental. Gracias a determinados sistemas de control, llamados pilotos automáticos, se puede facilitar el manejo de estas naves a su piloto humano, de manera que la aeronave tenga una cierta autonomía. En este trabajo se propone uno de estos sistemas basado en algoritmos recientes, orientado a UAV de ala fija, capaz de lidiar con las no linealidades propias de estas aeronaves, detectar algunos fallos que se puedan dar en ellas durante el vuelo y adaptarse a estas circunstancias.

A lo largo del documento se aborda el diseño y las pruebas en simulación del piloto automático, explicando cada uno de sus componentes en detalle, probándolos en conjunto y realizando un pequeño estudio de la electrónica y sensores usados en estos aeromodelos. El sistema de control se compone de 2 bloques funcionales, observador y controlador, que fueron probados gracias a un simulador de vuelo desarrollado ex profeso para este proyecto.

Finalmente, se comentan otras aplicaciones de los algoritmos usados en este texto, tales como el cálculo de trayectorias de vuelo óptimas (mínimo gasto energético, máxima velocidad, etc.), predicción del comportamiento de un avión en fase de diseño o la estimación de coeficientes aerodinámicos del modelo del avión a partir de los registros (*logs*) de pruebas de vuelo.

ABSTRACT

Unmanned Aerial Vehicles (UAVs) are an emerging market in which automatic control plays an essential role. By means of some control systems called autopilots, the human pilot can be freed of many tasks required for handling these vehicles, giving the aircraft some autonomy. In this project, an autopilot based on recent algorithms is proposed. It is directed to fixed-wing aircraft and is capable of dealing with the nonlinearities of these systems, detect some of the faults that may arise during flight and adapt to these circumstances.

This document includes the design and simulation tests of the control system, explanations of each component in detail, tests of the complete control loop and a brief study of the electronics and sensors of these aircraft. The control system is composed of 2 functional blocks, called observer and controller, that were tested using a flight simulator developed for this project.

Finally, other applications of the developed algorithms are presented, such as the calculation of optimal flight trajectories (i.e. minimum energy usage, maximum velocity, etc.), prediction of the airplane's behaviour at its design phase or the identification of aerodynamic coefficients of the aircraft model using flight data.

ÍNDICE GENERAL

Índice de figuras

Índice de tablas

Nomenclatura

Aeronáutica

Sím.	Descripción	Unidades
b	Envergadura	m
$\bar{c}$	Cuerda media aerodinámica	m
C_D	Coeficiente de arrastre	-
C_L	Coeficiente de sustentación	-
C_l	Coeficiente de momento de alabeo	-
C_m	Coeficiente de momento de cabeceo	-
C_n	Coeficiente de momento de guiñada	-
C_X, C_Y, C_Z	Coeficientes de fuerzas aerodinámicas en ejes cuerpo	-
δ_a	Deflexión de los alerones	rad
δ_e	Deflexión del elevador	rad
δ_f	Deflexión de los flaps	rad
δ_p	Posición de la palanca de potencia	-
δ_r	Deflexión del timón de dirección	rad
I_p	Momento de inercia de la hélice	$\mathrm{kg} \cdot \mathrm{m}^2$
i_t	Ángulo de incidencia del estabilizador horizontal	rad
I_x, I_y, I_z	Momentos de inercia respecto de los ejes cuerpo	$\mathrm{kg} \cdot \mathrm{m}^2$
g	Aceleración de la gravedad	$\mathrm{m/s}^2$
m	Masa de la aeronave	kg
p	Velocidad de alabeo	rad/s
$\bar{q}$	Presión dinámica	Pa
q	Velocidad de cabeceo	rad/s
r	Velocidad de guiñada	rad/s
S	Superficie alar	m^2

T	Empuje	N
V	Velocidad relativa al viento (*airspeed*)	m/s
V_0	Velocidad de referencia (*trim speed*)	m/s
α	Ángulo de ataque	rad
β	Ángulo de derrape o deslizamiento	rad
γ	Ángulo de pendiente de la trayectoria	rad
Ω_p	Velocidad angular de la hélice	rad/s
ϕ	Ángulo de alabeo (*roll*)	rad
ψ	Ángulo de guiñada (*yaw*)	rad
ρ	Densidad del aire	kg/m^3
θ	Ángulo de cabeceo (*pitch*)	rad

Control automático

Sím.	**Descripción**
$\mathbf{D}$	Matriz de dispersión
$\mathbf{f}$	Función de transición de estado
$\mathbf{H}_{dr}$	Matriz hessiana relativa diagonal
$\mathbf{H}_r$	Matriz hessiana relativa
$\mathbf{H}$	Matriz hessiana
$\boldsymbol{h}$	Función de salida
$\mathbf{J}$	Función de coste
N	Horizonte de predicción
$\mathbf{K}_k$	Ganancia del filtro
$\mathbf{M}$	Matriz de información de Fisher
n_u	Dimensión del vector de entrada
n_x	Dimensión del vector de estado
$\mathbf{P}_{k\|k}$	Covarianza de la estimación del estado actual
$\mathbf{P}_{k-1\|k-1}$	Covarianza de la estimación del estado anterior
$\mathbf{P}_{k\|k-1}$	Covarianza de la predicción del estado
$\mathbf{P}_{y,k}$	Covarianza de la predicción de la observación

$\mathbf{Q}$	Covarianza de ruido de proceso
$\mathbf{R}$	Covarianza de ruido de medida
$\mathbf{s}$	Variables de holgura
$\mathbf{S_x}$	Matriz de sensibilidad del estado
$\mathbf{S_y}$	Matriz de sensibilidad de salida
$\mathbf{u}_k$	Entrada actual del sistema
$\boldsymbol{v}$	Ruido de medida
$\mathbf{w}$	Ruido de proceso
$\mathbf{W_s}$	Pesos de las variables de holgura
$\boldsymbol{w}$	Vector de pesos de la UT
$\mathbf{W_u}$	Pesos de las variables de entrada
$\mathbf{W_x}$	Pesos de las variables de estado
$\hat{\mathbf{x}}_{k\|k}$	Estado actual del filtro
$\hat{\mathbf{x}}_{k-1\|k-1}$	Estimación previa del estado
$\hat{\mathbf{x}}_{k\|k-1}$	Predicción del estado
$\hat{\mathbf{y}}_{k\|k-1}$	Predicción de la observación
$\boldsymbol{\Psi}$	Gradiente del modelo
$\mathbf{z}_k$	Observación actual

1. INTRODUCCIÓN

Los UAV (*Unmanned Aerial Vehicles*, aeronaves no tripuladas, drones) son artefactos con un gran potencial en usos civiles. En las últimas décadas, están siendo aplicados con éxito en diversos ámbitos: vigilancia de cultivos, exploración, salvamento, transporte, etc. Ofrecen vías novedosas de actuación y extienden la capacidad de intervención humana en escenarios poco accesibles [1].

Estas aeronaves suelen tener algún sistema de control asistido que facilita el trabajo del piloto. El estudio y diseño de este tipo de sistemas es una actividad contemplada entre las competencias de la titulación académica de grado en Ingeniería Electrónica.

Para su manejo eficaz, un UAV debe ser tratado como un sistema dinámico bajo control, como una planta industrial. La teoría de sistemas de control brinda las herramientas y técnicas para poder operar estas aeronaves de forma precisa y segura, incluso en escenarios cambiantes o ante situaciones peligrosas.

Este Trabajo de Fin de Grado (TFG) se centra en el estudio de un UAV de ala fija con propulsión eléctrica por hélice. Sus cualidades—autonomía, eficiencia energética, velocidad, capacidad de carga—los diferencian de los UAV tipo multirotor, cuya principal cualidad es la capacidad de mantenerse quietos en el espacio.

Para estudiar un UAV como planta a controlar se necesita, en primer lugar, un modelo matemático que relacione acciones y resultados en función del tiempo. Las entradas de control y las salidas medibles serán las vías por las que se ejerce el control.

Para controlar la aeronave, en esta memoria se desarrollará un lazo de control completo, basado en algoritmos recientes, que procese las señales recibidas de unos sensores instalados en la nave y genere señales de control para el avión adecuadas para que este alcance y mantenga una determinada referencia o consigna (como una altitud o rumbo predefinidos).

El lazo de control contiene técnicas que podrían calificarse de *experimentales* o poco probadas, lo que permite etiquetar al presente trabajo como proyecto de investigación según lo contemplado en la normativa vigente de la Universidad de Cantabria y así debe interpretarse.

Como cualidad distintiva de este TFG, cabe destacar el tratamiento cuidadoso que se ha querido dar al estudio de las siguientes facetas:

1. Creación del modelo matemático, con sus entradas, salidas y estados.
2. Creación de un simulador de vuelo, que utiliza el anterior modelo para predecir la respuesta del sistema ante diversas entradas.
3. Creación de un observador/estimador de estados de tipo UKF, capaz de estimar estos a partir de medidas de sensores.
4. Resolver la identificación de parámetros del sistema en tiempo real, a partir de las mediciones durante el vuelo (aplicable en tiempo real o a registros *logs* de pruebas de vuelo).

5. Crear un controlador del sistema, que asegure la estabilidad y sea robusto frente a incidentes varios.

6. Detección de fallos, a partir de la monitorización de los parámetros identificados, detección de sus variaciones y diagnóstico de tales sucesos.

1.1. MOTIVACIÓN

Durante el curso 2023-2024, el autor participó en un intercambio académico en la Universidad Politécnica de Valencia (UPV), en el marco del programa SICUE, donde cursó las asignaturas propias del 3.^{er} curso de grado.

Como actividades no académicas pero vinculadas a la universidad, el autor formó parte de un grupo de estudiantes relacionado con los aviones de radiocontrol (RC). El grupo, llamado Xtra2 UPV (pronunciado *extradós*), está liderado por alumnos de la Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial (ETSIADI) y durante ese curso académico quería reforzar su solvencia en el ámbito de la electrónica. De este modo, el autor fue invitado a unirse, entrando en este mundo.

El equipo estaba inmerso en una competición internacional entre universidades, la *Air Cargo Challenge* (ACC) [2], en la que grupos de estudiantes diseñan y construyen aeronaves no tripuladas de ala fija que transporten la mayor carga útil (*payload*) posible, conforme a una serie de reglas. En la figura 1.1 se muestra el prototipo construido por Xtra2, el XTRA25, conocido como «Chimuelo».

Figura 1.1: Prototipo XTRA25 «Chimuelo» en vuelo durante la Air Cargo Challenge 2024.

Durante la competición, los pilotos de cada equipo demuestran las bondades de estos prototipos, que son calificados en función de la carga útil que transportan, su eficiencia y su velocidad.

Con cada nueva edición se plantean nuevas reglas y retos de diseño y construcción. El equipo Xtra2 UPV había participado ya en las ediciones de 2019 y 2022 y, como novedad, en la de 2024, la normativa abría la posibilidad de incluir asistencias automáticas al piloto.

Tanto en el diseño como en la construcción y las posteriores pruebas de vuelo, se plantearon cuestiones recurrentes sobre cómo aprovechar las asistencias automáticas para mejorar al máximo el rendimiento del avión y, así, la puntuación en la competición. Mediante estas ayudas se podría controlar el avión con más suavidad, perfeccionar o automatizar maniobras de aterrizaje y despegue, garantizar virajes coordinados, estabilizar el vuelo en crucero…

Por otro lado, durante el curso académico los miembros más veteranos del grupo impartían sesiones teórico-prácticas de diseño y fabricación de aeromodelos. Estas estaban pensadas para que los nuevos miembros aprendieran todo lo necesario para empezar en el mundillo. Cada año ingresan nuevos miembros de distintas titulaciones de grado y máster de todos los cursos y, gracias a estas sesiones, pueden empezar a diseñar estos prototipos con muy pocos conocimientos previos.

Este entorno de entusiasmo y ganas de trabajar resultó muy provechoso para aprender sobre aeronáutica y reforzar los conocimientos de automática dados en el grado, al darles una aplicación totalmente práctica. Además, sirvió para despertar en el autor un gran interés por todos aquellos proyectos que reuniesen estas dos disciplinas.

Una vez terminada la construcción del prototipo y de finalizar la competición con un meritorio 5.º puesto de entre 31 equipos participantes, quedaban pendientes de implementar algunas de las ideas que se plantearon durante el diseño del avión.

Una de ellas era el anhelo de obtener los valores efectivos de algunos parámetros de diseño del avión de forma experimental mediante los registros de telemetría de las pruebas de vuelo. Esto ya era posible mediante experimentación en túnel de viento, pero obtenerlos con sensores de bajo coste y en pruebas de vuelo resultaba una idea atractiva. De ahí surge este proyecto, aunque del trabajo realizado en el equipo Xtra2 surgieron muchas otras ideas que podrían dar lugar a otros tantos proyectos de complejidad similar.

1.2. OBJETIVOS

Durante el desarrollo de este trabajo, se han ido expandiendo los requisitos y necesidades de las herramientas a desarrollar. Del mismo modo, el interés por profundizar en determinados temas ha ido variando con el tiempo, condicionando el enfoque y la dirección del esfuerzo dedicado al proyecto. Una vez terminada la memoria del mismo, se puede decir que los objetivos que se pretendían alcanzar fueron:

- Por los intereses propios del autor:
  - Desarrollar un sistema de control completo que sea multivariable y moderno, aprovechando las últimas técnicas disponibles.
  - Desarrollar un sistema de identificación de parámetros, que pudiera utilizarse en tiempo real o no, para obtener estimaciones de los valores de parámetros aerodinámicos de aeronaves.
- Por los temas de interés destacados por José Morcillo [3], miembro de Xtra2 UPV:

  - Programación de un simulador de vuelo para optimizar la trayectoria del avión en la competición.
  - Realización de un estudio de los sensores de las aeronaves en general y su aplicación a los aeromodelos desarrollados en el equipo Xtra2.
  - Desarrollo de un piloto automático para un UAV.
  - Desarrollo de un sistema de medición del coeficiente de arrastre de un UAV (estimación de parámetros aerodinámicos).
- De las tareas propuestas por Jorge Castillo [4]:
  - Redacción de documentos explicativos (esta memoria) sobre los servos usados como actuadores.
  - Desarrollo de un sistema de control de vuelo que evite la entrada en pérdida. Queda incluido en el desarrollo del sistema de control moderno y multivariable.

1.3. ESTRUCTURA DEL TRABAJO

El desarrollo de este trabajo no ha sido lineal. Durante el proceso, se han encontrado varios problemas que han requerido buscar soluciones no consideradas al principio, ramificando el desarrollo del trabajo. La figura 1.2 pretende representar algunos de los diversos frentes que se fueron abriendo y que requirieron atención intensificada. Sin embargo, en esta memoria la exposición se ha organizado en un orden lógico y lineal, que permita su comprensión a cualquier lector ajeno al proyecto. Por ello, en esta sección se explica en qué orden se describen los progresos del trabajo en este texto, que es distinto del progreso natural de la actividad de un trabajo.

En el presente capítulo únicamente se plantean los objetivos del proyecto, la motivación y la metodología. Sirve de introducción al lector para el alcance del trabajo, la visión del autor y las razones que han llevado a crear este documento.

La primera parte del trabajo se compone de dos capítulos, que sirven de introducción a los temas tratados y establecen los conocimientos básicos necesarios para comprender el resto de la memoria.

- A lo largo del capítulo 2 se presentan nociones básicas de la teoría de sistemas dinámicos. Aquí se definen brevemente los conceptos de sistema, parámetros, estado y salida. También se mencionan las representaciones matemáticas de estos sistemas, llamadas modelos, y la teoría conocida como identificación, que busca obtener estos modelos de forma experimental.
- En el capítulo 3 se introducen conceptos elementales de ingeniería aeronáutica, necesarios para comprender el sistema dinámico tratado en el trabajo y sus ecuaciones diferenciales.

La segunda parte del trabajo consta de otros tres capítulos, en los que se abordan con cierto detalle el estudio del sistema a controlar: el avión.

- El capítulo 4 está dedicado a desarrollar el modelo matemático del avión. Se plantean sus ecuaciones diferenciales y se acota la validez de estas según los regímenes de trabajo de la aeronave. También se relacionan los términos de las ecuaciones con los conceptos ya establecidos en la teoría de sistemas.
- En el capítulo 5 se desarrollan los actuadores del sistema, sus limitaciones, sus modelos matemáticos y las simplificaciones hechas para obtener estos últimos.
- Por último, en el capítulo 6 se hace lo propio con los sensores a bordo de la aeronave, tratando sus limitaciones, precios e inconvenientes.

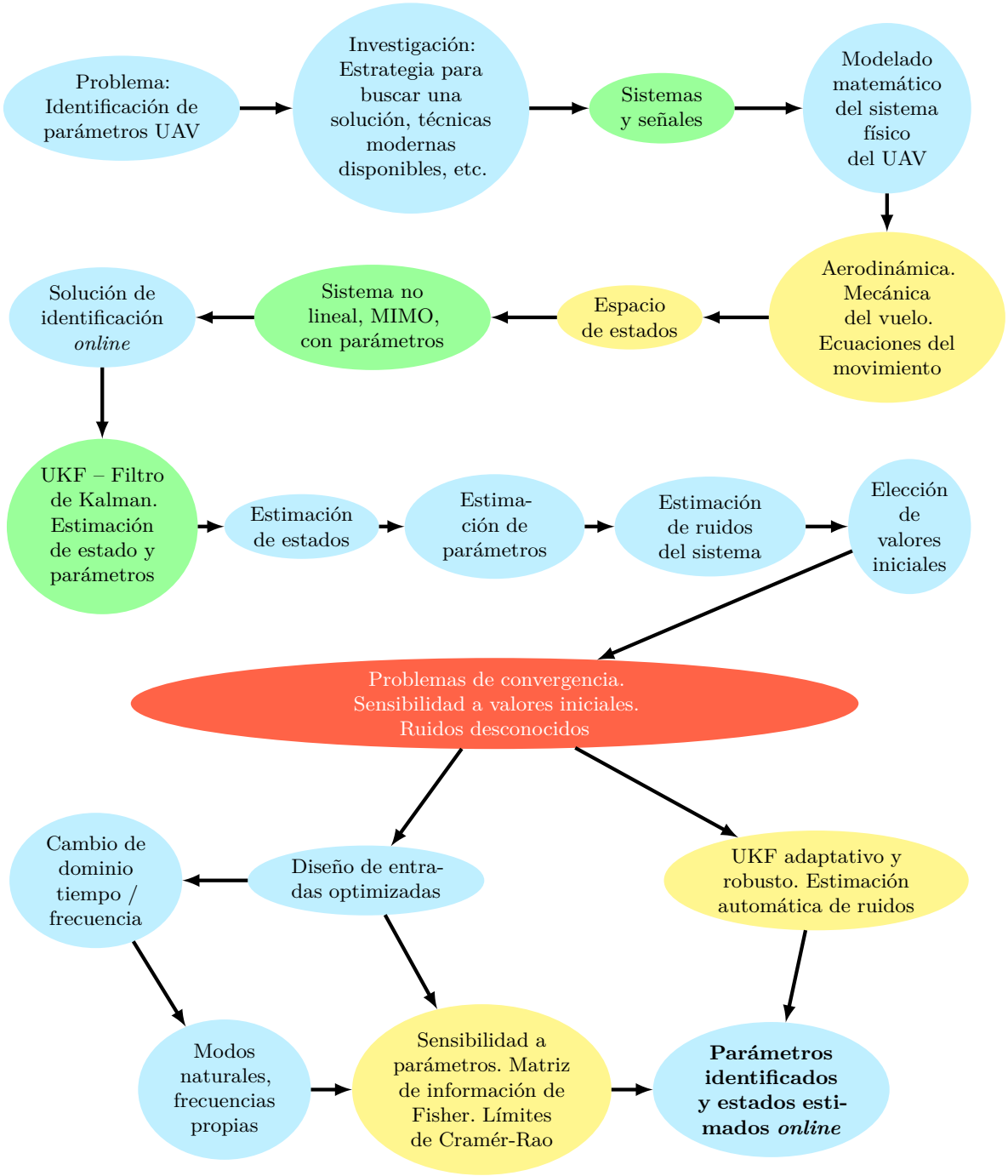


Figura 1.2: Ideograma o mapa de ideas del trabajo.

La tercera parte se dedica a la solución propuesta y a describir cada una de sus partes en detalle, repartidos en 3 capítulos.

- En el capítulo 7 se exponen los métodos numéricos y las herramientas de *software* que permiten simular el comportamiento del avión a partir del modelo explicado en la parte anterior.
- El capítulo 8 se centra en el algoritmo matemático que permite estimar estados y parámetros. Este es un filtro de Kalman, en particular su variante *unscented* (sin aroma). A lo largo del capítulo se describe su comportamiento y características, además de justificar su elección frente a otras alternativas y desarrollar su definición matemática.
- El capítulo 9 versa sobre el controlador de vuelo elegido, su funcionamiento, características y otras alternativas consideradas.

En la cuarta y última parte del cuerpo del trabajo se presentan los resultados obtenidos con el sistema propuesto, probándolo en distintas situaciones con distintas suposiciones iniciales.

- A lo largo del capítulo 10 se muestran el desempeño del controlador diseñado a la hora de realizar distintas maniobras. También sirve para comprobar las capacidades del simulador.
- El capítulo 11 demuestra las prestaciones del observador elegido, el UKF, para estimar el estado completo del sistema a partir de unas medidas de sensores contaminados con ruido que no incluyen todas las variables de estado.
- En el capítulo 12 se muestra el procedimiento seguido para identificar parámetros aerodinámicos del UAV, además del propio estado del sistema, a partir de medidas ruidosas.
- Finalmente, se aportan unas conclusiones sobre las técnicas de control exploradas en el trabajo y los problemas o inconvenientes que presentan. Esto se hace en el capítulo 13.

Al final de la memoria se encuentran los apéndices, que incluyen información relacionada con el trabajo que pudiera ser de interés para el lector, pero cuya lectura no es necesaria para comprender el trabajo.

- En el apéndice A se recogen algunos de los programas de MATLAB desarrollados para el trabajo, que incluyen el código de un UKF y otras funciones auxiliares para realizar las simulaciones expuestas en esta memoria.
- Por último, en el apéndice B se recopila una lista de las herramientas usadas para maquetar este documento así como la fecha de producción o compilación.

1.4. METODOLOGÍA

El trabajo presentado en esta memoria se enmarca en los TFG de tipo proyectos de investigación según la normativa de la Universidad de Cantabria (UC). Por ello presenta la estructura y el contenido que tiene, que es distinto de la organización *clásica* de un proyecto de ingeniería.

La parte de investigación del trabajo ha consistido en recopilar información sobre cómo otros autores han resuelto el problema de control automático no lineal de UAV de ala fija. Sobre este tema, si bien existen algunos libros sobre la materia, la mayor parte de conocimiento se encuentra en forma de artículos publicados por editoriales científicas como Springer Nature, Elsevier, el *American Institute of Aeronautics and Astronautics* (AIAA), etc. Los usuarios de la UC tienen acceso a algunas de estas publicaciones, mientras que otras fuentes solo fueron accesibles mediante la cuenta de usuario de la UPV durante el curso 2023-2024.

La bibliografía del capítulo 14 contiene todas las referencias a los artículos y fuentes consultadas con título, número de identificación (DOI, ISBN, ISSN, etc.), una dirección URL de referencia y la fecha de consulta de esta, siguiendo el estándar de citas de IEEE.

La investigación ha permitido conocer los últimos avances en los campos de estudio del trabajo, que se resumen en cada uno de los capítulos de la parte III. Además, en los artículos y referencias consultadas se han podido ver aplicaciones de los algoritmos aquí comentados y comprender sus capacidades y potenciales problemas. De esta manera se descubrieron y eligieron los métodos usados para controlar el avión y estimar su estado y parámetros, como se comenta en sus respectivos capítulos.

Una vez elegidas las técnicas para realizar el control, se diseñan los bloques funcionales recurriendo, nuevamente, a la literatura especializada y reciente, en la que se describen técnicas de ajuste o sintonización automáticos y algunas soluciones a posibles problemas. Los subsistemas se prueban en simulación y se hacen reiteradas correcciones hasta obtener los resultados deseados.

Para la programación de todos los algoritmos se hace uso de MATLAB y Simulink. Este lenguaje permite un equilibrio entre rapidez de ejecución y simplicidad del código que lo hace adecuado para hacer pruebas rápidas con los métodos numéricos que fueron necesarios durante el proyecto. Esto se alinea con la filosofía del trabajo de explorar posibles aplicaciones de ciertos algoritmos sin considerar una implementación en *hardware* específico. La familiarización con el lenguaje de MATLAB adquirida durante los cursos del grado también ayudó a acelerar el proceso de escritura de código, que con otras alternativas hubiera sido más laborioso.

Parte I

CONCEPTOS BÁSICOS

2. SISTEMAS DINÁMICOS Y SU CONTROL

Para el control automático de cualquier sistema, se suele recurrir a conceptos relacionados con la teoría de control. Por ello, este capítulo sirve de resumen de algunos conceptos básicos sobre sistemas dinámicos y teoría de control, buena parte de ellos impartidos en asignaturas de automática del grado. Estos conocimientos son básicos y se han tratado en multitud de publicaciones extensas. Algunas de las consultadas fueron [5-9].

2.1. INTRODUCCIÓN A LOS SISTEMAS DINÁMICOS

En esta sección se resumen las nociones básicas de la llamada *teoría de sistemas* que se necesitan para contextualizar los algoritmos y expresiones contenidas en los capítulos siguientes. En las próximas páginas, el lector encontrará una introducción a los modelos de sistemas, la nomenclatura y notación matemática comúnmente usada para tratar estos conceptos y unas ideas preliminares sobre identificación de sistemas.

Un *sistema* consiste en un conjunto de elementos que trabajan de forma coordinada. Una aeronave es un sistema de tipo físico, consistente en varios actuadores mecánicos y unas superficies capaces de generar fuerzas y momentos de origen aerodinámico que tienden a cambiar la orientación y posición de la nave. Se llama *modelo* a la representación matemática de un sistema que permite predecir la evolución de este en el futuro a partir de unas condiciones dadas [5].

Los modelos resultan útiles para comprender el comportamiento de los sistemas ante distintos estímulos, conocidos como *entradas*. La información mínima que, junto a las entradas, permite predecir la salida futura de un sistema se conoce como *estado del sistema*. Este estado consiste en una o varias variables que condicionan el comportamiento del sistema en los instantes siguientes. Para el caso de las aeronaves, las entradas son los mandos del piloto como la posición de los actuadores, mientras que el estado consiste en la posición del avión en el espacio y de su velocidad instantánea. Las salidas serán cualquier variable del sistema que pueda medirse con sensores.

En un modelo de un sistema, la evolución o *dinámica* del estado está gobernada por la *ecuación de estado*. Para sistemas definidos en *tiempo continuo*, esta toma la forma de una ecuación diferencial, que en su versión más general se puede escribir como [5]

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right) , \tag{2.1}$$

donde $\mathbf{x}$ denota el vector de estado, $\dot{\mathbf{x}}$ representa su derivada respecto del tiempo y $\mathbf{u}$ indica el vector de entrada. Estos vectores contienen las variables de estado y de entrada del sistema y sus dimensiones se denotan por n_x y n_u respectivamente. Cuando los tamaños de ambos vectores son 1 se dice que el sistema es SISO (*single-input single-output*), mientras que si son mayores se dice que es MIMO (*multiple-input multiple-output*) [8].

Si el sistema es lineal, la función $\mathbf{f}$ se simplifica a productos de matrices y vectores y la dinámica del sistema se puede expresar como [7]

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) , \tag{2.2}$$

donde **A** y **B** son matrices conocidas como matriz de estado y de entrada respectivamente [8].

Otra forma de expresar la dinámica del sistema es en *tiempo discreto*, de forma que las variables del sistema solo están definidas en instantes concretos y periódicos de tiempo. Los valores de los vectores en un instante dado se denota por el subíndice k, el anterior instante se denota por $k-1$, etc. Esta forma de considerar el tiempo en los sistemas resulta útil cuando se trabaja con sistemas digitales (computadores), en los que las medidas se procesan a intervalos regulares de tiempo T_s, llamado período de muestreo (*sample time*) [9].

En esta forma de representación, la dinámica del estado se define por una *ecuación en diferencias*, que en su forma genérica se escribe como [9]

$$\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) . \tag{2.3}$$

Nótese que se ha usado el mismo nombre para la función **f** tanto aquí como en el sistema continuo de la ecuación (2.1). Por simplicidad, se ha adoptado esta notación en este texto, aunque ambas funciones son distintas, no deben confundirse y así se debe interpretar según el contexto.

Hasta ahora, las ecuaciones relacionaban la evolución del estado del sistema con la entrada. No obstante, el estado del sistema no siempre es accesible o tiene sentido físico. El conjunto de variables de un sistema que se pueden medir se recoge en el llamado *vector de salida*, que se relaciona con el estado y entrada actuales mediante la *ecuación de salida*, dada en tiempo discreto por [9]

$$\mathbf{y}_k = \boldsymbol{h}(\mathbf{x}_k, \mathbf{u}_k) \tag{2.4}$$

y de forma equivalente en tiempo continuo. En la práctica, esta salida del sistema no se puede conocer con absoluta exactitud. Esto es porque, para medirla, se usan sensores que introducen ruido en sus medidas. La expresión que relaciona las medidas $\mathbf{z}_k$ de los sensores con la auténtica salida del sistema se conoce como *ecuación de medida* y en este texto tomará la forma [10]

$$\mathbf{z}_k = \mathbf{y}_k + \boldsymbol{v}_k \,, \tag{2.5}$$

donde $\boldsymbol{v}_k$ representa el ruido de medida introducido por los sensores. Esta es una variable aleatoria que sigue una distribución normal de media nula, es decir, se asume que el ruido de medida es Gaussiano. La covarianza de este ruido viene dado por la matriz de covarianza de ruido de medida **R**, que generalmente es diagonal. Cuando el ruido interviene de esta forma en el sistema, se conoce como *ruido aditivo*, mientras que si interviene de forma no lineal se dice que no es aditivo [11]. Por simplicidad, en este trabajo se ha optado por considerar solo ruidos aditivos.

En algunas publicaciones no se hace distinción entre el vector de salida del sistema y el vector de observaciones o medidas [10], de modo que se plantea una única expresión que mezcla salidas y medidas, resultado de combinar las ecuaciones (2.4) y (2.5), obteniendo

$$\mathbf{z}_k = \boldsymbol{h}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{v}_k \,. \tag{2.6}$$

Por ello, en estos textos la función $\boldsymbol{h}(\mathbf{x}_k, \mathbf{u}_k)$ se suele llamar función de medida. En este proyecto se ha optado por hacer una distinción entre los vectores **y** y **z**, por lo que se usarán las ecuaciones (2.4) y (2.5) por separado.

Existen variantes no deterministas de las ecuaciones de estado, en las que la dinámica del sistema está afectada por un ruido aleatorio que recibe el nombre de *ruido de proceso* [10]. Este se denota por **w** y puede afectar a las ecuaciones en tiempo continuo o en tiempo discreto

indistintamente. En este supuesto, la ecuación de estado del sistema en tiempo discreto pasa a ser:

$$\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) + \mathbf{w}_k\,. \tag{2.7}$$

La covarianza de este ruido viene dada por la *matriz de covarianza de ruido de proceso*, denotada por $\mathbf{Q}$. Generalmente, esta es una matriz diagonal cuyos elementos son las covarianzas del ruido de proceso para cada variable de estado.

2.2. ESTABILIDAD DE SISTEMAS DINÁMICOS

La estabilidad es un concepto importante de la teoría de sistemas que merece especial atención, principalmente en el diseño de controladores. Existen varias definiciones formales de estabilidad que se comentarán a continuación pero, en pocas palabras, podría decirse que un sistema es estable si, ante una entrada acotada, su respuesta también es acotada [12].

Estudiar la estabilidad de los sistemas puede llegar a ser un proceso complejo, con una gran carga matemática, más aún si el sistema es no lineal. En términos formales, podrían mencionarse los teoremas de estabilidad de Lyapunov, que permiten estudiar la estabilidad de sistemas dinámicos en general y, dentro de estos, los sistemas lineales.

El método más sencillo para estudiar la estabilidad de un sistema es conocido como **método de linealización** [13, 14] y consiste en calcular el jacobiano $\mathbf{J}$ del sistema respecto del vector de estado $\mathbf{x}$, que viene dado por la expresión

$$\mathbf{J}(\mathbf{x},\mathbf{u}) = \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_{n_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}_{n_x}}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{f}_{n_x}}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{f}_{n_x}}{\partial \mathbf{x}_{n_x}} \end{bmatrix}, \tag{2.8}$$

donde n_x es el orden del sistema (dimensión del vector $\mathbf{x}$).

Un punto de equilibrio del sistema es aquel en el que la función $\mathbf{f}$ vale $\vec{0}$ y viene dado por los vectores $\mathbf{x}^*$ y $\mathbf{u}^*$. Un sistema será estable localmente si se verifica que los autovalores del jacobiano evaluado en el punto de equilibrio $(\mathbf{x}^*, \mathbf{u}^*)$ tienen todos parte real negativa, inestable si alguno de ellos tiene parte real positiva e indeterminado si al menos uno de ellos está en el eje imaginario y el resto tiene parte real negativa.

Este método es el más utilizado para estudiar la estabilidad de sistemas lineales, puesto que la ecuación de estado de estos en tiempo continuo es

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.9}$$

y la matriz $\mathbf{A}$ es el jacobiano del sistema respecto del estado.

Aunque el método es sencillo y fácil de aplicar, no garantiza la estabilidad en sistemas no lineales o lo hace en una región de estados cercanos al punto de equilibrio. Este concepto de cercanía depende de la naturaleza del sistema y no se puede determinar por este método. Por ello, cuando los autovalores del jacobiano tiene parte real negativa se dice que el sistema es localmente estable [14].

En el caso de las aeronaves, y en particular las de ala fija, este método se explica en [15, 16] y de hecho es el método utilizado para estudiar la estabilidad de las aeronaves de la ACC, como

se ilustra en [3, 17, 18]. Esto es así por las características específicas que tienen los aviones, que permiten realizar esta clase de estudio con éxito.

El otro criterio ampliamente conocido es el llamado **método de Lyapunov** [13, 14], que consiste en buscar una función $V(\mathbf{x})$, llamada *función de Lyapunov*, que debe cumplir ciertas propiedades para que el sistema sea estable. Sin embargo, este método no se tratará en este trabajo.

Estabilidad en aeronaves

Para estudiar la estabilidad de aeronaves se recurre, como ya se ha dicho, a la linealización del sistema. Sin entrar en detalles sobre el modelo del avión, que se verá más adelante en el capítulo 4, se puede comentar algunas particularidades del estudio de estabilidad en ingeniería aeronáutica.

En esta disciplina, para simplificar el cálculo de la estabilidad, este se divide en una primera parte, conocida como estabilidad estática, y otra conocida como estabilidad dinámica. Este tema se trata con más profundidad en [3, 17-19].

La **estabilidad estática** se refiere al cálculo de algunos coeficientes (en concreto derivadas aerodinámicas) que determinan la capacidad de la aeronave para generar fuerzas y momentos recuperadores en caso de que se produzca una perturbación. Es la primera fase del estudio de la estabilidad y, si se verifican unos criterios basados en los signos de estos coeficientes, se determina que el avión es estáticamente estable y se procede a estudiar la estabilidad dinámica. De forma sencilla, la estabilidad estática responde a la pregunta de si el avión tiene tendencia a recuperarse en caso de perturbación; admite como respuestas *sí* o *no*.

La **estabilidad dinámica** consiste en el criterio de estabilidad de linealización mencionado anteriormente. Básicamente, se linealiza el sistema y se determina la posición en el plano complejo de los autovalores del jacobiano evaluado en una condición de trimado del avión, la cual es elegida por el diseñador del aparato. Como previamente se ha hecho un estudio de la estabilidad estática, generalmente (no es condición suficiente) el sistema es dinámicamente estable, aunque la posición de los autovalores o polos puede no ser la deseada por el equipo de diseño, y por ello interesa cambiar ciertos parámetros del avión para mejorar o aumentar la estabilidad de la aeronave, jugando con la posición de estos autovalores. Podría decirse que el estudio de esta estabilidad responde a la pregunta de *cómo* se recupera el avión en caso de perturbación y admite una respuesta más abierta.

2.3. IDENTIFICACIÓN DE SISTEMAS

En teoría de sistemas, se conoce como *identificación* al proceso de obtener modelos matemáticos de sistemas a partir de datos experimentales. En el caso de las aeronaves, como la tratada en este trabajo, para elaborar el modelo se suele partir de ecuaciones diferenciales basadas en leyes físicas. Estas expresiones son conocidas como ecuaciones de Bryan y se desarrollarán en el capítulo 4. El libro de referencia sobre identificación de aeronaves y que se citará varias veces en esta memoria es [10]. Otra obra destacable es [6], sobre técnicas de identificación de sistemas dinámicos en general.

Para un sistema dinámico cualquiera, el modelo suele venir determinado por alguna expresión que relaciona las entradas $\mathbf{u}$ con el estado $\mathbf{x}$ y las salidas $\mathbf{y}$ del sistema. Estas expresiones dependen de ciertos coeficientes, llamados *parámetros*, que se asumen constantes o cuya evolución no

está definida en el modelo. En estos casos, la identificación de sistemas consiste en determinar, primero, una *estructura* para el modelo (unas ecuaciones); segundo, el valor de estos parámetros, generalmente representados mediante un vector $\boldsymbol{\theta}$, tal que el modelo se comporte de la forma más parecida posible al sistema original [20].

Para los UAV de ala fija de este trabajo, la estructura del modelo se ha fijado previamente y consiste en unas ecuaciones diferenciales obtenidas de la literatura aeronáutica, recomendadas expresamente para este propósito. Estas conforman el modelo del avión tratado en el proyecto y se presentan en el capítulo 4.

Formalmente, la identificación busca el conjunto de parámetros $\hat{\boldsymbol{\theta}}$ que minimice el error cuadrático del modelo, dado por [10]:

$$J(\hat{\boldsymbol{\theta}}) = \sum_{k=1}^{N} \left[\mathbf{z}_k - \boldsymbol{h}(k, \hat{\boldsymbol{\theta}})\right]^\intercal \left[\mathbf{z}_k - \boldsymbol{h}(k, \hat{\boldsymbol{\theta}})\right] \tag{2.10}$$

donde $\mathbf{z}_k$ denota el vector de medidas experimentales en el instante k, $\boldsymbol{h}(k, \hat{\boldsymbol{\theta}})$ representa el vector de salida predicho por el modelo en el instante k evaluado en el vector de parámetros supuesto $\hat{\boldsymbol{\theta}}$ y N es el número de muestras tomadas.

Existen múltiples técnicas para resolver este problema, como se puede comprobar en cualquier obra extensa sobre identificación de sistemas [6, 10]. El filtro de Kalman, comentado más adelante en el capítulo 8, es un estimador bayesiano recursivo que puede usarse para estimar el vector de estado $\mathbf{x}$ y también para identificar el sistema encontrando el valor óptimo de $\hat{\boldsymbol{\theta}}$.

2.4. SISTEMAS DE CONTROL

Los sistemas de control surgen de la necesidad de automatizar determinados procesos o tareas mediante algún sistema ya existente que se puede controlar. Una aeronave es un sistema cuyo comportamiento se puede controlar por el piloto, pero sus acciones también se pueden automatizar. En este capítulo se hará una breve introducción a los sistemas y se orientará al lector hacia los conceptos clave para entender las decisiones tomadas durante el desarrollo de este trabajo.

Para representar las arquitecturas, estructuras o lazos de control se suele recurrir a diagramas de bloques como el de la figura 2.1 [5, 8]. En ella se representan mediante bloques los elementos que intervienen en un piloto automático moderno: el avión, el controlador, los sensores y el observador; así como las señales intercambiadas entre ellos mediante flechas. El controlador recibe una señal de error $\mathbf{e}_k$ y, con ella, calcula la entrada $\mathbf{u}(t)$ a aplicar al avión. Algunas variables del avión $\mathbf{y}(t)$ se miden con sensores, generando las medidas $\mathbf{z}_k$. El observador recibe estas medidas junto a la entrada y obtiene una estimación del estado del sistema $\hat{\mathbf{x}}$ y de la salida $\hat{\mathbf{y}}_{k|k}$. Esta última se puede restar de la referencia $\mathbf{r}_k$ para obtener el error $\mathbf{e}_k$, que el controlador usa para calcular las entradas de la planta.

Esta arquitectura es adecuada para controlar sistemas multivariable en los que se desee controlar variables del estado que no se puedan medir. Para alcanzar el objetivo de este trabajo es necesario diseñar los bloques del observador y el controlador, a los que se les dedicarán los capítulos 8 y 9. Para probar estos algoritmos sin una planta real, se sustituirá esta por un simulador de vuelo, explicado en el capítulo 7.

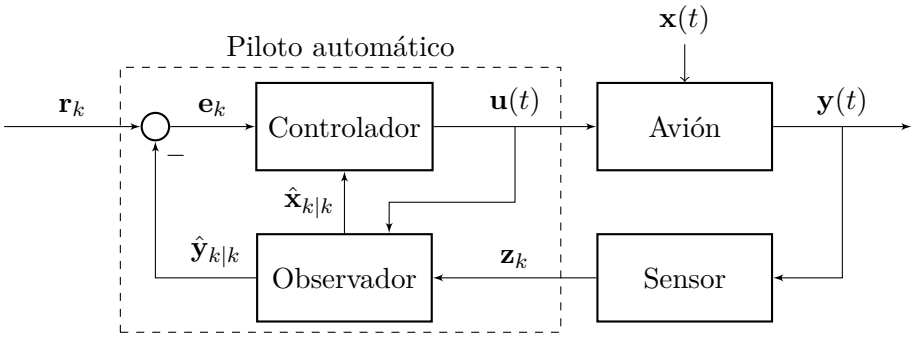


Figura 2.1: Diagrama de bloques de un sistema de control con observador o estimador del estado del sistema. Adaptado de [8]

3. FUNDAMENTOS DE AERONÁUTICA

Como el sistema a controlar es una aeronave, se usarán términos relacionados con aerodinámica y mecánica del vuelo. En este capítulo se describirán brevemente algunos de ellos.

3.1. ANATOMÍA DE UN AVIÓN

En esta sección se comentarán brevemente los elementos que componen un avión. Estos son conceptos básicos sobre aeronáutica y han sido tratados en multitud de publicaciones [1, 21, 22].

Por avión se entiende a toda aeronave de ala fija que genera sustentación por medio de esta y que avanza por la acción de uno o varios motores. Los elementos que suelen componer un avión se muestran en la figura 3.1, y son los siguientes [21, 23]:

- **Fuselaje**: Contiene en su interior parte de la carga y la electrónica.
- **Ala fija**: Consta de una superficie con una geometría tal que genera la fuerza de sustentación para mantener el avión en vuelo.
- **Empenaje**: Consiste en las superficies horizontal (estabilizador horizontal) y vertical (deriva o estabilizador vertical) situadas en la cola del avión y que aportan estabilidad longitudinal y direccional.
- **Tren de aterrizaje**: Es la estructura situada entre el fuselaje y el suelo, y permite que el avión circule por este durante el rodaje por pista, el despegue y el aterrizaje.
- **Planta propulsiva**: Conjunto de elementos que generan el empuje del avión y permiten que este acelere y alcance las velocidades requeridas para que el ala genere sustentación para permanecer en el aire.

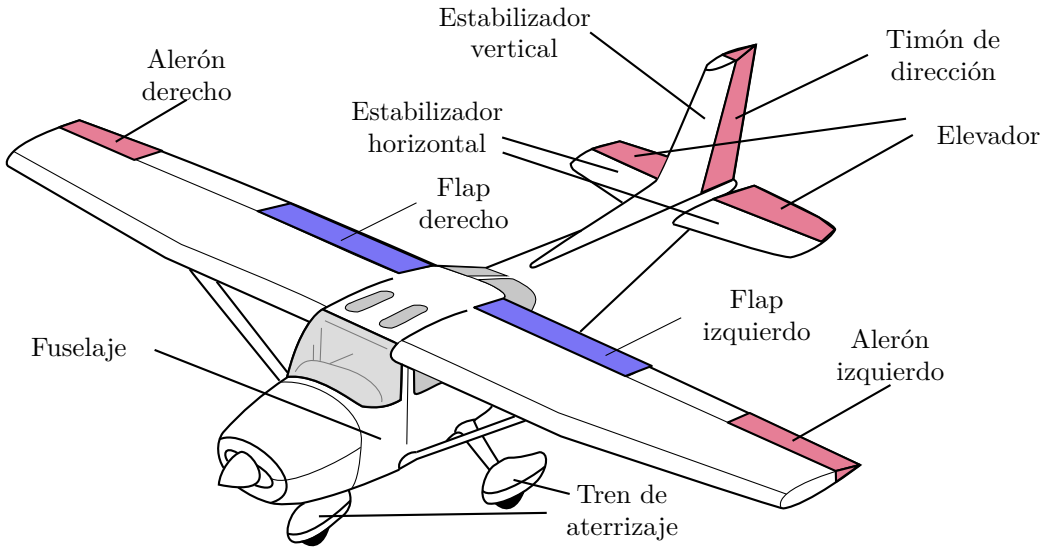


Figura 3.1: Partes de un avión. Rojo: superficies de control. Azul: superficies hipersustentadoras. Fuente: [23]

3.2. SUPERFICIES DE CONTROL

Una aeronave cuenta con una serie de superficies móviles, llamadas superficies de control, que ya se mostraron en la anterior figura 3.1 destacadas en rojo. Al hacerlas girar respecto a su eje de giro, estas crean fuerzas y momentos que alteran la orientación, rumbo o cabeceo de la nave. Generalmente, las superficies de control se colocan en la parte trasera de alguna superficie fija, como las alas o los planos de cola, y están unidas a ellas por un eje. Principalmente son 3: Timón de profundidad, timón de dirección y alerones:

- **Elevador o timón de profundidad**: El timón de profundidad, también llamado elevador, es la parte móvil del estabilizador horizontal. Produce un momento que causa un movimiento de cabeceo de la aeronave.
- **Timón de dirección**: También conocido como *rudder* por su nombre en inglés, esta superficie es la parte móvil del estabilizador vertical o deriva. Cuando se deflecta produce momentos que hacen que el avión guiñe. Durante el vuelo se suele usar para coordinar los virajes junto con los alerones, y se puede usar en los aterrizajes para contrarrestar el viento cruzado.
- **Alerones**: Situados cerca de los extremos del ala, estas superficies producen momentos para provocar un alabeo de la aeronave. Junto con el timón de dirección o *rudder*, permiten que el avión realice virajes coordinados.

Además de estas, existen los **flaps**, que son unas superficies móviles parecidas a los alerones, pero que se engloban en las llamadas superficies hipersustentadoras, resaltadas en azul en la figura 3.1. Su misión es aumentar la sustentación durante las maniobras a baja velocidad, como el despegue y el aterrizaje. El resto del tiempo permanecen fijos y alineados con el ala.

Existe un criterio de signos para las deflexiones de las superficies de control. Este se muestra en la figura 3.2.

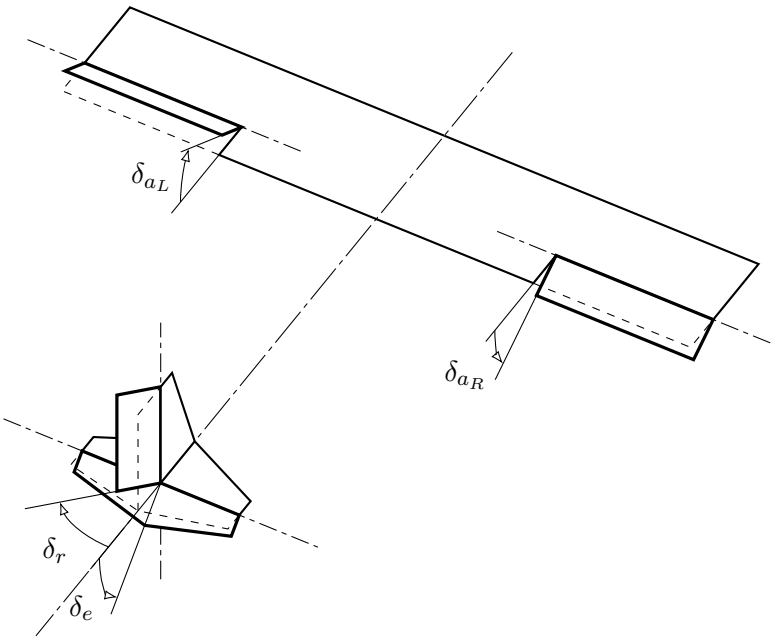


Figura 3.2: Criterio de signos de las deflexiones de las superficies de control. Fuente: [10]

La deflexión de los alerones izquierdo y derecho se denota por δ_{a_L} y δ_{a_R} respectivamente,

con los signos positivos según se indican en la figura. Estos se unen en un único término δ_a que viene dado por la expresión [10]

$$\delta_a = \frac{\delta_{a_R} - \delta_{a_L}}{2}. \tag{3.1}$$

3.3. FUERZAS Y MOMENTOS SOBRE EL AVIÓN

Una aeronave suele presentar varias superficies aerodinámicas con distintos perfiles capaces de generar fuerzas y momentos. En este momento, resulta de interés considerar los 3 planos relevantes del avión [22]:

- En primer lugar, el plano longitudinal o de cabeceo (*pitch plane*), que es vertical y contiene al eje longitudinal de la aeronave.
- En segundo lugar, el plano direccional o de guiñada (*yaw plane*), que es horizontal y contiene el movimiento conocido como guiñada (*yaw*), que coincide con el rumbo.
- Finalmente, el plano lateral o de alabeo (*roll plane*), perpendicular a los anteriores y que contiene el ángulo de alabeo o de escora (*roll* o *bank angle*).

En estos planos aparecen fuerzas y momentos que provocan los movimientos antes mencionados. Como ejemplo, la figura 3.3 ilustra las fuerzas que aparecen en el plano longitudinal. El ala principal es la mayor fuente de sustentación L y mantiene al avión en el aire. Al moverse el avión por el aire con una velocidad V, este encuentra una resistencia a su avance D paralela al vector velocidad. El peso, producto de la masa m y la aceleración de la gravedad g, siempre apunta hacia abajo y el empuje de la hélice T debe vencer al arrastre para que la aeronave mantenga una velocidad constante. En este texto se asume que el empuje T es paralelo al eje longitudinal de la nave.

La *cuerda* del ala es una línea imaginaria que une el punto más adelantado del perfil del ala, situado en el llamado *borde de ataque*, con el punto más atrasado, perteneciente al *borde de fuga* [22]. El ángulo formado por esta línea y el vector velocidad V se llama ángulo de ataque y se denota por α.

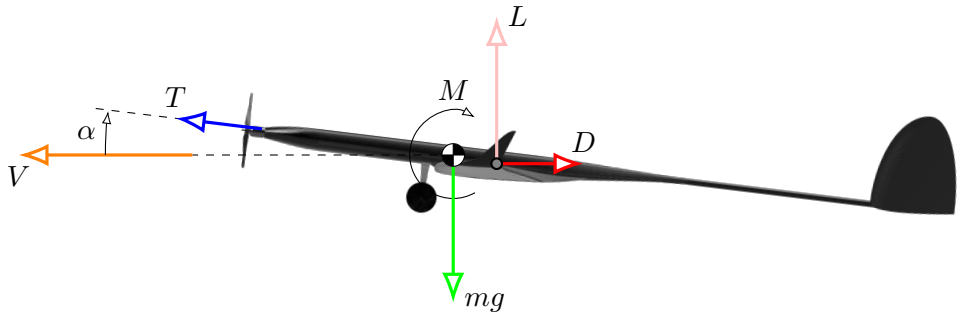


Figura 3.3: Fuerzas en el plano longitudinal del avión. Fuente: [22]

El momento neto alrededor del centro de gravedad, resultado de todas las fuerzas y momentos de origen aerodinámico en el plano longitudinal, se conoce como momento de cabeceo y se suele denotar por M en la literatura. Este viene dado por la expresión [15]

$$M = \frac{1}{2}\rho V^2 S c C_m\,, \tag{3.2}$$

donde C_m es un coeficiente aerodinámico adimensional de momento de cabeceo; S es la superficie alar; c es la cuerda media aerodinámica (una longitud que depende de la geometría del ala); ρ la densidad del aire y V la velocidad del avión respecto al aire. El valor de C_m depende de las características del avión, del ángulo de ataque α y de la deflexión del timón de profundidad, como se verá en el capítulo 4.

La fuerza aerodinámica resultante se puede desglosar en dos componentes perpendiculares: una hacia arriba y perpendicular al flujo, que constituye la sustentación L, y otra componente hacia atrás que constituye el arrastre D. Estas fuerzas vienen dadas por las expresiones [24]

$$L = \frac{1}{2}\rho V^2 S C_L \,, \tag{3.3}$$

$$D = \frac{1}{2}\rho V^2 S C_D \,, \tag{3.4}$$

siendo C_L y C_D coeficientes aerodinámicos que dependen de la geometría del perfil y del ángulo de ataque α.

Es posible simplificar los sistemas de fuerzas en los otros dos planos de la aeronave. En el plano de la guiñada *yaw* se simplifica a una fuerza lateral denotada por Y y un momento aerodinámico de guiñada denotado por N. Estos vienen dados por las expresiones [10, 24]

$$Y = \frac{1}{2}\rho V^2 S C_Y \,, \tag{3.5}$$

$$N = \frac{1}{2}\rho V^2 S b C_n \,, \tag{3.6}$$

donde b es la envergadura, la longitud del ala de extremo a extremo, y C_Y y C_n coeficientes aerodinámicos.

De igual modo, en el plano de alabeo, el momento neto se llama momento de alabeo (*roll*) y se denota por L [25], igual que la sustentación, aunque no debe confundirse con esta. Su expresión es [10]

$$L = \frac{1}{2}\rho V^2 S b C_l \,. \tag{3.7}$$

Además de estas fuerzas aerodinámicas, sobre los UAV aparece una fuerza debida al empuje de la hélice que llevan acoplada al motor. Esta se denota por T, y si el eje del motor no está alineado con el centro de gravedad, crea un momento sobre este denotado por M_T [4]:

$$M_T = z_T T \,, \tag{3.8}$$

siendo z_T la distancia entre la línea de acción del empuje T y el centro de gravedad.

3.4. MOVIMIENTO DEL AVIÓN

Un avión en vuelo controlado se mueve hacia delante y puede seguir una trayectoria rectilínea o girar, dependiendo de la posición de las superficies de control y de las fuerzas y momentos debidos a esta posición y el movimiento relativo del aire sobre ellas.

La velocidad relativa del avión respecto al aire se conoce como *airspeed*, se denota por V e influye en la magnitud todas las fuerzas y momentos aerodinámicos. La relación entre las

fuerzas y la deflexión de las superficies de control es objeto de estudio de la aerodinámica y sus resultados se pueden resumir en las ecuaciones del modelo aerodinámico del apartado 4.3.

Normalmente, el avión se mueve de forma que la cuerda del ala principal forma un cierto ángulo con el flujo de aire aparente, llamado ángulo de ataque[1] y denotado por α. Dependiendo del diseño de la aeronave, en algunos casos, la cuerda del ala forma un ángulo con el eje longitudinal del avión. No existe convenio unificado sobre el nombre de este ángulo. Algunos autores lo llaman ángulo de incidencia [21, 25] o de montaje (*mounting angle* en [26] y *rigging angle* en [15]). Esta es una cuestión geométrica que condiciona poco el comportamiento del avión y en el modelo de los capítulos siguientes no se tiene en cuenta. En esta memoria se asumirá que la cuerda y el eje longitudinal están alineados.

En determinadas condiciones de vuelo, es posible que el avión se mueva con una componente de velocidad lateral, de forma que el plano longitudinal forme un ángulo β con el viento aparente. Este ángulo se conoce como ángulo de deslizamiento o de derrape.

De forma similar, el plano del estabilizador horizontal puede formar un cierto ángulo con el eje longitudinal del avión. Esté ángulo puede llamarse ángulo de incidencia de la cola y se denotará por i_t [4, 15, 25].

Las fuerzas aerodinámicas son función de los ángulos aerodinámicos de ataque y derrape, así como la velocidad relativa al aire. Desde el punto de vista estructural, resulta de interés relacionar la magnitud de las fuerzas que actúan sobre el avión con el peso de la propia nave. Esta relación se conoce como factor de carga, dado por [27]

$$n = \frac{L}{W}\,, \tag{3.9}$$

donde L es la sustentación y W es el peso. El factor de carga es una métrica de los esfuerzos que está soportando la estructura y debe mantenerse a un valor limitado para no ponerla en riesgo. Un límite razonable podría ser 2 [22], aunque depende de las características de cada avión. Con este límite en mente se pueden planificar maniobras que resulten suaves y será relevante cuando se diseñe el controlador.

[1] *Ángulo de ataque* es una forma bastante inequívoca de referirse al ángulo formado por la proyección del vector velocidad del viento relativo a un perfil y la cuerda de este cuando el flujo se proyecta sobre el plano del propio perfil. En algunas obras como [10, 15] se usa el término *ángulo de incidencia*, pero es un término ambiguo que se puede referir a cualquier ángulo relacionado con un perfil, como el ángulo de montaje o el propio ángulo de ataque.

Parte II

MODELADO DEL SISTEMA

4. MODELO MATEMÁTICO DE LA AERONAVE

Para el diseño de cualquier controlador de vuelo o para predecir el comportamiento de una aeronave se necesita un modelo. Este tendrá varias simplificaciones, pero servirá para aproximar razonablemente la realidad y comprender de forma analítica las respuestas del sistema.

Dependiendo de las necesidades de precisión de la predicción, se suele recurrir a modelos no lineales de gran complejidad o bien a modelos linealizados. En la práctica, es más común recurrir a modelos lineales, puesto que son más sencillos y establecen relaciones claras entre las variables implicadas.

En este texto, se tomará un modelo bastante completo de la aeronave y se harán una serie de simplificaciones razonadas que reduzcan la complejidad del modelo manteniendo una precisión aceptable. Este modelo será posteriormente usado en la parte III.

El modelo se basa en unas ecuaciones diferenciales muy tratadas en la literatura. Estas se conocen como ecuaciones del movimiento de las aeronaves o ecuaciones de Bryan y se tratan con especial detalle en las publicaciones [4, 10, 15, 16, 24, 26-28].

En general, estas ecuaciones se incluyen en libros relacionados con la Mecánica del Vuelo, que es una asignatura impartida en el grado en Ingeniería Aeroespacial. Los términos de estas ecuaciones se pueden calcular mediante otras ecuaciones derivadas de la teoría aerodinámica y también se tratan en las publicaciones antes citadas.

En las anteriores referencias también se incluyen versiones linealizadas de las ecuaciones del movimiento, pero en este trabajo se ha procurado trabajar con el modelo no lineal completo, que permite cubrir o modelizar un rango más grande de condiciones de vuelo para una aeronave dada, a coste de mayor complejidad y coste computacional.

4.1. SISTEMAS DE REFERENCIA

Antes de empezar a desarrollar el modelo matemático de la aeronave, es necesario presentar los distintos sistemas de coordenadas que se usarán en las secciones siguientes.

Las magnitudes vectoriales del modelo que se presentará a continuación están referidas a distintos sistemas de coordenadas. Todos ellos son ortonormales, siguen la regla de la mano derecha y son conocidos en la literatura aeronáutica [10, 15, 26, 27].

4.1.1. Sistema de ejes tierra (*earth axes*)

El primero de los sistemas de referencia es aquel ligado a la tierra. A continuación se presentan dos alternativas.

El primer sistema de ejes tierra tiene su origen o_0 en la superficie, el eje z_0 apunta hacia el centro de la tierra y el plano x_0y_0 es tangente a la superficie, con el eje x_0 generalmente apuntando al norte y el y_0 hacia el este, como se muestra en la figura 4.1 [15].

Este sistema de coordenadas se conoce como NED (*North East Down*) y sirve para vuelos en los que la curvatura de la tierra es despreciable [15]. Como los UAVs de la Air Cargo Challenge

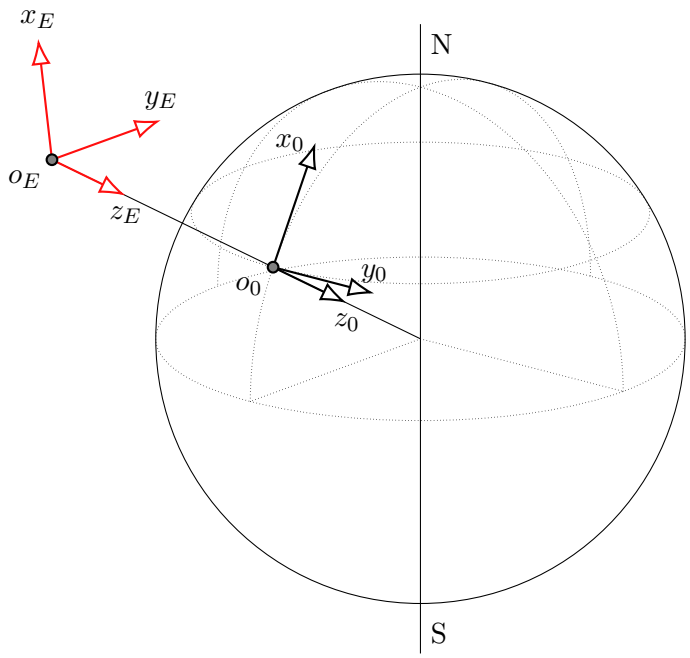


Figura 4.1: Sistemas de ejes tierra. Fuente: [15]

rara vez superan los 100 o 120 m de altitud y el alcance está limitado por la vista del piloto, esta simplificación es aceptable.

El otro sistema de referencia comúnmente usado es aquel compuesto por o_E y los ejes x_E, y_E y z_E. El plano $x_E y_E$ es paralelo al plano $x_0 y_0$, con la diferencia de que el eje x_E puede apuntar a una dirección arbitraria [15]. Esto permite establecer un sistema de referencia inercial para así calcular el desplazamiento del avión durante una maniobra relativo a un origen arbitrario.

Es importante destacar que la altitud del avión sobre el suelo, h, viene dada por la coordenada z_E del centro de gravedad del UAV, tomando un origen de ejes tierra en el suelo, cambiada de signo. Es decir, $h = -z_E$.

4.1.2. Sistema de ejes cuerpo (*aircraft body-fixed axes*)

El segundo sistema de referencia es aquel ligado a la geometría del propio avión, llamado el sistema de ejes cuerpo (*aircraft body-fixed axes* en inglés)[1].

En este sistema de referencia el origen o se toma, por lo general, en el centro de gravedad de la aeronave, el eje longitudinal x_b se define hacia delante, paralelo al eje de rotación de la hélice o bien el eje principal de inercia longitudinal; el eje y_b se toma perpendicular al plano de simetría longitudinal del avión y hacia la derecha; el eje z_b se toma hacia abajo, perpendicular a los anteriores [15]. Este sistema de ejes se muestra en la figura 4.2.

Este sistema de referencia se relaciona con el de ejes tierra mediante la actitud del avión (*aircraft attitude*), que consta de los ángulos de cabeceo θ, alabeo ϕ y guiñada ψ, representados

[1]Realmente, los ejes viento y de estabilidad también son ejes cuerpo puesto que van ligados al cuerpo del avión y tienen su origen en el centro de gravedad de este. Se ha hecho esta distinción para simplificar la comprensión y la traducción del inglés, donde los ejes cuerpo se llaman *body-fixed axes* y se engloban en los llamados *body axes* junto con los de estabilidad y los ejes viento [10].

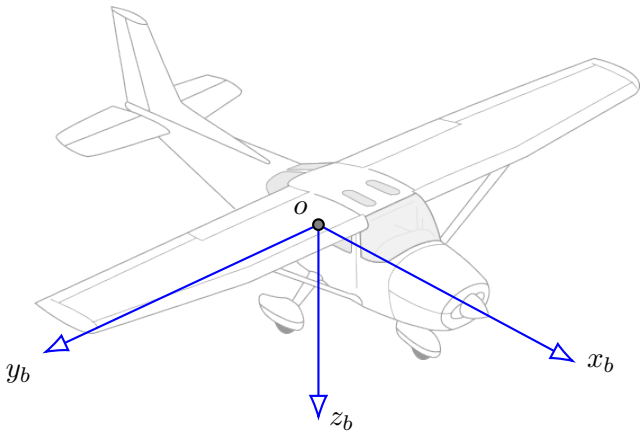


Figura 4.2: Sistema de ejes cuerpo. Fuente: [15]

en la figura 4.3. La matriz de cambio de sistema de referencia de ejes tierra a ejes cuerpo es [26]:

$$\mathbf{M}_{be} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\theta \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi - \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi + \cos\psi\sin\theta \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{4.1}$$

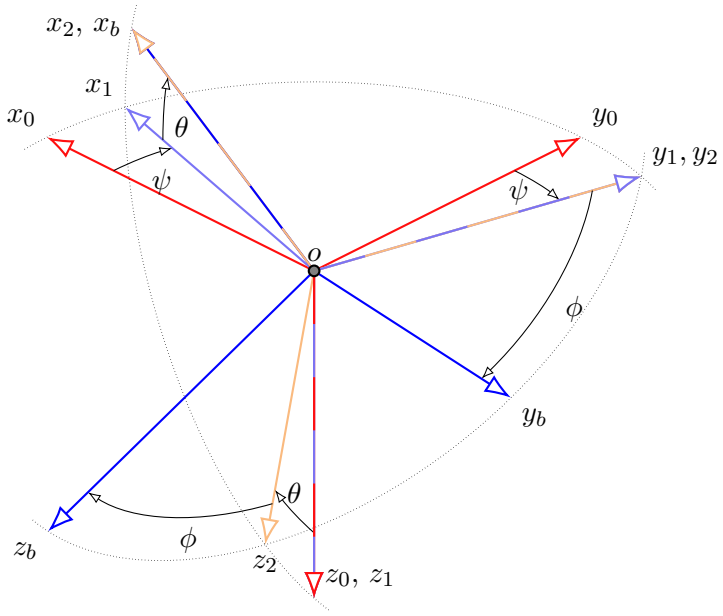


Figura 4.3: Relaciones angulares entre ejes cuerpo y ejes tierra. Fuente: [15]

Los ejes cuerpo son el sistema de referencia habitual para formular las ecuaciones del movimiento del avión. Sin embargo, en estas ecuaciones intervienen términos con magnitudes que se suelen calcular relativos a otros dos sistemas de referencia, explicados a continuación.

4.1.3. Sistemas de ejes viento (*wind axes*) y ejes de estabilidad (*stability axes*)

El sistema de ejes viento ($ox_wy_wz_w$) tiene su origen en el centro de gravedad de la aeronave y se mueve con este. El eje x_w apunta hacia delante, en sentido contrario al vector viento relativo respecto al avión (viento aparente). El ángulo formado por x_w y el plano x_by_b es el ángulo de ataque α. El eje y_w apunta hacia la derecha, formando un ángulo β, llamado ángulo de deslizamiento o de derrape (*sideslip*), con el eje y_b. El eje z_w está contenido en el plano de simetría longitudinal de la aeronave y apunta hacia abajo, formando un ángulo α con z_b [15].

Los ejes de estabilidad ($ox_sy_sz_s$), por otra parte, también tienen su origen en el centro de gravedad del avión. Se diferencian de los ejes viento en que el eje x_s coincide con la proyección del eje x_w sobre el plano longitudinal. De este modo, el eje y_s coincide con y_b y z_s coincide con z_w. Estas relaciones se pueden apreciar en la figura 4.4.

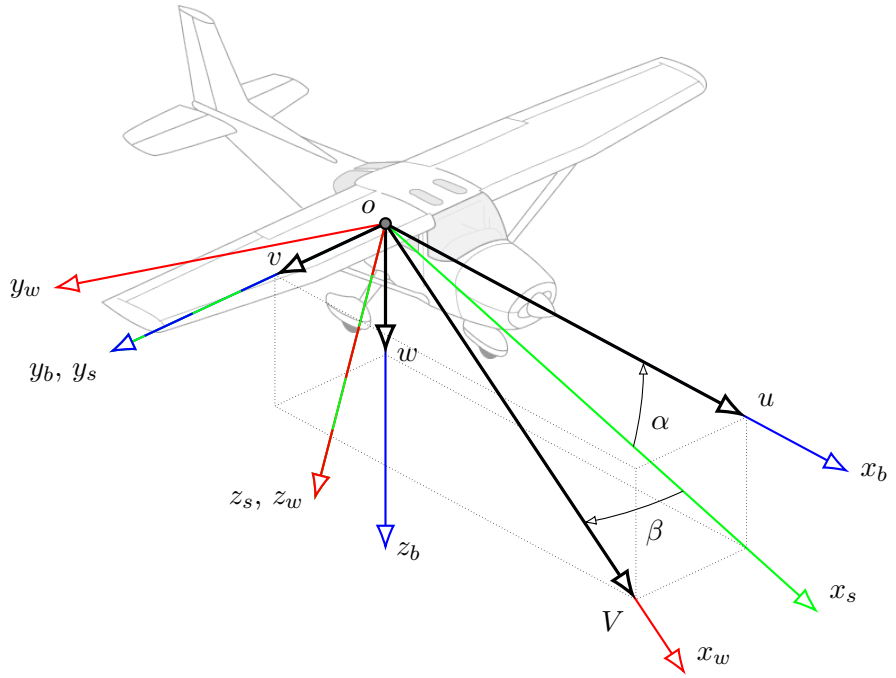


Figura 4.4: Relación ejes cuerpo, ejes viento y ejes de estabilidad. Fuente: [10]

Examinando con detenimiento la anterior figura, se pueden obtener las expresiones:

$$\mathbf{V} = \begin{bmatrix} u & v & w \end{bmatrix}^\intercal , \tag{4.2}$$

$$u = V \cos\alpha \cos\beta , \tag{4.3}$$

$$v = V \sin\beta , \tag{4.4}$$

$$w = V \sin\alpha \cos\beta , \tag{4.5}$$

$$V \equiv |\mathbf{V}| = \sqrt{u^2 + v^2 + w^2} , \tag{4.6}$$

siendo u, v y w las componentes en ejes cuerpo del vector velocidad aerodinámica $\mathbf{V}$, cuyo módulo se denota por V. Las relaciones entre los distintos ejes también se pueden ver en la figura 4.5.

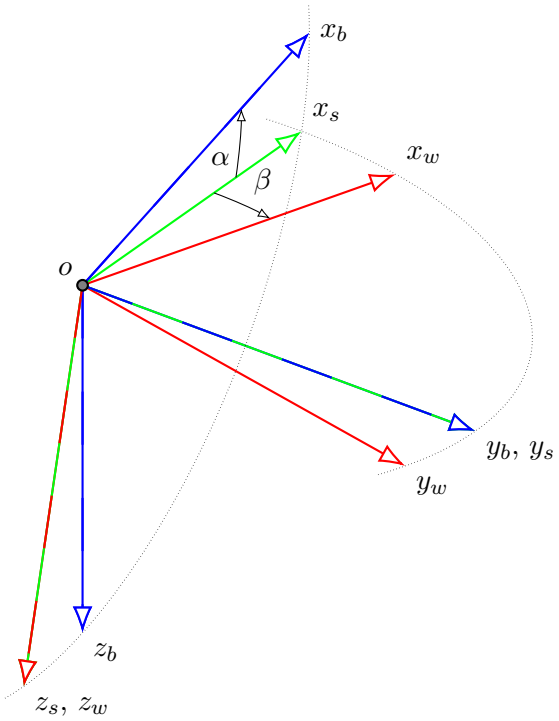


Figura 4.5: Relación entre ejes cuerpo, viento y de estabilidad. Fuente: [10]

Los parámetros aerodinámicos, como se explica más adelante en el apartado 4.3, suelen estar referidos a los ejes de estabilidad, mientras que el *software* Xflr5, comentado en el apartado 4.6, calcula algunos en ejes cuerpo. Para convertirlos, basta usar la matriz del cambio de sistema de referencia $\mathbf{M}_{bs}$ [26]:

$$\mathbf{M}_{bs} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}. \tag{4.7}$$

Si se considera solo el plano longitudinal y se asume β nulo, los ejes de estabilidad y viento coinciden, reduciéndose el problema al mostrado en la figura 4.6.

Aquí se puede ver claramente el ángulo formado entre el horizonte local (paralelo al plano $x_E y_E$) y la velocidad V. A este ángulo se lo conoce como el ángulo de pendiente de la trayectoria (*path angle*) γ, que se define como [10]:

$$\gamma = \theta - \alpha \tag{4.8}$$

Una vez comentados todos los sistemas de referencia de interés, se puede proceder a formular las ecuaciones del movimiento de la aeronave en el sistema que más convenga.

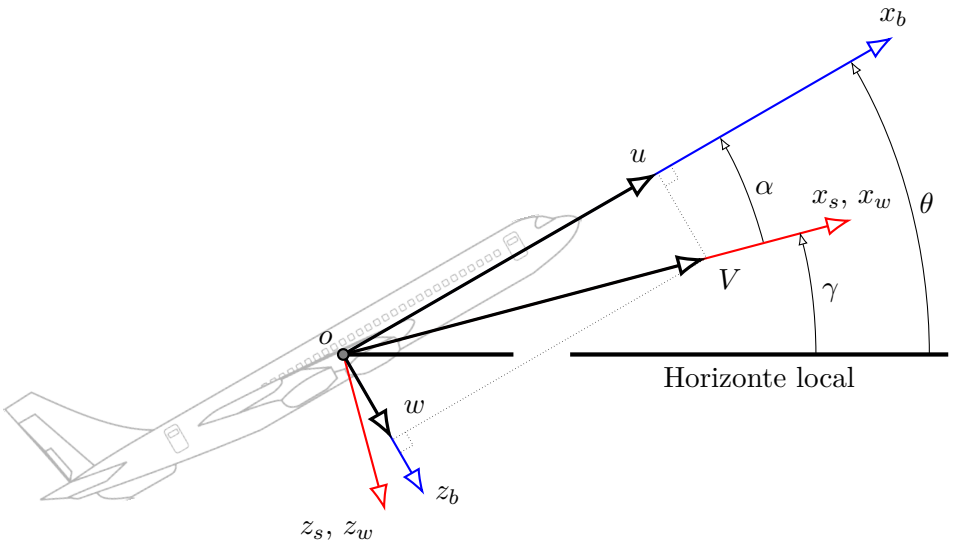


Figura 4.6: Relación entre ejes cuerpo, ejes viento y el horizonte en el plano de simetría del avión. Fuente: [15]

4.2. ECUACIONES DEL MOVIMIENTO

Las ecuaciones del movimiento de la aeronave son aquellas que relacionan las aceleraciones lineales y angulares del avión con las fuerzas y momentos aplicados. Se derivan de la segunda ley de Newton.

Dependiendo del grado de exactitud que se pretenda conseguir con el modelo y de las variables que se quieran relacionar, se pueden plantear 9 ecuaciones para definir la dinámica completa del avión, 4 para expresar solo la dinámica longitudinal y 3 para relacionar las velocidades del avión en ejes cuerpo con las velocidades en ejes tierra. Para plantear estas ecuaciones, se parte de las siguientes suposiciones [10, 26]:

- El avión es un sólido rígido y su distribución de masa es invariante con el tiempo. No se consideran los efectos aeroelásticos.
- La aeronave es simétrica respecto del plano $x_b z_b$.
- El empuje debido al motor es paralelo al eje longitudinal x_b.
- Los momentos giroscópicos debidos a la planta propulsiva son despreciables.
- La tierra es un sistema de referencia fijo. Los efectos de la rotación de la tierra son despreciables.
- No hay viento. La velocidad del aire con respecto a la tierra es nula.
- Atmósfera en calma, sin turbulencia.
- El avión vuela a una altitud tal que el efecto suelo es despreciable.
- La altitud del vuelo es suficientemente baja como para no considerar la curvatura de la tierra (tierra plana).
- La gravedad es constante para cualquier altitud. Los centros de gravedad y de masa son coincidentes.
- Ángulos pequeños de ataque y derrape. No está considerada la entrada en pérdida.

4.2.1. Ecuaciones con 6 grados de libertad

La forma más general de expresar las ecuaciones del movimiento es con 6 grados de libertad. Estas se pueden plantear en ejes cuerpo y en ejes viento, obteniendo en cada caso las expresiones de distintas variables de interés. Aquí se han tomado las ecuaciones que relacionan aquellas variables más fáciles de medir con los sensores disponibles.

La forma más habitual de expresar las ecuaciones de la dinámica es en ejes cuerpo. Estas permiten calcular las derivadas[2] de las velocidades en ejes cuerpo (u, v y w) mediante las siguientes expresiones, obtenidas de Klein y Morelli [10],

$$\dot{u} = rv - qw + \frac{\bar{q}S}{m}C_X - g\sin\theta + \frac{T}{m}\,, \tag{4.9}$$

$$\dot{v} = pw - ru + \frac{\bar{q}S}{m}C_Y + g\cos\theta\sin\phi\,, \tag{4.10}$$

$$\dot{w} = qu - qv + \frac{\bar{q}S}{m}C_Z + g\cos\theta\cos\phi\,, \tag{4.11}$$

donde p, q y r denotan las velocidades angulares de alabeo (*roll*), cabeceo (*pitch*) y guiñada (*yaw*) respectivamente; $\bar{q}$ es la *presión dinámica*, de valor $\frac{1}{2}\rho V^2$ con ρ la densidad del aire; S es la superficie alar; C_X, C_Y y C_Z son coeficientes de fuerzas aerodinámicas en ejes cuerpo; T es el empuje generado por la hélice; m es la masa de la aeronave y g es la aceleración de la gravedad. Estas expresiones se pueden expresar de forma más compacta usando notación matricial, como se pueden encontrar en otros textos [24, 25], pero aquí se ha optado por las variantes escalares por simplicidad y claridad.

Desde el punto de vista de la identificación, son más convenientes las expresiones en ejes viento, que relacionan directamente la velocidad relativa al viento V o *airspeed* y los ángulos aerodinámicos α y β con las fuerzas y momentos que actúan sobre el avión. Esto es porque las variables aerodinámicas se pueden medir directamente con sensores, cosa que no es factible con las velocidades en ejes cuerpo. Las ecuaciones [10] de las derivadas respecto del tiempo de estas magnitudes aerodinámicas son:

$$\begin{aligned}\dot{V} = &-\frac{\bar{q}S}{m}C_{D_w} + \frac{T}{m}\cos\alpha\cos\beta \\ &+ g(\cos\phi\cos\theta\sin\alpha\cos\beta + \sin\phi\cos\theta\sin\beta - \sin\theta\cos\alpha\cos\beta)\,,\end{aligned} \tag{4.12}$$

$$\begin{aligned}\dot{\alpha} = &-\frac{\bar{q}S}{mV\cos\beta}C_L + q - \tan\beta(p\cos\alpha + r\sin\alpha) - \frac{T\sin\alpha}{mV\cos\beta} \\ &+ \frac{g}{V\cos\beta}(\cos\phi\cos\theta\cos\alpha + \sin\theta\sin\alpha)\,,\end{aligned} \tag{4.13}$$

$$\begin{aligned}\dot{\beta} = &\frac{\bar{q}S}{mV}C_{Y_w} + p\sin\alpha - r\cos\alpha + \frac{g}{V}\cos\beta\sin\phi\cos\theta \\ &+ \frac{\sin\beta}{V}\left(g\cos\alpha\sin\theta - g\sin\alpha\cos\phi\cos\theta + \frac{T\cos\alpha}{m}\right)\,,\end{aligned} \tag{4.14}$$

donde C_L es el coeficiente de sustentación y C_{D_w} y C_{Y_w} son los coeficientes de arrastre y de fuerza lateral en ejes viento, dados por

$$C_{D_w} = C_D\cos\beta - C_Y\sin\beta\,, \tag{4.15}$$

$$C_{Y_w} = C_Y\cos\beta + C_D\sin\beta\,. \tag{4.16}$$

[2]En los libros de mecánica del vuelo, es habitual encontrar la notación de Newton para las derivadas de variables respecto del tiempo ($\dot{x}$), en lugar de otras como la de Leibniz ($\frac{dx}{dt}$) o de Lagrange (x').

Por otro lado, las aceleraciones angulares de cabeceo $\dot{q}$, alabeo $\dot{p}$ y guiñada $\dot{r}$ vienen dadas por las expresiones [10]:

$$\dot{p} = (c_1 r + c_2 p - c_4 I_p \Omega_p) q + \bar{q} S b (c_3 C_l + c_4 C_n)\,, \tag{4.17}$$

$$\dot{q} = (c_5 p + c_7 I_p \Omega_p) r - c_6 (p^2 - r^2) + c_7 \bar{q} S \bar{c} C_m\,, \tag{4.18}$$

$$\dot{r} = (c_8 p - c_2 r - c_9 I_p \Omega_p) q + \bar{q} S b (c_9 C_n + c_4 C_l)\,, \tag{4.19}$$

donde C_l, C_m y C_n son los coeficientes aerodinámicos de momentos de alabeo (L), cabeceo (M) y guiñada (N) respectivamente; b es la envergadura de la aeronave; $\bar{c}$ es la cuerda media aerodinámica; Ω_p es la velocidad angular de la hélice (*propeller* en inglés, de ahí el subíndice p); I_p es el momento de inercia de la hélice, que de aquí en adelante se considera despreciable, y los coeficientes c_i son constantes de inercia que siguen las ecuaciones

$$\begin{aligned}
c_1 &= [(I_y - I_z) I_z - I_{xz}^2]/\Gamma\,, & \Gamma &= I_x I_z - I_{xz}^2\,,\\
c_2 &= [(I_x - I_y + I_z) I_{xz}]/\Gamma\,, & c_3 &= I_z/\Gamma\,,\\
c_4 &= I_{xz}/\Gamma\,, & c_5 &= (I_z - I_x)/I_y\,,\\
c_6 &= I_{xz}/I_y\,, & c_7 &= 1/I_y\,,\\
c_8 &= [(I_x - I_y) I_x - I_{xz}^2]/\Gamma\,, & c_9 &= I_x/\Gamma\,.
\end{aligned} \tag{4.20}$$

Aquí, I_x, I_y e I_z son momentos de inercia referidos a los ejes cuerpo del avión e I_{xz} es el producto de inercia asociado a los ejes x_b y z_b, considerando un tensor de inercia en ejes cuerpo dado por [25]

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & -I_{xz} \\ 0 & I_y & 0 \\ -I_{xz} & 0 & I_z \end{bmatrix}. \tag{4.21}$$

Por su parte, las derivadas de los ángulos de actitud del avión respecto del tiempo se calculan mediante las expresiones siguientes [10, 25], conocidas como ecuaciones cinemáticas (*kinematic equations*):

$$\dot{\phi} = p + (q \sin\phi + r \cos\phi) \tan\theta\,, \tag{4.22}$$

$$\dot{\theta} = q \cos\phi - r \sin\phi\,, \tag{4.23}$$

$$\dot{\psi} = \frac{q \sin\phi + r \cos\phi}{\cos\theta}\,. \tag{4.24}$$

Ecuaciones de navegación

En ocasiones es útil escribir la velocidad del avión relativa a los ejes tierra. Integrando estas se puede calcular la distancia recorrida respecto al suelo, la misma que se puede obtener mediante un sensor GPS. Estas ecuaciones se conocen en inglés como *navigation equations*, lo que se podría traducir como ecuaciones de navegación.

Las velocidades en ejes tierra se relacionan con las velocidades en ejes cuerpo por la actitud de la aeronave. Las expresiones son [10, 25]:

$$\begin{aligned}
\dot{x}_E = {} & u \cos\psi \cos\theta + v(\cos\psi \sin\theta \sin\phi - \sin\psi \cos\phi)\\
& + w(\cos\psi \sin\theta \cos\phi + \sin\psi \sin\theta)\,,
\end{aligned} \tag{4.25}$$

$$\begin{aligned}
\dot{y}_E = {} & u \sin\psi \cos\theta + v(\sin\psi \sin\theta \sin\phi - \cos\psi \cos\phi)\\
& + w(\sin\psi \sin\theta \cos\phi + \cos\psi \sin\theta)\,,
\end{aligned} \tag{4.26}$$

$$\dot{h} \equiv -\dot{z}_E = u \sin\theta - v \cos\theta \sin\phi - w \cos\theta \cos\phi\,. \tag{4.27}$$

Nótese que $\dot{h} = -\dot{z}_E$, indicando, como ya se dijo en el apartado 4.1, que la altitud h de la aeronave sobre la tierra viene dada por $h = -z_E$.

En forma matricial, estas ecuaciones se escriben como

$$\begin{bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{z}_E \end{bmatrix} = \mathbf{M}_{eb} \begin{bmatrix} u \\ v \\ w \end{bmatrix} , \tag{4.28}$$

donde $\mathbf{M}_{eb}$ es la matriz del cambio de sistema de referencia de ejes cuerpo a ejes tierra, dada por la ecuación (4.1) [25].

4.2.2. Simplificación a dinámica longitudinal

En algunos casos de estudio, no es conveniente considerar la dinámica completa de la aeronave, sino que se puede ceñir a la dinámica longitudinal. Esta es una simplificación habitual para empezar con la identificación, ya que reduce considerablemente el número de parámetros y variables del sistema. De este modo es más sencillo evaluar el rendimiento del proceso de identificación y encontrar posibles fallos.

Para reducir el modelo del anterior apartado a la dinámica longitudinal, hay que considerar que no se aplica el timón de dirección ni los alerones, que se parte de vuelo nivelado (sin alabeo), que no existen fuerzas ni momentos laterales y que los ángulos de alabeo, guiñada y deslizamiento son nulos en todo momento.

Con esta premisa, las ecuaciones de la dinámica de la aeronave se reducen a [29-31]:

$$\dot{V} = -\frac{\bar{q}S}{m} C_D + \frac{T}{m} \cos\alpha + g \cos\theta \sin\alpha \,, \tag{4.29}$$

$$\dot{\alpha} = -\frac{\bar{q}S}{mV} C_L + q - \frac{T \sin\alpha}{mV} + \frac{g}{V} (\cos\theta \cos\alpha + \sin\theta \sin\alpha) \,, \tag{4.30}$$

$$\dot{q} = \frac{\bar{q}S\bar{c}}{I_y} C_m \,, \tag{4.31}$$

$$\dot{\theta} = q \,, \tag{4.32}$$

$$\dot{x}_E = V \cos(\theta - \alpha) \,, \tag{4.33}$$

$$\dot{h} \equiv -\dot{z}_E = V \sin(\theta - \alpha) \,. \tag{4.34}$$

4.3. MODELO AERODINÁMICO

Los coeficientes aerodinámicos son difíciles de modelar de forma analítica, por lo que resulta complicado fijar una estructura para su modelo. La estrategia habitual consiste en definirlo mediante un desarrollo en serie de Taylor multivariable.

Dependiendo de la precisión que se pretenda obtener, los coeficientes aerodinámicos se pueden escribir como función de las variables aerodinámicas y de sus derivadas respecto del tiempo, deflexión de las superficies de control, etc. Cuánto más complejo sea el modelo, más parámetros se requerirán para definirlo y más difícil será estimar sus valores en el proceso de identificación. Este tema se trata en profundidad en [10]. Para este trabajo se tomará un modelo simplificado, dado por [10]

$$C_i = C_i \left(\alpha, \beta, \boldsymbol{\delta}, \frac{\boldsymbol{\Omega} l}{V} \right)$$

para $i = D$, Y, L, l, m, n, donde

- $\boldsymbol{\delta}$ es la deflexión de cualquier superficie de control
- $\boldsymbol{\Omega}$ es la velocidad angular en ejes de estabilidad
- l es una longitud característica

Las expresiones de cada coeficiente aerodinámico son las siguientes y están definidas alrededor de una condición de vuelo concreta dada por la velocidad de referencia V_0 [10]:

$$\begin{aligned} C_L = {} & C_{L0} + C_{L\alpha} \cdot \alpha + C_{L\dot{\alpha}} \cdot \frac{\dot{\alpha}\bar{c}}{2V_0} + C_{Lq} \cdot \frac{q\bar{c}}{2V_0} + C_{LV} \cdot \frac{V - V_0}{V_0} \\ & + C_{L\delta_e} \cdot \delta_e + C_{L\delta_f} \cdot \delta_f + C_{Li_t} \cdot i_t, \end{aligned} \tag{4.35}$$

$$\begin{aligned} C_D = {} & C_{D0} + C_{D\alpha} \cdot \alpha + C_{D\alpha^2} \cdot \alpha^2 + C_{Dq} \cdot \frac{q\bar{c}}{2V_0} + C_{DV} \cdot \frac{V - V_0}{V_0} \\ & + C_{D\delta_e} \cdot \delta_e + C_{D\delta_f} \cdot \delta_f + C_{Di_t} \cdot i_t, \end{aligned} \tag{4.36}$$

$$\begin{aligned} C_Y = {} & C_{Y\beta} \cdot \beta + C_{Y\dot{\beta}} \cdot \frac{\dot{\beta}b}{2V_0} + C_{Yp} \cdot \frac{pb}{2V_0} + C_{Yr} \cdot \frac{rb}{2V_0} \\ & + C_{Y\delta_a} \cdot \delta_a + C_{Y\delta_r} \cdot \delta_r, \end{aligned} \tag{4.37}$$

$$\begin{aligned} C_l = {} & C_{l\beta} \cdot \beta + C_{l\dot{\beta}} \cdot \frac{\dot{\beta}b}{2V_0} + C_{lp} \cdot \frac{pb}{2V_0} + C_{lr} \cdot \frac{rb}{2V_0} \\ & + C_{l\delta_a} \cdot \delta_a + C_{l\delta_r} \cdot \delta_r, \end{aligned} \tag{4.38}$$

$$\begin{aligned} C_m = {} & C_{m0} + C_{m\alpha} \cdot \alpha + C_{m\dot{\alpha}} \cdot \frac{\dot{\alpha}\bar{c}}{2V_0} + C_{mq} \cdot \frac{q\bar{c}}{2V_0} + C_{mV} \cdot \frac{V - V_0}{V_0} \\ & + C_{m\delta_e} \cdot \delta_e + C_{m\delta_f} \cdot \delta_f + C_{mi_t} \cdot i_t, \end{aligned} \tag{4.39}$$

$$\begin{aligned} C_n = {} & C_{n\beta} \cdot \beta + C_{n\dot{\beta}} \cdot \frac{\dot{\beta}b}{2V_0} + C_{np} \cdot \frac{pb}{2V_0} + C_{nr} \cdot \frac{rb}{2V_0} \\ & + C_{n\delta_a} \cdot \delta_a + C_{n\delta_r} \cdot \delta_r, \end{aligned} \tag{4.40}$$

donde δ_e es la deflexión del timón de profundidad, δ_f es la deflexión de los flaps, i_t es la posición angular del estabilizador horizontal (generalmente fija durante el vuelo), δ_a es la deflexión media de los alerones (véase la ecuación (3.1)) y δ_r es la deflexión del timón de dirección.

Los términos C_{ij} con $j = \alpha$, β, $\dot{\alpha}$, $\dot{\beta}$, p, q, r, V, δ de las ecuaciones (4.35) a (4.40) se conocen como derivadas aerodinámicas y son los parámetros a identificar con el algoritmo detallado en el capítulo 8. Existen formas dimensionales y adimensionales de estos coeficientes, y no existe un convenio unificado sobre la forma de obtener estas últimas. En el caso de este TFG, se han tomado las variantes adimensionales siguiendo las expresiones de [4, 10], aunque otros autores recurren a expresiones ligeramente diferentes [30-32].

Los anteriores coeficientes están referidos al sistema de ejes de estabilidad. Al pasarlos a ejes cuerpo, se convierten en los coeficientes C_X, C_Y y C_Z de las ecuaciones (4.9) a (4.11). La transformación viene dada por:

$$\begin{bmatrix} C_X \\ C_Y \\ C_Z \end{bmatrix}_b = \mathbf{M}_{bs} \begin{bmatrix} -C_D \\ C_Y \\ -C_L \end{bmatrix}_s, \tag{4.41}$$

donde $\mathbf{M}_{bs}$ es la matriz de cambio de sistema de referencia de ejes de estabilidad a ejes cuerpo y quedó definida en la ecuación (4.7).

4.4. MODELO DE EMPUJE

El modelo de empuje es una representación matemática del comportamiento del grupo propulsivo del avión, y permite obtener la magnitud del empuje T (*thrust*) en función de la velocidad de la aeronave V y la posición de la palanca de potencia (*throttle*) δ_p por medio de alguna ecuación

$$T = f(V, \delta_p)\,. \tag{4.42}$$

En esta sección se aborda el modelado desde un enfoque experimental relativamente simple. Para un modelo más completo y con mayor fundamento teórico, puede consultarse el apartado 5.2.

Dependiendo del tipo de propulsión que presente una aeronave (hélice, turbohélice, turbofán, etc.) se puede recurrir a distintas estructuras del modelo. En el caso de los UAV tratados en este texto, el grupo propulsivo se compone de una hélice bipala de paso fijo movida por un motor eléctrico de tipo *brushless* DC (sin escobillas, BLDC). Para los aeromodelos de la ACC, normalmente se recurre a hélices con diámetros de entre 10 y 15 pulgadas y pasos de entre 5 y 9 pulgadas, como la mostrada en la figura 4.7. De esta forma se obtienen empujes máximos de entre 10 y 20 N que decaen rápidamente con la velocidad [33].

Figura 4.7: Fotografía de una hélice bipala T-motor 13x6,5. Cortesía de T-motor: [34]

Una forma relativamente sencilla de obtener modelos de empuje es mediante un banco de pruebas como el de la figura 4.8, que permite medir el empuje generado por una combinación de hélice y motor. Para comprobar la dependencia del empuje con la velocidad, el equipo Xtra2 pudo disponer de un túnel de viento en la UPV, mostrado en la figura 4.8b, para realizar los ensayos probando distintas hélices. Para la medida del empuje, el banco cuenta con una célula de carga; mientras que la velocidad se puede obtener de un tubo de Pitot instalado en el bastidor, cuya calibración se detalla en el apartado 6.2.1, o bien de la instrumentación propia del túnel de viento.

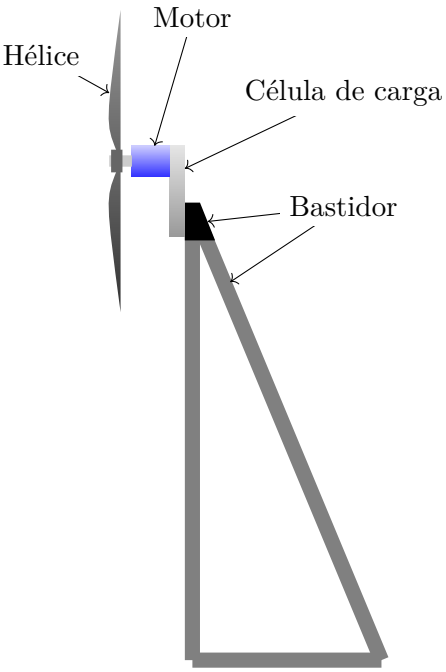
Una vez completado el ensayo para una planta propulsiva concreta, se pueden graficar los datos experimentales y ajustar estos a curvas o superficies como la mostrada en la figura 4.9, que se corresponde con el modelo de empuje del «Favara» obtenido en [4]. En esta referencia, el empuje en unidades del sistema internacional (N) viene dado por la expresión

$$T(\delta_p, V) = \begin{cases} T_S(\delta_p) - 0{,}05V - 0{,}014V^2 & \text{si} \quad V \leq 27{,}654\,\text{m/s} \\ 0 & \text{si} \quad V > 27{,}654\,\text{m/s} \end{cases}, \tag{4.43}$$

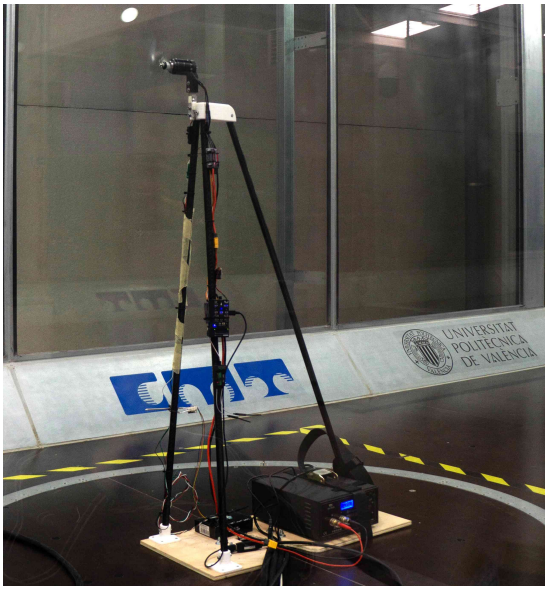
siendo T_S el *empuje estático* (empuje a velocidad nula), definido como

$$T_S(\delta_p) = \frac{12{,}089}{1 + \exp(-9{,}721\delta_p + 4{,}022)} - 0{,}212 \quad \text{donde} \quad \delta_p \in [0, 1]. \tag{4.44}$$

Examinando la figura 4.9 y la ecuación (4.43), se puede comprobar que el empuje decae con la velocidad desde su valor máximo—correspondiente al empuje estático—hasta hacerse nulo. A velocidades superiores, la hélice solo genera arrastre y las lecturas de la célula de carga se vuelven



(a) Esquema del banco de pruebas.



(b) Fotografía del banco de pruebas en el túnel de viento del CMT de la UPV.

Figura 4.8: Banco de pruebas de motores de Xtra2.

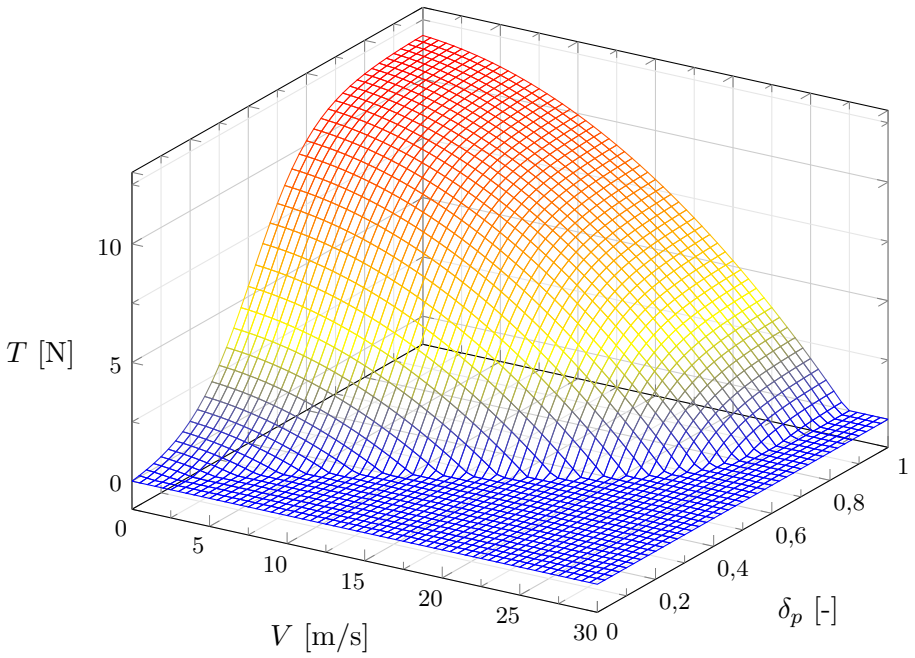


Figura 4.9: Representación 3D del modelo de empuje del «Favara». Fuente: [4]

negativas (empuje negativo). Sin embargo, el modelo no pretende reflejar este comportamiento y estos valores negativos se cuentan como nulos.

El empuje estático realmente no sigue una función suave con respecto a la palanca de gases como indica la ecuación (4.44). Esta es una aproximación a una sigmoide obtenida mediante mínimos cuadrados que representa de forma aceptable la zona muerta de la palanca de potencia a valores bajos (de 0 a 0,1) y la saturación del empuje a valores altos de δ_p (de 0,7 a 1).

Una forma más precisa hubiera sido considerar estos tres tramos como funciones distintas y, por tanto, que el empuje estático fuera una función a trozos. El ajuste de la ecuación (4.44), presentado en [4] y que se usará en este texto tiene la ventaja de que permite modelar de forma aproximada con una única función continua y suave el comportamiento no lineal del empuje estático, y por eso se ha tomado como referencia.

4.5. VECTORES DE ESTADO Y DE ENTRADA

Una vez planteadas las ecuaciones de la dinámica de la aeronave, se pueden concentrar todas las variables de estado del avión en un único vector de 12 componentes de la forma

$$\mathbf{x} = \begin{bmatrix} V & \alpha & \beta & p & q & r & \phi & \theta & \psi & x_E & y_E & h \end{bmatrix}^\intercal , \tag{4.45}$$

donde:

- V es el módulo del vector velocidad aerodinámica en m/s.
- α y β son los ángulos aerodinámicos de ataque y deslizamiento respectivamente, ambos en rad.
- p, q y r son las velocidades angulares de alabeo (*roll*), cabeceo (*pitch*) y guiñada (*yaw*) en ejes cuerpo respectivamente en rad/s.
- ϕ, θ y ψ son los ángulos de alabeo (*roll*), cabeceo (*pitch*) y guiñada (*yaw*) respectivamente en rad, conocidos en conjunto como la actitud del avión (*aircraft attitude*).
- x_E, y_E y h son las coordenadas del centro de gravedad de la aeronave en el sistema de ejes tierra, donde $h = -z_E$ es la altitud respecto del suelo, todas en m.

Si se considera solo la dinámica longitudinal, el vector de estado se simplifica a

$$\mathbf{x} = \begin{bmatrix} V & \alpha & q & \theta & x_E & h \end{bmatrix}^\intercal . \tag{4.46}$$

De igual modo, el vector de entrada asociado al modelo completo es

$$\mathbf{u} = \begin{bmatrix} \delta_e & \delta_p & i_t & \delta_f & \delta_a & \delta_r \end{bmatrix}^\intercal , \tag{4.47}$$

donde:

- δ_e es la deflexión del timón de profundidad o elevador en rad.
- δ_p es la posición angular de la palanca de potencia o de gases, comprendida entre 0 y 1. Es adimensional.
- i_t es el ángulo de incidencia del estabilizador horizontal, generalmente fijo durante el vuelo, en rad.
- δ_f es la posición angular de los flaps en rad.
- δ_a es la deflexión media de los alerones, definida por la ecuación (3.1), en rad.
- δ_r es la deflexión del timón de dirección o *rudder* en rad.

La versión reducida a la dinámica longitudinal es

$$\mathbf{u} = \begin{bmatrix} \delta_e & \delta_p & i_t & \delta_f \end{bmatrix}^\mathsf{T} . \tag{4.48}$$

Estos vectores serán tenidos en cuenta para los capítulos siguientes, en los que se plantearán más ecuaciones del modelo que depende de ellos y, especialmente, en el diseño de los bloques funcionales de la siguiente parte de la memoria.

4.6. MODELOS MATEMÁTICOS DE UAV DE XTRA2

El modelo matemático de una aeronave consiste en las ecuaciones diferenciales descritas en las secciones anteriores junto a los valores numéricos de todos los parámetros. Los valores numéricos se pueden obtener por métodos experimentales—túnel de viento o identificación—o bien mediante aproximaciones teóricas. Se puede llamar modelo *a priori* a todos los valores de los coeficientes que se tienen antes de realizar un proceso de identificación del avión. Estos valores constituyen el punto de partida para la identificación, los valores iniciales.

Para obtener los valores *a priori* de todos los parámetros es necesario recurrir a varios programas de cálculo o bien a pruebas experimentales.

Por un lado, los parámetros inerciales se pueden obtener de programas de CAD (*Computer-Aided Design*), introduciendo un modelo completo del avión y añadiendo la densidad del material de cada pieza. Con esto se pueden calcular la masa total, los momentos y productos de inercia y la posición del centro de gravedad.

Para obtener el modelo de empuje, es necesario recurrir a datos del fabricante o bien realizar pruebas estáticas y dinámicas en banco de ensayos. Con esto se estima el empuje máximo, su variación con la velocidad y su dependencia de la posición de la palanca de potencia.

La obtención de los parámetros aerodinámicos es la más compleja. Una primera estimación se puede obtener mediante el programa libre Xflr5 [35]. Este programa permite obtener los coeficientes aerodinámicos en distintos regímenes de trabajo, así como las derivadas aerodinámicas de control en ejes cuerpo. También permite obtener la superficie alar, cuerda media aerodinámica, envergadura, etc.

El mayor problema de Xflr5 es que su método de cálculo se basa en la teoría de las alas de Lanchester-Prandtl [36] y otros métodos relacionados, que pueden llevar a grandes discrepancias con la realidad en casos de flujo compresible o viscoso y con ciertas plantas alares. Un método más exacto pero de gran coste computacional es el CFD (*Computer Fluid Dynamics*, Mecánica de fluidos computacional), que consiste en simular mediante métodos numéricos el comportamiento del fluido en un dominio en el espacio. Esta técnica permite obtener mejores estimaciones, especialmente del arrastre aerodinámico y su valor a α nulo, C_{D0}.

En las páginas siguientes se incluyen los parámetros teóricos o *a priori* de dos aeromodelos diseñados por Xtra2, sin haberse identificado en vuelo. La metodología seguida para obtener estos valores se puede consultar en [4].

4.6.1. Modelo del XTRA23 «Favara»

En la ACC de 2022, Xtra2 participó con su prototipo XTRA23 «Favara», mostrado en la figura 4.10. Este avión permitió al equipo alcanzar un 4.º puesto en esta edición, celebrada en la

Figura 4.10: Fotografía del XTRA23 «Favara» en la Air Cargo Challenge 2022. Fuente: [38]

ciudad alemana de Múnich. El informe técnico presentado para la competición, que contiene muchos detalles del diseño y fabricación, está disponible para su consulta en [37].

El antiguo alumno de la UPV Jorge Castillo Torres, por entonces miembro de Xtra2 UPV, incluyó en su TFG [4] un modelo no lineal de este avión[3]. En su trabajo el modelo se usa para diseñar un controlador de vuelo LQR, y utiliza las mismas ecuaciones que las propuestas en este capítulo.

Dada la procedencia y consiguiente fiabilidad de este modelo, se ha decidido emplearlo como referencia para desarrollar el controlador y el UKF expuestos en la parte siguiente. No obstante, hay que aclarar que las diferencias entre modelos de esta clase de UAV están en los valores de los parámetros del modelo, por lo que resulta inmediato, dados unos valores de estos parámetros, acondicionar los algoritmos de control para otra aeronave.

El de Jorge Castillo no fue el único TFG inspirado por el «Favara». Otros desarrolladores del avión aprovecharon el trabajo invertido para su tema de proyecto fin de carrera, como en el diseño de los flaps, tema central de [38]. De forma similar, José Morcillo, un año antes, documentó los avances del equipo y las lecciones aprendidas sobre la fabricación de aeromodelos en su proyecto de fin de grado [3]. Con este trabajo, se preparó el camino para los siguientes prototipos que fabricaría el equipo hasta la actualidad.

El «Favara» tenía las características inerciales y geométricas de la tabla 4.1. Estos valores se conocen con exactitud porque son parámetros de diseño del avión, geométricos, invariantes o fácilmente medibles, de forma que se consideran conocidos y no es necesario identificarlos.

Como particularidad, el modelo de este avión considera una distancia vertical entre el eje del motor y el centro de gravedad de 9 mm, de modo que el empuje crea un momento de cabeceo hacia abajo (el avión tiene tendencia a picar) respecto a este punto. Este fenómeno condiciona el manejo del avión y altera completamente las condiciones de trimado esperadas por el equipo y calculadas en Xflr5. Así, se refleja la importancia de considerar la altura del centro de gravedad respecto del motor y no solo de su posición longitudinal en la aeronave.

[3]Realmente, el modelo que se incluye en el TFG es de un demostrador, una versión de prueba del avión final. Además, el modelo considera una masa de 2,1 kg y sin carga útil, pero por simplicidad se asumirá que es un modelo del XTRA23.

Tabla 4.1: Valores característicos del «Favara». Fuente: [4]

Nombre	**Símb.**	**Valor**	**Unidades**
Envergadura	b	2,02	m
Cuerda media aerodinámica	$\bar{c}$	0,207	m
Superficie alar	S_w	0,4242	m^2
Masa en vacío (OEW)	m	2,107	kg
Momentos de inercia	I_x	0,277	$\mathrm{kg}\cdot\mathrm{m}^2$
	I_y	0,2387	$\mathrm{kg}\cdot\mathrm{m}^2$
	I_z	0,5121	$\mathrm{kg}\cdot\mathrm{m}^2$
	I_{xz}	−0,0051	$\mathrm{kg}\cdot\mathrm{m}^2$
Dist. vert. motor-c.d.g	z_T	−0,009	m

Los parámetros a identificar del «Favara» son las derivadas aerodinámicas, cuyos valores para este aeromodelo se recogen en las tablas 4.2 y 4.3. La tabla 4.4 contiene otras derivadas aerodinámicas relevantes, las llamadas *derivadas de control*, que relacionan los valores de las entradas (δ_e, δ_f, δ_a, etc.) con los coeficientes aerodinámicos.

Tabla 4.2: Derivadas aerodinámicas longitudinales «a priori» del XTRA23 «Favara» en ejes de estabilidad. Fuente: [4]

Coef.	0	α	α^2	$\dot{\alpha}$	q	V
C_D	0,0334	0,0641	1,1023	0	0	−0,0228
C_L	0,3811	5,3057	0	0,9559	7,2904	0,1595
C_m	0,0348	−0,3518	0	−3,4858	−14,201	0

Tabla 4.3: Derivadas aerodinámicas lateral-direccionales «a priori» del XTRA23 «Favara» en ejes de estabilidad. Fuente: [4]

Coef.	β	$\dot{\beta}$	p	r
C_Y	−0,2739	0	−0,0742	0,1883
C_l	−0,0644	0	−0,6244	0,2516
C_n	0,0475	0	−0,1484	−0,0618

Tabla 4.4: Derivadas de control del XTRA23 «Favara» en ejes cuerpo. Fuente: [4]

Coef.	δ_e	δ_f	i_t	δ_a	δ_r
C_X	−0,0231	−0,059	−0,0277	0	0
C_Z	−0,4153	−1,3365	−0,4974	0	0
$C_{m,b}$	−1,5047	0,1195	−1,7496	0	0
C_Y	0	0	0	−0,0418	0,11
$C_{l,b}$	0	0	0	−0,3447	0,0005
$C_{n,b}$	0	0	0	−0,0031	−0,0535

Una forma habitual de representar gráficamente algunos de estos parámetros es mediante las llamadas *polares del avión*. Estas gráficas representan en función de α algunos coeficientes aerodinámicos como C_L, C_D o relaciones entre ellos como el cociente C_L/C_D, conocido como *eficiencia aerodinámica*. Generalmente, estas se obtienen en régimen estacionario, de modo

que solo entran en juego los coeficientes con subíndice 0 y las derivadas estacionarias, que son las derivadas de los coeficientes aerodinámicos respecto de α, α^2 y β. Obtener estas curvas es inmediato conociendo los valores de todas estas constantes y aplicando las ecuaciones (4.35), (4.36) y (4.39) considerando el resto de términos nulos. Al hacerlo, se obtienen las gráficas de la figura 4.11.

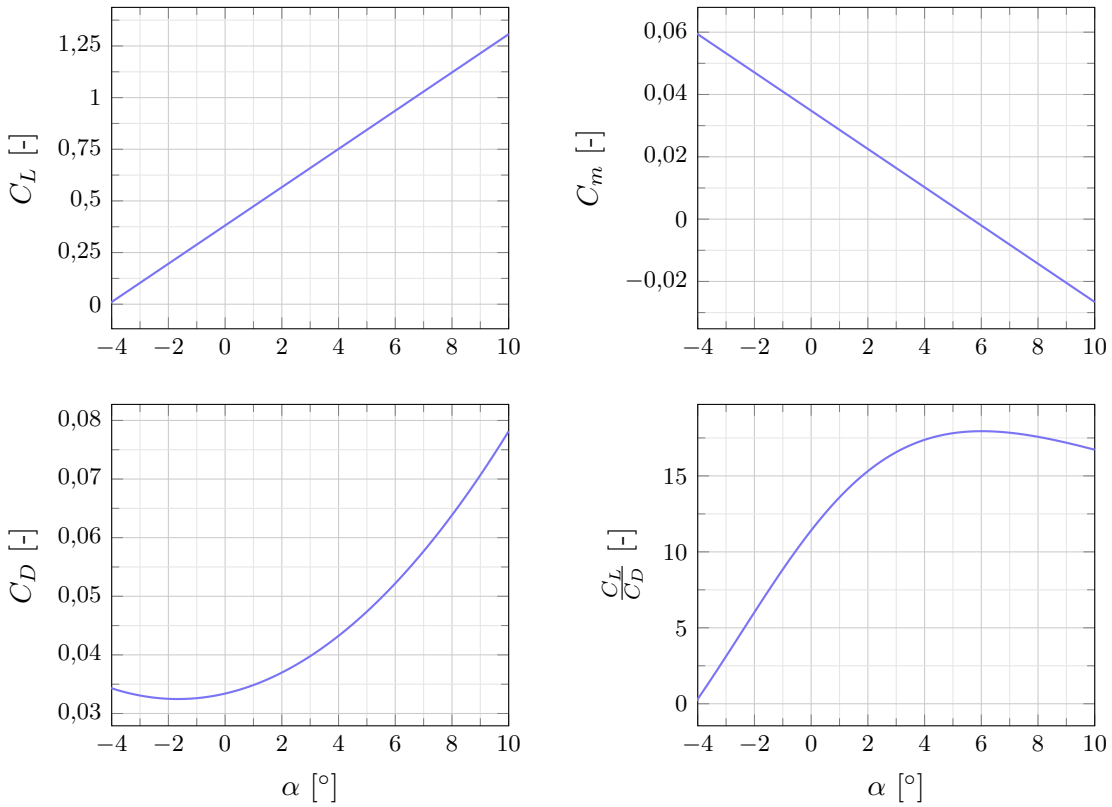


Figura 4.11: Curvas polares de C_D, C_L y C_m del XTRA23 «Favara» obtenidas mediante las ecuaciones (4.35), (4.36) y (4.39).

Trimado del modelo

Como consecuencia de los valores de las anteriores tablas, el avión tiene unas tendencias naturales y unos *puntos de trimado* concretos. Un punto de trimado viene dado por aquella condición de vuelo (valores de $\mathbf{x}$ y de $\mathbf{u}$) en la que el vector de aceleraciones lineales y angulares es $\vec{0}$, a diferencia del punto de equilibrio, en el que todo el vector $\dot{\mathbf{x}}$ es nulo [4]. Este estado de trimado se usará como estado inicial de las simulaciones.

Para determinar el estado de trimado del avión en una condición de vuelo concreta, se puede modificar todo el vector de entrada. Es posible trimar el avión para cualquier condición de vuelo como vuelo horizontal, ascenso, descenso o un viraje. En este caso, se quiso trimar el avión para vuelo horizontal, con velocidad vertical nula ($\dot{h} = 0$).

El trimado se puede cambiar en vuelo modificando ligeramente la deflexión del timón de profundidad. Sin embargo, para este trabajo se usó la metodología descrita en [16] y aplicada en [4], en la que se calculan los valores de la incidencia del estabilizador horizontal y la posición de la palanca de gases resolviendo un problema de optimización.

El problema consiste en minimizar una *función de coste* que depende de aquellas variables que se pretende anular (como las aceleraciones). Para minimizarla, hay que encontrar el estado del avión y la entrada adecuados. Formalmente, el problema de optimización se plantea como [4]

$$\begin{aligned} \underset{\mathbf{x},\,\mathbf{u}}{\text{minimize}} \quad & \mathbf{Q}(\mathbf{x},\dot{\mathbf{x}})\mathbf{H}\mathbf{Q}^{\intercal}(\mathbf{x},\dot{\mathbf{x}}) \\ \text{sujeto a} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) \end{aligned} \tag{4.49}$$

donde

$$\mathbf{Q}(\mathbf{x},\dot{\mathbf{x}}) = \begin{bmatrix} V-10, & \mathbf{x}_3, & \dot{\mathbf{x}}_{1:9}, & \dot{\mathbf{x}}_{12}, & h-50, & \theta-\alpha, & \mathbf{x}_{4:7}, & \mathbf{x}_{9:11}, & \delta_e, & \delta_f, & \delta_a, & \delta_r \end{bmatrix},$$

$$\mathbf{H} = \begin{bmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \alpha_r \end{bmatrix},$$

siendo $\dot{\mathbf{x}}_{1:9}$ las 9 primeras componentes del vector $\dot{\mathbf{x}}$ y $\mathbf{x}_{4:6}$ las componentes 4 a 6 del vector de estado. Nótese que en el vector $\mathbf{Q}$, se está ajustando la velocidad ($V = \mathbf{x}_1$) a 10 m/s y la altitud ($h = \mathbf{x}_{12}$) a 50 m. Estos valores están tomados de [4], donde 10 m/s es, aproximadamente, la velocidad con mayor eficiencia aerodinámica para el modelo del XTRA23 y 50 m es una altitud razonable para los aeromodelos de la ACC. En la matriz $\mathbf{H}$, los pesos α_i permiten ajustar el peso relativo de cada variable, aunque con todos a 1 es posible resolver el problema de optimización con éxito.

Al usar MATLAB con el comando `fmincon` [39] para resolver la ecuación (4.49), se obtienen los siguientes vectores de estado y entrada para la condición de trimado de vuelo horizontal:

$$\mathbf{x}_0 = \begin{bmatrix} 10 & 4{,}4114^\circ & 0 & 0 & 0 & 0 & 0 & 4{,}4114^\circ & 0 & 0 & 0 & 50 \end{bmatrix}$$
$$\mathbf{u}_0 = \begin{bmatrix} 0 & 0{,}3124^\circ & 0 & 0{,}187^\circ & 0 & 0 \end{bmatrix}$$

4.6.2. Modelo del XTRA25 «Chimuelo»

Tras un año de descanso, la siguiente edición de la ACC, en 2024, tuvo novedades en la normativa que forzaron a los equipos participantes a actualizar sus diseños. Esta vez, el equipo apostó por un avión mucho más grande, capaz de transportar una gran carga útil, creando el XTRA25, que pasó a ser conocido como «Chimuelo». Este prototipo ya se mostró en la figura 1.1 del apartado 1.1.

Del desarrollo de este avión se puede destacar el TFG de David Silla [40], sobre el diseño de los dispositivos de punta alar (*winglets*) de la aeronave, o el de Adrián Ortega [41], que versa acerca del diseño estructural.

Los valores característicos geométricos e inerciales para este aeromodelo se recogen en la tabla 4.5. Como se puede ver, este UAV tenía una envergadura considerablemente mayor que su predecesor de la ACC 2022.

Mediante programas como Xflr5, como se menciona en [4], se pueden obtener las derivadas aerodinámicas y de control para este avión, recopiladas en las tablas 4.6 y 4.7. Estos valores se incluyen únicamente a modo de referencia, ya que, como se ha dicho en el epígrafe anterior, el modelo usado para el proyecto ha sido el del «Favara».

Tabla 4.5: Valores característicos del «Chimuelo». Fuente: [42]

Nombre	**Símb.**	**Valor**	**Unidades**
Envergadura	b	3,30	m
Cuerda media aerodinámica	$\bar{c}$	0,28	m
Superficie alar	S_w	0,90	m^2
Relación de aspecto	AR_w	12,125	-
Masa en vacío	OEW	3,755	kg
Máxima carga útil	MPL	4,080	kg
Máximo peso al despegue	MTOW	7,835	kg
Momentos de inercia	I_x	1,2241	$kg \cdot m^2$
	I_y	1,5310	$kg \cdot m^2$
	I_z	2,7167	$kg \cdot m^2$
	I_{xz}	0,0890	$kg \cdot m^2$
Dist. vert. motor-c.d.g	z_T	0	m

Tabla 4.6: Derivadas aerodinámicas longitudinales del XTRA25 «Chimuelo» en ejes de estabilidad. Fuente: [42]

Coef.	0	α	α^2	$\dot{\alpha}$	q	V
C_D	0,0179	0,05	1,1	0	0	−0,0228
C_L	0,2249	5,6862	0	0,9559	8,3834	0,1595
C_m	0,0162	−0,6996	0	0	−27,4880	0

Tabla 4.7: Derivadas aerodinámicas lateral-direccionales del XTRA25 «Chimuelo» en ejes de estabilidad.

Coef.	β	$\dot{\beta}$	p	r
C_Y	−0,2091	0,0000	−0,0374	0,1865
C_l	−0,0269	0,0000	−0,5991	0,101 69
C_n	0,0848	0,0000	−0,0476	−0,0731

Finalmente, en la figura 4.12 se muestran las curvas polares de este UAV. Puede comprobarse que su eficiencia aerodinámica (C_L/C_D) máxima es considerablemente mayor que la del «Favara», además de que el ángulo de ataque para el que el coeficiente de momento de cabeceo se anula es también menor, de poco más de 1° frente a los casi 6° de su predecesor.

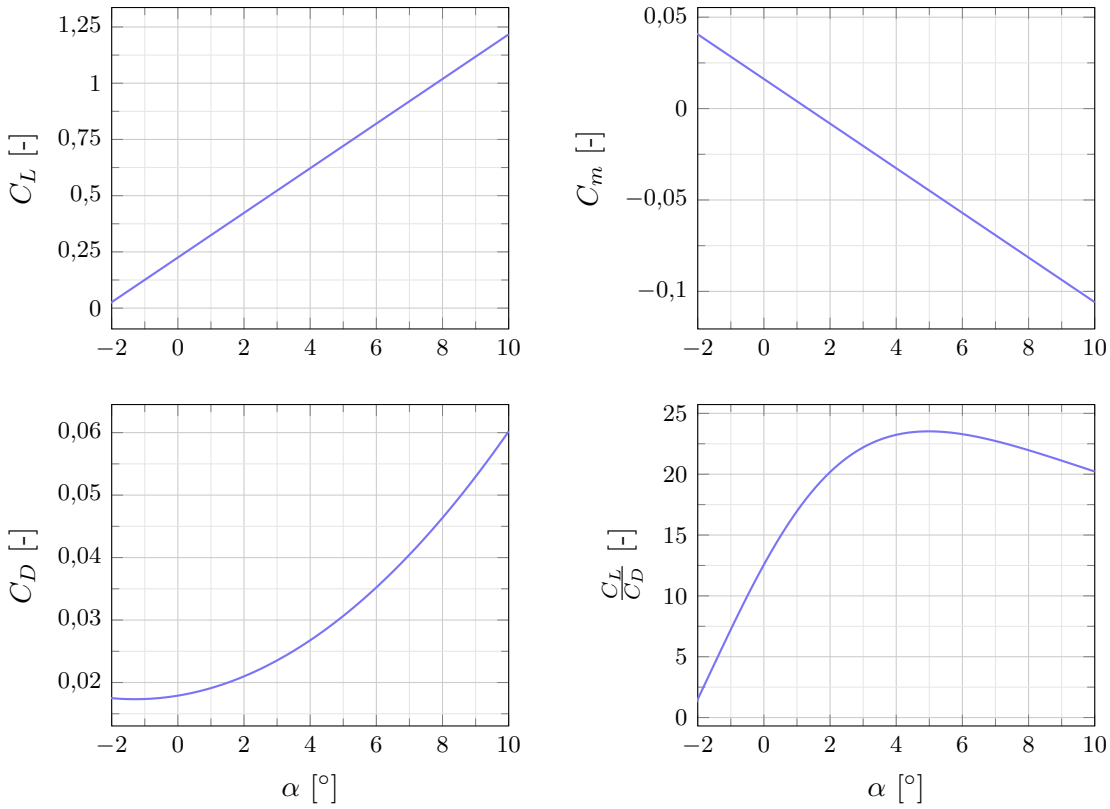


Figura 4.12: Curvas polares de C_D, C_L y C_m del XTRA25 «Chimuelo» obtenidas mediante las ecuaciones (4.35), (4.36) y (4.39).

5. MODELOS DE ACTUADORES

En el capítulo anterior se presentaron las entradas de control del sistema, que consistían en la deflexión de ciertas superficies y la posición de la palanca de potencia. Lo cierto es que estas entradas tienen una dinámica. Es decir, sus variaciones no son instantáneas, sino que son lentas y se pueden modelar mediante ciertas ecuaciones. La rapidez o lentitud de la variación de estas señales depende de los actuadores instalados en el avión. Este capítulo se dedica al estudio de estos dispositivos.

5.1. SERVOMOTORES

Los actuadores comúnmente usados en radiocontrol para alterar el comportamiento del sistema son los servomotores, conocidos habitualmente como *servos*. Estos consisten en alguna clase de motor eléctrico acompañado de una electrónica que permite un control preciso de la posición angular del eje. Normalmente, también cuentan con una reductora para aumentar el par. Pueden verse como una caja negra que recibe el ángulo deseado de rotación del eje y como salida el propio eje. A modo ilustrativo, la figura 5.1 muestra uno de estos dispositivos.

Figura 5.1: Fotografía del servomotor KST DS589MG V8.0. Cortesía de KST: [43]

Cuando uno se dispone a comprar estos dispositivos y busca en internet, encuentra que existen varias especificaciones para caracterizar estos actuadores. Algunas de las más importantes son [44]:

- **Servo analógico o digital**: Esto se refiere a la tecnología del lazo de control incorporado. Los analógicos utilizan amplificadores operacionales para realizar el control y un potenciómetro para detectar la posición angular del eje. Los digitales usan microcontroladores

y *encoders* digitales en el lazo de control. Estos últimos son bastante más caros y existe cierto debate sobre su utilidad en aplicaciones de radiocontrol [45, 46].
- **Tensión nominal (operating voltage)**: La tensión de alimentación admisible por el dispositivo.
- **Velocidad de rotación (operating speed)**: La velocidad angular que alcanza el eje cuando el servo opera sin carga (sin par resistente). Se suelen medir por el tiempo que se tarda en recorrer una distancia angular (incremento de posición angular) de 60°. Por ejemplo, 0,16 s/60°
- **Par máximo (stall torque)**: El momento máximo que puede generar el servomotor, generalmente medido en kg · cm.

Con estas características, se puede construir un modelo sencillo del servomotor que permita predecir su seguimiento de la referencia de posición. No obstante, una forma habitual y algo más directa de obtener estos modelos consiste en tomar medidas experimentales del propio servo y ajustarlas a un modelo. Es decir, *identificar* el servo, tratándolo como un sistema dinámico.

En [4] se hace una identificación sencilla de unos servos SG90. Mediante una cámara rápida y una guía visual para el ángulo, se propone un método para identificar la respuesta del servo sin carga. El modelo usado en el citado trabajo es el de un sistema de primer orden que relaciona el ángulo de entrada (la orden) y el ángulo de salida en el eje del servo, cuya *función de transferencia* en la teoría de control clásica toma la forma

$$G(s) = \frac{\Theta_{\text{out}}(s)}{\Theta_{\text{cmd}}(s)} = \frac{k}{1 + \tau s}\,, \tag{5.1}$$

donde s es la variable de la transformada de Laplace, k es la ganancia del servo (relacionado con el error en régimen permanente) y τ es la constante de tiempo del sistema, el tiempo que este tarda en alcanzar el 63 % del valor final de la respuesta y aproximadamente la cuarta parte del tiempo que se tarda en alcanzar el régimen permanente.

Tras ajustar las medidas al sistema de la ecuación (5.1), el modelo resultante obtenido en [4] para los servos SG90 queda dado por

$$G(s) = \frac{0{,}93}{1 + 0{,}14s}\,. \tag{5.2}$$

Nótese que este modelo no tiene en cuenta los límites de velocidad máxima del servo ni del par generado. La constante de tiempo obtenida, de valor 0,14 s, es aproximadamente 1,5 veces la velocidad máxima indicada en las especificaciones del servo [47], de 0,1 s/60°.

En la literatura se pueden encontrar otras técnicas mucho más elaboradas para el modelado de estos dispositivos. En [48] se propone un sistema adaptativo relativamente sofisticado para identificar en tiempo real unos servos. Para el modelo del servomotor, los autores toman un sistema de tercer orden, de forma que ajustan 7 parámetros del modelo en lugar de 2. Esta idea de identificación con modelos de orden mayor que 1 se repite en [49, 50], donde los autores recurren también a modelos de tercer orden para la relación entre las posiciones angulares de referencia (entrada) y salida, aunque con distintos objetivos.

En general, estos modelos reflejan bastante bien la dinámica real de los servos, pero no tienen en cuenta los datos dados por el fabricante. Por esta razón, para simplificar el modelo y hacerlo adaptable de forma sencilla para cualquier modelo de servo, se opta por un modelo sencillo de primer orden con ganancia unitaria y constante de tiempo dependiente de las características del

servo, de manera que los modelos resultantes sean muy parecidos al de la ecuación (5.1). La simplificación de ganancia unitaria es razonable porque cualquier error de ganancia se puede corregir mediante *software*. Se asumirá que el servo está suficientemente sobredimensionado como para que la variación de velocidad en vacío y en carga sea despreciable.

Esta tendencia se puede encontrar en [51], donde se obtiene un modelo de primer orden con saturación de velocidad para el servo. La saturación depende del límite de velocidad máxima del servo obtenido de las especificaciones, mientras que la constante de tiempo se identifica experimentalmente, tomando un valor $\tau = 0{,}01\,\mathrm{s}$. En [16], nuevamente, se usan modelos de primer orden para los actuadores, considerando una constante de tiempo de $\tau = 0{,}05\,\mathrm{s}$.

La constante de tiempo de los servos depende del propio servo y de la tensión de alimentación. En el caso de algunos modelos, estos admiten un rango amplio de tensiones de alimentación que consiguen grandes rangos de pares y velocidades máximas. A modo de ejemplo, en la tabla 5.1 se comparan las especificaciones de distintos modelos de servo, algunos de ellos usados por los equipos de la ACC 2024. Las especificaciones que faltan en la tabla no estaban en las hojas de datos de los servos correspondientes.

Tabla 5.1: Comparativa de servos usados por diferentes equipos de la ACC 2024.

Modelo		S90	ES3054HV	X08H V6.0 Plus	HV95
Fabricante		Tower Pro	Emax	KST	Chaservo
Reductora		Plástica	Metálica	Metálica	Metálica
Par máximo (*stall torque*) [kg · cm]	5 V	1,8	-	-	6,5
	6 V	-	3,4	3,8	8,5
	7,4 V	-	-	4,84	10,5
	8,4 V	-	4,7	5,3	11,5
Corriente máxima (*stall*) [A]	5 V	-	-	-	0,9
	6 V	-	1	-	1,16
	7,4 V	-	-	0,9	1,32
	8,4 V	-	1,3	-	1,58
Velocidad [s/60°]	5 V	0,1	-	-	0,21
	6 V	-	0,13	0,15	0,17
	7,4 V	-	-	0,12	0,15
	8,4 V	-	0,09	0,09	0,12
Rango		180	180	120	100
Peso [g]		9	20,5	9	20
Precio [€]		3,5	13,99 €	41,99 €	62,99 €
Equipo (ACC 2024)		-	AeroTéc Atlas	AeroUD, UBI, Xtra2	Akamodell Stuttgart e. V.

Además de las propiedades de los servomotores, para elaborar modelos más precisos se podría considerar el mecanismo que une el eje del servo con la superficie de control, ya que es posible que su relación cinemática sea compleja y no lineal (por ejemplo, por usar mecanismos de tipo cuadrilátero articulado con manivela y balancín de distinto tamaño).

Para este TFG se optó por el modelo de servo simplificado de primer orden de [4], que es el más sencillo, para no complicar en exceso el simulador. Se asume que el ángulo de giro del servo coincide con la deflexión de la superficie de control correspondiente.

5.2. AMPLIACIÓN DEL MODELO PROPULSIVO

En el apartado 4.4 ya se introdujo un modelo de empuje sencillo, que relacionaba la posición de la palanca de potencia y la velocidad aerodinámica con el empuje generado por la hélice. Este modelo se obtenía por identificación de los datos medidos en un ensayo en túnel de viento y sus expresiones eran sencillas. No obstante, en otras circunstancias o como ampliación de este trabajo, podría ser deseable trabajar con un modelo más complejo de la planta propulsiva. En ese caso, el modelo se puede ampliar disociando el motor eléctrico y la hélice y considerando sus efectos por separado.

En primer lugar, el empuje T generado por la hélice se puede modelar a partir de la teoría de hélices mediante la expresión [24, 33]

$$T = \rho n^2 D^4 C_T \,, \tag{5.3}$$

siendo ρ la densidad del aire, n la velocidad de giro de la hélice en revoluciones por segundo, D el diámetro de la hélice y C_T un coeficiente aerodinámico adimensional. Como ya ocurría con las fuerzas aerodinámicas del capítulo 4, la dificultad del modelado radica en obtener expresiones para los coeficientes adimensionales.

De forma similar, el par Q_p y la potencia mecánica P en el eje de la hélice se pueden modelar como [24, 33]:

$$Q_p = \rho n^2 D^5 C_Q \,, \tag{5.4}$$

$$P = \rho n^3 D^5 C_P \,, \tag{5.5}$$

donde C_Q y C_P son, nuevamente, coeficientes adimensionales. La potencia mecánica también se puede obtener como [33]

$$P = 2\pi n Q_p = \Omega_p Q_p \,, \tag{5.6}$$

siendo Ω_p la velocidad angular en rad/s. Es habitual adimensionalizar también la velocidad del viento aparente V mediante el grado de avance (*advance ratio*), dado por [33]

$$J = \frac{V}{nD} \,. \tag{5.7}$$

Por último, la eficiencia de la hélice η se calcula como la relación entre la potencia útil (TV) y la potencia mecánica de entrada (P):

$$\eta = \frac{TV}{P} \,, \tag{5.8}$$

o, en términos de coeficientes adimensionales,

$$\eta = \frac{C_T J}{C_P} \,. \tag{5.9}$$

En [33] se obtienen en túnel de viento las características de distintas hélices bipala del fabricante Aero-Naut, con diámetros de 9 a 16 pulgadas. Como resultados del estudio, en la anterior referencia se grafican los coeficientes experimentales C_T, C_P y η para distintas velocidades (grados de avance) y velocidades de giro. A modo de ejemplo, en la figura 5.2 se muestran las gráficas de los coeficientes característicos de una hélice de 13 pulgadas de diámetro y 8 de paso (13x8). Como la potencia y el par están relacionados mediante la ecuación (5.6), se podría obtener una curva del coeficiente de par C_Q a partir de las gráficas presentadas.

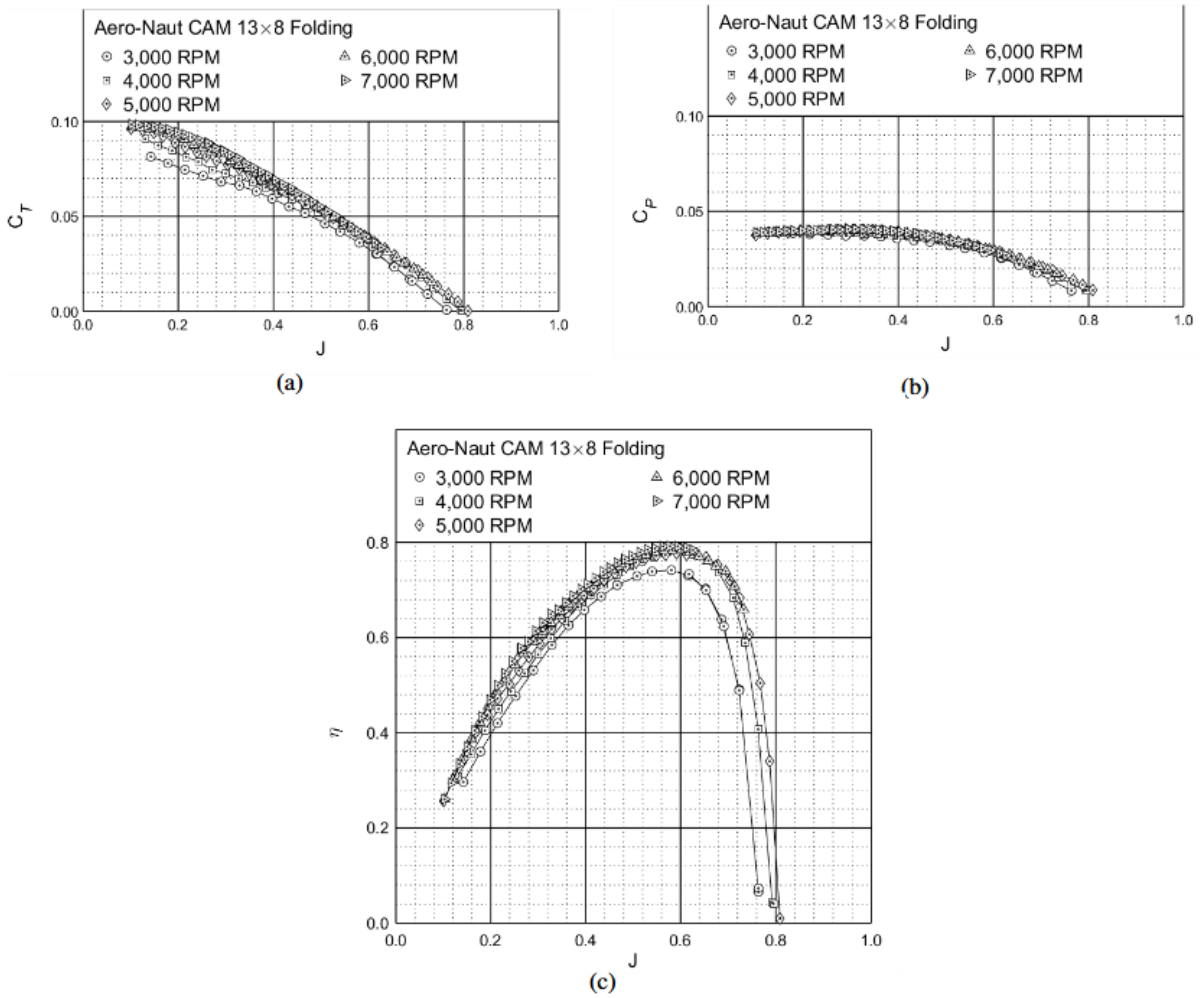


Figura 5.2: Coeficientes adimensionales de la hélice Aero-Naut CAM 13x8 plegable (*folding*). (a) coeficiente de empuje, (b) coeficiente de potencia, (c) eficiencia. Fuente: [33]

Estas gráficas se pueden aproximar como funciones cuadráticas de J. De esta manera, los coeficientes de empuje y de par se definen como [24]

$$C_T(J) = C_{T0} + C_{T1}J + C_{T2}J^2\,, \tag{5.10}$$

$$C_Q(J) = C_{Q0} + C_{Q1}J + C_{Q2}J^2\,, \tag{5.11}$$

donde los coeficientes C_{T*} y C_{Q*} se pueden obtener de un ajuste mediante mínimos cuadrados de los datos experimentales.

Por otra parte, el elemento que mueve la hélice suele ser un motor eléctrico de tipo *brushless* (sin escobillas, BLDC), cuyas conexiones están unidas a la salida de un elemento electrónico que regula la velocidad de giro. En castellano se llama a este elemento *variador* (de frecuencia), aunque en inglés se denomina *electronic speed controller* (ESC). A modo ilustrativo, la figura 5.3 muestra el ESC T-motor AM66A, que fue el utilizado en el XTRA25.

En [16] se propone un modelo sencillo para motores eléctricos y ESC. La dinámica de la

Figura 5.3: Fotografía del ESC T-motor AM66A. Cortesía de T-motor: [52]

velocidad angular Ω_p del motor viene dada por

$$\dot{\Omega}_p = (\Omega_{\text{cmd}} - \Omega_p)\frac{1}{\tau_{\text{ESC}}}\,, \tag{5.12}$$

donde τ_{ESC} es la constante de tiempo del conjunto motor-ESC (aunque se asume que depende principalmente del ESC) y Ω_{cmd} es la velocidad de consigna que el ESC trata de alcanzar y que este recibe mediante el comando δ_p.

Es decir, en [16] se modela el motor como un sistema de primer orden. El valor propuesto para su constante de tiempo τ_{ESC} es 0,05 s. Esta es una opción sencilla que permite representar de forma básica los transitorios de velocidad angular del motor y la hélice, de forma que el empuje no cambie instantáneamente.

En [24] se desarrolla otro modelo más completo de la propulsión, considerando un par motor cuya expresión es [1]

$$Q_m = K_Q\left[\frac{1}{R}(V_{\text{in}} - K_V\Omega_m) - i_0\right]\,, \tag{5.13}$$

siendo K_Q la constante de par del motor en $\text{N}\cdot\text{m/A}$, R la resistencia interna, V_{in} la tensión aplicada al motor (regulada por el ESC mediante modulación PWM), K_V la constante de velocidad del motor en $\text{V}\cdot\text{s/rad}$, Ω_m la velocidad angular del motor en rad/s e i_0 la corriente en vacío o sin carga del motor.

Estos parámetros se pueden obtener de las hojas de características de los motores. A modo de ejemplo, en la tabla 5.2 se recogen algunas de las especificaciones del T-motor AT2826 3D F3A 900KV, el motor reglamentario de la ACC2024, cuya fotografía se muestra en la figura 5.4.

De estos datos, la constante de velocidad de las especificaciones, $K_V^* = 900\,\text{rpm/V}$, se relaciona con K_V de la ecuación (5.13) por $K_V = 60/(2\pi K_V^*)$. Además, según [24], $K_V = K_Q$.

[1] Los motores DC sin escobillas funcionan con una especie de corriente alterna trifásica generada por el ESC. Para controlar el motor, se varía el valor eficaz de la tensión aplicada y su frecuencia [53]. El modelo propuesto en [24] no entra en los detalles de este funcionamiento, por lo que es un modelo relativamente simplificado de estos motores, pero sigue siendo más complejo que los encontrados en otras obras relacionadas con los UAV y por eso se ha mencionado en esta memoria.

Tabla 5.2: Especificaciones del T-motor AT2826 3D F3A 900KV. Fuente: [54]

Magnitud	Valor	Unidades
Masa	175	g
Constante de velocidad (KV)	900	rpm/V
Corriente en vacío (10 V)	2,2	A
Corriente máxima (180 s)	57	A
Potencia máxima (180 s)	820	W
Resistencia interna	24	mΩ

Figura 5.4: Fotografía del T-motor AT2826 3D F3A 900KV. Cortesía de T-motor: [54]

La tensión V_{in} aplicada al motor se controla por el ESC mediante la expresión [24]

$$V_{\text{in}} = \delta_p V_{\text{DC}} \,, \tag{5.14}$$

donde V_{DC} es la tensión de la batería y δ_p es el *throttle* o la posición de la palanca de potencia. Como δ_p varía entre 0 y 1, la tensión máxima aplicada al motor es la propia de la batería.

En régimen estacionario, el par motor y el resistente son iguales, es decir, $Q_m = Q_p$. Igualando las ecuaciones (5.4) y (5.13) y sustituyendo previamente C_Q por la ecuación (5.11) en la ecuación (5.4), se puede calcular la velocidad de giro, n. Con esta variable se puede evaluar el empuje generado mediante la ecuación (5.3), habiendo sustituido previamente el coeficiente C_T por la ecuación (5.10). De esta manera, queda calculado el empuje considerando las características del motor y un modelo más completo de la hélice.

Por simplicidad, para este trabajo se optó por el modelo simplificado del apartado 4.4, aunque se dejan las explicaciones de la presente sección como anticipo de una posible ampliación o mejora del modelo de empuje.

6. SENSORES USADOS EN AEROMODELOS

Hasta ahora, se ha tratado la aeronave como un sistema cuya dinámica viene gobernada por un modelo matemático y cuya respuesta depende de este y del estado y entrada actuales. Lo que suele ocurrir con los sistemas dinámicos—y el avión no es excepción—, es que algunas variables del estado no se pueden medir con precisión o no se puede medir en absoluto, lo que dificulta el control, que considera que se conoce con precisión el estado completo. Además, por cuestiones económicas, en determinados ámbitos no se puede disponer de algunos sensores para medir ciertas magnitudes. Por estas razones, este capítulo se dedica a estudiar el estado actual del mercado de sensores de aeronaves no tripuladas y a explicar brevemente el funcionamiento de algunos de estos dispositivos.

6.1. MODELO MATEMÁTICO DE UN SENSOR

La ecuación de medidas de un sensor genérico se puede describir mediante la expresión [10]

$$z_k = (1+\lambda)y_k + b + v_k \quad \text{donde} \quad k = 1, 2, \ldots, N \tag{6.1}$$

siendo z_k la k-ésima salida del sensor, y_k el valor auténtico de la variable medida, v_k el ruido aleatorio de medida, λ un parámetro constante de error de escala y b el parámetro de sesgo constante. Nótese que si b y λ son nulos la ecuación (6.1) se convierte en una versión escalar de la ecuación (2.5).

La ecuación (6.1) está en tiempo discreto, puesto que los sensores se van a integrar en un sistema de control digital en el que las medidas se toman a intervalos regulares de tiempo. La dinámica del sensor y los posibles retardos no se han contemplado en el modelo porque se consideran despreciables frente a la dinámica de las variables medidas, como se hace en [10] y otras muchas obras.

Este modelo será tomado como referencia para los sensores que se traten en el trabajo. Esto requerirá hacer varias simplificaciones, como por ejemplo tomar como aditivos ruidos que intervienen de forma no lineal en las medidas (véase el apartado 2.1). Además, las medidas de los sensores se considerarán ya corregidas, de forma que los factores de calibración b y λ puedan despreciarse en los siguientes capítulos de esta memoria.

6.2. MEDIDAS DE MAGNITUDES Y SU INTERÉS

Antes de comentar los sensores que se utilizan en los aeromodelos, hay que aclarar qué variables se pueden medir y el interés que tiene medirlas. Para ello, se recuerda que el vector de estado del avión tiene 12 componentes:

$$\mathbf{x} = \begin{bmatrix} V & \alpha & \beta & p & q & r & \phi & \theta & \psi & x_E & y_E & h \end{bmatrix}^\intercal .$$

De estas, todas pueden resultar de interés de una u otra forma, dependiendo de la clase de control que se quiera hacer del avión. A continuación se recuerda su significado y utilidad:

- Magnitudes aerodinámicas:
  - La velocidad aerodinámica V (*airspeed*) es la velocidad del avión con respecto al viento. Resulta importante para generar la sustentación y no debe caer por debajo de la velocidad de entrada en pérdida.
  - El ángulo de ataque α es el ángulo entre la cuerda del ala y el viento aparente (o el vector velocidad de la aeronave). Si se supera su valor crítico el avión entra en pérdida.
  - El ángulo de deslizamiento β. Un valor distinto de 0 indica que la aeronave no está realizando un viraje coordinado o está derrapando, lo que resulta poco eficiente.
- Las velocidades angulares p, q y r son las velocidades angulares de alabeo, cabeceo y guiñada en ejes cuerpo respectivamente. Resultan importantes por las aceleraciones centrípetas que se dan en el avión, que podrían superar los esfuerzos admisibles por la estructura.
- La actitud del avión, dada por los ángulos ϕ, θ y ψ, es la posición angular de los ejes cuerpo de la aeronave respecto a los ejes tierra en ángulos de Tait-Bryan. Pueden indicar escora, pendiente de la trayectoria y rumbo si se conocen los ángulos de deslizamiento y de ataque.
- Las posiciones x_E, y_E y h indican la posición del centro de gravedad del avión respecto a un punto fijo de la tierra. Son importantes para navegar por el espacio y seguir trayectorias.

Todas las anteriores variables se pueden medir de una u otra forma, aunque los sensores no siempre están disponibles o su uso en esta clase de aeromodelos no justifica su elevado precio. Las secciones siguientes arrojarán luz acerca de las distintas alternativas comerciales.

6.2.1. Tubo de Pitot

El tubo de Pitot es un sensor relativamente sencillo para medir la velocidad aerodinámica. Funciona a base de medir la diferencia de presiones entre dos puntos distintos, como muestra la figura 6.1. El sensor consiste en un tubo hueco, orientado en la dirección del flujo y sellado en su extremo. La presión en este tubo es la llamada presión de remanso o presión total P_t. Por otro lado, la presión estática P_s se mide mediante un agujero (o 2 en el caso de la figura) perpendicular al flujo.

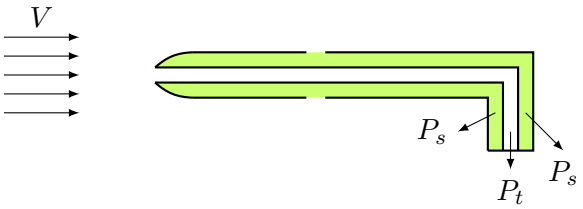


Figura 6.1: Esquema de un tubo de Pitot. Fuente: [21]

La diferencia de presiones se puede obtener con un manómetro diferencial. Con esta medida, es posible calcular la velocidad mediante la ecuación [55]

$$V = \sqrt{\frac{2(P_t - P_s)}{\rho}}, \tag{6.2}$$

donde ρ es la densidad del aire. La anterior expresión solo es válida si se considera el flujo incompresible, puesto que se deriva de la ecuación de Bernoulli. Esta simplificación es aceptable para vuelo a bajos números de Mach, como es el caso de las aeronaves de la ACC.

Esta forma de medir la velocidad tiene el pequeño inconveniente de que depende de la densidad del aire. Por tanto, cualquier incertidumbre asociada al conocimiento de su valor afectará a la precisión de la estimación de la velocidad. Esto puede resultar asumible teniendo en cuenta que las altitudes de vuelo serán pequeñas (poca variación de la densidad). En cualquier caso, se puede utilizar la ecuación de los gases ideales [55, 56] para obtener el valor de la densidad dada la temperatura y la presión (medidas con sus correspondientes sensores). De esta forma, la densidad viene dada por

$$\rho = \frac{p}{RT}\,, \tag{6.3}$$

donde p es la presión absoluta, T es la temperatura y R es la constante de los gases ideales, de valor 287,053 $\mathrm{m^2/(s^2 \cdot K)}$ [55]. Otra forma de aproximar la densidad es usar el modelo atmosférico estándar (ISA), como se describe en [3, 4, 16].

Para aeromodelos del estilo de la ACC, es habitual recurrir al MS4525DO de TE Connectivity. Este dispositivo consiste en un transductor de presión diferencial que permite medir directamente la diferencia de presiones de la ecuación (6.2) para, suponiendo un valor de la densidad del aire, calcular la velocidad. Este modelo también incorpora un sensor de temperatura y su conexión con otros dispositivos es mediante el protocolo $\mathrm{I^2C}$. El sensor se puede comprar en distintos kits, como el de la figura 6.2, que incluyen el tubo de Pitot y todo lo necesario para hacerlo funcionar.

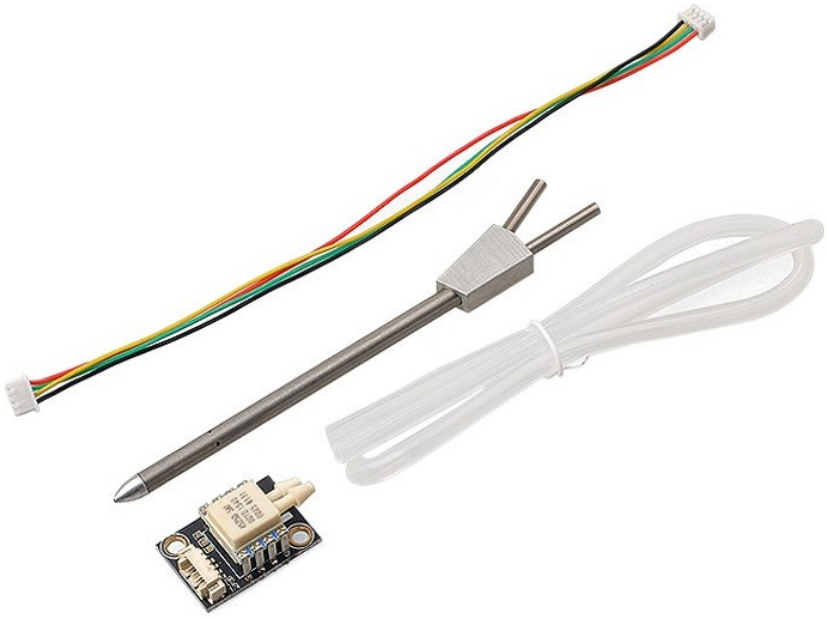


Figura 6.2: Fotografía del sensor MS4525DO junto al tubo de Pitot, cables y tubos flexibles para su conexión. Fuente: [57]

Como la velocidad es una magnitud estimada a partir de una medida de presión mediante una relación no lineal, realmente el ruido de medida no es aditivo. No obstante, se asumirá que sí lo es por simplicidad. En la hoja de datos del fabricante [58] no se especifican los niveles de ruido del sensor, por lo que se deben estimar, por ejemplo, por experimentación.

En su variante más común, este sensor admite una diferencia de presiones de hasta 1 psi (6,9 kPa), lo que permite medir velocidades de hasta 100 m/s, mucho mayores que las esperadas para esta clase de vehículos.

Para conocer más datos de este sensor, se pudo probar durante el curso 2023-2024 en túnel de viento a distintas velocidades. Este será el único sensor tratado en esta memoria con un gran nivel de detalle, debido al considerable sesgo que presentaba y al proceso llevado a cabo para calibrarlo.

La figura 6.3 muestra algunas medidas del sensor durante un experimento en el túnel de viento del CMT de la UPV. Durante el ensayo, se fue variando la velocidad del aire en el túnel en incrementos de unos 5 m/s desde 0 hasta 25 m/s. Esto demuestra que las medidas tienen un gran error de *offset*, además de un nivel de ruido elevado.

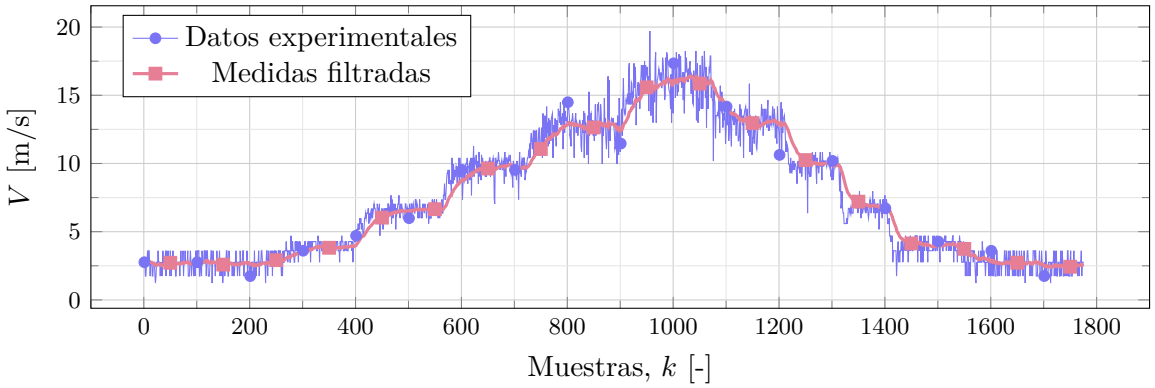


Figura 6.3: Medidas tomadas con el tubo de Pitot durante un ensayo en túnel de viento.

La curva con las medidas filtradas se obtiene aplicando un filtro paso bajo discreto a las medidas del sensor. Este filtro viene dado por la expresión

$$y_k = \alpha y_{k-1} + (1-\alpha)u_k\,, \tag{6.4}$$

siendo y_k la salida del filtro en el instante actual, y_{k-1} la salida en el instante anterior, u_k la entrada actual (medida sin filtrar) y α el factor de olvido, cuyo valor se fijó a 0,95. Este ajuste consigue un filtro muy agresivo, que modifica la dinámica de la señal original, pero permite obtener fácilmente el valor medio de las medidas una vez alcanzado el régimen estacionario (velocidad constante).

Durante el experimento, se anotaron a mano las medidas tomadas por la instrumentación incorporada en el propio túnel de viento, de forma que se pudo calibrar *a posteriori* el sensor, obteniendo la recta de calibración de la figura 6.4, cuya ecuación es

$$V_{\text{medida}} = 0{,}5962 \cdot V_{\text{real}} + 0{,}7302\,. \tag{6.5}$$

De esta forma, se pueden corregir las medidas del sensor despejando la V_{real} de la ecuación (6.5), obteniendo:

$$V_{\text{real}} = \frac{V_{\text{medida}} - 0{,}7302}{0{,}5962}\,. \tag{6.6}$$

Aplicando la ecuación (6.6) a los valores de la anterior figura 6.3, se obtienen unas medidas corregidas, que se muestran en las gráficas de la figura 6.5. Se puede comprobar que el ajuste es muy bueno para velocidades entre 5 y 25 m/s, pero que, con aire quieto (muestras 0 a 250 y de 1550 hasta el final), la medida del Pitot sigue sin ser nula. Esto no será un problema cuando el avión esté en el aire y no se considerará relevante.

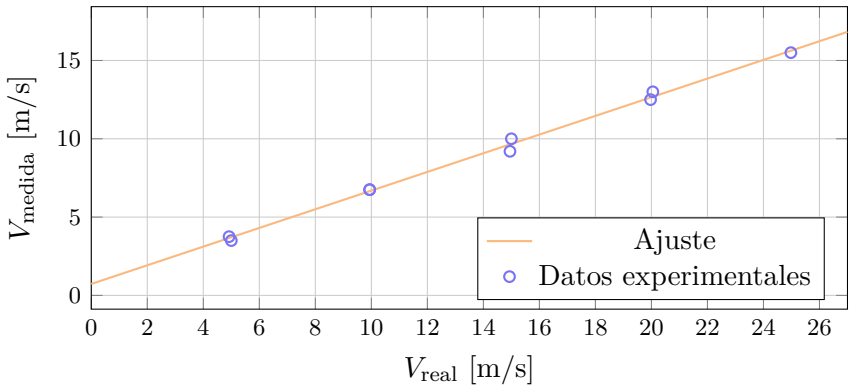


Figura 6.4: Recta de calibración del tubo de Pitot.

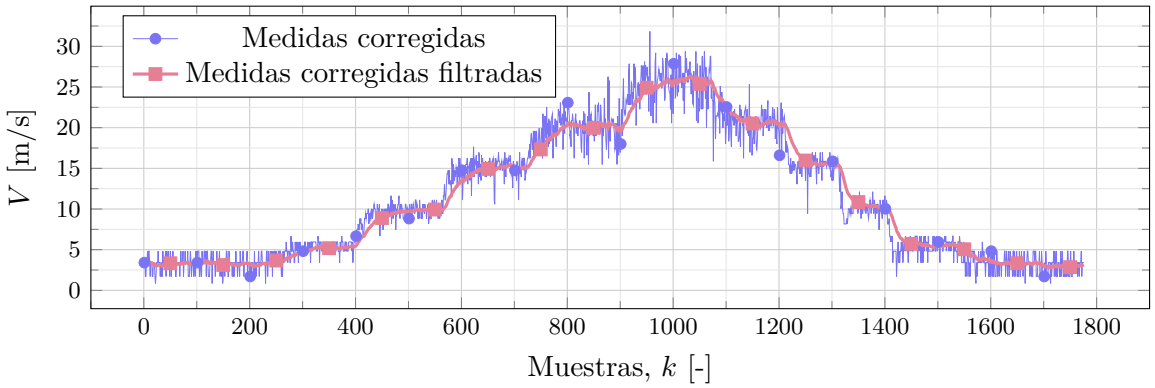


Figura 6.5: Medidas del tubo de Pitot corregidas.

Para modelizar el ruido del sensor, se asume que sigue una distribución normal y que es aditivo. Con esta premisa, la varianza (o covarianza) del ruido de medida se puede obtener como la media de las varianzas de las medidas para distintos tramos a velocidad constante, en los que la media de las muestras sea, por tanto, relativamente constante. La varianza de un vector de muestras se obtiene en MATLAB mediante el comando `var`, mientras que la media de las varianzas de cada tramo se puede obtener rápidamente con el comando `mean`. Tras estas operaciones, se obtiene una varianza media de aproximadamente 1,5762. Por tanto, se tiene:

$$\mathbf{R}_V = \mathbb{E}\left[\boldsymbol{v}^\intercal \boldsymbol{v}\right] \approx 1{,}5762\,. \tag{6.7}$$

A partir de este punto de la memoria, se supondrá que las medidas del Pitot ya han sido corregidas aplicando la ecuación (6.6) y que presentan un error aditivo gaussiano de media nula ($\mathbb{E}[\boldsymbol{v}] = 0$) cuya varianza viene dada por la ecuación (6.7).

6.2.2. Sensores de ángulos aerodinámicos

Los ángulos aerodinámicos de ataque α y derrape β son importantes durante el vuelo y un controlador necesita conocer sus valores en todo momento para operar correctamente. Desafortunadamente, los sensores para estos ángulos no están muy extendidos entre los UAV pequeños o de bajo coste. Estas dos magnitudes, al ser de naturaleza aerodinámica, se pueden medir con el mismo tipo de sensor, si bien será necesario uno para cada ángulo.

En [59] se realiza un estudio bastante exhaustivo de los métodos o dispositivos comúnmente usados para medir el ángulo de ataque en aeronaves no tripuladas. Los dispositivos comentados en el anterior texto, en su mayor parte, no están al alcance de este trabajo por complejidad de construcción o por precio.

En [60, 61] se exponen técnicas algo artesanales para fabricar sensores de ángulo de ataque. En estas referencias se propone la fabricación de un dispositivo similar a una veleta en miniatura, cuyo eje está unido a un *encoder* (sensor de ángulo absoluto) magnético de gran resolución. En [61], el modelo elegido es el AS5048A [62] de ams-OSRAM. Al soplar el viento sobre la veleta, este produce una fuerza aerodinámica que pasa por su centro de presiones. Si el eje del *encoder* no pasa por este punto, entonces esta fuerza generará momento respecto al eje que tenderá a alinear la veleta con el viento, determinando así el ángulo de ataque. En la anterior referencia [61] se dedica un apartado a detallar el proceso de diseño de la forma del dispositivo para que produzca un momento adecuado a bajas velocidades, junto con el ejemplo numérico de su caso de aplicación. En [63] se hace un carenado, mostrado en la figura 6.6, para incorporar un sensor de este tipo junto a un tubo de Pitot en un UAV de ala fija.

Figura 6.6: Ejemplo de montaje de un sensor de ángulo de ataque y un tubo de Pitot en el borde de ataque de un ala de UAV. Fuente: [63]

En lugar de recurrir a dispositivos físicos para obtener directamente una medida, se puede usar la información de otros sensores para reconstruir los ángulos aerodinámicos. Para ello, se suele recurrir a *observadores*, algoritmos que permiten estimar el valor de magnitudes a partir de medidas directas o indirectas de sensores. Esta técnica para estimar α y β se describe en [59, 64] y en [60] se utiliza junto a los sensores desarrollados para filtrar las medidas.

Durante el curso 2023-2024, el equipo Xtra2 UPV no pudo disponer de sensores de ángulo de ataque y tampoco se fabricaron siguiendo las anteriores publicaciones, ya que la investigación sobre estos dispositivos ha sido posterior. Por esta razón, para esta memoria se ha optado por estimar los ángulos aerodinámicos mediante reconstrucción de los datos con medidas de otros sensores. Para hacerlo, se propone el uso de un filtro de Kalman de tipo UKF, como se comentará en el capítulo 8. En [59] se mencionan técnicas similares basadas en el EKF, un algoritmo muy similar al UKF propuesto.

6.2.3. Unidad inercial (IMU)

La unidad inercial es un dispositivo que mide la aceleración y la rotación de un objeto. En el caso de los UAV, se utilizan para determinar las fuerzas específicas en ejes cuerpo (a_x, a_y y a_z), velocidades angulares (p, q y r) y la actitud (ϕ, θ y ψ) del prototipo. Las fuerzas específicas

(fuerzas por unidad de masa) son las aceleraciones debidas a la suma neta de fuerzas que actúan sobre el avión. Sus expresiones son [10]:

$$a_x = \frac{1}{m}\left(\bar{q}SC_X + T\right) , \tag{6.8}$$

$$a_y = \frac{1}{m}\left(\bar{q}SC_Y\right) , \tag{6.9}$$

$$a_z = \frac{1}{m}\left(\bar{q}SC_Z\right) . \tag{6.10}$$

En la práctica, es habitual encontrar estos dispositivos electrónicos como circuitos integrados que forman parte de las controladoras de vuelo comerciales. Internamente, estos circuitos integrados suelen incluir acelerómetros, giroscopios y magnetómetros, que permiten medir aceleraciones o fuerzas específicas, velocidades angulares y campos magnéticos, respectivamente. El campo magnético detectado por el magnetómetro se puede relacionar con la orientación espacial del avión respecto a la tierra, de manera que se puede considerar un sensor de actitud [65].

La tabla 6.1 muestra algunos ejemplos de circuitos integrados IMU conocidos. La tabla recoge el número de ejes (6 ó 9) de cada sensor: 3 para acelerómetro, 3 para giróscopo (6 ejes) y 3 para magnetómetro (los 3 ejes restantes). También incluye un precio orientativo obtenido de la tienda online Mouser Electronics [66] para los modelos no descatalogados y ejemplos de controladoras de vuelo comerciales que los incorporan. Los sensores MPU-6050 y MPU-9250, aunque están descatalogados, son especialmente conocidos entre los aficionados a la robótica con Arduino por su uso en diversas aplicaciones.

Tabla 6.1: Ejemplos de IMU comerciales y de controladoras de vuelo que las integran. Precios obtenidos de [66] para los modelos aún en venta.

Modelo	N.º ejes	Precio (€)	Ejemplo de controladora de vuelo
MPU-6000 [67]	6	-	Pixhawk 2.4.8 [68]
MPU-6050 [67]	6	-	-
ICM-42688-P [69]	6	4,97	Matek F405 Wing V2 [70]
ICM-20689 [71]	6	-	Pixhawk 4 [72, 73]
MPU-9250 [74]	9	-	-
ICM-20948 [75]	9	6,75	Pixhawk 6X [76]

Las IMU listadas en la tabla 6.1 presentan diferentes características de ruido. Los parámetros más relevantes para modelarlo suelen ser la desviación estándar del ruido en acelerómetros y giróscopos, típicamente especificados como *noise density* en las hojas de datos.

Por ejemplo, para el ICM-42688-P [69], la IMU de la controladora de vuelo usada por Xtra2 UPV durante el curso 2023-2024, la densidad espectral de ruido del acelerómetro es de $70\,\mu g/\sqrt{Hz}$ y la del giróscopo es de $0{,}0028\,^\circ/s/\sqrt{Hz}$.

Este sensor no tiene magnetómetro, de manera que no sirve para obtener una idea orientativa del ruido que podría presentar. Por ello, se ha decidido tomar como valores de referencia los del ICM-20948 [75], que es un sensor de 9 ejes bastante parecido al MPU-9250 [74].

Para estimar la matriz de covarianza del ruido de medida $\mathbf{R}$, se asume que el ruido es gaussiano y que los sensores son independientes. La matriz de ruido $\mathbf{R}$ será una matriz diagonal

de 9×9 (tres acelerómetros, tres giróscopos y tres magnetómetros) de la forma:

$$\mathbf{R}_{\text{IMU}} = \text{diag}(\sigma_{a_x}^2, \sigma_{a_y}^2, \sigma_{a_z}^2, \sigma_{g_x}^2, \sigma_{g_y}^2, \sigma_{g_z}^2, \sigma_{m_x}^2, \sigma_{m_y}^2, \sigma_{m_z}^2)\,, \tag{6.11}$$

donde σ_a, σ_g y σ_m son las desviaciones típicas del ruido de acelerómetro, giróscopo y magnetómetro, respectivamente. Para una frecuencia de muestreo f_s, la desviación típica del ruido se puede estimar como [65, 77, 78]

$$\sigma = (\text{Noise Density}) \cdot \sqrt{\frac{f_s}{2}}\,. \tag{6.12}$$

Para el ICM-20948 a $f_s = 20\,\text{Hz}$:

$$\sigma_a \approx 0{,}7273\,\mu\text{g} = 7{,}123 \times 10^{-3}\,\text{m/s}^2\,, \tag{6.13}$$

$$\sigma_g \approx 0{,}0474\,^\circ/\text{s} = 827{,}3 \times 10^{-6}\,\text{rad/s}\,, \tag{6.14}$$

$$\sigma_m \approx 0{,}5^\circ\,. \tag{6.15}$$

En la hoja de datos no aparecen datos de ruido del magnetómetro, así que, para las simulaciones de este trabajo, se ha supuesto una desviación típica σ_m considerablemente más alta en comparación a los anteriores sensores. En la práctica, esta desviación típica se podría calcular a partir de las medidas experimentales.

Sustituyendo valores en la ecuación (6.11), se obtiene el valor de la matriz $\mathbf{R}_{\text{IMU}}$ para el ruido de la IMU.

6.2.4. Altímetro

Además de los sensores anteriormente comentados, uno de los más básicos o elementales es el altímetro. Este sensor incorpora un transductor de presión que mide la presión atmosférica y, a partir de ella, calcula la altitud. Como la presión atmosférica disminuye con la altitud, el altímetro puede utilizar esta información para determinar la altura del objeto respecto al nivel del mar.

Al igual que ocurría con las unidades inerciales, los altímetros son dispositivos comúnmente integrados en controladoras de vuelo. La tabla 6.2 muestra algunos ejemplos de altímetros y ejemplos de controladoras de vuelo comerciales que los integran, junto con los precios de los modelos aún en venta. El BMP180 se ha incluido por ser un sensor conocido entre los aficionados a la robótica con Arduino, aunque el circuito integrado está descatalogado.

Tabla 6.2: Ejemplos de altímetros integrados comerciales y de controladoras de vuelo comerciales que los integran. Precios obtenidos de [66] para los modelos aún en venta.

Fabricante	Modelo	Precio (€)	Ejemplo de controladora de vuelo
Bosch	BMP180 [79]	-	-
Bosch	BMP388 [80]	-	Pixhawk 6X [76]
TDK InvenSense	ICP-10111 [81]	2,57	-
STMicroelectronics	LPS22HB [82]	2,51	-
Infineon	DPS310 [83]	2,38	Matek F405 Wing V2 [70]

En la hoja de datos del altímetro de la Matek F405 Wing V2, el ruido viene dado por el valor RMS (*root mean square*) y su valor máximo es 1 Pa. Tomando el modelo atmosférico estándar [3]

y suponiendo altitudes bajas, este ruido se corresponde con una variación de altitud de inferior a 1 metro. No obstante, para la simulación de este trabajo, se optó por considerar un ruido de altitud relativamente grande, con una desviación típica de 1 m, de manera que se tiene:

$$\mathbf{R}_h = \sigma_h^2 = 1^2 \tag{6.16}$$

6.2.5. Módulos GPS

Los módulos GPS permiten determinar la posición geográfica de un objeto en movimiento, como un UAV. De esta manera, estos dispositivos permiten obtener valores de x_E, y_E y h.

Algunas controladoras de vuelo comerciales incluyen módulos GPS integrados, mientras que otras requieren la compra de un módulo GPS externo. La controladora de vuelo Matek F405 Wing V2 no incluye un sensor GPS, por lo que, si se requiere, debe ser comprado aparte. Algunos modelos conocidos son el NEO-6M o el NEO-M8N, ambos del fabricante u-blox.

Para satisfacer los objetivos de control de este trabajo, no es estrictamente necesario conocer las coordenadas tridimensionales del avión en el espacio. Sin embargo, las magnitudes medidas por distintos sensores facilita la reconstrucción de las variables medidas y facilita el proceso de estimación de estados o parámetros.

La precisión del GPS depende, entre otras cosas, del número de satélites a los que se conecte el módulo, por lo que es difícil estimar el ruido de estas medidas. Como referencia destacable, en [84] se aborda el problema de la estimación adaptativa de los ruidos de sensores GPS. Para el presente trabajo, se supuso una desviación típica de ruido de posición de 2 m. Por tanto, la matriz de ruido de medida del GPS resulta

$$\mathbf{R}_{\text{GPS}} = \begin{bmatrix} \sigma_{x_E}^2 & \sigma_{y_E}^2 \end{bmatrix} = \begin{bmatrix} 2^2 & 2^2 \end{bmatrix} . \tag{6.17}$$

La medida de altitud h se tomará del altímetro, que tiene mayor precisión. Un GPS también permite obtener la orientación y velocidad, pero por simplicidad no se consideraron para este proyecto.

6.3. ECUACIONES DE SALIDA

En el capítulo 4 se describieron las ecuaciones que gobiernan la dinámica de la aeronave y permiten predecir la evolución de su estado. En este capítulo se han expuesto los sensores que permiten obtener mediciones de algunas componentes del vector de estado y en esta sección se mostrarán las ecuaciones que determinan las variables medidas por estos sensores. Estas son las llamadas ecuaciones de salida.

6.3.1. Modelo con 6 grados de libertad

A semejanza del apartado 4.2, aquí se describen las ecuaciones de las variables que pueden medirse del modelo completo con 6 grados de libertad. El vector con las variables medidas es

$$\mathbf{y} = \begin{bmatrix} V & p & q & r & \phi & \theta & \psi & x_E & y_E & h & a_x & a_y & a_z \end{bmatrix}^\intercal \tag{6.18}$$

donde V, p, q, r, ϕ, θ, ψ, x_E, y_E y h son variables de estado que se pueden medir directamente mediante un tubo de Pitot (V), un giroscopio (p, q y r), un magnetómetro (ϕ, θ y ψ), un GPS (x_E e y_E) y un altímetro o barómetro (h). Las aceleraciones a_x, a_y y a_z son las fuerzas específicas

medidas por un acelerómetro situado en el centro de gravedad de la aeronave y alineado con los ejes cuerpo.

Partiendo de que los equipos estén debidamente calibrados, las medidas de V, p, q, r, ϕ, θ, ψ, x_E, y_E y h no requieren mayor procesado y se pueden introducir directamente en el lazo de realimentación sin incurrir en errores significativos [10]. Para afinar más, se podría considerar la distancia del tubo de Pitot al centro de gravedad, pero aquí se ha asumido nula.

Las fuerzas específicas en ejes cuerpo a_x, a_y y a_z, por su parte, suelen recibir un tratamiento especial cuando el acelerómetro que permite obtenerlas no está situado en el centro de gravedad de la aeronave. Si el acelerómetro está situado en una posición relativa al centro de gravedad de la aeronave dado por $\begin{bmatrix} x_a & y_a & z_a \end{bmatrix}^\intercal$ en ejes cuerpo, las medidas corregidas son [10]

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{\text{c.g.}} = \begin{bmatrix} a_{x_E} \\ a_{y_E} \\ a_{z_E} \end{bmatrix} + \begin{bmatrix} q^2+r^2 & -(pq+\dot{r}) & -(pr+\dot{q}) \\ -(pq+\dot{r}) & p^2+r^2 & -(qr-\dot{p}) \\ -(pr+\dot{q}) & -(qr-\dot{p}) & p^2+q^2 \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} \tag{6.19}$$

donde $\begin{bmatrix} a_{x_E} & a_{y_E} & a_{z_E} \end{bmatrix}^\intercal$ denota las medidas experimentales (subíndice E) tomadas por el sensor.

En función de las variables de estado del sistema, las medidas $\begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^\intercal$ del acelerómetro se pueden predecir mediante las expresiones

$$a_x = \frac{1}{m}\left(\bar{q}SC_X + T\right) , \tag{6.20}$$

$$a_y = \frac{1}{m}\left(\bar{q}SC_Y\right) , \tag{6.21}$$

$$a_z = \frac{1}{m}\left(\bar{q}SC_Z\right) . \tag{6.22}$$

6.3.2. Modelo simplificado a dinámica longitudinal

Al igual que en el apartado anterior, si las ecuaciones se reducen a la dinámica longitudinal, el vector de salida con las variables que pueden medirse directamente es:

$$\mathbf{y} = \begin{bmatrix} V & \alpha & q & \theta & x_E & h & a_x & a_z \end{bmatrix}^\intercal \tag{6.23}$$

donde a_x y a_z son las aceleraciones longitudinal y vertical en ejes cuerpo, medidas por un acelerómetro situado en el centro de gravedad de la aeronave, cuyas expresiones en función del resto de variables del sistema son [10]:

$$a_x = \dot{u} + qw + g\sin\theta = \frac{1}{m}\left(\bar{q}SC_X + T\right) \tag{6.24}$$

$$a_z = \dot{w} + qu - g\cos\theta = \frac{1}{m}\left(\bar{q}SC_Z\right) \tag{6.25}$$

Parte III

DESARROLLO DE BLOQUES FUNCIONALES

7. SIMULADOR DE VUELO

Con el modelo desarrollado en los anteriores capítulos, se puede predecir la respuesta de la aeronave ante distintas entradas. Para ello, hay que resolver las ecuaciones diferenciales que componen el modelo, lo que puede resultar una tarea complicada y computacionalmente exigente. Se puede llamar *simulador* al conjunto de programas que permiten obtener numéricamente la solución de estas ecuaciones diferenciales con precisión. En las secciones siguientes, se explican los fundamentos que hacen posible este simulador.

7.1. ANTECEDENTES

El problema de la simulación del vuelo de aeronaves ha sido muy estudiado en el siglo XX y sigue siendo un tema relevante en el XXI. Desde el simulador académico descrito en [32, 85] hasta los simuladores para entrenar a pilotos profesionales, todos ellos usan métodos numéricos para integrar las ecuaciones diferenciales y simular la respuesta del avión. La dificultad del problema está en obtener datos fidedignos de la aeronave a simular, encontrar un modelo con unas expresiones precisas y en resolver estas con rapidez [86].

Entre las fuentes disponibles para encontrar información y guía para crear el simulador, cabe mencionar dos TFG realizados en la UPV: [32, 85]. En ellos se desarrolla un simulador de vuelo académico para la asignatura *Mecánica del vuelo* del grado en ingeniería aeroespacial que ha servido de gran ayuda para desarrollar este trabajo.

En las obras antes citadas se presenta un modelo general de la aeronave, como el incluido en el capítulo 4 de este proyecto, y se explican los algoritmos disponibles para simular su respuesta. MATLAB dispone de varias funciones, más precisas o más rápidas como `ode113` [87], que implementa el método Adams-Bashforth-Moulton y es la elegida en [85]; `ode89` [88] o métodos programados a mano como Runge Kutta 4 (RK4) [89, 90].

Partiendo del modelo expuesto en la parte II, el desarrollo del simulador consiste en crear algún entorno cómodo que integre numéricamente las ecuaciones del modelo y permita representar los resultados de formas atractivas o útiles para el usuario. Generalmente, cuando se piensa en simuladores se esperan interfaces gráficas, representaciones realistas del interior de la cabina o de los paisajes, mandos virtuales interactivos, etc. Lo cierto es que esto son añadidos que, si bien hacen al programa más vistoso, no contribuyen a un cálculo más preciso de las actuaciones el avión y, por tanto, no son estrictamente necesarios.

Explorando las distintas aproximaciones que se han dado en los últimos tiempos, se decide desarrollar un entorno de simulación híbrido, que permita un vuelo manual, interactivo y en tiempo real del avión y también un vuelo programado en modo *batch* o *headless*, que no requiera de una interfaz gráfica y que reciba una secuencia de vectores de entrada de otro programa, como el controlador de vuelo, y simule la respuesta para examinarla posteriormente.

7.2. TÉCNICAS DE INTEGRACIÓN NUMÉRICA

Las ecuaciones diferenciales del capítulo 4 son no lineales y su solución analítica es difícil de obtener. Por ello, se puede recurrir a métodos numéricos para obtener una aproximación

relativamente buena de la solución.

Los métodos matemáticos para integrar numéricamente ecuaciones diferenciales ordinarias son bien conocidos en la literatura [16, 32, 85, 86] y se han aplicado con éxito a multitud de problemas a lo largo de los años, incluida la simulación de sistemas dinámicos.

Una familia de algoritmos muy extendida son los conocidos como métodos de Runge-Kutta. Estos son un conjunto de técnicas para, dada una ecuación diferencial explícita, obtener un cálculo aproximado de sus variables dependientes. El método más sencillo de esta familia es el de Euler, presentado a continuación [86].

Se considera un sistema dinámico dado por una ecuación diferencial de la forma

$$\dot{x}(t) = f(x(t))\,, \tag{7.1}$$

y se conoce un valor inicial $x(t_0) = x_0$. Se puede obtener el valor de $x(t)$ en un instante $t_0 + h$, siendo h el llamado *paso de integración*, mediante la expresión

$$x(t_0 + h) = x(t_0) + \dot{x}(x_0) \cdot h\,.$$

Generalizando, se tiene

$$x(t_{k+1}) = x\left(t_k\right) + \dot{x}\left(x(t_k)\right) \cdot h\,, \tag{7.2}$$

o, con una notación más compacta,

$$x_{k+1} = x_k + \dot{x}(x_k) \cdot h\,. \tag{7.3}$$

Este método es muy sencillo, pero tiene una precisión moderada que se ve muy reducida si h toma valores altos. Por ello, se suele recurrir a métodos orden superior, como Runge-Kutta 4 (RK4) o Dormand-Prince [86].

Como se dijo anteriormente, MATLAB incorpora estos métodos numéricos y son accesibles mediante la familia de comandos `ode`. La elección de un comando u otro depende del coste computacional que se pueda asumir y de la precisión requerida. Para este trabajo, se optó por el comando `ode45` [91], que aplica el método de Dormand-Prince a una función que represente la dinámica de la aeronave de forma explícita ($\dot{x} = f(x, u)$). No obstante, otros métodos de la familia, como los analizados en [85] pueden obtener mejores resultados.

7.3. SIMULADOR DE VUELO EN MATLAB/SIMULINK

Mediante los métodos numéricos mencionados previamente, es factible desarrollar un simulador de vuelo en tiempo real utilizando Simulink. La evolución temporal de las respuestas se obtienen mediante integración numérica de las ecuaciones diferenciales del avión de la parte II. Las entradas pueden ser generadas por el usuario de forma manual o provenir de un controlador. Independientemente del origen de las entradas, la figura 7.1 muestra el diagrama de bloques que representa la forma de operar del simulador.

Como se puede comprobar, se trata de integrar numéricamente los modelos dinámicos de los actuadores ($\dot{\mathbf{u}} = \mathbf{f_u}(\cdot)$) y de la nave ($\dot{\mathbf{x}} = \mathbf{f}(\cdot)$), de manera que el resultado de los primeros sirva de entrada para el segundo. A nivel de resolución, se puede componer un vector ampliado dado por $\begin{bmatrix}\mathbf{x}^\intercal & \mathbf{u}^\intercal\end{bmatrix}^\intercal$ y concatenar las ecuaciones de modelo para resolver todas las ecuaciones con un

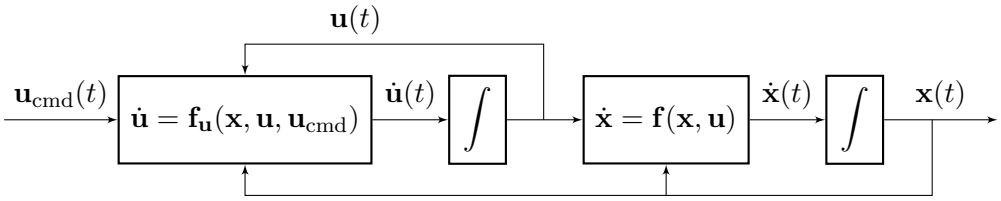


Figura 7.1: Diagrama de bloques del simulador de vuelo

solo *solver*. Dentro de Simulink, gracias a su simplicidad, basta plantear el diagrama de bloques de la anterior figura 7.1 y el propio programa se ocupa de su resolución.

La interfaz gráfica de Simulink también facilita la creación de un cuadro de mando virtual, provisto de deslizadores para los controladores, que permiten modificar los valores de las variables de control, así como de indicadores gráficos para visualizar parámetros como la altitud, la velocidad aerodinámica, entre otros. Adicionalmente, la *Aerospace Toolbox* de MATLAB [92] ofrece instrumentos gráficos, entre ellos un horizonte artificial, que ayudan a representar otras variables de interés como el asiento o el alabeo. En las figuras 7.2 y 7.3 se muestran estos elementos.

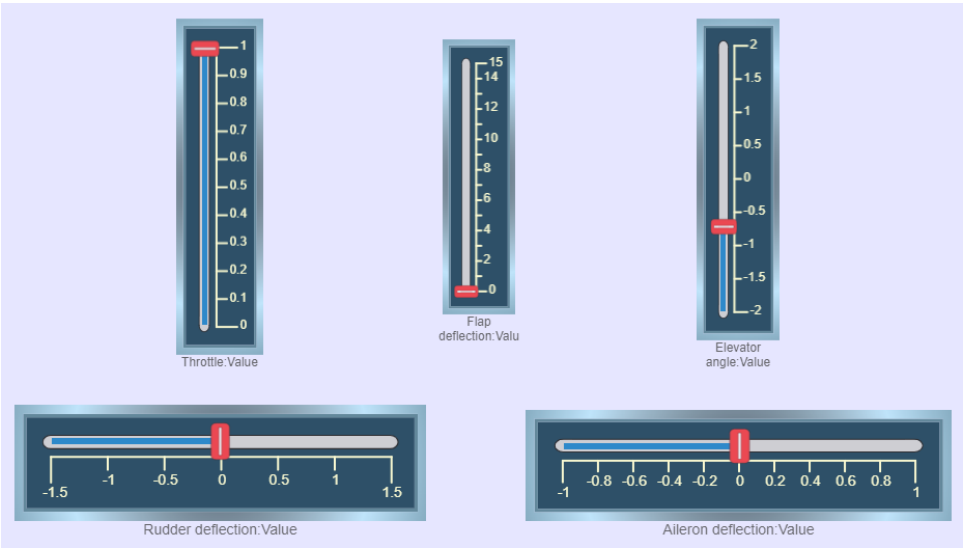


Figura 7.2: Mandos virtuales en el simulador de vuelo.

Los mandos virtuales se pueden sustituir con un mando físico (un *joystick* o similar), que facilitaría mucho el control del avión y aportaría realismo a la experiencia. Simulink soporta la interacción con esta clase dispositivos, aunque esto no se probó durante el desarrollo del proyecto.

También hay que destacar que Simulink posibilita ralentizar el tiempo de ejecución de la simulación y establecer una relación específica entre la progresión del tiempo real y el simulado. Al fijar esta relación en 1, se obtiene una simulación en tiempo real.

Con estos elementos, el simulador resultante permite emular un vuelo puramente instrumental, sin mostrar gráficamente la posición del avión en el espacio ni una interfaz gráfica realista y envolvente, pero es suficiente para comprobar las capacidades de un modelo de avión concreto y, aunque no se grafique en tiempo real, el simulador registra los históricos de los valores de

Figura 7.3: Indicadores e instrumentación virtual en el simulador de vuelo.

todas las variables durante la simulación para inspeccionarlos posteriormente. De esta manera se simulan los vuelos manuales e interactivos, en los que el usuario toma el control del avión.

Gracias a la forma de trabajar de MATLAB, se puede reaprovechar el código desarrollado para el simulador de Simulink y crear un simulador en modo *headless*, que no requiera de una interfaz gráfica para funcionar. Esta herramienta resulta más adecuada para probar el desempeño del controlador de vuelo, que permite automatizar las maniobras del avión. En este modo, las simulaciones no se ejecutan en tiempo real, sino lo más rápido posible para ahorrar tiempo.

Una vez simulada la maniobra, ya fuera controlada por el usuario de forma manual o por el controlador de vuelo de forma automática, los valores históricos de todas las variables se guardan para ser examinados posteriormente. Estos datos se pueden graficar mediante el comando `plot` de MATLAB, generalmente para visualizar la evolución temporal de variables; no obstante, también se puede usar el comando `plot3`, la variante tridimensional de `plot`, para representar la trayectoria seguida por la aeronave durante la simulación, así como la trayectoria de referencia fijada por el controlador, si la hubiese.

Como los buenos resultados del vuelo manual dependen enormemente de la pericia del usuario, todas las gráficas obtenidas del simulador en esta memoria pertenecen a vuelos automáticos guiados por el controlador, que genera respuestas repetibles y mucho más precisas que cualquier piloto inexperto.

8. FILTRO DE KALMAN

El *filtro de Kalman* (*Kalman filter*, KF) es un algoritmo que permite obtener estimaciones de los valores de magnitudes físicas a partir de las medidas ruidosas obtenidas mediante unos sensores, también llamadas observaciones. La palabra *filtro* hace referencia a que, si las variables a estimar son las medidas por los sensores, el algoritmo actúa como un filtro de ruido. No obstante, este método también permite estimar los valores de magnitudes que no se pueden medir directamente, de modo que pertenece a los llamados *observadores de estado*. Estos son un conjunto de métodos matemáticos usados para estimar el estado del sistema a partir de su salida [8].

La capacidad de estimar el estado se puede ampliar a la estimación de parámetros del modelo del sistema. De esta manera, se puede ir actualizando el modelo en tiempo real y modificando el controlador en consecuencia. Con esta propiedad, el piloto automático puede ser tolerante a fallos, ya que estos se podrían reflejar en forma de cambios en los parámetros del modelo, el KF los detecta y propaga estos nuevos coeficientes al controlador, cambiando las acciones de control aplicadas al avión o notificando al usuario de estos cambios. Este aspecto se tratará en detalle en los apartados 8.5 y 8.6.

El filtro de Kalman, visto como un bloque funcional, tiene unas entradas y salidas que se muestran en la figura 8.1. Como se ve, recibe las medidas u observaciones $\mathbf{z}_k$ contaminadas con ruido junto a la entrada del sistema actual $\mathbf{u}_k$ para obtener una estimación del estado, indicada como $\hat{\mathbf{x}}_{k|k}$.

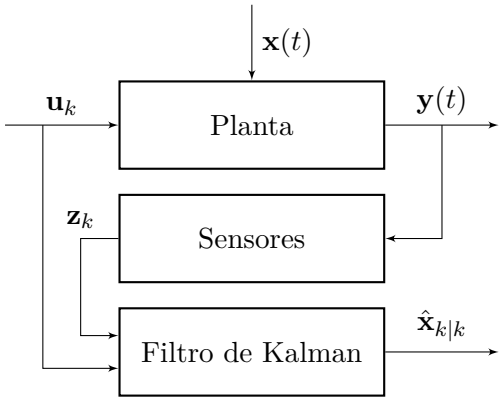


Figura 8.1: Diagrama de bloques de las entradas y salidas de un filtro de Kalman.

El auténtico estado $\mathbf{x}(t)$ del sistema es desconocido en la práctica, y solo se puede hacer estimaciones de su valor. Estas estimaciones del estado se denotarán como $\hat{\mathbf{x}}_{k|k}$, indicando que están definidas en tiempo discreto (se procesan en un computador). En la literatura, este valor se puede ver escrito también como $\hat{\mathbf{x}}_k$ [93] o $\hat{\mathbf{x}}_k^+$ [94].

Conviene introducir en este punto algunos términos que se explicarán en las siguientes páginas de este capítulo y que determinan el funcionamiento del filtro. Estos son la *predicción*, las *innovaciones* y los *residuos*.

La *estimación del estado* $\hat{\mathbf{x}}_{k|k}$ corresponde con el valor obtenido a la salida del filtro después de aplicar los pasos del filtro. Antes de obtener de esta estimación, el filtro calcula una *predicción del estado*, que se representará por $\hat{\mathbf{x}}_{k|k-1}$ en este texto. En otras publicaciones, este vector se puede ver escrito como $\hat{\mathbf{x}}_k^-$ [93, 94].

La diferencia entre la predicción de la salida $\hat{\mathbf{y}}_{k|k-1}$ y de la medida u observación $\mathbf{z}_k$ se llama *innovación* (*innovation* en inglés):

$$\boldsymbol{\mu}_k = \mathbf{z}_k - \boldsymbol{h}(\hat{\mathbf{x}}_{k|k-1}) = \mathbf{z}_k - \hat{\mathbf{y}}_{k|k-1} \tag{8.1}$$

Por otro lado, a la diferencia entre la observación $\mathbf{z}_k$ y la estimación de la salida $\hat{\mathbf{y}}_{k|k}$ a partir de la estimación del estado se la conoce como *residuo* (*residual*):

$$\boldsymbol{\varepsilon}_k = \mathbf{z}_k - \boldsymbol{h}(\hat{\mathbf{x}}_{k|k}) = \mathbf{z}_k - \hat{\mathbf{y}}_{k|k} \tag{8.2}$$

En algunas fuentes, como la documentación de MATLAB [11], estos dos términos se consideran iguales y se definen únicamente por la ecuación (8.2).

La tabla 8.1 recoge algunas notaciones comunes para diversas variables relacionadas con el filtro de Kalman y su variante UKF. Los símbolos usados en este texto se han elegido entre la literatura consultada con la intención de que sean suficientemente explícitos, aunque pueda acabar resultando en una simbología algo más recargada.

Tabla 8.1: Notaciones comunes en el UKF.

Descripción	En este texto	Otros autores
Vector de estado	$\mathbf{x}$	ídem [94-96]
Vector de salida predicha	$\hat{\mathbf{y}}_{k\|k-1}$ [96]	$\bar{\mathbf{z}}_n$ [95, 97]
Vector de medidas	$\mathbf{z}$	ídem [94-96]
Vector de pesos de la UT	$\boldsymbol{w}$ [95, 97]	$\boldsymbol{\eta}^m$, $\boldsymbol{\eta}^c$ [96]
Sigma points de la UT	$\mathcal{X}$	ídem [95, 96]
Innovaciones	$\boldsymbol{\mu}_k$ [31]	$\mathbf{e}_k$ [98], $\boldsymbol{\Delta}_k$ [99]
Residuos	$\boldsymbol{\varepsilon}_k$ [97]	$\boldsymbol{\Delta}_k$ [98]

8.1. EXPLICACIÓN INTUITIVA

El filtro de Kalman es un algoritmo que permite corregir las mediciones $\mathbf{z}$ de un conjunto de sensores imprecisos o con ruido que se usan para muestrear la salida $\mathbf{y}$ de un sistema dinámico (caracterizado como un modelo matemático en el espacio de estados), cuando se tiene algún modelo matemático que explique la dinámica de estados de ese sistema partiendo del estado actual ($\mathbf{x}$) y las entradas controladas por el usuario ($\mathbf{u}$).

Otra forma de ver el mismo problema es considerar cómo corregir las predicciones de un modelo matemático de un sistema dinámico cuando, además, se pueden realizar mediciones de algunas cantidades $\mathbf{z}$, relacionadas con los estados del sistema real a modelizar.

En cualquier caso, se trata de ponderar adecuadamente tanto el modelo predictivo como las mediciones de los sensores para conseguir una estimación del estado que mejore la precisión de cualquiera de ambas fuentes de información (medidas o modelo matemático) por separado. Para esta ponderación se tiene en cuenta la incertidumbre de las medidas u observaciones y la incertidumbre del propio modelo matemático del sistema.

La corrección a las mediciones se calcula con el teorema de Bayes de la probabilidad condicionada. En principio se puede demostrar con distribuciones normales, pero después se puede generalizar para cualquier distribución estadística.

Para comprender cómo se realizan estas correcciones y estimaciones, se puede examinar con detenimiento la figura 8.2.

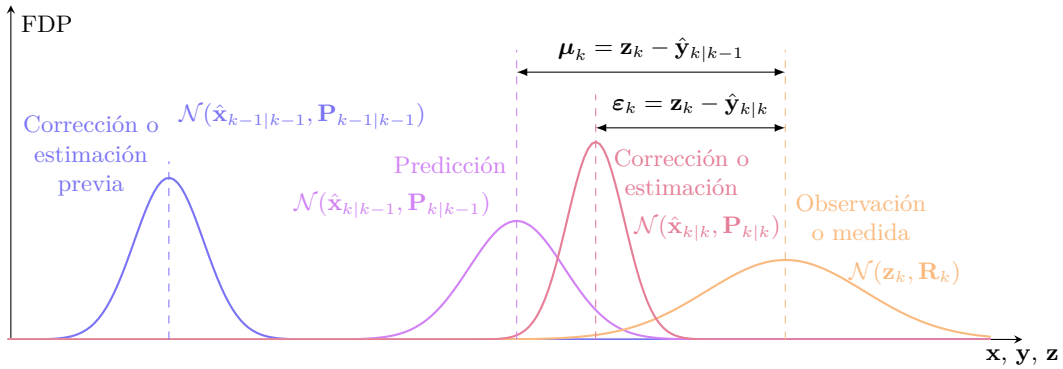


Figura 8.2: Distribuciones normales de las variables del filtro de Kalman, con $\mathbf{x} = \mathbf{y}$ creciente con el tiempo. Figura adaptada de [95]

En el eje horizontal se representa el estado $\mathbf{x}$, que podría ser la posición de un móvil en una trayectoria unidimensional. En el eje vertical se representa la función de densidad de probabilidad (FDP). El conocimiento del estado (posición $\mathbf{x}$) se adquiere basado en un modelo matemático, o bien por mediciones. Cada una de las posiciones dibujadas en el diagrama queda caracterizada por el tiempo, que no está reflejado en el diagrama, pero se asume que aumenta en la dirección positiva del eje horizontal; y por la certeza o probabilidad, que sigue una distribución normal dada por su media μ y varianza o covarianza σ^2: $\mathcal{N}(\mu, \sigma^2)$.

El primer paso es la *predicción* del estado del sistema $\hat{\mathbf{x}}_{k|k-1}$, basándose en un modelo matemático que predice el siguiente valor esperado partiendo del estado anterior $\hat{\mathbf{x}}_{k-1|k-1}$. La covarianza (o incertidumbre) de esta predicción, $\mathbf{P}_{k|k-1}$, se obtiene como una transformación de las covarianzas del estado actual y del modelo.

El segundo paso consiste en obtener una *observación* o medición $\mathbf{z}_k$ del estado del sistema, que también tiene una incertidumbre $\mathbf{R}_k$.

El último paso es la *corrección*, que obtiene el valor más verosímil $\hat{\mathbf{x}}_{k|k}$ y su incertidumbre $\mathbf{P}_{k|k}$, a partir de la combinación lineal de los estados predichos y medidos y sus correspondientes incertidumbres, de tal manera que la corrección tiene menos incertidumbre que cualquiera de los valores anteriores por separado.

A medida que se van recibiendo observaciones $\mathbf{z}_k$ y se van corrigiendo las estimaciones, se acaba llegando a un equilibrio en el que las salidas corregidas $\hat{\mathbf{y}}_{k|k}$ están alrededor de la salida real del sistema, es decir, que los valores estimados y reales coinciden. En la práctica, este fenómeno se puede comprobar mediante la evolución de los residuos $\boldsymbol{\varepsilon}_k$. Cuando el filtro funciona correctamente, la media de estos es pequeña (idealmente nula), mientras que su desviación típica o covarianza está relacionada con las propiedades del ruido de medida, parametrizado por la matriz de covarianza $\mathbf{R}$.

8.2. VARIANTES, MEJORAS Y APLICACIONES

El filtro original de Kalman, ideado en 1960 [100], solo es válido para sistemas lineales [94], por lo que no es adecuado para la mayoría de sistemas en la práctica. Para solucionar este inconveniente, con el tiempo se han ido creando variantes de este filtro que sí se pueden aplicar a sistemas no lineales.

En esta sección se exploran las distintas variantes que han surgido con el tiempo, los añadidos o mejoras que se han hecho a los algoritmos básicos y las aplicaciones principales que ha tenido. De esta forma, este epígrafe constituye un estudio de la evolución temporal del filtro y de los últimos avances en el campo de aplicación del KF.

Variantes del filtro y alternativas

Desde su presentación a finales del siglo pasado [93, 101], ha ganado bastante popularidad la variante *unscented Kalman Filter* (UKF), que tiene un desempeño mucho mejor que la variante más usada hasta entonces, el *Extended Kalman Filter* (EKF), para la estimación de sistemas no lineales [98, 102, 103]. De acuerdo con muchas publicaciones recientes [104, 105], el UKF es la mejor opción en muchos aspectos para este propósito.

El EKF, que ha sido aplicado durante muchos años a la estimación de sistemas no lineales, opera con el jacobiano del sistema dinámico a estimar. Con este jacobiano, se linealiza el sistema alrededor del punto de trabajo, de manera que se pueden aplicar las técnicas de estimación del KF original. Esta forma de aproximar las no linealidades funciona bien cuando el sistema es relativamente *suave*, pero con sistemas altamente no lineales, esta técnica deja de ofrecer resultados aceptables [93]. Además, el EKF tiene algunos problemas de estabilidad en caso de ruidos mal estimados, de gran amplitud o suposiciones de valor inicial muy alejadas del valor real, como se explica en [106].

La variante *unscented* del KF (UKF), introducida en el año 1997 por Julier y Uhlmann [107], pretende eliminar estos problemas del EKF mediante una aproximación distinta de las no linealidades, basada en la llamada *transformada unscented* (*unscented transform*, UT). Esta consiste en evaluar el modelo en unos cuantos puntos cercanos al estado estimado para determinar cómo se comporta el sistema alrededor del punto de trabajo. Esto elimina la necesidad del jacobiano y permite aproximar la dinámica del sistema con más precisión que la simplificación lineal del EKF. Por esta razón, siguiendo la corriente de los últimos tiempos, se ha decidido usar el UKF como bloque observador para el sistema de control de este trabajo.

La última vertiente de desarrollo del KF para sistemas no lineales es el *Cubature Kalman Filter* (CKF), presentado en 2009 [108], aunque no ha tenido tanta repercusión como el UKF. De hecho, el CKF también trabaja de una forma similar al UKF con el modelo del sistema, evaluando sus funciones en varios puntos cercanos al punto de trabajo. Por esta razón, el CKF ofrece resultados bastante similares al UKF, como se demuestra en [109, 110]. Este algoritmo también se probó durante el desarrollo de este proyecto y también consiguió resultados similares al UKF, por lo que se descartó en favor del último debido a su uso más extendido.

Existen otras alternativas más modernas para la estimación de estados de sistemas dinámicos, como aquellas basadas en redes neuronales, que también permiten la identificación de sistemas y se han aplicado con éxito a la aeronáutica, como se puede ver en [111]. Sin embargo, en este proyecto se opta por la vía más clásica y tradicional, basada en la inferencia bayesiana, por la gran cantidad de literatura que hay escrita hasta el momento.

Aplicaciones

El uso más habitual del KF es el que ya se ha explicado: estimar el auténtico estado del sistema a partir de medidas ruidosas y un modelo matemático. Para esto, el modelo usado debe reflejar de forma correcta el comportamiento del sistema, lo que requiere que las ecuaciones del modelo y sus coeficientes o parámetros tomen valores adecuados. Para obtener este modelo con sus parámetros se puede recurrir a la identificación de sistemas, como ya se explicó en el apartado 2.3. Esta técnica consiste en obtener modelos de sistemas mediante medidas experimentales y se puede conseguir con varios algoritmos. El filtro de Kalman es uno de estos algoritmos.

Si se considera que los parámetros de un modelo son estados de un modelo no lineal cuya dinámica es constante (los parámetros no varían), entonces se puede utilizar alguna variante (EKF, UKF, etc.) del filtro para estimar sus valores. Este procedimiento se conoce como estimación de parámetros, que es una parte de la identificación de sistemas. La idea de reutilizar el UKF para identificar aeronaves en tiempo real resulta sumamente atractiva a efectos de simplicidad y aprovechamiento de recursos y se quiso probar para este trabajo.

Tanto el EKF como el UKF se han usado en numerosas ocasiones para este propósito, como se demuestra en la abundante literatura escrita al respecto. Algunas referencias destacables pueden ser el artículo original de van der Merwe sobre la variante *Square-root* del UKF [101], en el que se comenta esta utilidad del algoritmo; otro artículo del mismo autor [93] en el que se menciona la estimación simultánea de estados y parámetros con un solo filtro (llamado *estimación dual* en el texto); el artículo [112], en el que se describen las capacidades de los filtros UKF, EKF y de partículas de la *Toolbox* de identificación de MATLAB [113], proponiendo el uso de estos para la identificación en tiempo real de sistemas, o [29, 31], en los que se usa un UKF para estimar estados y parámetros de UAV de ala fija.

Otra publicación reciente digna de mención es [114], en la que se aborda la identificación de aeronaves de ala fija considerando ruidos de proceso y medida (como hace el KF), pero se usa un método de identificación distinto de un KF. En esta obra se comenta la posibilidad de usar el UKF para estimar estados y parámetros, pero sus autores la descartan debido a sus posibles problemas de convergencia en el caso de estimación de parámetros. También se explica que el desempeño del filtro depende en gran medida del ajuste de las matrices de ruidos de proceso $\mathbf{Q}$ y medidas $\mathbf{R}$.

Estos comentarios no son nuevos, sino que pertenecen a una corriente de detractores de los algoritmos recursivos de estimación de parámetros, entre los que se encuentra el UKF. Otro ejemplo es [115]. Los autores de este artículo sostienen que esta clase de métodos ofrecen resultados peores que los métodos *offline* o *batch*, que trabajan sobre el *log* completo, y que, por tanto, solo es recomendable el uso de técnicas como el UKF en aplicaciones en los que una identificación *online* o recursiva es obligatoria y no hay alternativas.

Técnicas de adaptabilidad

De forma paralela al desarrollo de filtros para sistemas no lineales, a lo largo de las últimas décadas se han desarrollado métodos adaptativos, que permiten modificar dinámicamente los parámetros del filtro de Kalman para garantizar la convergencia de los parámetros o las variables estimadas. Esto es muy deseable porque permite un ajuste parcialmente automático del algoritmo. En concreto, existen multitud de publicaciones tratando la sintonización automática de las matrices de covarianzas de ruidos de proceso $\mathbf{Q}$ y medida $\mathbf{R}$ en tiempo real. Por citar algunas, se pueden mencionar los trabajos [116-121], en los que se proponen distintas técnicas para la

estimación de los ruidos de proceso y medida a partir de las medidas recibidas y las métricas del KF.

En las anteriores referencias no se tratan vehículos aéreos como el UAV de este trabajo, sino que se usan distintas variantes del filtro (EKF, UKF y CKF) para estimar o identificar otros sistemas dinámicos como baterías, embarcaciones de superficie no tripuladas, vehículos submarinos, etc. No obstante, las técnicas adaptativas propuestas en todas estas publicaciones tienen ciertas suposiciones en común que también son aplicables al UKF y al UAV de ala fija tratado en el presente proyecto.

8.3. EL UNSCENTED KALMAN FILTER

El *Unscented Kalman Filter* (UKF) es un algoritmo capaz de estimar estados de un sistema en tiempo discreto dado por una ecuación de transición de estados (recuerde el apartado 2.1):

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \tag{8.3}$$

La novedad de este filtro respecto a otras variantes como el EKF está en cómo aproximar las no linealidades del modelo. El UKF recurre a una transformada, llamada transformada *Unscented* (sin aroma, UT), que permite muestrear una variable aleatoria mediante un conjunto de puntos. El funcionamiento del UKF es un tema conocido en la literatura de control, de manera que la información que se detalla a continuación está muy extendida. Algunas referencias consultadas fueron [11, 95, 101, 122, 123].

El paso 1 del UKF consiste en tomar unos puntos, llamados *sigma-points*, de la variable aleatoria. De esta forma se muestrea su distribución estadística. Existen varios criterios acerca del número de puntos a tomar, como se explica en [124] pero el más extendido consiste en tomar $2N + 1$ puntos, siendo N la dimensión de la variable aleatoria. Estos puntos se presentan en una matriz $\mathcal{X}$ de N filas y $2N + 1$ columnas. A continuación hay que decidir qué puntos tomar. El primero es la media de la distribución. Este es el punto 0:

$$\mathcal{X}^{(0)}_{k-1|k-1} = \hat{\mathbf{x}}_{k-1|k-1} \tag{8.4}$$

donde el superíndice (0) indica el número del punto y la columna 1 de la matriz $\mathcal{X}_{k-1|k-1}$. El resto de puntos se toman a una distancia de este que depende de la desviación típica σ de la variable aleatoria. De ahí el nombre de *sigma points*. De este modo, el resto de puntos se calcula como

$$\begin{aligned} \mathcal{X}^{(i)}_{k-1|k-1} &= \hat{\mathbf{x}}_{k-1|k-1} + \left(\sqrt{(N+\lambda)\mathbf{P}_{k-1|k-1}}\right)_i, \qquad i = 1, \cdots, N \\ \mathcal{X}^{(i)}_{k-1|k-1} &= \hat{\mathbf{x}}_{k-1|k-1} - \left(\sqrt{(N+\lambda)\mathbf{P}_{k-1|k-1}}\right)_i, \qquad i = N+1, \cdots, 2N \end{aligned} \tag{8.5}$$

siendo λ un parámetro de la transformada y $\left(\sqrt{(N+\lambda)\mathbf{P}_{k-1|k-1}}\right)_i$ la i-ésima columna de la matriz cuadrada $\sqrt{(N+\lambda)\mathbf{P}_{k-1|k-1}}$, calculada mediante la factorización Cholesky. $\mathbf{P}_{k-1|k-1}$ es la covarianza de la última estimación del filtro, de manera que su raíz cuadrada es la desviación típica.

El paso 2 consiste en propagar los *sigma points* por la función de transición de estados, obteniendo una matriz $\mathcal{X}_{k|k-1}$ del mismo tamaño que $\mathcal{X}_{k-1|k-1}$:

$$\mathcal{X}_{k|k-1} = \mathbf{f}(\mathcal{X}_{k-1|k-1}, \mathbf{u}_{k-1}) \tag{8.6}$$

El paso 3 consiste en calcular un vector de pesos para ponderar los *sigma points*. En este momento hay que introducir los parámetros que determinan el comportamiento del filtro. Estos son 3: α, β y κ:

- El parámetro α determina la dispersión de los *sigma points* y suele variar entre 10^{-4} y 1. Cuanto más pequeño es el valor, más concentrados están los puntos tomados. Se suele cambiar en función de las no linealidades de la función $\mathbf{f}$.
- El parámetro β sirve para aportar información acerca de la distribución de la variable aleatoria. Para distribuciones Gaussianas, toma un valor de 2.
- El parámetro κ se suele calcular tal que $N + \kappa = 3$ cuando la distribución de la variable aleatoria es Gaussiana.

El parámetro λ, usado en la ecuación (8.5) para calcular los *sigma points*, se obtiene a partir de los anteriores mediante la expresión

$$\lambda = \alpha^2(N + \kappa) - N\,. \tag{8.7}$$

Dicho esto, los pesos se calculan como:

$$\begin{aligned} \boldsymbol{w}_i &= \frac{\lambda}{N+\lambda}, \qquad i = 0 \\ \boldsymbol{w}_i &= \frac{\lambda}{2(N+\lambda)}, \quad i > 0 \end{aligned} \tag{8.8}$$

de forma que $\boldsymbol{w}$ es un vector de la forma

$$\boldsymbol{w} = [w_0 \quad w_1 \quad \cdots \quad w_{2N}]\,. \tag{8.9}$$

Este cálculo solo se hace una vez, cuando ya se han asignado valores a los parámetros, por lo que forma parte de la inicialización del filtro y se suele calcular al inicio del proceso. Se ha explicado en este punto por cuestiones didácticas.

El paso 4 consiste en calcular la media y la covarianza de la distribución resultante. Estos se calculan como

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=0}^{2N} w_i \mathcal{X}^{(i)}_{k|k-1}\,, \tag{8.10}$$

$$\mathbf{P}_{k|k-1} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}^{(i)}_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}\right)\left(\mathcal{X}^{(i)}_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}\right)^{\mathsf{T}} + \mathbf{Q}_k\,, \tag{8.11}$$

siendo $\mathbf{Q}_k$ la covarianza del ruido de proceso, que se añade a la covarianza de la predicción del estado para tener en cuenta la incertidumbre del modelo. De esta manera, queda calculada la predicción del estado siguiente $\hat{\mathbf{x}}_{k|k-1}$ usando el modelo y su covarianza $\mathbf{P}_{k|k-1}$.

El paso 5 consiste en calcular la predicción de la observación. Para ello se usa la función de observación $\boldsymbol{h}$:

$$\mathcal{Y}_{k|k-1} = \boldsymbol{h}(\mathcal{X}_{k|k-1}, \mathbf{u}_k)\,. \tag{8.12}$$

donde $\mathcal{Y}_{k|k-1}$ es una matriz de M filas y $2N+1$ columnas, siendo M la dimensión de la observación. A continuación se calcula la media y la covarianza de la observación:

$$\hat{\mathbf{y}}_{k|k-1} = \sum_{i=0}^{2N} w_i \mathcal{Y}_{k|k-1}^{(i)}\,, \tag{8.13}$$

$$\mathbf{P}_{y,k} = \sum_{i=0}^{2N} w_i \left(\mathcal{Y}_{k|k-1}^{(i)} - \hat{\mathbf{y}}_{k|k-1}\right)\left(\mathcal{Y}_{k|k-1}^{(i)} - \hat{\mathbf{y}}_{k|k-1}\right)^{\mathsf{T}} + \mathbf{R}_k\,. \tag{8.14}$$

El término $\mathbf{R}_k$ es la covarianza del ruido de observación, que se añade a la covarianza de la predicción de la salida para tener en cuenta la incertidumbre de la medida. De esta forma, se obtiene una predicción de la observación $\hat{\mathbf{y}}_{k|k-1}$ y su covarianza $\mathbf{P}_{y,k}$.

El paso 6 consiste en calcular la covarianza cruzada entre el estado y la observación:

$$\mathbf{P}_{xy,k} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1}\right)\left(\mathcal{Y}_{k|k-1}^{(i)} - \hat{\mathbf{y}}_{k|k-1}\right)^{\mathsf{T}}\,. \tag{8.15}$$

Este término es importante para calcular la ganancia del filtro o ganancia de Kalman, que se calcula en el paso 7:

$$\mathbf{K}_k = \mathbf{P}_{xy,k}\left(\mathbf{P}_{y,k}\right)^{-1}\,. \tag{8.16}$$

La ganancia del filtro es un término que pondera la influencia de la observación en la estimación del estado. Cuanto mayor es la ganancia, más se ajusta la estimación a la observación.

El paso 8 consiste en actualizar la estimación del estado y su covarianza. La estimación del estado se actualiza como:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{y}}_{k|k-1})\,. \tag{8.17}$$

donde $\mathbf{z}_k$ es la observación actual. La covarianza de la estimación del estado se actualiza como:

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{P}_{y,k}\mathbf{K}_k^{\mathsf{T}}\,. \tag{8.18}$$

De esta forma, se obtiene una estimación del estado actual $\hat{\mathbf{x}}_{k|k}$ y su covarianza $\mathbf{P}_{k|k}$. Este proceso se repite para cada iteración del filtro, actualizando el estado y la covarianza en cada paso.

8.4. ADAPTABILIDAD

El rendimiento del filtro de Kalman depende en gran medida de la exactitud de las matrices de covarianza de los ruidos de proceso y medida ($\mathbf{Q}$ y $\mathbf{R}$). Unas estimaciones malas de estas matrices llevan a malas estimaciones de parámetros, mal seguimiento de los estados e incluso a divergencias [98].

Tener un conocimiento de las matrices de ruido puede ser complejo. Por un lado, para estimar $\mathbf{Q}$ se requiere un conocimiento preciso del sistema y una verificación con pruebas experimentales, puesto que se trata de determinar la exactitud del modelo. Por otro lado, la matriz $\mathbf{R}$ cuantifica el ruido de medida, lo que hace necesario medir los niveles de ruido de los sensores, también de forma experimental. Todo esto debe hacerse de forma previa a la estimación de estados o parámetros mediante el KF.

Debido a los problemas derivados de estimar estas matrices, existen métodos para estimarlas sobre la marcha durante el funcionamiento del filtro partiendo de unas suposiciones iniciales. Esto ahorra el estudio previo del proceso y los sensores y también permite detectar cambios (generalmente incrementos) en estos niveles de ruido, lo que podría indicar un fallo de alguno de estos sistemas. Aquellos filtros que actualizan sus estimaciones de estas matrices sobre la marcha se llaman *adaptativos* en la literatura.

A lo largo del tiempo se han propuesto distintos métodos para conseguir la adaptabilidad de los filtros de Kalman. Mehra [125] los divide en 4 categorías: métodos bayesianos, de máxima verosimilitud (*maximum likelihood*, ML), de correlación y de emparejamiento de covarianzas (*covariance matching*). Los más extendidos son los de correlación y emparejamiento de covarianzas [98], y en particular aquellos que operan con los residuos y las innovaciones.

Mediante estos métodos, estimar el valor de cualquiera de estas matrices fijando el valor de la otra es posible, pero intentar estimar ambas es complicado porque los cambios de $\mathbf{Q}$ influyen en la estimación de $\mathbf{R}$ y viceversa [126]. Por ello, es necesaria una cuidadosa selección de sus valores iniciales o el uso de sistemas que permitan detectar la fuente del fallo (*fault detection*). Para este trabajo se usó el método de estimación de las covarianzas de [97], desarrollado a continuación.

Ajuste de Q

La matriz $\mathbf{Q}$ es la matriz de covarianza de ruido de proceso $\mathbf{w}_k$. Este ruido se puede escribir como [97]:

$$\hat{\mathbf{w}}_k \approx \mathbf{K}_k \boldsymbol{\mu}_k \,, \tag{8.19}$$

siendo $\mathbf{K}_k$ la ganancia de Kalman y $\boldsymbol{\mu}_k$ las innovaciones. Por tanto la matriz $\mathbf{Q}_k$ se podría obtener como [97]

$$\mathbf{Q}_k = \operatorname{cov}(\hat{\mathbf{w}}_k) = \mathbf{K}_k \, \mathbb{E}\left[\boldsymbol{\mu}_k \boldsymbol{\mu}_k^\intercal\right] \mathbf{K}_k^\intercal \,, \tag{8.20}$$

donde $\operatorname{cov}(\cdot)$ denota el operador de covarianza (covarianza de una variable aleatoria) y $\mathbb{E}[\cdot]$ indica la esperanza matemática de una variable aleatoria.

La esperanza de una variable se suele estimar de dos formas. Se puede aproximar como la media de un conjunto de muestras en una ventana móvil [98] o bien por un filtro paso bajo con factor de olvido λ [97].

En este trabajo se ha optado por el método del filtro paso bajo, de forma que la nueva matriz $\mathbf{Q}_k$, denotada como $\mathbf{Q}_{k+1}$, se obtiene como

$$\mathbf{Q}_{k+1} = (1-\lambda)\mathbf{Q}_k + \lambda \left(\mathbf{K}_k \boldsymbol{\mu}_k \boldsymbol{\mu}_k^\intercal \mathbf{K}_k^\intercal\right) \,. \tag{8.21}$$

El valor de λ se actualiza dinámicamente mediante un criterio estadístico que se verá a continuación

Ajuste de R

De forma similar a $\mathbf{Q}$, la matriz $\mathbf{R}$ se puede estimar mediante la secuencia de residuos $\boldsymbol{\varepsilon}_k$ y con un filtro paso bajo. La matriz $\mathbf{R}_k$ se puede definir mediante la expresión

$$\mathbf{R}_k = \operatorname{cov}(\boldsymbol{v}_k) = \mathbb{E}\left[\boldsymbol{\varepsilon}_k \boldsymbol{\varepsilon}_k^\intercal\right] + \mathbf{S}_{\mathbf{y},k|k} \,, \tag{8.22}$$

donde $\boldsymbol{v}_k$ es el ruido de medida y $\mathbf{S}_{\mathbf{y},k|k}$ se obtiene como

$$\mathbf{S}_{\mathbf{y},k|k} = \sum_{i=0}^{2N} w_i \left(\boldsymbol{h}(\mathcal{X}_{k|k}^{(i)}) - \hat{\mathbf{y}}_{k|k}\right) \left(\boldsymbol{h}(\mathcal{X}_{k|k}^{(i)}) - \hat{\mathbf{y}}_{k|k}\right)^\intercal \,, \tag{8.23}$$

y, por tanto, la nueva matriz $\mathbf{R}_k$, denotada como $\mathbf{R}_{k+1}$, se calcula mediante el filtro paso bajo con factor de olvido δ de la forma

$$\mathbf{R}_{k+1} = (1-\delta)\mathbf{R}_k + \delta\left(\boldsymbol{\varepsilon}_k\boldsymbol{\varepsilon}_k^\intercal + \mathbf{S}_{\mathbf{y},k|k}\right), \tag{8.24}$$

donde δ, al igual que λ, se regula mediante un criterio estadístico que se verá a continuación.

Criterio de actualización de matrices

Para actualizar las matrices de ruidos, en [97] se propone un método que usa un criterio estadístico para determinar si es necesario modificar las matrices. La función usada es

$$\varphi_k = \boldsymbol{\mu}_k^\intercal(\mathbf{P}_{y,k})^{-1}\boldsymbol{\mu}_k\,, \tag{8.25}$$

donde φ_k sigue una distribución χ^2 con M grados de libertad, siendo M el número de salidas del sistema. Si se quiere detectar un fallo con una propabilidad de $1-\sigma$, entonces hay un valor umbral $\chi^2_{\sigma,M}$ tal que

$$P(\varphi_k > \chi^2_{\sigma,M}) = \sigma\,. \tag{8.26}$$

De esta manera, sintonizando el valor de σ se puede controlar la sensibilidad del sistema a los fallos. Si $\varphi_k > \chi^2_{\sigma,M}$, entonces se considera que hay un fallo y se procede a actualizar las matrices de ruido.

Con este criterio, los factores de olvido vienen dados por las ecuaciones

$$\lambda = \text{máx}\left\{\lambda_0, \frac{\varphi_k - a\chi^2_{\sigma,M}}{\varphi_k}\right\}, \tag{8.27}$$

$$\delta = \text{máx}\left\{\delta_0, \frac{\varphi_k - b\chi^2_{\sigma,M}}{\varphi_k}\right\}, \tag{8.28}$$

siendo a, b, λ_0 y δ_0 otros parámetros a ajustar. En [97] se proponen los valores $a = b = 5$, $\lambda_0 = \delta_0 = 0{,}2$ y $\sigma = 0{,}5$ y fueron los valores usados en este trabajo, aunque se deberían haber hecho pruebas para el caso específico del avión tratado y de los sensores considerados.

8.5. DISEÑO DEL EXPERIMENTO PARA IDENTIFICACIÓN

Los buenos resultados en la identificación de parámetros dependen de las maniobras que se hagan o, más concretamente, de la evolución de las variables medidas durante el experimento. Para el caso de un aeromodelo o de una aeronave a tamaño real, no es posible realizar cualquier maniobra, ya que podría poner en peligro la integridad de los tripulantes o de la propia estructura del avión. Por tanto, se debe estudiar cómo realizar la identificación de forma óptima con un cuidadoso diseño del experimento, lo que se traduce en una elección meticulosa de la entrada a aplicar al avión.

Para preparar el experimento se requiere cierto conocimiento previo sobre el prototipo a estudiar. Esto implica un buen modelo *a priori* del avión con el que hacer simulaciones y encontrar las mejores acciones de control para el experimento. Como bien se dice en [127], aquí se da una paradoja, en tanto que se pueden diseñar muy buenas entradas si el modelo *a priori* es fidedigno y es entonces cuando el experimento es menos necesario.

El diseño de las entradas óptimas se suele plantear en la literatura como la resolución de un problema de optimización. Este consiste en la maximización o minimización de ciertas métricas que permiten evaluar o cuantificar la *información* presente en unas medidas obtenidas con unos sensores durante un experimento. Estas métricas suelen cuantificar de una u otra forma la llamada *sensibilidad* del sistema a los parámetros del modelo.

Una interpretación intuitiva es la siguiente: durante el experimento se pretende maximizar la influencia de los parámetros en la respuesta del sistema para que los valores de estos sean más fáciles de determinar. Esto se consigue variando las entradas aplicadas al sistema, provocando ciertas respuestas que evidencien los parámetros del modelo.

Esta influencia de los parámetros en la respuesta es la *sensibilidad*, que es instantánea y depende del estado actual; mientras que la *información* que haya en la respuesta depende de la sensibilidad (instantánea) al cuadrado integrada a lo largo de todo el experimento.

La sensibilidad del sistema respecto de los parámetros se puede cuantificar y representar de distintas formas. Una de ellas es la *matriz de sensibilidad de salida* o simplemente *matriz de sensibilidad*, dada por [128]

$$\mathbf{S_y} = \frac{\partial \mathbf{y}}{\partial \boldsymbol{\theta}} \,. \tag{8.29}$$

Es decir, la matriz de sensibilidad es el jacobiano de la salida $\mathbf{y}$ respecto del vector de parámetros $\boldsymbol{\theta}$. El valor de esta matriz varía con el tiempo (no se ha indicado esta dependencia para simplificar la notación) y depende de los valores de los estados en cada instante, a los que se llega aplicando unas determinadas entradas al sistema.

El cálculo de la ecuación (8.29) puede no ser trivial. Cuando no se logra obtener una expresión analítica, se recurre a diferenciación numérica o al desarrollo de la expresión mediante la regla de la cadena [129]. Para este trabajo, se optó por el método descrito en [130], donde se considera la siguiente expresión para la sensibilidad de salida:

$$\mathbf{S_y} = \frac{\partial \boldsymbol{h}}{\partial \mathbf{x}} \mathbf{S_x} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta}} \,. \tag{8.30}$$

Aquí, $\mathbf{S_x} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}}$ y se conoce como sensibilidad del estado respecto de los parámetros. Esta se puede obtener mediante la siguiente ecuación diferencial [129, 130]

$$\dot{\mathbf{S}}_\mathbf{x} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{S_x} + \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \,. \tag{8.31}$$

Al integrar numéricamente la ecuación (8.31) y sustituir cada $\mathbf{S_x}$ en la ecuación (8.30) se puede obtener la matriz de sensibilidad para todos los instantes de tiempo de una simulación. El resto de jacobianos ($\frac{\partial \boldsymbol{h}}{\partial \mathbf{x}}$, $\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta}}$, $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}$) no resultan muy complejos de calcular de forma analítica para el modelo tratado en este trabajo.

Una vez calculada la matriz de sensibilidad, se pueden obtener otras métricas de sensibilidad o de identificabilidad que permiten evaluar si un experimento es adecuado para identificación. Hay que decir en este punto que existen varias tendencias para el diseño de experimentos en la literatura, como se indica en [128]. Aquí se presentan dos: una basada en la *matriz de información de Fisher* (FIM, *Fisher Information Matrix*) y otra que utiliza la matriz hessiana aproximada $\mathbf{H}$. La figura 8.3 ilustra las matrices y cálculos requeridos en cada una de las dos estrategias para estudiar la identificabilidad del sistema.

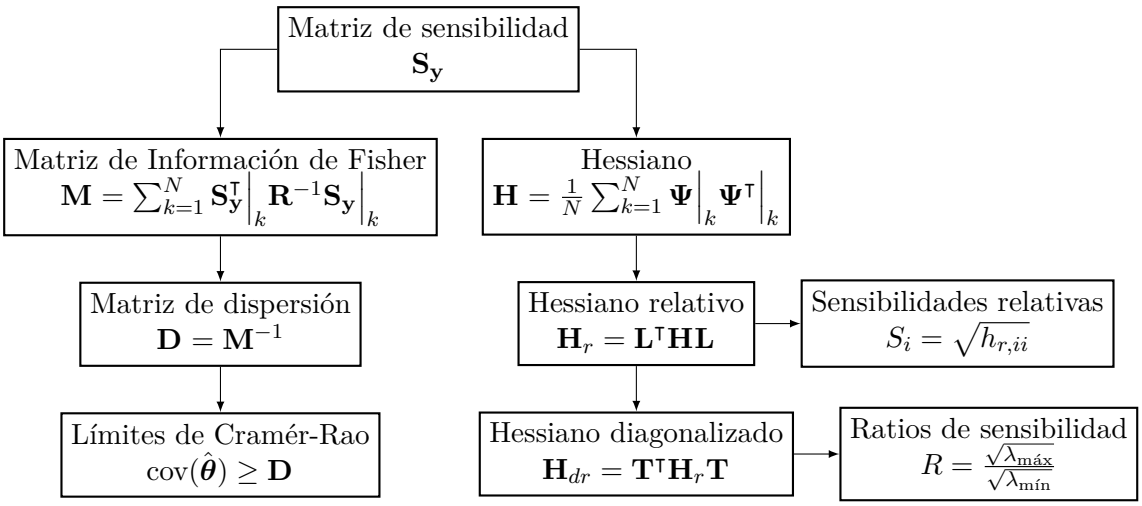


Figura 8.3: Relaciones entre matrices y otras métricas de sensibilidad.

La FIM se puede obtener mediante la expresión [130]

$$\mathbf{M} = \sum_{k=1}^{N} \mathbf{S_y^\intercal}\Big|_k \mathbf{R}^{-1}\mathbf{S_y}\Big|_k , \tag{8.32}$$

donde $\mathbf{R}$ denota la matriz de covarianza de ruido de medida (recuérdese el apartado 2.1) y k representa cada una las N muestras obtenidas en la simulación del experimento. La inversa de esta matriz se conoce como matriz de dispersión $\mathbf{D}$ [131]

$$\mathbf{D} = \mathbf{M}^{-1} , \tag{8.33}$$

y representa la menor covarianza que es posible obtener de unas estimaciones de parámetros a partir de un experimento dado. Es decir, si la covarianza asociada a la estimación del vector de parámetros $\hat{\boldsymbol{\theta}}$ se denota por $\mathrm{cov}(\hat{\boldsymbol{\theta}})$, entonces se verifica [131]

$$\mathrm{cov}\left(\hat{\boldsymbol{\theta}}\right) \geq \mathbf{D} . \tag{8.34}$$

Esta covarianza mínima se conoce como el límite inferior de Cramér-Rao (*Cramér-Rao lower bound*) [10]. La matriz $\mathbf{D}$ es usada por Gupta [131] y Morelli [127, 132, 133], dos investigadores de la NASA, para el diseño de experimentos de identificación. Su función de coste a minimizar consiste en alguna transformación de la matriz de dispersión $\mathbf{D}$, como su determinante, traza o traza ponderada[1].

En [135], se plantea otra estrategia para el diseño de experimentos, que se aplica con éxito en [136, 137]. En estos artículos, en lugar de trabajar con la matriz de sensibilidad $\mathbf{S_y}$, trabajan con el *gradiente del modelo* (*model gradient*), al que se denota por $\mathbf{\Psi}$, que corresponde con una sola fila de la matriz de sensibilidad $\mathbf{S_y}$; es decir, se debe aplicar el procedimiento a cada salida (fila de $\mathbf{S_y}$) del sistema. Este gradiente se utiliza para calcular una matriz hessiana aproximada mediante la ecuación

$$\mathbf{H} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{\Psi}\Big|_k \mathbf{\Psi}^\intercal\Big|_k . \tag{8.35}$$

[1]Se entiende por traza de una matriz cuadrada a la suma de elementos de su diagonal principal [134]. La traza ponderada consiste en una suma ponderada de estos.

Con este hessiano, en [135, 136] se calcula una nueva matriz normalizada o relativizada a los valores de los parámetros, dada por

$$\mathbf{H}_r = \mathbf{L}^\intercal \mathbf{H} \mathbf{L}\,, \tag{8.36}$$

donde

$$\mathbf{L} = \mathrm{diag}\,(\boldsymbol{\theta}) \tag{8.37}$$

es una matriz cuya diagonal principal es el vector de parámetros. A partir de este hessiano relativo, se puede obtener la *sensibilidad relativa* de cada parámetro θ_i a identificar como

$$S_i = \sqrt{h_{r,ii}} \quad \text{con} \quad h_{r,ii} = \{\mathbf{H}_r\}_{ii} \tag{8.38}$$

donde $\{\mathbf{H}_r\}_{ii}$ denota el i-ésimo elemento diagonal de $\mathbf{H}_r$. De este modo, para mejorar las estimaciones se busca maximizar la sensibilidad relativa S de todos los parámetros. Sin embargo, esto no es suficiente para garantizar una buena identificación; se desea también que las sensibilidades de todos los parámetros sean parecidas. Esto se puede comprobar con los autovalores de la matriz hessiana diagonal $\mathbf{H}_{dr}$, dada por

$$\mathbf{H}_{dr} = \mathbf{T}^\intercal \mathbf{H}_r \mathbf{T} \tag{8.39}$$

donde $\mathbf{T}$ es una matriz de transformación cuyas columnas son los autovectores de $\mathbf{H}_r$. La ecuación (8.39) devuelve una matriz cuyos elementos diagonales λ_i permiten obtener el ratio de sensibilidad R dado por

$$R = \frac{\sqrt{\lambda_{\text{máx}}}}{\sqrt{\lambda_{\text{mín}}}}\,. \tag{8.40}$$

Cuando más cercano a 1 sea R, más parecidas en tamaño serán las sensibilidades relativas de todos los parámetros, lo que indica baja correlación entre parámetros.

A partir de estas métricas, en [136] se propone un problema de optimización para diseñar experimentos, que consiste en mejorar de forma simultánea todos los indicadores de sensibilidad. El método debe aplicarse a cada salida del sistema j. Formalmente, se expresa como

$$\min_{\mathbf{u}} \max \left\{ R_j\,, \quad R_{j,i}\,, \frac{1}{S_{j\,\text{mín}}}\,, \frac{1}{S_{j,i\,\text{mín}}} \right\}\,, \tag{8.41}$$

donde R_j se obtiene mediante la ecuación (8.40) para la j-ésima salida del sistema; $S_{j,\text{mín}}$ es la mínima sensibilidad de los parámetros para la salida j, obtenida mediante la ecuación (8.38); $S_{j,i\,\text{mín}}$ es la mínima sensibilidad del i-ésimo parámetro para la salida j, obtenida mediante la expresión

$$S_{j,i\,\text{mín}} = \sqrt{\left\{\mathbf{H}_r^{-1}\right\}_{ii}^{-1}}\,, \tag{8.42}$$

que denota la raíz cuadrada del inverso del i-ésimo elemento de la diagonal de la inversa de la matriz $\mathbf{H}_r$; por último, $R_{j,i} = \frac{S_{j,i\,\text{mín}}}{S_{j\,\text{mín}}}$ es la relación entre la sensibilidad mínima del parámetro θ_i y la sensibilidad mínima de la j-ésima salida del sistema.

8.6. DETECCIÓN DE FALLOS

Como se ha observado, el UKF puede emplearse para identificar los parámetros del modelo de un sistema. Esta capacidad resulta sumamente útil porque permite actualizar el modelo de la planta en tiempo real, reconociendo sus variaciones a lo largo del tiempo y corrigiendo las

imprecisiones del modelo. Un fallo en el sistema, como la rotura de algún actuador, se manifiesta en un cambio en la dinámica del mismo y, por tanto, puede ser detectado por un UKF utilizado de esta manera. De este modo, se introduce el concepto de *detección de fallos*. Hay que recordar que los parámetros del modelo se consideran constantes, de manera que cualquier cambio en ellos se traduce en una variación en la dinámica del sistema, lo que puede ser interpretado como un fallo.

La detección de fallos es un campo muy estudiado en la literatura sobre control, incluyendo su aplicación a los UAV [138, 139]. En [140] se recopilan y analizan diversos métodos propuestos para la detección de fallos en estos vehículos. En dicho artículo, se establece una distinción entre los métodos basados en modelo, como la estimación de parámetros y estados, y aquellos basados en datos, tales como pruebas estadísticas o redes neuronales. En el presente trabajo, se estudió la alternativa basada en identificación de parámetros, dado que permite reutilizar el UKF previamente usado para estimar el estado del sistema. Esta parece ser una técnica muy extendida según [141].

En el anterior artículo [140], se exponen los distintos tipos de fallos más estudiados: fallos de actuadores y fallos de sensores:

- Dentro de los fallos de actuadores se encuentran:
  - Pérdida de efectividad (*loss of effectiveness*)
  - Desviación al extremo del rango (*handover*)
  - Bloqueo en una posición (*lock-in-place*)
  - Posición flotante sin control (*float*)
- Entre los fallos de sensores se incluyen:
  - Error de sesgo (*bias*)
  - Error de deriva (*drift*)
  - Pérdida de precisión (*loss of accuracy*)
  - Congelamiento del valor medido (*freezing*)

Por simplicidad, en este trabajo se consideraron únicamente los fallos de actuadores. En particular, se estudiaron los fallos de pérdida de efectividad, que son relativamente fáciles de detectar mediante el UKF, una vez planteado el modelo de la parte II.

La pérdida de efectividad significa que un actuador no rinde como se esperaba, de manera que su efecto en el sistema es menor al previsto. Por ejemplo, si una superficie de control (o su servomotor asociado) pierde efectividad, esta producirá momentos menores sobre el avión, manifestándose en forma de respuestas más lentas de la aeronave.

Independientemente de la causa de este fenómeno, el efecto producido en el modelo es una reducción del coeficiente aerodinámico de momento asociado a una superficie de control ($C_{m\delta_e}$, $C_{l\delta_a}$ o $C_{n\delta_r}$). Por tanto, la pérdida de efectividad se traduce en un cambio en los parámetros del modelo del sistema, que pueden ser identificados con un UKF de forma recursiva en tiempo real. Según lo dicho anteriormente, si los parámetros (considerados constantes) del modelo cambian sus valores, es posible que se trate de un fallo. De esta manera, el UKF y su capacidad de estimación de parámetros del modelo también permite detectar fallos en un sistema, todo de forma simultánea y en tiempo real.

De forma similar, los errores de deriva o sesgo de sensores también podrían detectarse mediante un UKF. En su caso, bastaría añadir como parámetros del modelo los sesgos o derivas de los sensores (recuérdese el apartado 6.1), que se estimarían de forma recursiva. Esto también

permitiría, en teoría, calibrar sensores de forma automática. Sin embargo, en este trabajo no se ha implementado esta funcionalidad.

En el caso de las variaciones de parámetros, se podría considerar un umbral arbitrario para determinar cuándo se ha producido un fallo. Por ejemplo, si el valor estimado de un parámetro cae por debajo de un cierto umbral, se podría considerar que el sistema ha sufrido una pérdida de efectividad. Una versión de este método se encuentra en [142], donde se usa un modelo para las entradas de la forma

$$u_{fi} = (1 + \gamma_i)u_i \,, \qquad i \in \{1, \ldots, n_u\} \tag{8.43}$$

donde u_{fi} es la i-ésima entrada del sistema con fallo (la que de verdad se aplica al sistema), u_i es la i-ésima entrada nominal (la orden enviada al actuador), $\gamma_i \in [-1, 0]$ es el parámetro de pérdida de efectividad y n_u es el número de entradas del sistema. Si el valor estimado de γ_i es cercano a 0, el actuador funciona correctamente, mientras que si su valor es negativo y cercano a -1, el actuador ha sufrido una pérdida de efectividad.

De forma similar, en [139] se propone un modelo de fallo de actuador basado en un parámetro de efectividad ω_a que multiplica la entrada del actuador, de forma que la entrada aplicada al avión toma la forma

$$\mathbf{u}_f = \omega_a \mathbf{u} + \sigma_a \,, \tag{8.44}$$

donde $\mathbf{u}_f$ es la entrada aplicada al sistema, $\mathbf{u}$ es la entrada nominal y σ_a es el sesgo del actuador, otro parámetro a estimar. $\omega_a \in [0, 1]$ es el parámetro de efectividad del actuador, con 1 indicando un funcionamiento correcto y 0 un fallo total. El parámetro σ_a es un sesgo que podría servir para detectar la posición flotante del actuador cuando este falle. En el citado artículo también se aplica una técnica similar para los modelos de los sensores, añadiendo sus parámetros de sesgo y deriva como variables a estimar.

Otras técnicas de detección de fallos son los test estadísticos, como el propuesto en [97], que se basa en la comparación de las innovaciones $\boldsymbol{\mu}_k$ del UKF con una distribución χ^2. Otros parecidos se pueden encontrar en [143, 144]. Las ecuaciones para realizar el test propuesto en [97] ya se incluyeron en el apartado 8.4, donde se describió el método de adaptación de las matrices de ruido del UKF. En cierto modo, cuando se detecta un fallo, el modelo preestablecido deja de ser válido y el UKF debe tenerlo en cuenta, por lo que se debería actualizar su matriz de covarianza $\mathbf{Q}$.

Los métodos anteriormente descritos se podrían aplicar al piloto automático desarrollado en este trabajo con relativa facilidad. Sin embargo, dada la gran extensión y complejidad que ha terminado teniendo el trabajo en su conjunto, se ha optado por dejar este aspecto para el futuro.

9. CONTROLADOR DE VUELO

En este capítulo se aborda el estudio y diseño del bloque controlador basado en la técnica de control predictivo por modelo, conocida por sus siglas en inglés *Model Predictive Control* (MPC) y su variante no lineal, el NMPC (*Nonlinear* MPC). Este enfoque se distingue por su capacidad para anticipar el comportamiento de un sistema y determinar las acciones de control óptimas, teniendo en cuenta restricciones explícitas en las variables de entrada y estado. A lo largo del capítulo, se explicará en detalle el principio de funcionamiento de esta técnica de control, se comparará con otras estrategias tradicionales y se presentarán los aspectos específicos de su implementación en el marco del proyecto.

Para la redacción de este capítulo, ha servido de gran ayuda el libro *Small Unmmanned Aircraft: Theory and Practice* [24], que trata, entre otros temas, el control automático de UAV mediante distintos controladores, aunque no incluye el MPC.

Para el desarrollo del NMPC, fueron especialmente útiles las referencias [12, 145-147], que tratan sobre controladores predictivos, su funcionamiento y su implementación. También se han consultado los artículos [148, 149], que tratan sobre la implementación de controladores MPC en MATLAB y la librería `acados`, respectivamente. Esta última ha sido la librería usada para implementar el NMPC en este trabajo.

Otras referencias destacables son los TFG de Jorge Castillo [4] y de Álvaro Ortiz [150], ambos desarrollados en la UPV. El primero se ha citado ya en numerosas ocasiones en los anteriores capítulos y trata del diseño de un controlador LQR, mientras que el segundo es una comparativa entre el MPC y el LQR, desarrollando ambos controladores para un avión concreto. El resto de fuentes usadas han sido en su mayor parte artículos y se irán mencionando en las siguientes páginas.

9.1. ESTADO DE LA CUESTIÓN

El MPC está ganando protagonismo como una de las técnicas más avanzadas en el ámbito del control automático debido a su capacidad para manejar sistemas complejos, multivariables y con restricciones [146]. Estas son sus principales diferencias con el control PID (proporcional-integral-derivativo), que hasta ahora sigue siendo el algoritmo de control más extendido [147, 151].

A pesar de su gran popularidad—sobre todo en sistemas SISO—, las limitaciones del PID se hacen evidentes cuando se trata de controlar sistemas MIMO en los que las entradas están acopladas entre sí, como es el caso de las aeronaves. Además, toda la teoría de control clásica, de la que forma parte el PID, se basa en sistemas lineales. Para controlar sistemas altamente no lineales, donde las linealizaciones no consiguen prestaciones aceptables, el control moderno como el MPC aporta nuevas herramientas que permiten trabajar con esta clase de sistemas sin recurrir a la linealización ni a simplificaciones del modelo.

En los últimos 10 años, la comunidad académica se ha dedicado a estudiar las aplicaciones de estos controles predictivos en vehículos aéreos no tripulados en todas sus variantes. Por citar algunas referencias, se pueden mencionar las obras [152-155]. Estas tratan la implementación de

controladores MPC en drones de 4 rotores (los más comunes), aeronaves de despegue y aterrizaje vertical (*vertical take-off and landing*, VTOL) y aeronaves de ala fija como la tratada en el presente trabajo.

Dentro de los MPC aplicados a aviones de ala fija, existen distintas variantes en lo referente a las variables controladas. Las tendencias dominantes se reducen a dos:

- Por un lado, en [156-158] se proponen controladores con un modelo no lineal en los que las variables controladas son la velocidad y dos de los tres ángulos de la actitud del avión; a saber: cabeceo (*pitch*) y alabeo (*roll*) o cabeceo y guiñada (*yaw*). Estos son controles relativamente sencillos que han podido ejecutarse en tiempo real, como se comenta en [159]. Estos controladores no requieren de un GPS, puesto que las variables controladas se pueden medir con una unidad inercial (actitud) y con un tubo de Pitot (velocidad aerodinámica).
- Por otro lado, en [160, 161] (y también en [153] para un dron de 4 rotores) se implementa un control predictivo controlando la posición del prototipo en el espacio, lo que se conoce como un control de seguimiento de trayectoria (*path-following*). Este es un control de más *alto nivel*, en el sentido de que se deja al controlador decidir la actitud y velocidad necesarios para alcanzar los puntos del espacio que sirven como referencia, llamados comúnmente *waypoints*. Para estos controladores se requiere conocer la posición de la nave con precisión, lo que hace necesario un GPS y una adecuada fusión de datos con la unidad inercial para mejorar la precisión de las estimaciones.

En este trabajo, se optó por una variante de la primera tendencia, de manera que se diseñó un controlador que controle la velocidad, el rumbo o guiñada y la altitud. Estas tres variables son suficientemente intuitivas como para permitir un control sencillo del avión, simplificando las pruebas.

Como ampliación futura del trabajo, se podría diseñar un sistema de seguimiento de trayectorias que recibiera la lista de *waypoints* y generara las referencias de velocidad, rumbo y altitud para este controlador. Esto queda como trabajo pendiente.

El MPC tampoco está exento de problemas. Si bien permite trabajar con sistemas no lineales, su coste computacional puede ser muy superior al PID, de modo que no puede implementarse en el mismo *hardware* y es muy posible que se requieran microcontroladores específicos de gran potencia para ejecutarlo, especialmente si la dinámica del sistema es rápida y se requiere una frecuencia de muestreo alta. Por ejemplo, en [162] se realiza un control de MAV de ala fija con una placa Khadas VIM3 [163], un ordenador compacto (un *single-board computer*, SBC) que incorpora un procesador con 4 núcleos de arquitectura ARM Cortex A73 a 2,2 GHz y 2 Cortex A53 a 1,8 GHz.

Otro problema o característica del MPC y su variante no lineal NMPC es que su estabilidad es difícil de estudiar o garantizar, como se puede comprobar en [164-166]. A diferencia de los controladores *clásicos*, el MPC no se puede diseñar mediante el lugar de las raíces o diagramas de Bode o Nyquist. En este paradigma de control no se trata de ubicar polos en lazo cerrado, sino de resolver en cada período de muestreo un problema de optimización mediante métodos numéricos, como se verá más adelante.

Estos inconvenientes son los que mantienen al MPC en un segundo plano en lo referente a pilotos automáticos de aeronaves. Las soluciones comerciales siguen usando controladores PID para estabilizar y guiar el vuelo de aeronaves no tripuladas, ya sean multirotor o de ala fija. De estas alternativas se pueden mencionar las más populares para UAV de ala fija: Ardupilot [167]

y PX4 [168]. Estos dos proyectos de código abierto consisten en el desarrollo de *software* para pilotos automáticos de UAV. Estos programas están pensados para ejecutarse en controladoras de vuelo comerciales, esto es, placas electrónicas que incorporan un microcontrolador, sensores y varios puertos de entrada y salida. De entre estos dispositivos, los más conocidos quizás sean las de la familia Pixhawk.

Durante el curso 2023-2024, Xtra2 UPV estuvo experimentando con la controladora Matek F405 Wing V2 [70], mostrada en la figura 9.1. Esta placa incorpora una unidad inercial de 6 ejes (sin magnetómetro) y un altímetro, aunque admite la conexión con más sensores mediante UART e I^2C. De esta manera se pudieron conectar un tubo de Pitot con sensor MS4525DO (véase el apartado 6.2.1) y un GPS.

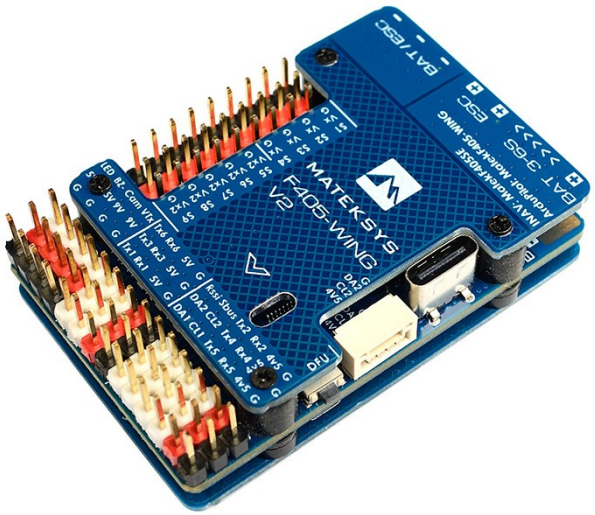


Figura 9.1: Fotografía de la controladora de vuelo Matek F405 Wing V2. Cortesía de Matek Systems [70]

9.2. FUNCIONAMIENTO DEL CONTROLADOR

El controlador predictivo por modelo tiene un principio de funcionamiento distinto al de muchos otros controladores convencionales, por lo que debe ser comprendido desde ese marco. Su funcionamiento no está ligado, al menos de forma explícita, a polos de lazo abierto o cerrado, ceros u otros conceptos tratados en la teoría de control clásica.

El enfoque predictivo del MPC se basa en utilizar un modelo matemático del sistema para anticipar su comportamiento futuro bajo diferentes condiciones. Este modelo permite evaluar el efecto de las acciones de control en un intervalo de tiempo comprendido entre el instante inicial (actual) y un instante futuro predefinido. A este intervalo se lo denomina horizonte de predicción [146].

El controlador utiliza las predicciones del modelo para calcular una secuencia de entradas que a su vez produzca una respuesta del sistema que minimice una función de coste, en la que se incorporan las penalizaciones necesarias para garantizar el seguimiento de la referencia. Esta secuencia de entradas tiene una duración temporal conocida como horizonte de control, mientras que las consecuencias de estas entradas se estudian hasta el horizonte de predicción,

que puede ser igual o mayor que el anterior [12]. En este trabajo, se ha optado por considerar ambos horizontes iguales.

La resolución del problema de optimización no solo considera el estado actual del sistema, sino también un conjunto de restricciones, como límites en las magnitudes y velocidades de cambio de las entradas y estados. El algoritmo numérico de optimización o *solver* debe hallar la secuencia de entradas óptima en el horizonte de control que minimice la función de coste evaluada en el horizonte de predicción cumpliendo estas restricciones.

Los horizontes de predicción y control definen el intervalo temporal en el que se estudia el comportamiento del sistema de forma predictiva, y deben ajustarse para equilibrar las prestaciones del sistema de control y la complejidad computacional. Este ajuste es importante para implementar el MPC en tiempo real, ya que el cálculo de las acciones de control debe completarse antes de que expire el período de muestreo y, al tratar con sistemas no lineales y multivariables, la obtención del óptimo se complica y puede no alcanzarse la convergencia con algunos *solvers* [169].

Formalmente, el problema de optimización del NMPC se define como [170]

$$\underset{\mathbf{u},\,\mathbf{s}}{\text{minimizar}} \quad J(\mathbf{u}) \tag{9.1a}$$

$$\begin{aligned}
\text{sujeto a} \qquad \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) && k = 0, \ldots, N-1, && \text{(9.1b)}\\
\mathbf{u}_{\text{mín}} &\leq \mathbf{u}_k \leq \mathbf{u}_{\text{máx}} && k = 0, \ldots, N-1, && \text{(9.1c)}\\
\mathbf{x}_{\text{mín}} - \mathbf{s}_k &\leq \mathbf{x}_k \leq \mathbf{x}_{\text{máx}} + \mathbf{s}_k && k = 1, \ldots, N, && \text{(9.1d)}\\
\mathbf{s}_k &\geq 0 && k = 1, \ldots, N, && \text{(9.1e)}\\
\mathbf{x}_0 &= \mathbf{x}_{\text{init}} && && \text{(9.1f)}
\end{aligned}$$

Siendo $J(\mathbf{u})$ la función de coste, $\mathbf{u}$ la matriz con las acciones de entrada, $\mathbf{u}_k$ la k-ésima columna de $\mathbf{u}$, $\mathbf{s}$ la matriz de variables de holgura, $\mathbf{s}_k$ la k-ésima columna de $\mathbf{s}$, $\mathbf{u}_{\text{mín}}$ y $\mathbf{u}_{\text{máx}}$ los límites inferior y superior de las entradas y $\mathbf{x}_{\text{mín}}$, $\mathbf{x}_{\text{máx}}$ los límites inferior y superior del vector de estado y N el horizonte de predicción. A continuación se explican uno por uno estos términos.

La función de coste se puede definir como [149]:

$$J(\mathbf{u}) = \sum_{k=0}^{N} \ell\left[\mathbf{x}_k, \mathbf{u}_k\right] + \phi\left[\mathbf{x}_N\right], \tag{9.2}$$

donde N es el horizonte de predicción, el subíndice 0 denota el instante inicial, $\mathbf{x}$ es el estado del sistema con dimensión n_x, $\mathbf{u}$ es la entrada del sistema de dimensión n_u, $\ell(\cdot)$ es la función de coste instantáneo y $\phi(\cdot)$ es el coste terminal, aquí considerado nulo.

Al tratar el problema en tiempo discreto, la evolución del estado $\mathbf{x}$ está gobernada por un modelo del sistema de la forma

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \tag{9.3}$$

donde $\mathbf{f}$ puede ser una función no lineal. El valor inicial $\mathbf{x}_0$ del estado al iniciar el problema viene dado por $\mathbf{x}_{\text{init}}$ y se impone como restricción en el problema de optimización. $\mathbf{x}_{\text{init}}$ se obtiene de la salida del UKF tratado en el capítulo 8, que estima el estado a partir de las medidas de los sensores y las entradas aplicadas al sistema.

La función de coste instantáneo puede tener cualquier forma, lineal o no lineal y depende de aquellas variables que se quieran controlar. Una estructura típica es la cuadrática [149], dada por

$$\ell = (\mathbf{x}_k - \mathbf{r})^\intercal \mathbf{W}_\mathbf{x} (\mathbf{x}_k - \mathbf{r}) + \mathbf{u}_k^\intercal \mathbf{W}_\mathbf{u} \mathbf{u}_k, \tag{9.4}$$

donde $\mathbf{W_x}$ y $\mathbf{W_u}$ son matrices que permiten ponderar cada variable de los vectores de estado y entrada respectivamente y $\mathbf{r}$ es la referencia a seguir.

El problema va acompañado de unas restricciones (*constraints*), que permiten especificar las limitaciones de los actuadores, del sistema o imposiciones arbitrarias que acotan la solución del problema. En este caso, la entrada debe estar acotada por unos límites inferiores y superiores [149]

$$\mathbf{u}_{\text{mín}} \leq \mathbf{u}_k \leq \mathbf{u}_{\text{máx}}\,, \tag{9.5}$$

lo que denota que cada una de las N columnas de la matriz $\mathbf{u}$ debe cumplir estas condiciones. Estas restricciones se conocen como rígidas (*hard*), puesto que no pueden violarse [170].

De forma similar, el estado también está acotado dentro de unos límites, pero estos pueden relajarse mediante una variable de holgura $\mathbf{s}_k$ que permite que el estado se desplace ligeramente fuera de los límites. Esto se expresa como [149]

$$\mathbf{x}_{\text{mín}} - \mathbf{s}_k \leq \mathbf{x}_k \leq \mathbf{x}_{\text{máx}} + \mathbf{s}_k \quad k = 1, \ldots, N \tag{9.6}$$

$$\mathbf{s}_k \geq 0 \quad k = 1, \ldots, N \tag{9.7}$$

Así, si el estado sale de los límites, la variable de holgura correspondiente tomará un valor positivo para cumplir la inecuación. Para convertir estas inecuaciones en restricciones propiamente dichas, se debe penalizar los valores de las holguras, lo que se hace en la función de coste instantáneo ℓ añadiendo un término de la forma [149]

$$\ell = (\mathbf{x} - \mathbf{r})^\intercal \mathbf{W_x} (\mathbf{x} - \mathbf{r}) + \mathbf{u}^\intercal \mathbf{W_u} \mathbf{u} + \mathbf{s}_k^\intercal \mathbf{W_s} \mathbf{s}_k, \tag{9.8}$$

donde $\mathbf{W_s}$ es una matriz de pesos que penaliza el uso de las holguras. Si el estado sobrepasa los límites, la función de coste aumentará y el *solver* buscará minimizar este valor, lo que hará que las holguras tiendan a cero. Como estas restricciones no son inviolables, se las conoce como restricciones flexibles o suaves (*soft*) [170].

Esta clase de restricciones se aplican al estado porque no se tiene un control absoluto de él y, si por alguna perturbación saliera de los límites y las restricciones aplicadas fueran rígidas, el valor inicial $\mathbf{x}_{\text{init}}$ saldría de los límites y el problema no tendría solución (*infeasible*). Las entradas son las variables controladas y a ellas sí se pueden aplicar restricciones rígidas sin estos problemas.

El problema de la ecuación (9.1) se debe resolver en cada período de muestreo, obteniendo una secuencia de entradas óptimas $\mathbf{u}$. De estas, el controlador toma la primera acción $\mathbf{u}_0$, la aplica a la planta y se espera un período de muestreo. Al expirar este, se recibe la nueva estimación del estado actual $\mathbf{x}_{\text{init}} = \hat{\mathbf{x}}_{k|k}$, actualizando la restricción, y se vuelve a resolver el problema de optimización con el nuevo estado, repitiendo el proceso indefinidamente.

La resolución del problema de optimización se realiza mediante métodos numéricos, ya que la función de coste y las restricciones pueden ser no lineales y el problema puede ser no convexo. Estos métodos numéricos se conocen como *solvers* y vienen con las distintas librerías disponibles para la implementación de estos controladores, como se verá en apartado 9.4.

9.3. SINTONIZACIÓN DEL MPC

La configuración del MPC depende, como se ha visto, de los horizontes de predicción y control (iguales en este trabajo), de las matrices de pesos de estado y entrada y de las restricciones. Las

restricciones y los horizontes vienen dados por limitaciones del sistema dinámico o del *hardware* en el que se implementen. Quedan pendientes de fijar las matrices de pesos.

La forma de ponderar el error de seguimiento ($\mathbf{r}_k - \mathbf{x}_k$) de cada variable puede condicionar la respuesta del sistema. En el caso específico del avión, por ejemplo, si se aplica peso a las velocidades angulares y sus referencias se fijan en 0, las respuestas del avión serán lentas y con pocas oscilaciones. Por otro lado, si se penaliza el error de altitud, el controlador dará más importancia a reducirlo frente a los errores de otras variables, lo que podría aumentar la velocidad con la que se alcanza la referencia, pero también producir sobreimpulsos o empeorar el seguimiento de otras variables en una maniobra combinada.

El peso asociado a las entradas penaliza su uso y puede servir para evitar entradas demasiado grandes que acaben produciendo sobreimpulsos (podría contrarrestar los sobreimpulsos debidos a grandes pesos en variables de estado), aunque también puede ralentizar las respuestas del sistema.

Los valores de estas matrices se pueden ajustar a mano, por tanteo, mediante un simulador del sistema controlado en lazo cerrado, introduciendo al controlador algunas referencias para las variables a controlar, juzgando las respuestas y corrigiendo las matrices en consecuencia. Sin embargo, este proceso se puede automatizar, como se propone en [171].

El algoritmo propuesto en este artículo consiste en un proceso iterativo en el que se calculan los errores medios en distintos períodos temporales de la respuesta, distinguiendo entre un error en régimen permanente y un error en tiempos intermedios del transitorio. Se busca minimizar el error en régimen permanente (garantiza el seguimiento de referencia), pero también penaliza un error medio positivo en el transitorio, lo que indicaría un sobreimpulso. En función de estos errores, el algoritmo cambia de forma iterativa y sistemática el peso asociado a una variable para mejorar el desempeño del controlador en su seguimiento.

Por simplicidad, en este trabajo se recurrió al método manual, ajustando los pesos hasta obtener la respuesta deseada, aunque el uso del citado algoritmo hubiera acelerado el proceso. No obstante, conviene comentar que la sintonización de esta clase de controlador resultó sencilla e intuitiva, de forma que el proceso no fue tedioso.

9.4. IMPLEMENTACIÓN

Existen varias alternativas para implementar el MPC. Hay que aclarar que la implementación es el último paso de diseño del MPC. Al estar sustentado en un problema de optimización, una vez definido adecuadamente el problema y establecidas todas las restricciones y ajustes, los resultados finales no deberían cambiar mucho de una implementación a otra, puesto que se está optimizando la misma función. No obstante, al tratarse de la minimización de una función no convexa, el número de *solvers* válidos se reduce significativamente y más aún cuando se pretende resolver el problema en pocas iteraciones y sin caer en mínimos locales. Aun así, se destaca la relativamente poca importancia que puede tener los detalles de implementación en las entradas óptimas generadas por el controlador, partiendo de que el *solver* encuentre el mínimo global.

Para este trabajo se usó MATLAB, que soporta varias librerías relacionadas con este controlador. Las más destacables son:

- `acados` [149], una librería para crear controladores MPC escrita en C con interfaces para los lenguajes Python y MATLAB u Octave.
- **MATMPC** [148], una librería hecha en MATLAB, similar a la anterior.

- La **MPC toolbox** de MATLAB [172], similar a las anteriores, pero forma parte del repositorio de *toolboxes* de MATLAB.
- **CasADi** [173], una biblioteca de optimización y diferenciación simbólica (no necesariamente enfocada a problemas de control óptimo) que sirve de base para la anterior librería `acados`, pero que también se puede usar directamente para diseñar un MPC a medida.

La librería `acados` ha sido usada con éxito para implementar controladores no lineales para UAV de ala fija, como demuestra Reinhardt [174, 175] (autor con publicaciones recurrentes sobre MPC) y otros autores [162, 164]. En cierto modo, es la alternativa popular para el desarrollo de estos controladores. También es posible construir el MPC desde cero, definiendo la función de coste con expresiones analíticas y teniendo un control preciso de la resolución del problema de optimización. Esto se puede conseguir mediante la biblioteca CasADi, que también se ha usado para diseñar controladores de UAV en [176, 177], algunos de ala fija como en [178], además de resolver otros problemas de optimización relacionados con este campo en [130].

Después de probar brevemente todas las alternativas, se eligió la librería `acados`, que resultó bastante sencilla de usar y permitió obtener un MPC funcional en poco tiempo una vez leídos varios ejemplos de uso. Esta implementación es provisional, en el sentido de que se ha hecho en un ordenador portátil (no una placa controladora de vuelo), por lo que resulta inviable para probar el controlador en un prototipo. Sin embargo, cumple con los requisitos de simulación de este trabajo y facilitó el estudio y aprendizaje sobre estos controladores.

Parte IV

RESULTADOS Y CONCLUSIONES

10. PRUEBAS DEL SIMULADOR Y EL CONTROLADOR MPC

Tras la descripción de la solución propuesta, este capítulo abre la última parte del documento con los resultados obtenidos. En las secciones siguientes, el lector encontrará las simulaciones hechas con el controlador MPC cuando este recibe el estado real completo del sistema. Con los resultados obtenidos se comprobará, primero, las capacidades del simulador y del modelo y, segundo, se verá cómo el controlador es estable y hace uso de todos los mandos del avión cumpliendo con las restricciones impuestas.

La figura 10.1 muestra el diagrama de bloques que representa las conexiones entre los bloques funcionales usados en este capítulo. La realimentación del estado $\mathbf{x}$ al controlador es directa, sin estar contaminada por ruidos, para comprobar que el controlador responde correctamente ante estados verosímiles. Además del estado, el MPC recibe la señal de referencia $\mathbf{x}_r$ y, con ambos, genera la señal de control $\mathbf{u}_{\text{cmd}}$.

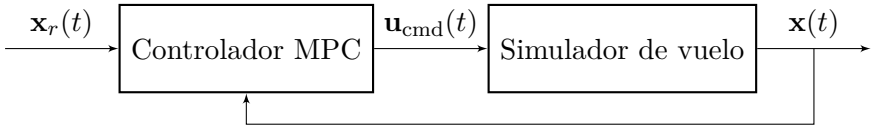


Figura 10.1: Diagrama de bloques de las pruebas del simulador y controlador.

10.1. CONFIGURACIÓN DEL REGULADOR

Las pruebas incluidas en este capítulo se han hecho con la misma configuración. Esta viene determinada por las matrices de pesos, restricciones, penalizaciones, el horizonte de predicción, de control, el período de muestreo y el modelo.

El período de muestreo depende del avión y para este caso se fijó en 0,05 s (20 Hz). Por otra parte, los horizontes de predicción y control, considerados iguales en este proyecto, también dependen del sistema a controlar y se establecieron en 5 s, que se corresponde con un número de muestras predichas $N = 100$.

El modelo de avión en tiempo discreto usado por el MPC fue el modelo no lineal del «Favara» desarrollado en la parte II y discretizado mediante Runge-Kutta 4. De esta manera, el MPC diseñado es, realmente, un MPC no lineal o NMPC.

10.1.1. Matrices de pesos

Para permitir un control sencillo del avión, se han establecido como variables de referencia las magnitudes siguientes: velocidad aerodinámica V, ángulo de guiñada ψ y altitud h. El resto de variables de estado serán modificadas según el controlador estime necesario para alcanzar los valores de referencia de las anteriores variables. Como recordatorio, el vector de estado del sistema tiene 12 componentes en el siguiente orden:

$$\mathbf{x} = \begin{bmatrix} V & \alpha & \beta & p & q & r & \phi & \theta & \psi & x_E & y_E & h \end{bmatrix}^\intercal. \tag{10.1}$$

La matriz de pesos asociada a este vector se fija en

$$\mathbf{W_x} = \text{diag}\left(\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 1\end{bmatrix}\right), \tag{10.2}$$

donde diag(·) denota una matriz diagonal cuya diagonal es el vector dado. De esta forma se asigna un peso a las variables a controlar para que el filtro tienda a minimizar su error respecto a la referencia.

La matriz de pesos de las variables de entrada se toma como:

$$\mathbf{W_u} = \text{diag}\left(\begin{bmatrix}1 & 0 & 0 & 0 & 2 & 2\end{bmatrix}\right). \tag{10.3}$$

Recordando que el vector de entrada tiene la forma

$$\mathbf{u} = \begin{bmatrix}\delta_e & \delta_p & \delta_f & i_t & \delta_a & \delta_r\end{bmatrix}^\intercal, \tag{10.4}$$

por lo que que se asigna un peso al timón de profundidad δ_e, los alerones δ_a y timón de dirección δ_r, penalizando su uso. Esto permite reducir los sobreimpulsos y algunas oscilaciones, especialmente de la dinámica lateral-direccional.

De forma intuitiva, asignando peso a las superficies de control, se pretende que el *solver* encuentre acciones de control óptimas en las que no sea necesario mover demasiado las superficies de control, lo que debería ir asociado a menos correcciones de la trayectoria y, por tanto, a menos oscilaciones.

Conviene aclarar en este punto que el ángulo de incidencia i_t del estabilizador horizontal se ha fijado en 0,187°; un valor obtenido mediante un trimado del avión (véase el apartado 4.6.1) y que permanecerá fijo durante el vuelo, por lo que no tiene sentido asignarle un peso. De forma similar, los flaps no se usarán durante la simulación y su posición δ_f estará fija en 0°. En lugar de asignarles un peso aquí, se limitará su valor mediante restricciones rígidas.

10.1.2. Restricciones

Para la elección de las restricciones a imponer al controlador se recurre a la envolvente de vuelo de la aeronave, tomando unos valores prudenciales para no alcanzar sus límites. Si bien el NMPC permite definir cualquier función de coste y, por tanto, cualquier forma de penalización por incumplimiento de las restricciones, se ha optado por una función de coste cuadrática convencional. El reparto de restricciones es el siguiente: restricciones rígidas (*hard*) para el vector de entrada y restricciones flexibles (*soft*) para el vector de estado.

Restricciones rígidas

La tabla 10.1 recoge los valores límite (superior e inferior) asociados a las variables de entrada. Todas las superficies de control disponibles tienen límites de ±8° para prevenir la entrada en pérdida de estas, mantenerlas en su rango habitual de deflexión y dejar un cierto margen para permitir posibles actuaciones del piloto sobre ellas además de las acciones de control del propio regulador. En las superficies que se mantienen fijas, sus límites superior e inferior coinciden, de modo que solo pueden tomar ese valor.

Tabla 10.1: Restricciones rígidas establecidas para las entradas.

Símb.	Lím. inf.	Lím. sup.	Unidades
δ_e	-8	8	°
δ_p	0	1	-
δ_f	0	0	°
i_t	0,187	0,187	°
δ_a	-8	8	°
δ_r	-8	8	°

Tabla 10.2: Restricciones flexibles establecidas para las variables de estado.

Símb.	Lím. inf.	Lím. sup.	Unidades	Penalización
V	8	25	m/s	1×10^4
α	-4	8	°	1×10^4
β	-1	1	°	1×10^4
p	-30	30	°/s	1×10^4
q	-30	30	°/s	1×10^4
r	-30	30	°/s	1×10^4
ϕ	-40	40	°	1×10^4
θ	-30	30	°	1×10^4
ψ	0	0	°	0
x_E	0	0	m	0
y_E	0	0	m	0
h	10	120	m	1×10^3

Restricciones flexibles

En la tabla 10.2 se muestran los valores límite de cada componente del vector de estado junto a su penalización en caso de violarse estos valores extremos. Aquellas variables que no tienen penalización, como x_E o y_E, indican que no tienen ninguna limitación en sus valores.

Los valores extremos establecidos vienen determinados por una simplificación de la envolvente de vuelo del avión. La velocidad mínima de 8 m/s, por ejemplo, es la velocidad de entrada en pérdida según [4], mientras que la máxima se ha fijado teniendo en cuenta las capacidades de la propulsión, dadas por el modelo de empuje de la figura 4.9. Otra forma de obtener una restricción de velocidad mínima pasa por igualar la sustentación al peso de la aeronave, considerando el valor de α máximo que se quiera asumir y despejando la velocidad.

De forma similar, los valores extremos de α se han fijado así no solo para prevenir la entrada en pérdida, sino para evitar salir de la zona lineal de la polar de C_L y que el modelo aerodinámico considerado siga siendo preciso. En el caso de β, como un ángulo de derrape distinto de cero indica una pérdida de eficiencia aerodinámica, se ha tratado de minimizar su valor acotándolo en un intervalo pequeño, de $\pm 1°$.

En el caso de los valores de las velocidades angulares, estas están limitadas por las aceleraciones centrípetas a las que se expone el avión al girar en estas condiciones.

Para las penalizaciones en el coste, se define la matriz de pesos $\mathbf{W_s}$, dado por:

$$\mathbf{W_s} = \text{diag}\begin{bmatrix} 1\times10^3 & 1\times10^3 & 1\times10^2 & 1\times10^3 & 1\times10^3 & 1\times10^3 \\ & 1\times10^3 & 1\times10^3 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{10.5}$$

10.2. MANIOBRAS SENCILLAS EN EL PLANO VERTICAL

Para demostrar el funcionamiento del controlador, se empieza realizando algunas maniobras sencillas en el plano vertical, siempre partiendo de las condiciones de vuelo trimado del avión. En esta clase de maniobras entran en juego el timón de profundidad δ_e y la palanca de potencia δ_p. Las variables afectadas serán las siguientes: velocidad V, ángulos de asiento longitudinal θ, de ataque α y altitud h.

Para facilitar la interpretación de los resultados, se recuerda al lector el significado de estas magnitudes mediante la figura 10.2, donde se representan gráficamente buena parte de ellas.

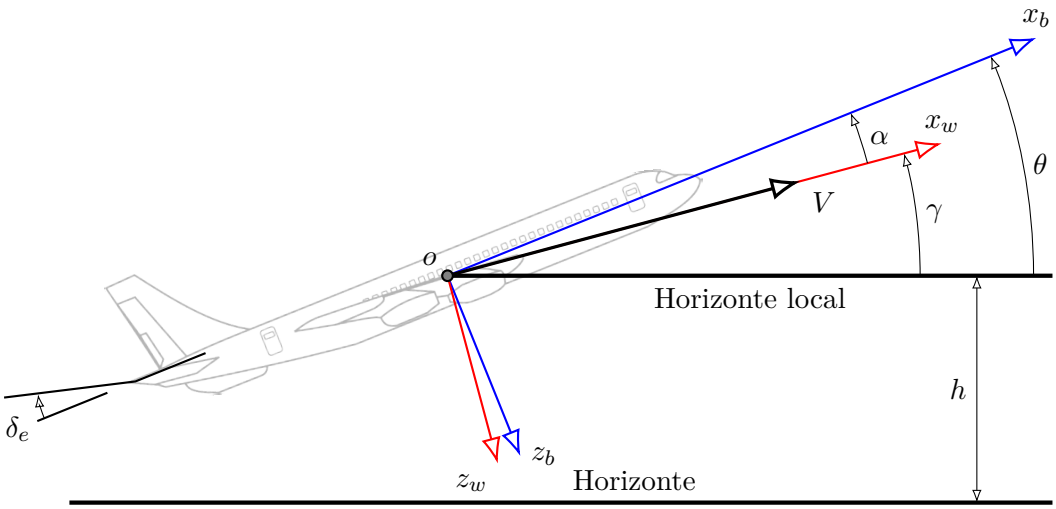


Figura 10.2: Resumen gráfico de las variables contenidas en el plano longitudinal de la aeronave.

10.2.1. Ascenso ante escalón en altitud

En la figura 10.3 se representa la respuesta del XTRA23 «Favara» ante un escalón de 10 m en la referencia de altitud partiendo de vuelo trimado. Como se puede comprobar, el escalón ocurre a partir de 1 segundo de simulación.

Examinando la evolución temporal de las variables, se puede observar cómo el controlador modifica a la vez la posición de la palanca de potencia δ_p y el elevador δ_e en cada instante para conseguir: primero, que el avión cabecee hacia arriba para comenzar el ascenso (segundos 1–2); segundo, que ascienda a una velocidad aproximadamente constante (segundos 2–3); y finalmente que cabecee hacia abajo para volver a vuelo nivelado (segundos 3–4).

Se puede apreciar la puesta al máximo de la palanca de potencia para aumentar la velocidad V y compensar por el aumento de la pendiente de la trayectoria γ (segundos 1–2); así como el uso del elevador para variar el ángulo de asiento longitudinal θ, controlar el valor del ángulo de ataque α y amortiguar las oscilaciones naturales de la aeronave (segundos 1–4).

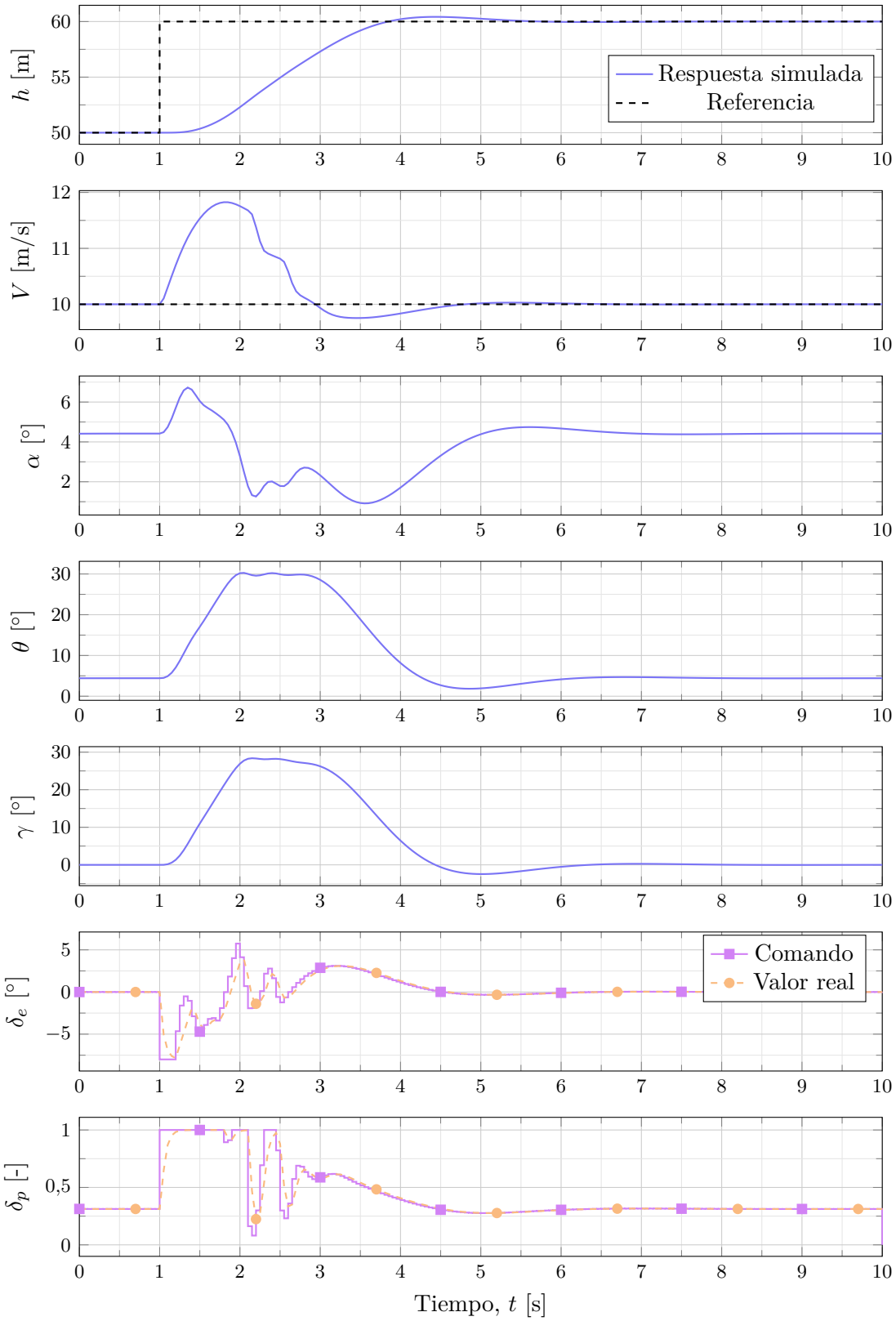


Figura 10.3: Simulación del MPC ante un escalón de 10 m en la referencia de altitud h.

Por último, cabe mencionar que el controlador cumple con las restricciones impuestas, de forma que el ángulo de ataque α no supera los 8°, el de asiento longitudinal apenar supera los 30° y las entradas se mantienen dentro de sus límites (0 a 1 para δ_p y $-8°$ a 8° para δ_e).

10.2.2. Descenso con escalón en altura

Esta vez, se aplica al avión un escalón de $-10\,\text{m}$ en la referencia de altitud. Como los mecanismos de ascenso y descenso son diferentes (no es un sistema lineal) y hay varias variables restringidas en sus valores límite, es de esperar un comportamiento y unas acciones de control distintas al caso anterior. Los resultados de la simulación se puede comprobar en la figura 10.4.

Esta vez, el controlador ha parado el motor para evitar que el avión se embale en el descenso, puesto que también se pretende controlar la velocidad (tiene un valor de consigna de $10\,\text{m/s}$). Por ello, el motor no se vuelve a usar hasta el segundo 14 (δ_p se mantiene a 0).

Para perder altitud, el controlador ha hecho cabecear el avión hacia abajo (reducir θ, que provoca una reducción de γ) para acelerar la caída, aunque esta se frena por la resistencia aerodinámica y la ausencia del empuje del motor. V aumenta en la bajada de 1 a 4 segundos y después se reduce por la resistencia aerodinámica hasta volver a la referencia. El elevador, nuevamente, se utiliza para cabecear la aeronave y compensar las oscilaciones.

En esta maniobra se puede comprobar cómo la energía cinética se convierte en energía potencial, puesto que al cabecear el avión hacia abajo su velocidad ha aumentado hasta casi $16\,\text{m/s}$. Como el UAV no tiene frenos aerodinámicos ni pretende alejarse de la consigna de altitud, lo único que puede hacer es dejarse frenar por la resistencia aerodinámica hasta volver a su valor de referencia manteniendo el motor parado.

Otra posible respuesta hubiera sido hacer un descenso más progresivo, no tan brusco, con mayor ángulo de asiento θ, que hubiera reducido el salto de velocidad en el descenso a costa de alcanzar la nueva referencia de altitud en algo más de tiempo (planear perdiendo altitud a una tasa controlada). La preferencia entre una respuesta y otra se puede ajustar mediante las matrices de pesos, asignando más peso a aquella variable que se quiera priorizar.

El ángulo de ataque α no se lleva a valores extremos porque no es una variable determinante en esta maniobra. Como se pretende reducir la sustentación para así descender, α se ha reducido hasta los $0°$ y se ha mantenido a valores bajos hasta recuperar la velocidad de consigna, ya que a mayores velocidades no se requiere tanto ángulo de ataque para conseguir la misma sustentación.

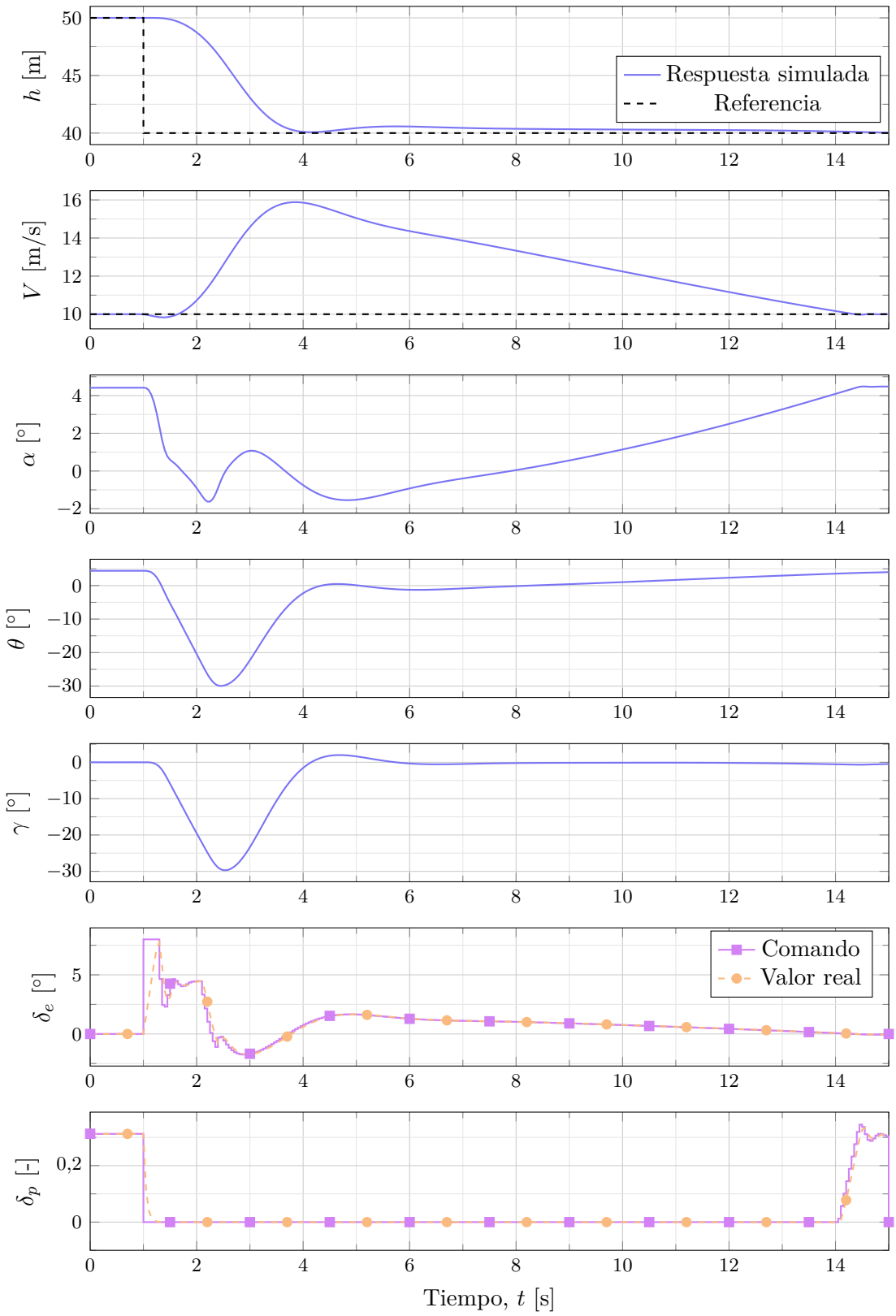


Figura 10.4: Simulación del MPC ante un escalón de $-10\,\text{m}$ en la referencia de altitud h.

10.2.3. Ascenso ante rampa de altitud

Las entradas escalón tienen la particularidad de que son muy bruscas. Hacen pasar al sistema a controlar de un estado de reposo a estar sometido a grandes acciones de control para alcanzar la nueva referencia en el menor tiempo posible. En general, exponen a la planta a grandes aceleraciones y fuerzas y las respuestas no suelen reflejar las necesidades reales de un control automático.

Las entradas rampa son entradas mucho más suaves, en las que la referencia cambia de forma continua, con una velocidad de cambio (pendiente de la rampa) constante. De forma intuitiva, en un instante cualquiera de la rampa el controlador desconoce cuánto falta para que esta acabe, por lo que aplica acciones de control mucho más suaves y lentas. Además, si el sistema es suficientemente rápido para seguir la rampa, el error instantáneo no es demasiado grande en ningún momento.

Para probar esta clase de entradas, se somete al avión a una entrada rampa en la referencia de altitud, de manera que se pretende realizar un ascenso de 15 m en 5 s (tasa de ascenso de 3 m/s). La evolución temporal del sistema se muestra en la figura 10.5.

Como se puede ver, esta vez las acciones de control se limitan a toques muy suaves del timón de profundidad δ_e al inicio y al final de la maniobra y un incremento mantenido en la palanca de potencia δ_p. Durante el ascenso, las entradas permanecen fijas, sin hacer movimientos innecesarios para compensar oscilaciones, mientras que el UAV gana altura de forma progresiva. El incremento de velocidad, de tan solo 0,2 m/s, se considera despreciable.

Para ascender, se incrementa el ángulo de asiento θ, forzando un cabeceo hacia arriba con el elevador (segundos 1–2). Este cabeceo también induce un incremento en el ángulo de ataque innecesario que se corrige inmediatamente (segundos 1,5–2,5).

La pendiente de la trayectoria γ, que demuestra el verdadero ascenso o descenso del avión, toma un valor de unos 17° durante el ascenso, necesarios para seguir la referencia indicada. Los ligeros sobreimpulsos (de 1 o 2°, a los 2,75 y 7,75 s) en sus transitorios se consideran despreciables.

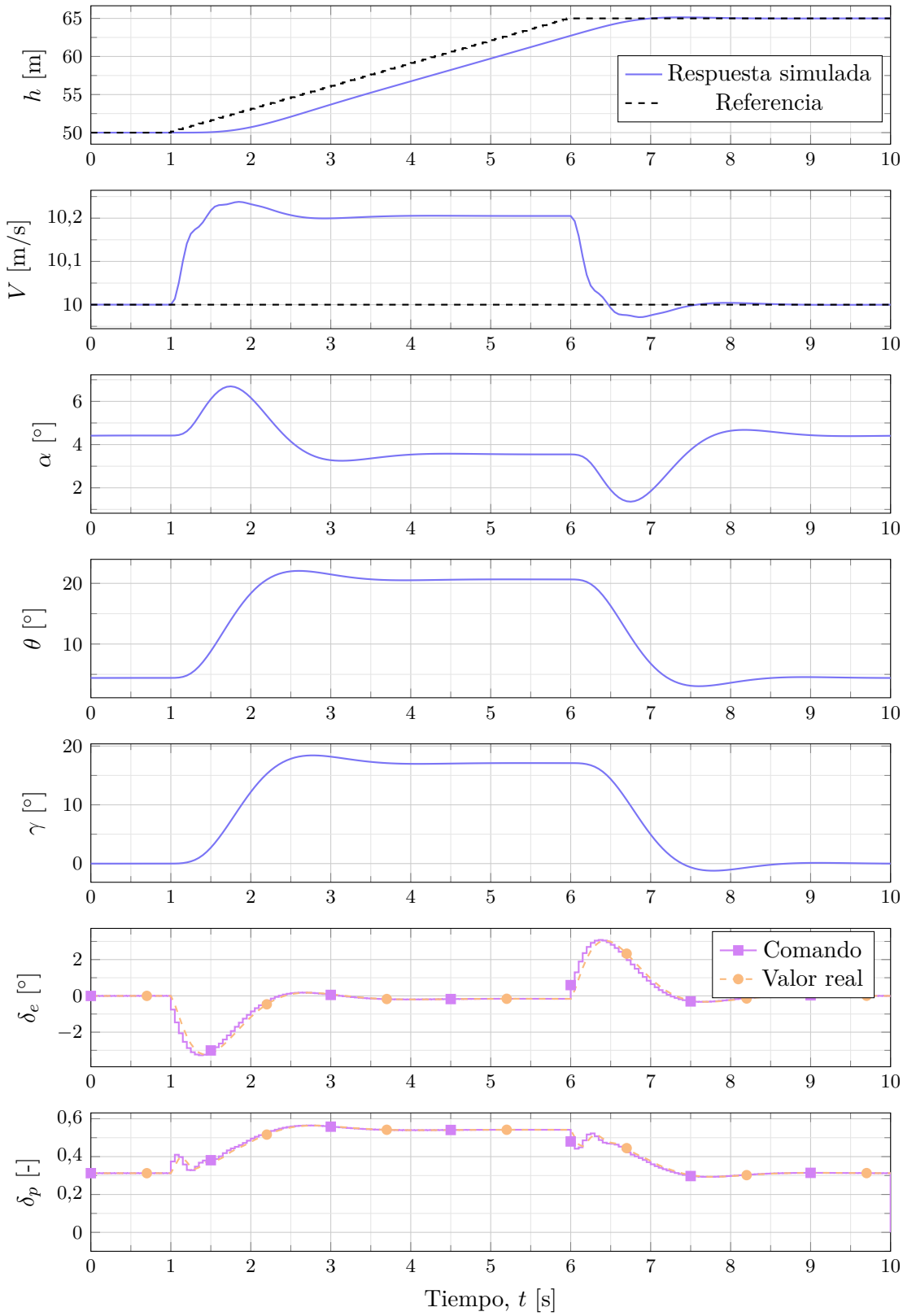


Figura 10.5: Simulación del MPC ante una entrada rampa de 3 m/s en 5 s en la referencia de altitud h.

10.2.4. Escalón en referencia de velocidad

Terminando con las pruebas en el plano longitudinal, aquí se muestran los resultados ante un escalón en la referencia de velocidad de 3 m/s, representando las variables en la figura 10.6.

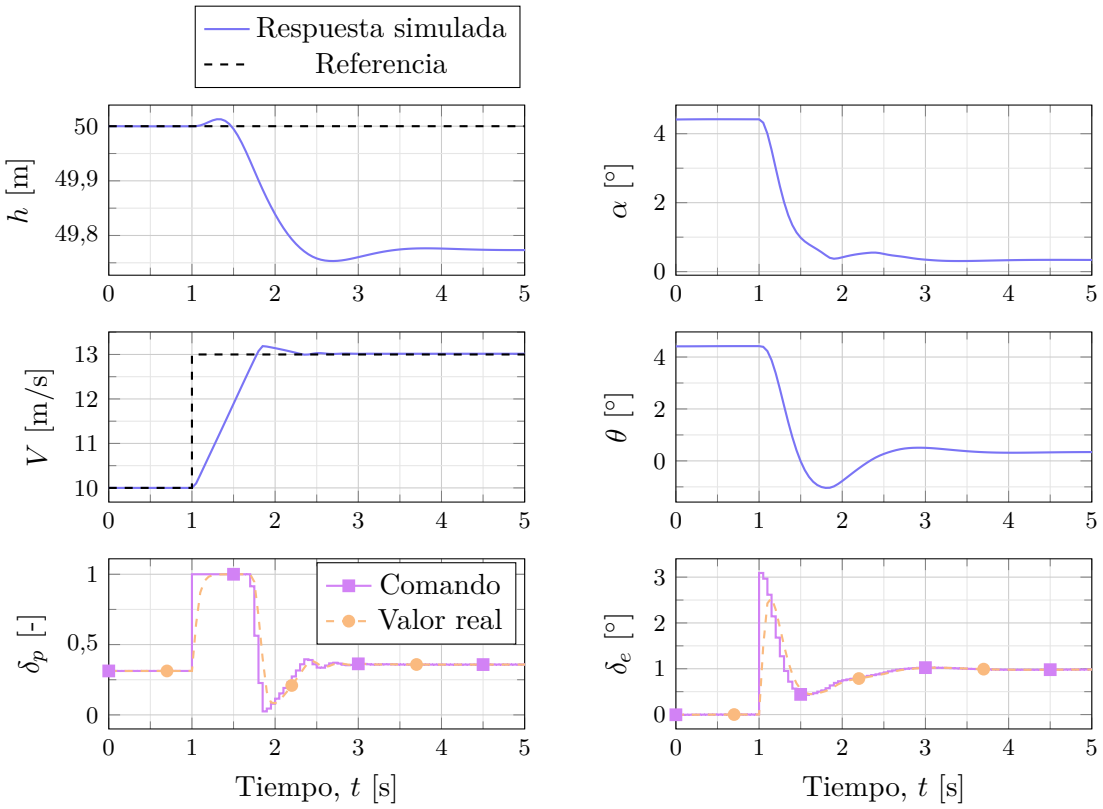


Figura 10.6: Simulación del MPC ante una entrada escalón de 3 m/s de incremento en la referencia de velocidad V.

Para incrementar la velocidad sin ascender se necesitan 2 cosas: primero, aumentar el empuje para vencer a la resistencia aerodinámica y acelerar; y segundo, reducir el ángulo de ataque para mantener constante la sustentación ante el incremento de velocidad. El controlador ha hecho ambas cosas de forma simultánea.

Para aumentar la velocidad con la máxima aceleración disponible, δ_p se pone al máximo hasta alcanzar la velocidad pedida (segundos 1–2), momento en el que vuelve a un valor ligeramente mayor del inicial, para el que se vence la resistencia aerodinámica con el empuje generado por el motor, que es la condición de velocidad constante.

Por otro lado, se debía reducir el ángulo de ataque y, para mantener una pendiente de la trayectoria nula ($\gamma = 0$), también se debe reducir θ. Esto se hace con el elevador, induciendo un cabeceo hacia abajo hasta estabilizar los anteriores ángulos en unos 0,3°.

Durante la maniobra, el avión pierde unos 20 cm de altitud, que se consideran despreciables frente a los 50 m de consigna y, como apenas contribuyen en la función de coste del controlador, este los ha pasado por alto. Como es un cambio muy pequeño, se considera irrelevante.

10.3. VIRAJE SIN PÉRDIDA DE ALTITUD

En esta sección se comentan algunas observaciones sobre la respuesta del avión en los cambios de rumbo.

Para probar con un viraje, se aplica al sistema una entrada rampa en la referencia de guiñada, pasando de 0 a 90° en 3 segundos manteniendo las referencias de velocidad y altitud constantes. Tras la simulación, se obtienen las gráficas de las figuras 10.7 y 10.8.

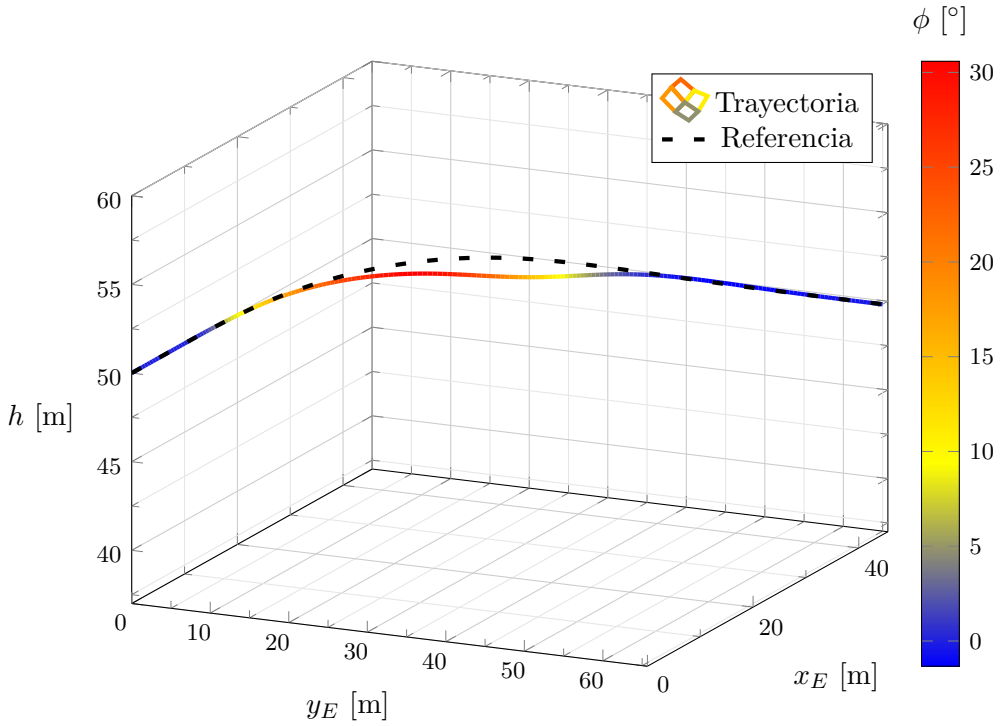


Figura 10.7: Trayectoria tridimensional recorrida por el avión.

Examinando las gráficas, se puede comprobar que, efectivamente, la variación en altitud es mínima (1 m en 50). Esta se debe a la pérdida de sustentación efectiva durante el viraje, al usarse parte de ella para producir la fuerza centrípeta necesaria para girar.

Por la mecánica del vuelo, el avión debe alabear o inclinarse hacia un lado para virar. En este caso, se ha inclinado 30° a la derecha ($\phi = 30°$ a los 4,2 segundos). Para conseguir esto, el controlador usa los alerones δ_a y el *rudder* δ_r, produciendo un momento de alabeo. Estas acciones también generan un ligero derrape del avión, que se manifiesta en forma de un incremento del ángulo β, limitado a $-1°$ por las restricciones blandas. Las oscilaciones en los controles, especialmente de los alerones, se deben a las variaciones en la función de coste del MPC debidas a estar en el valor límite de β, aunque la amplitud de estas oscilaciones es moderada y, por tanto, asumible.

Mirando la gráfica de δ_r, se puede comprobar que el controlador ha llevado a su extremo de $-8°$ la deflexión del timón de dirección. Esto es consecuencia de los coeficientes aerodinámicos asociados a esta superficie y de la maniobra pedida y no es alarmante. Además, como estaba previsto, el límite no se supera en ningún momento, puesto que está acotado por una restricción del controlador.

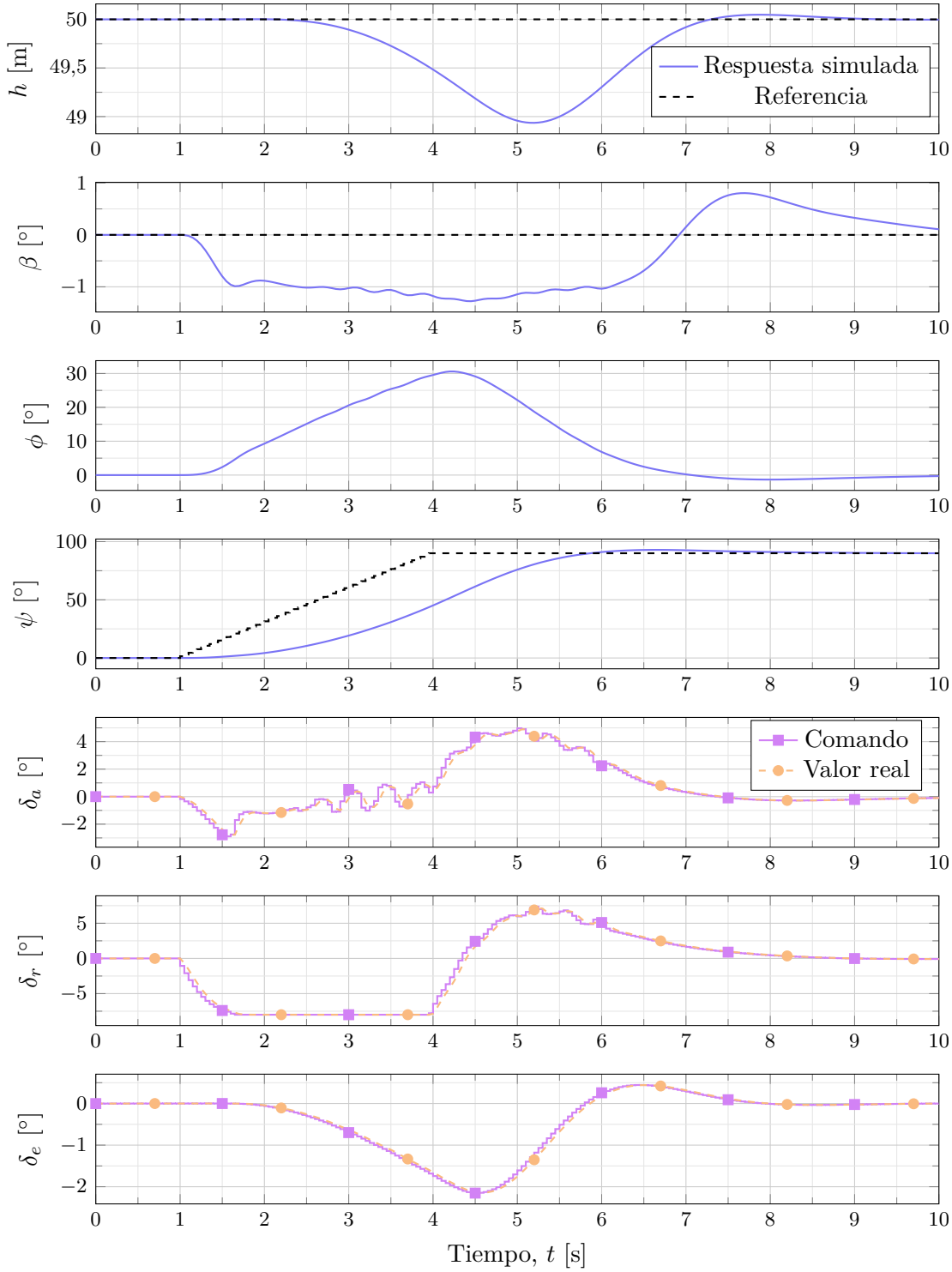


Figura 10.8: Simulación del MPC ante una entrada rampa de 30 °/s en 3 s en la referencia de guiñada ψ.

Comparando las gráficas del alabeo y la guiñada ψ, se puede ver que, efectivamente, primero el avión alabea (segundo 1,5) y esto induce, con cierto retraso, un incremento en el rumbo (incremento progresivo de ψ desde los segundos 1 a 4). Una vez alcanzado el máximo alabeo, el controlador lo reduce, pero el avión sigue virando hasta que el ángulo de alabeo se hace nulo (segundos 6–7).

Durante el viraje, el avión alabea hacia el interior de la curva, cambiando la dirección de la sustentación y reduciendo su componente vertical, que es la que mantiene el avión a altitud constante. Para compensar la reducción de esta componente, el controlador ha tratado de aumentar la sustentación total mediante un incremento en α (no representado), que se consigue mediante un ligero movimiento del elevador δ_e. Por la contribución de los distintos errores en la función de coste, el controlador ha priorizado el seguimiento de la referencia de guiñada o rumbo frente a la altitud, de manera que esta se ha visto ligeramente reducida puntualmente, aunque se ha recuperado hacia el final de la simulación.

10.4. ASCENSO HELICOIDAL

Una maniobra un poco más compleja consiste en realizar un ascenso con cambio de rumbo, lo que resulta en una trayectoria con forma de hélice. La figura 10.9 muestra la característica curva de la trayectoria durante esta maniobra.

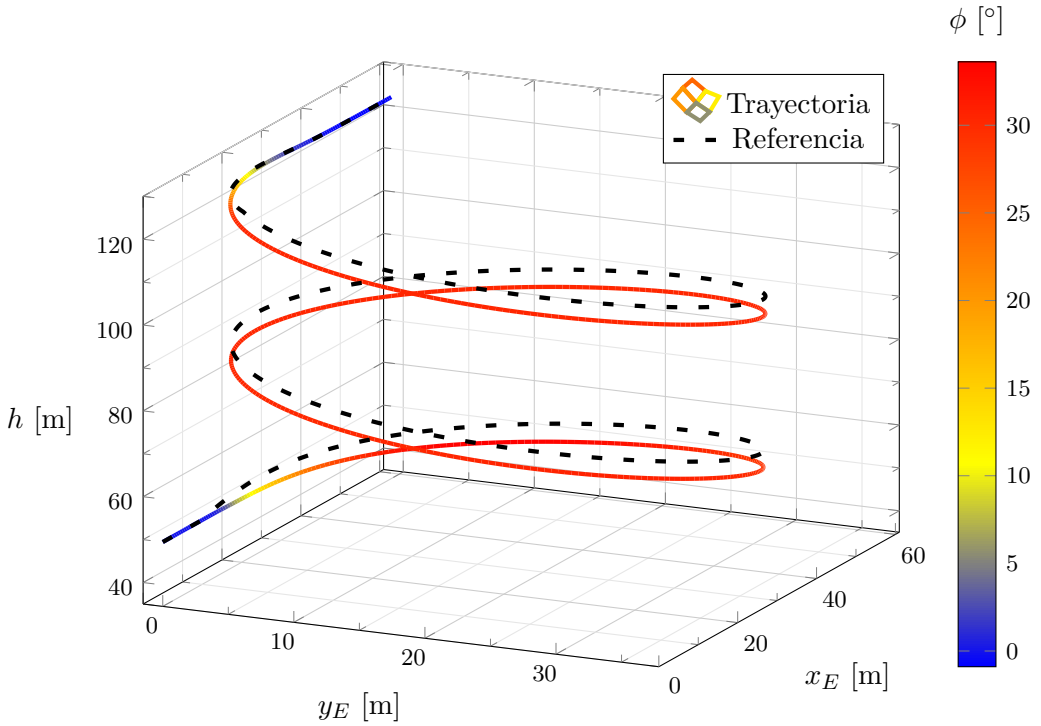


Figura 10.9: Trayectoria tridimensional recorrida por el avión durante el ascenso en hélice.

Durante la maniobra, el controlador debe combinar el manejo del elevador, alerones y timón de dirección para seguir la trayectoria y coordinar el viraje para minimizar el deslizamiento β. Las figuras 10.10 y 10.11 muestran la evolución temporal de los controles y las variables de estado más relevantes.

Como se ha dicho, en esta maniobra se combina un ascenso con un cambio de rumbo, de manera que el controlador debe, por un lado, cabecear hacia arriba el avión, aumentar la pendiente de la trayectoria y mantener una velocidad constante. Esto se hace, con algo más de dificultad que en los ejemplos anteriores, en los segundos 1 a 8. En esta simulación, el alabeo ha interferido con el ascenso, de manera que el controlador ha debido priorizar en cada momento las acciones de uno u otro, llevando a ligeras oscilaciones en estos primeros segundos.

Una vez alcanzado un equilibrio, los controles se mantienen fijos durante el ascenso helicoidal, ganando altitud y ascendiendo con una pendiente γ de unos 15°. Para conseguir esta pendiente, el motor ha tenido que acelerarse hasta casi un 60 % de su empuje máximo para esta velocidad ($\delta_p = 0{,}6$ desde los 6 segundos hasta los 25). Del mismo modo, los alerones y el *rudder* se ajustan para mantener la tasa de giro necesaria para la maniobra, permitiendo un derrape de $-1{,}5°$, algo superior al límite establecido, pero que sigue siendo un valor muy pequeño que no resulta alarmante.

Una vez terminado el ascenso, los controles prácticamente regresan de forma suave a sus valores iniciales, sin provocar apenas oscilaciones en ninguna variable.

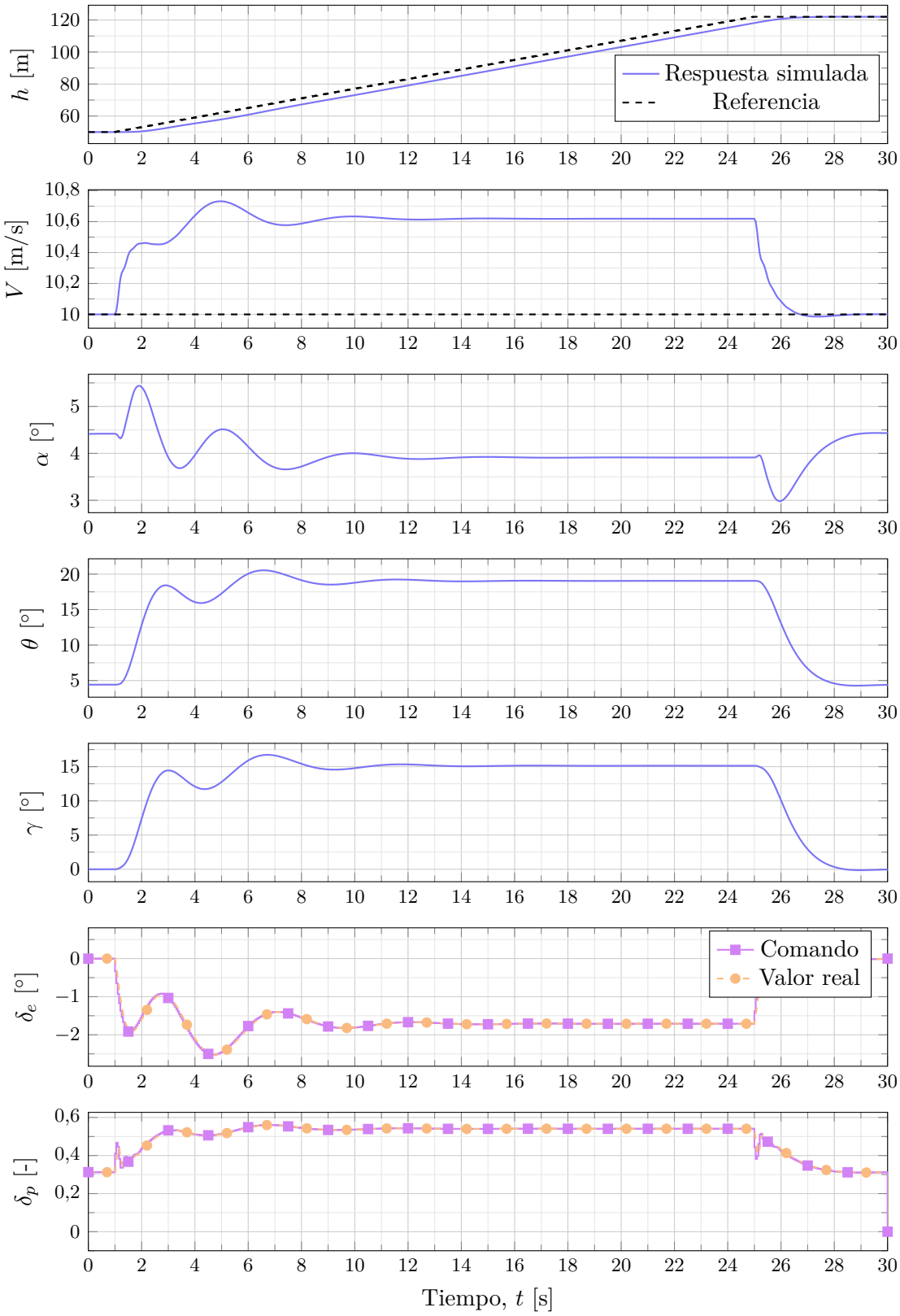


Figura 10.10: Evolución de las magnitudes de la dinámica longitudinal durante el ascenso helicoidal.

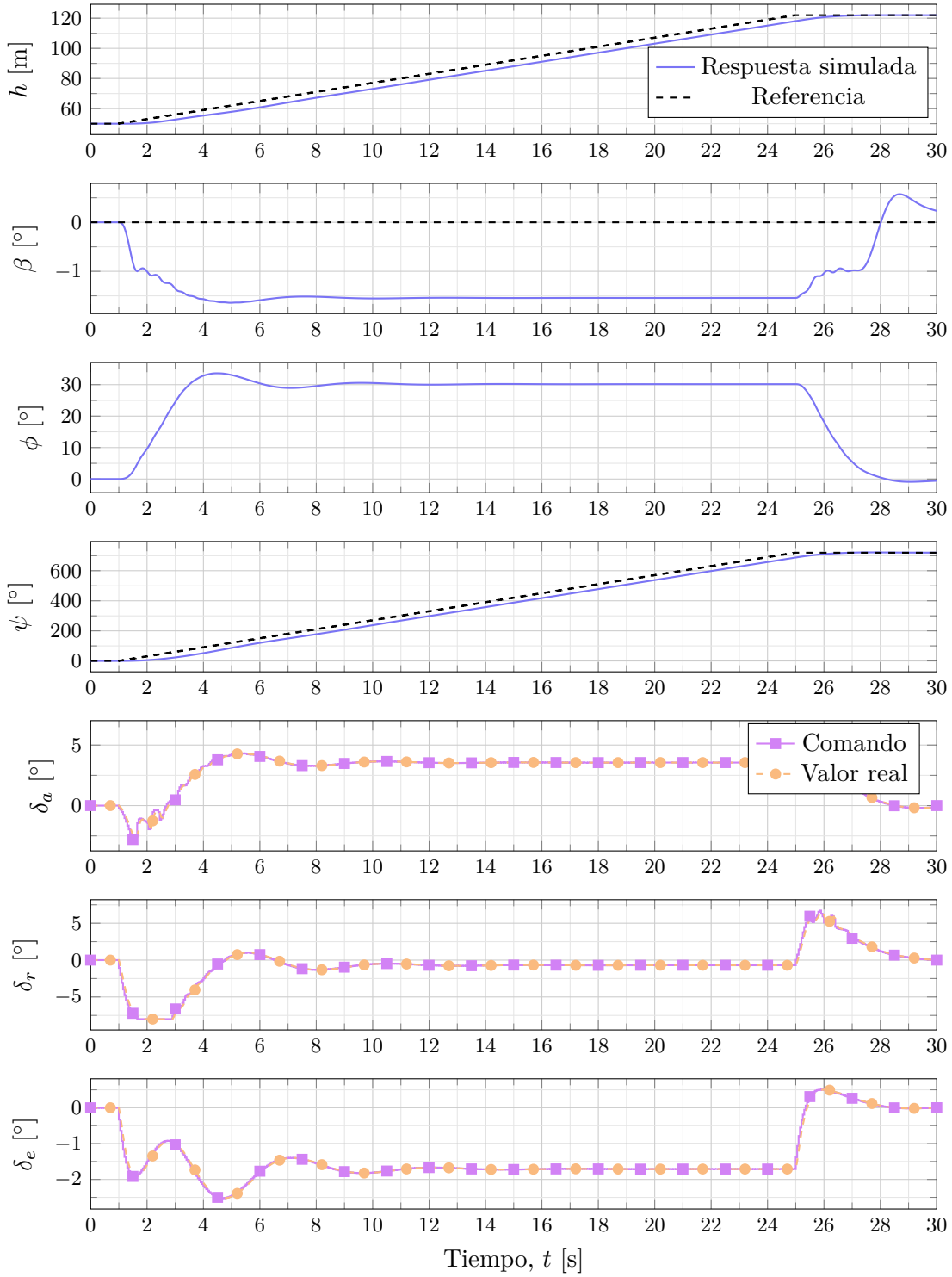


Figura 10.11: Evolución de las magnitudes de la dinámica lateral durante el ascenso helicoidal.

11. PRUEBAS DEL UKF Y DEL LAZO DE CONTROL COMPLETO

En este capítulo se demuestran las capacidades del filtro de Kalman desarrollado en el proyecto. En las páginas siguientes se incluyen gráficas en las que se puede comprobar la calidad de las estimaciones del estado que puede alcanzar el UKF trabajando como observador del estado del sistema. Estas pruebas fueron hechas con el lazo de control completo, con el MPC controlando el avión y el UKF en la realimentación.

11.1. CONFIGURACIÓN

Para demostrar el funcionamiento del filtro de Kalman y que este estime o filtre las medidas ruidosas, primero hay que establecer una maniobra para simular y una configuración del filtro. La maniobra elegida es una que combina un descenso de 15 m con un viraje de 90° a la derecha de forma simultánea. Esto garantiza variaciones en todas las variables del sistema y permitirá ver el seguimiento que el filtro hace de estas. La trayectoria completa seguida por el avión tras la maniobra, controlada por el MPC, se muestra en la figura 11.1.

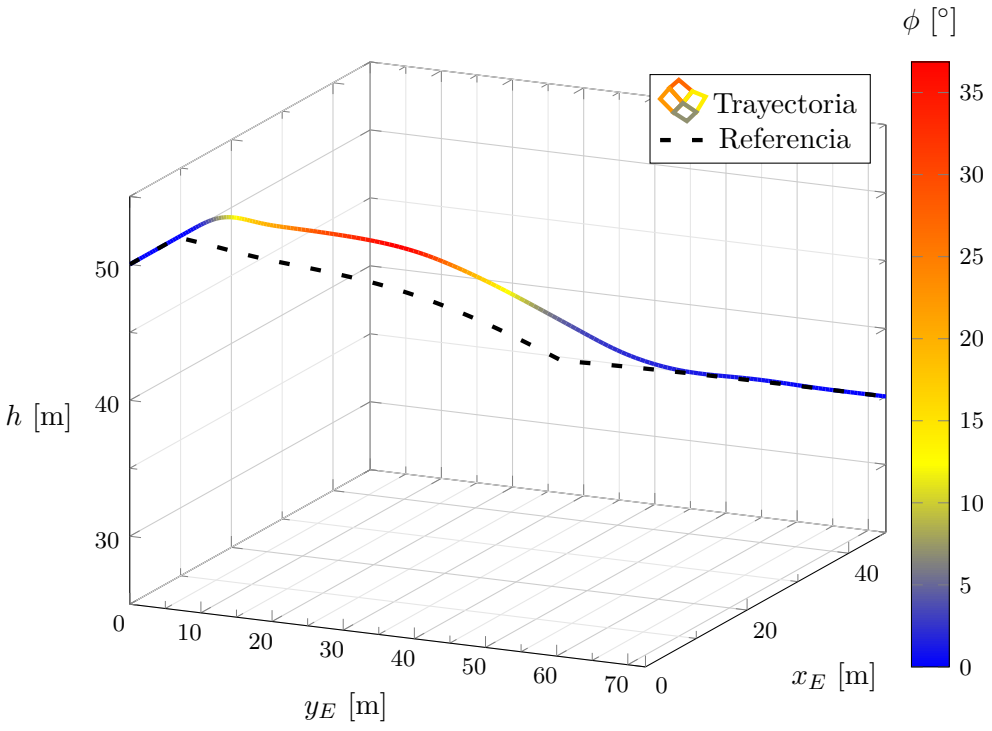


Figura 11.1: Trayectoria del avión durante la maniobra combinada.

Durante la simulación de la maniobra, las salidas se contaminan con un ruido gaussiano de media nula. La matriz de ruido de medida **R** usada es la que se definió en el capítulo 6.

El sistema no tiene ruido de proceso, pero, desde el punto de vista del filtro, se asume que el modelo del avión tiene un ruido aditivo de media nula cuya matriz de covarianza $\mathbf{Q}$ viene dada por

$$\mathbf{Q} = \text{diag}\left(0{,}01 \cdot \mathbf{x}_0 + 1 \times 10^{-5}\right) , \tag{11.1}$$

siendo $\mathbf{x}_0$ el vector de estado inicial de la simulación.

De igual modo, los parámetros de configuración del UKF se dejan a sus valores por defecto, con $\alpha = 1 \times 10^{-4}$, $\beta = 2$ y $\kappa = 0$.

La covarianza inicial del estado se establece en

$$\mathbf{P}_0 = 0{,}1 \cdot \hat{\mathbf{x}}_0 + 1 \times 10^{-4} . \tag{11.2}$$

El MPC usado mantiene la configuración establecida en el apartado 10.1, con sus matrices de pesos, restricciones y horizonte de control.

Como particularidad de estas pruebas, el período de muestreo se configuró a 0,1 s (10 Hz de frecuencia de muestreo), de manera que se recibían la mitad de muestras por segundo que en las pruebas del controlador del capítulo 10, reduciendo la información que recibía el filtro y dificultando su trabajo. El número de muestras del horizonte de control tuvo que cambiarse en consecuencia, pasando a ser $N = 50$ para mantener el horizonte de 5 s.

En general, las frecuencias de muestreo usadas en los controladores de aviones son mucho más altas. Por ejemplo, en [10] se sugieren frecuencias de muestreo de 50 o 100 Hz. Si bien estas pueden ser innecesariamente altas para los UAV tratados en este trabajo, lleva a indicar que las frecuencias de muestreo altas son aconsejables y resultan favorables para los sistemas de control.

11.2. ESTIMACIÓN DE ESTADOS

Durante la maniobra, las medidas obtenidas del simulador se hacen pasar al UKF junto a las entradas para estimar el estado real del sistema. Esto debe filtrar el ruido en aquellas variables que se pueden medir y crear estimaciones de aquellas variables para las que no se tenía sensores. Los resultados de la estimación se muestran en la figura 11.2. Estas estimaciones se hacen llegar al controlador, que genera entradas óptimas como en el anterior capítulo.

Como se puede comprobar, a pesar de que el ruido en algunas medidas sea relativamente grande, como es el caso de la velocidad V, el filtro consigue eliminar el ruido de estas y generar estimaciones muy cercanas al valor real. En general, las estimaciones de las variables medidas son excelentes.

En el caso particular de los ángulos aerodinámicos α y β, al no disponer de sensores para medirlos, el filtro debe estimar sus valores usando el modelo. A pesar de este inconveniente, las estimaciones se parecen mucho a sus valores auténticos, gracias a la gran precisión del modelo incluido en el UKF. Las estimaciones iniciales de estas variables, desde el segundo 0 al 3, no son demasiado buenas, pero una vez el filtro ha conseguido las muestras suficientes, las estimaciones terminan convergiendo a los valores correctos.

La conclusión que se puede sacar de esto es que el filtro de Kalman permite un filtrado muy eficaz de medidas contaminadas con ruido, incluso si este es de gran amplitud. Como gran diferencia frente a los filtros tradicionales, como los de tipo paso bajo o pasa banda, este sistema permite, además de filtrar, estimar valores de variables cuyo valor no se puede medir directamente.

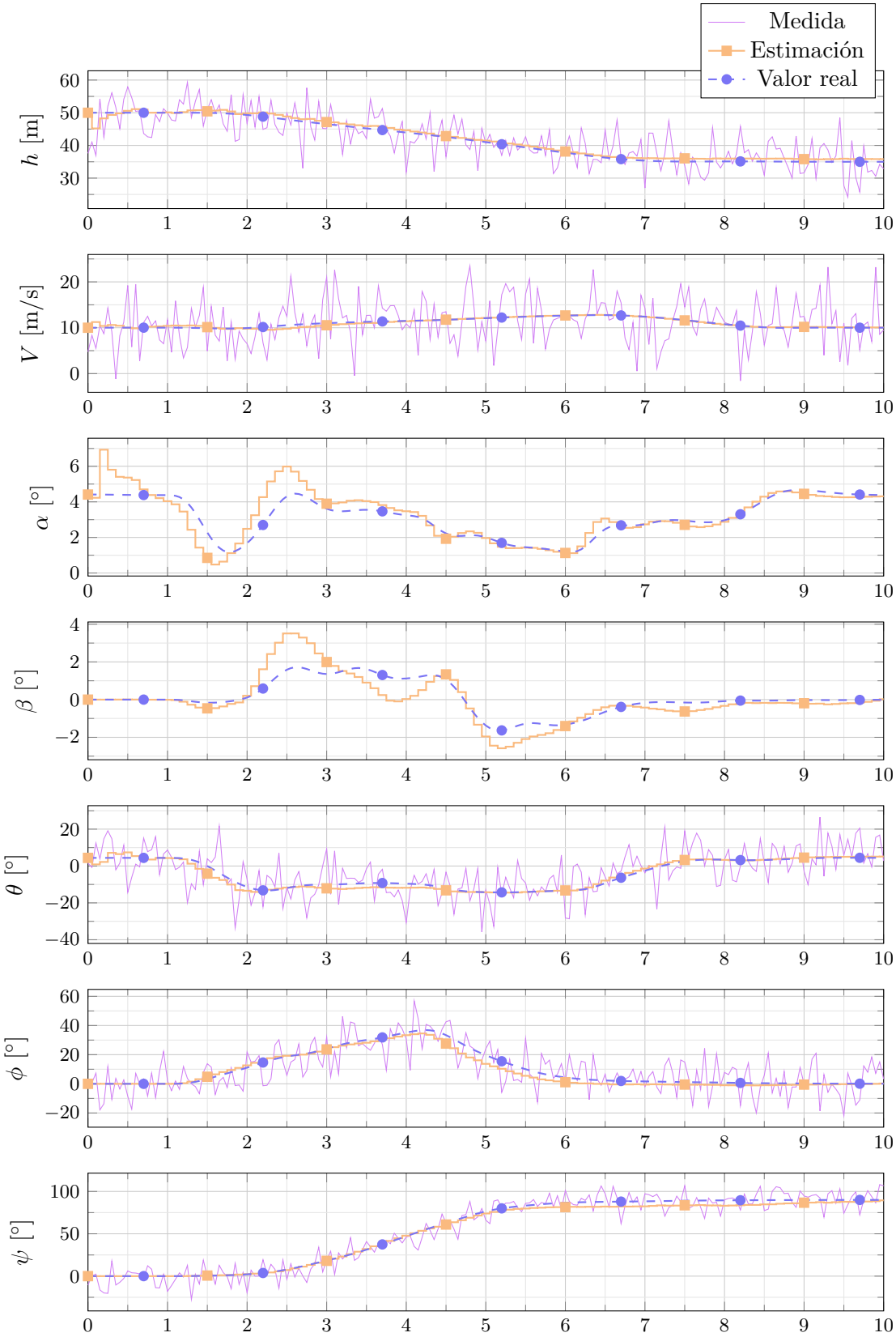


Figura 11.2: Estimaciones del estado del sistema en maniobra combinada.

Esto posibilita crear lazos de control en los que el controlador recibe como realimentación el estado completo del sistema (*full-state feedback*), y permite implementar controladores modernos como el MPC en esta clase de sistemas.

11.3. COMPORTAMIENTO DEL LAZO COMPLETO

Una vez comprobado que el UKF es capaz de estimar el estado correctamente, se puede pasar a estudiar cómo el controlador ha gestionado la maniobra. Este análisis es similar a los del capítulo 10, con la variación de que el controlador realmente no conoce el auténtico estado del sistema, sino una estimación. Las figuras 11.3 y 11.4 muestran los valores auténticos del estado durante la maniobra y las acciones de control aplicadas al avión. En cada figura se dividen, como en el capítulo anterior, en dinámicas longitudinal y lateral-direccional.

En lo referente a la dinámica longitudinal, se trata de un descenso, por lo que el controlador para el motor δ_p con la intención de no embalarse y, como no tiene frenos aerodinámicos, la velocidad aumenta ligeramente.

El seguimiento de la altura h, cuya referencia se ha variado mediante una rampa, es parecido a los realizados en el anterior capítulo, alcanzando la consigna al final de la maniobra sin ningún problema.

La principal diferencia de esta maniobra con las anteriores está en las oscilaciones del ángulo de cabeceo θ, con un sobreimpulso más acentuado y ciertas oscilaciones. Estas son especialmente visibles en el caso del ángulo de ataque α, aunque en ningún momento se ha aproximado al límite superior de 8°.

Pasando a la dinámica lateral direccional, se pueden apreciar unas oscilaciones en el valor de β durante el viraje, en parte debidas a las estimaciones inicialmente erráticas del UKF. No obstante, su valor se ha mantenido cercano a los límites establecidos, según lo previsto.

En cuanto al seguimiento de la referencia de guiñada, este ha sido bueno también, sin sobreimpulso y con un acercamiento muy suave a la referencia final de 90°. Para girar, se ha alabeado hasta casi 40° a la derecha (curva de ϕ en el segundo 4,25) mediante una coordinación de los alerones δ_a y el timón de dirección δ_r.

En general, la maniobra es suave y sigue adecuadamente la referencia, por lo que el piloto automático cumple su cometido y el sistema de control completo es estable en lazo cerrado. Para determinar su robustez, se podrían hacer pruebas más complejas, considerando viento u otras perturbaciones, fallos en los actuadores, etc. Esto se deja como una posible ampliación para el futuro, puesto que el objetivo del proyecto era descubrir, estudiar y probar técnicas de control modernas, cosa que ya se ha conseguido con las simulaciones aquí realizadas.

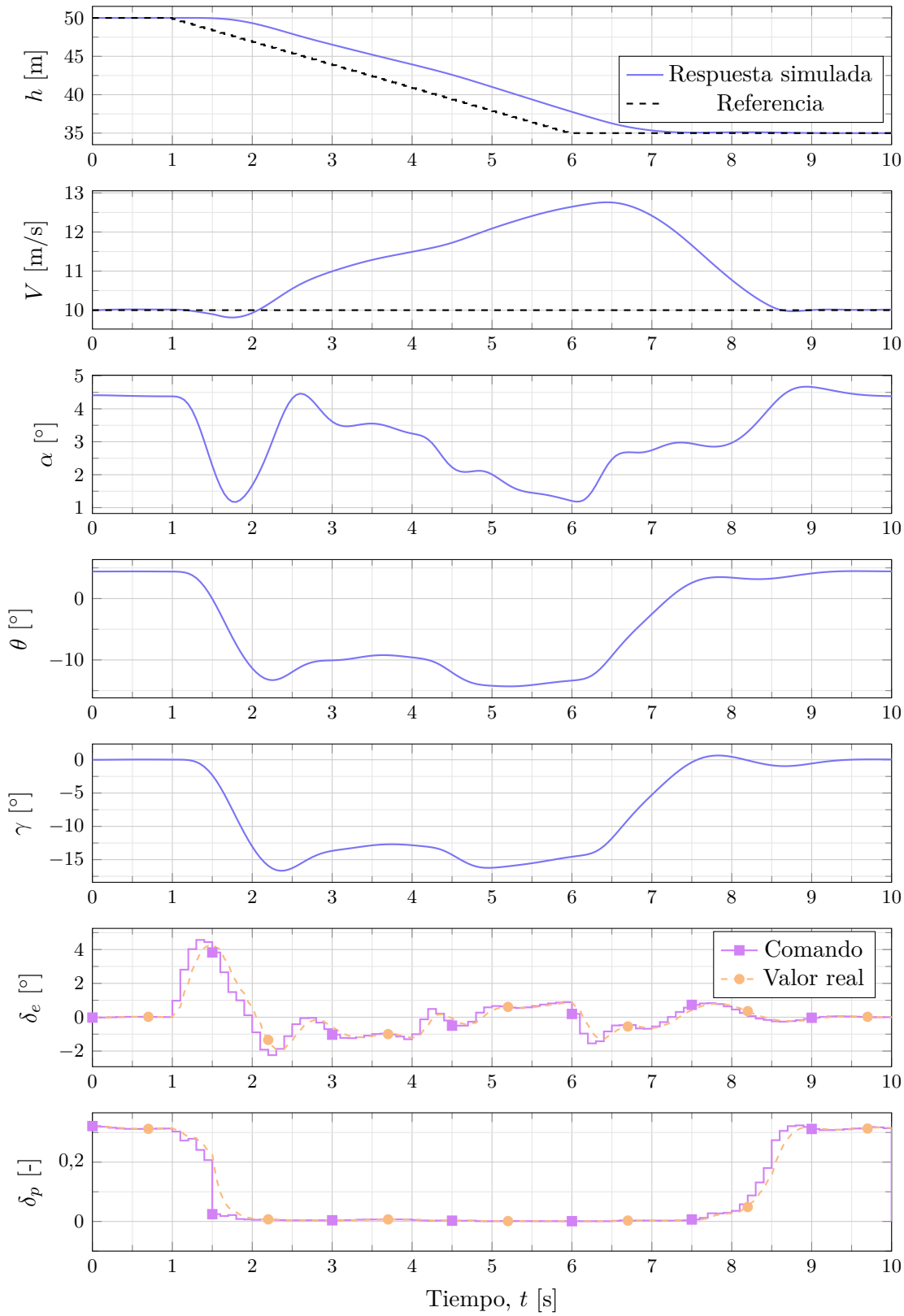


Figura 11.3: Valores reales de las variables de estado y acciones de control durante la maniobra.

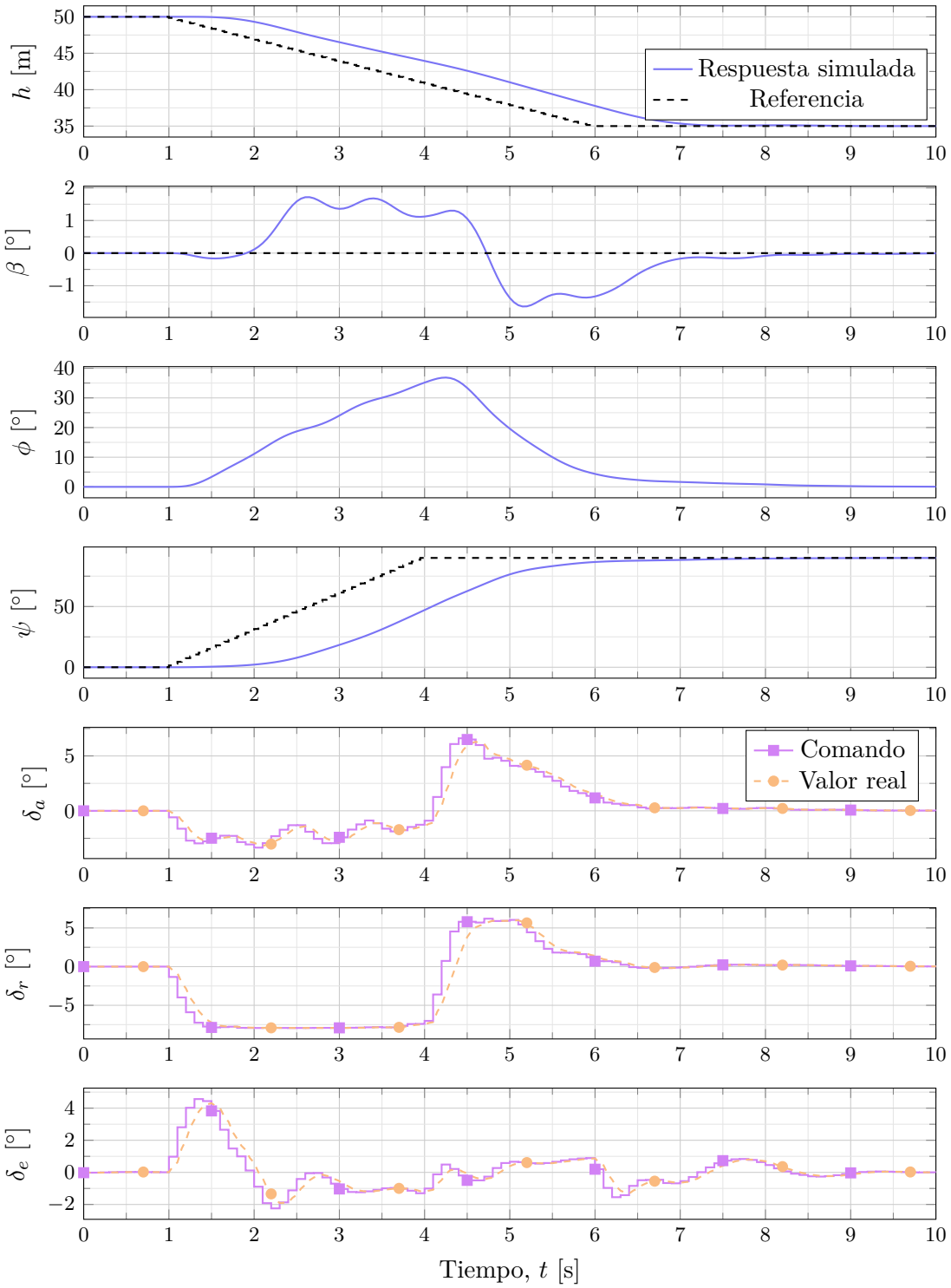


Figura 11.4: Valores reales de las variables de estado y acciones de control durante la maniobra.

12. IDENTIFICACIÓN DE PARÁMETROS

En este capítulo se recogen los resultados, análisis y comentarios de la identificación de parámetros del modelo usando el UKF. Los parámetros identificados son las derivadas aerodinámicas longitudinales, es decir, aquellas que intervienen en el cálculo de C_L, C_D y C_m (véanse las ecuaciones (4.35), (4.36) y (4.39)). Primero, se exponen las señales de entrada usadas y cómo han sido obtenidas para, seguidamente, mostrar los resultados obtenidos y exponer los comentarios pertinentes.

12.1. DISEÑO DE ENTRADAS

Como ya se dijo en el apartado 8.5, la identificación de parámetros es un proceso complejo que requiere de un estudio previo del sistema y una preparación de las entradas adecuadas. Por ello, consultando en la literatura sobre entradas adecuadas para identificación, se llega al artículo [179]. En esta obra se propone una técnica para generar de forma sistemática entradas adecuadas para identificación, cuya efectividad se demuestra experimentalmente en el anterior artículo y también en [180].

El fundamento del método son las señales multiseno (*multisine*, suma de senos), tal que sean ortogonales en los dominios del tiempo y de la frecuencia. La idea es introducir al sistema estas señales con la intención de que produzcan una respuesta que sea una especie de suma de respuestas a las distintas ondas de seno, de manera que en el mismo tiempo se obtiene una respuesta más *rica* en el dominio de la frecuencia y más variada en el dominio del tiempo.

Esta respuesta debería ser más efectiva para identificar los parámetros que las entradas secuenciales tradicionales, como los barridos en frecuencia, dobletes o entradas 3-2-1-1. No obstante, las entradas 3-2-1-1 se han aplicado con éxito en numerosas publicaciones [30, 111, 181] y han sido usadas por varios autores en los últimos años, incluido Morelli [127], el autor que ahora recomienda la entrada aquí usada.

La propuesta del citado artículo es generar unas entradas de la forma

$$\mathbf{u}_i = \sum_{n\in\{1,2,\ldots,M\}} A\sqrt{P_n}\sin\left(\frac{2\pi n\mathbf{t}}{T}+\phi_n\right) \quad \text{con} \quad i\in\{1,2,\ldots,n_u\} \tag{12.1}$$

donde M es el número total de armónicos que componen la entrada, T es el tiempo total de excitación (duración de la entrada), A es la amplitud de la onda multiseno, P_n es la fracción de potencia del n-ésimo armónico, $\mathbf{t}$ es el vector de tiempo, ϕ_n es el desfase de cada armónico y n_u es el número de entradas.

Las frecuencias angulares de estos armónicos serán

$$\omega_n = \frac{2\pi n}{T} \qquad n = 1, 2, \ldots, M \tag{12.2}$$

de tal forma que la máxima frecuencia será $\omega_M = 2\pi M/T$. El intervalo $[\omega_1, \omega_M]$ es el rango de frecuencias de las sinusoides que compondrán estas entradas.

El concepto de ortogonalidad de las ondas de seno se puede simplificar a que son señales multivariable (cada variable es una de las entradas del sistema), de tal forma que sus espectros en frecuencia no se solapan, lo que significa que las respuestas debidas a estas entradas tampoco se solapan, lo que las hace distinguibles y separables en el proceso de identificación, ahorrando tiempo. En [179] se realizan las demostraciones matemáticas pertinentes para justificar este fenómeno.

La propiedad de ortogonalidad es particularmente útil para la identificación simultánea de parámetros longitudinales y lateral-direccionales, ya que en ese caso es necesario introducir perturbaciones en los alerones, timón de profundidad y el timón de dirección, que podría excitar respuestas acopladas que complicarían el proceso. Mediante las entradas comentadas, estos problemas no se dan. No obstante, esto tampoco resulta un problema para el trabajo, ya que solo se han identificado los parámetros longitudinales.

Los desfases ϕ_n de la ecuación (12.1) pueden tomar cualquier valor, pero en [179] se propone optimizarlos para reducir el factor relativo de pico (*relative peak factor*, RPF) de la señal multiseno resultante. Las ondas de seno puras tienen un RPF de 1 y un valor bajo (cercano a 1) indica pocas perturbaciones alrededor del punto de equilibrio del avión (para no perturbarlo demasiado). Como se dice en el citado artículo, este indicador no afecta a la ortogonalidad ni al espectro en frecuencia. El RPF viene dado por la expresión

$$\mathrm{RPF}(\mathbf{u}) = \frac{\mathrm{máx}(\mathbf{u}) - \mathrm{mín}(\mathbf{u})}{2\sqrt{2}\,\mathrm{rms}(\mathbf{u})}\,, \tag{12.3}$$

donde el operador rms denota el valor eficaz. Para minimizar esta métrica, se debe resolver un problema de optimización, que se puede describir como

$$\begin{aligned} &\underset{\boldsymbol{\phi}}{\text{minimizar}} && \mathrm{RPF}(\mathbf{u}) \\ &\text{sujeto a} && -\pi \leq \phi_n \leq \pi, \quad \forall\, n \in \{1, 2, \ldots, M\} \end{aligned}$$

siendo $\boldsymbol{\phi}$ un vector que contiene las fases de las M componentes de la onda multiseno. Este es un problema de programación no lineal, no convexo que puede resolverse en MATLAB por algoritmos genéticos.

Una vez optimizadas las fases de cada componente, es muy probable que la señal obtenida no tome un valor nulo en el instante inicial. Para solucionar este problema, sabiendo que la señal es periódica y de período T, se pueden desplazar todas las senoides que componen la señal por τ segundos, siendo este el tiempo desde el instante inicial hasta el primer cruce por 0 de la entrada óptima calculada. Con este desfase temporal, se puede calcular el desfase a aplicar a cada componente mediante la expresión

$$\Delta\phi_n = \tau\omega_n\,. \tag{12.4}$$

Este desfase debe añadirse a los desfases obtenidos mediante el anterior problema de optimización. De esta manera, la entrada empieza en un valor nulo y puede sumarse a las acciones de control de un regulador sin perturbar demasiado la aeronave durante el vuelo.

Para computar los valores numéricos en cada instante de estas entradas en MATLAB se puede recurrir a varios paquetes o bibliotecas existentes. Por un lado, existe el conjunto de herramientas SIDPAC de la NASA [182, 183], de difícil obtención, que incluye unos 400 programas [184] de MATLAB para realizar distintas tareas relacionadas con la identificación mediante diversos

algoritmos. Como no se pudo disponer de ella, hubo que recurrir a otras alternativas o programar manualmente las mismas utilidades incluidas en este paquete de *software*.

MATLAB cuenta con comandos como `idinput` [185], de la *Toolbox* de identificación [113], que permite generar entradas de varios tipos adecuadas para identificación. Una de estas son señales multiseno como las comentadas anteriormente, que pueden crearse dada su amplitud, su contenido en frecuencia y el número de muestras mediante un solo comando. Esta herramienta gestiona de forma aleatoria los desfases entre sinusoides y reparte las frecuencias entre los canales que se desee. El inconveniente es que no optimiza los desfases para minimizar el RPF.

También existe una *Toolbox* libre de MATLAB llamada MUMI, disponible en GitHub [186], que permite crear estas entradas con una interfaz gráfica muy fácil de entender y usar, que permitió familiarizarse con ellas. Después de probarla intensamente y terminar comprendiendo mejor cómo funcionan las entradas propuestas, esta herramienta terminó descartándose en favor de una utilidad programada a medida para los propósitos de este trabajo, cuyo código se incluye en el listado A.4. No obstante, la facilidad de uso de esta herramienta frente a otras como el comando `idinput` facilitó mucho el entendimiento de estas entradas ortogonales y la hace muy aconsejable.

Como los parámetros que se quieren identificar son los longitudinales, las entradas más relevantes son el timón de profundidad y la palanca de potencia. Esta última no tiene demasiada influencia en la dinámica, por lo que se mantendrá fija.

Con este planteamiento, se linealiza el modelo del «Favara» alrededor de un punto para obtener sus frecuencias naturales. Las frecuencias mínima y máxima marcarán los límites del espectro en frecuencia de la entrada. Al linealizar el modelo en el punto de trimado, calculando el jacobiano del sistema $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, se obtiene una matriz cuadrada cuyos autovalores representan la dinámica del sistema. Realmente, estos valores propios son los *polos* del sistema [5, 8]. Para el caso de las aeronaves, estos polos suelen estar situados en determinados lugares del plano complejo y tienen nombres específicos.

Considérese un polo dado por un número complejo de la forma

$$p = \sigma \pm j\omega_d \,,$$

donde $j = \sqrt{-1}$ es la unidad imaginaria. Este polo tiene una frecuencia natural asociada que, en radianes por segundo, viene determinada por su módulo [5]

$$\omega_n = |p| = \sqrt{\sigma^2 + \omega_d^2} \,.$$

Para pasarla a Hz, basta con dividirla entre 2π. Además, un polo tiene un *factor de amortiguamiento* ζ que se puede calcular mediante la expresión [5]

$$\zeta = -\frac{\sigma}{\sqrt{\sigma^2 + \omega_d^2}} \,. \tag{12.5}$$

Las frecuencias naturales y los factores de amortiguamiento se pueden obtener directamente en MATLAB mediante el comando `damp` [187].

La tabla 12.1 recoge los valores numéricos de los polos del XTRA23 «Favara», con sus nombres, frecuencias naturales para aquellos que oscilan (tienen parte imaginaria no nula) y factor de amortiguamiento. Los modos *corto período* y *fugoide* o *largo período* corresponden con

la dinámica longitudinal, mientras que el *alabeo del holandés*, el *modo espiral* y la *convergencia en balance* corresponden a la dinámica lateral-direccional [3]. Además de estos, la matriz jacobiana tiene otros autovalores, pero estos están en el origen o tan próximos a este que se consideran integradores y no se han tenido en cuenta, como se hace en [4].

Tabla 12.1: Modos naturales del XTRA23 «Favara».

Nombre del modo	Valor del polo	Factor de amortiguamiento	Frecuencia natural [Hz]
Corto período	$-5{,}2793 \pm j1{,}2904$	0,9714	0,8650
Largo período o fugoide	$-0{,}1196 \pm j0{,}7665$	0,1542	0,1235
Convergencia en balance	$-11{,}9397$	1	-
Alabeo del holandés	$-0{,}7512 \pm j2{,}7114$	0,2670	0,4478
Modo espiral	0,1622	-1	-

Una forma habitual de representar estos polos es el llamado *lugar de las raíces*, que muestra los valores de estos puntos en el plano complejo. La figura 12.1 muestra esta clase de gráfica, con los polos divididos entre dinámica longitudinal y lateral-direccional.

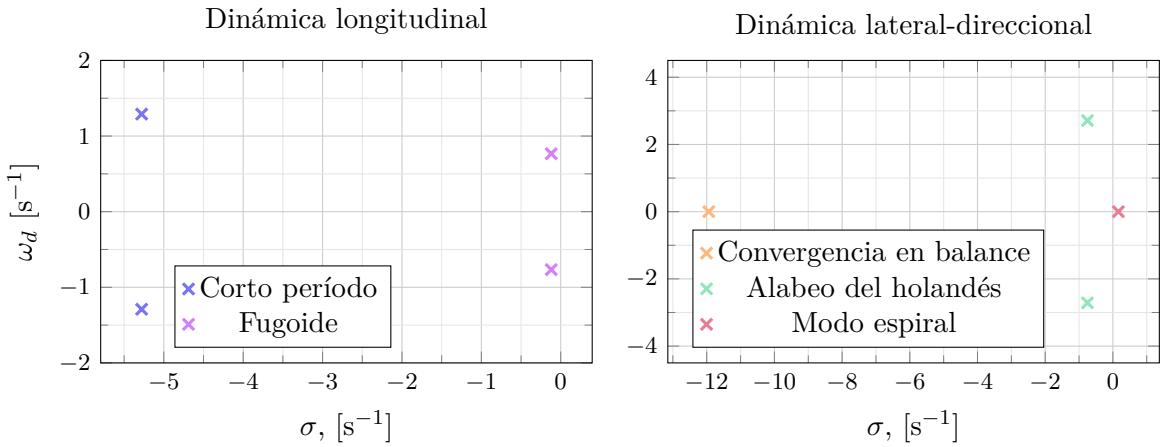


Figura 12.1: Polos del XTRA23 «Favara» linealizado alrededor de su punto de trimado.

Como ya se ha dicho, la parte relevante es la dinámica longitudinal, por lo que las frecuencias naturales a excitar en la maniobra son las del corto período y del fugoide. El corto período tiene una frecuencia natural de casi 1 Hz, pero tiene un factor de amortiguamiento muy cercano a 1, indicando que es muy amortiguado, lo que dificulta que se manifieste en las respuestas y lo convierte en prácticamente un polo real doble.

12.1.1. Entrada diseñada

Utilizando la ecuación (12.1), con un tiempo total T de 60 segundos y un espectro comprendido entre las frecuencias del fugoide y del corto período, se obtienen los valores de las entradas.

Mediante el problema de optimización expuesto en el apartado 8.5, se ajusta la amplitud de la entrada resultante. Esta viene limitada, a su vez, por el ángulo de ataque máximo admisible, que se fijó en 8° en el capítulo 10 (véase tabla 10.2) y aquí también se ha considerado.

De esta manera, la entrada óptima obtenida se muestra en la figura 12.2. Esta tiene el espectro en frecuencia de la figura 12.3, cuyas frecuencias están alrededor de la frecuencia del modo fugoide. Después de varias iteraciones, se concluyó que el corto período de este avión está demasiado amortiguado y que excitarlo no contribuye a mejorar las estimaciones, por lo que se decidió eliminar del espectro en frecuencia de la entrada.

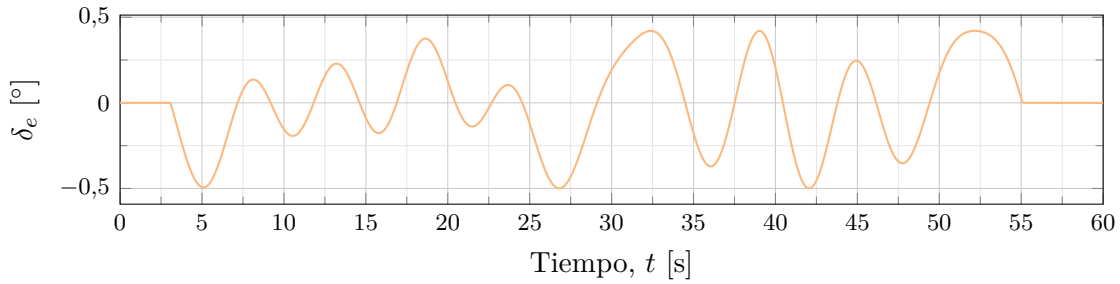


Figura 12.2: Entrada óptima aplicada al XTRA23 «Favara» para su identificación.

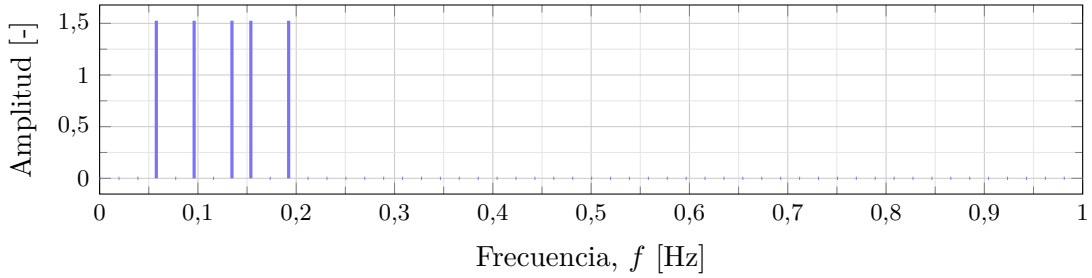


Figura 12.3: Espectro en frecuencia de la entrada óptima generada.

12.2. SIMULACIÓN DE LA MANIOBRA

Al aplicar la entrada diseñada al simulador, se obtiene la respuesta dinámica de la figura 12.4. Esta entrada excita aquellas frecuencias naturales del avión, generando oscilaciones apreciables en la figura que ponen de manifiesto los valores de ciertos parámetros, haciéndolos identificables. Estas oscilaciones están acotadas deliberadamente, de manera que el ángulo de ataque nunca supera el límite de 8°, lo que asegura que el modelo siga siendo válido para identificación. Además, como la única entrada que varía es el timón de profundidad, la dinámica lateral-direccional no se excita y β se mantiene a valor nulo.

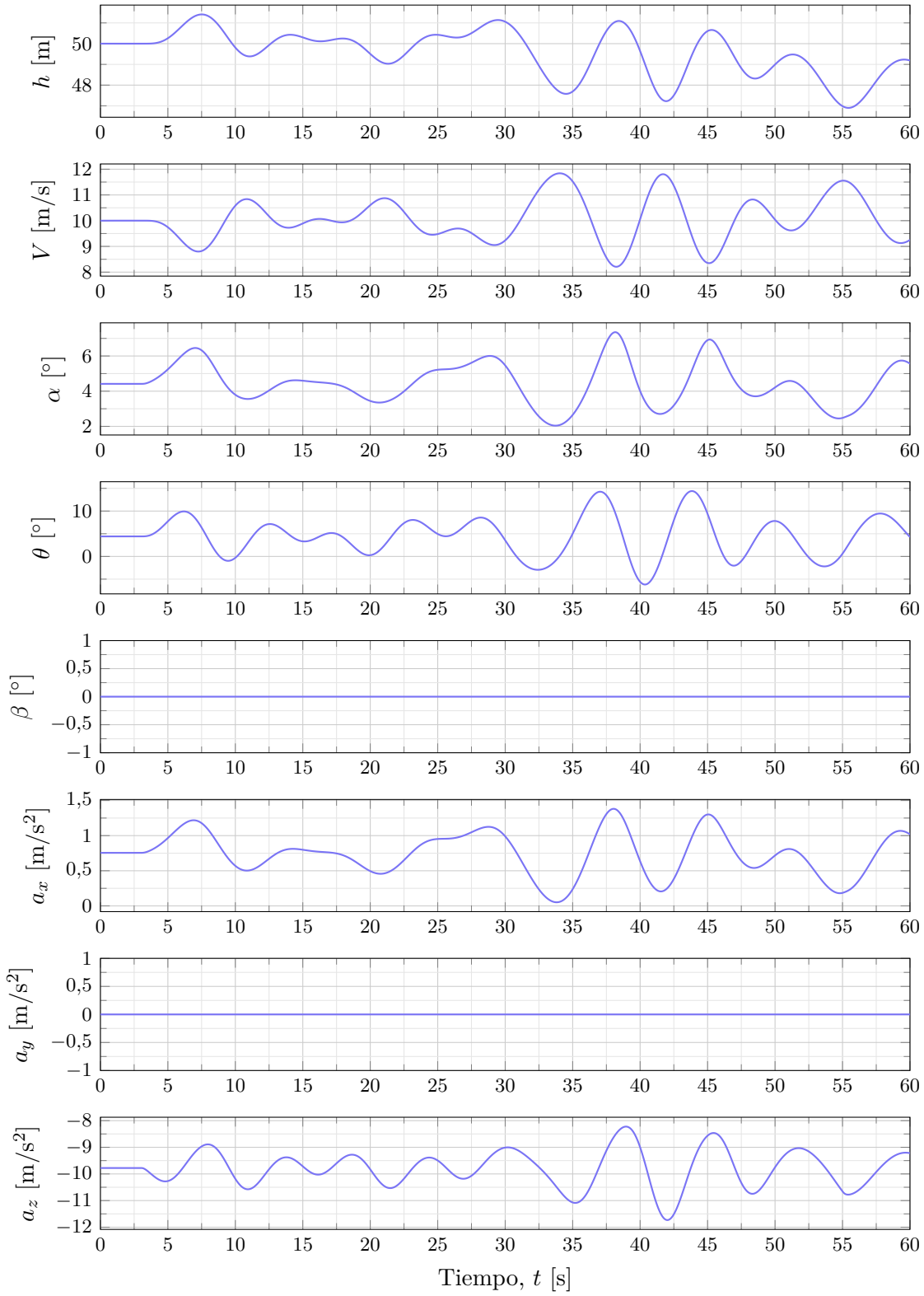


Figura 12.4: Evolución temporal de variables de estado durante la identificación.

12.3. VALORES IDENTIFICADOS

Una vez simulada la respuesta, se introducen los datos en el UKF para la identificación de parámetros. Cabe mencionar en este punto que, desde el punto de vista del filtro, resulta indiferente realizar la identificación después de haber realizado la maniobra a partir de los registros de vuelo (identificación *offline*) que durante la misma mediante la telemetría en tiempo real (identificación *online*), lo que resalta la flexibilidad del algoritmo.

Para realizar una estimación simultánea de estados y parámetros, el UKF trabaja con un vector de estado aumentado, dado por

$$\hat{\mathbf{x}}^a_{k|k} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\boldsymbol{\theta}}_{k|k} \end{bmatrix} , \tag{12.6}$$

donde $\hat{\boldsymbol{\theta}}_{k|k}$ es la estimación instantánea del vector de parámetros $\boldsymbol{\theta}$, que contiene las siguientes derivadas aerodinámicas:

$$\boldsymbol{\theta} = \begin{bmatrix} C_{D0} & C_{D\alpha} & C_{D\alpha^2} & C_{DV} & \\ C_{L0} & C_{L\alpha} & C_{L\dot{\alpha}} & C_{Lq} & C_{LV} \\ C_{m0} & C_{m\alpha} & C_{m\dot{\alpha}} & C_{mq} & C_{m\delta_e} \end{bmatrix}^\intercal \tag{12.7}$$

Los valores numéricos de cada coeficiente se pueden encontrar en las anteriores tablas 4.2 y 4.4 del apartado 4.6.1.

La suposición inicial para la estimación es que el vector de estado es un 20 % mayor que su valor real. Es decir,

$$\hat{\mathbf{x}}^a_0 = 1{,}2 \cdot \begin{bmatrix} \mathbf{x}_0 \\ \boldsymbol{\theta} \end{bmatrix} , \tag{12.8}$$

mientras que la matriz de covarianza del estado inicial es:

$$\mathbf{P}_0 = 0{,}1 \cdot \mathrm{diag}\left((\hat{\mathbf{x}}^a_0)^2 \right) ,$$

donde $(\hat{\mathbf{x}}^a_0)^2$ denota elevar al cuadrado cada componente del vector $\hat{\mathbf{x}}^a_0$, lo que en MATLAB se consigue con la operación `.^2`. Por otra parte, diag(·) denota una matriz diagonal construida con el vector $(\hat{\mathbf{x}}^a_0)^2$.

Nótese cómo en la ecuación (12.8) el vector de parámetros auténtico se denota por $\boldsymbol{\theta}$ y no lleva subíndice k (es constante), mientras que la estimación se representa por $\hat{\boldsymbol{\theta}}_{k|k}$ (como en la ecuación (12.6)) y siempre lleva subíndice.

Como los parámetros son de tamaños muy dispares, el UKF es propenso a dar problemas de estabilidad numérica, llevando a estimaciones pobres de algunos parámetros. Para solucionar este problema, se recurrió a una técnica muy sencilla que consiste en premultiplicar cada parámetro del vector $\boldsymbol{\theta}$ por un escalar y, dentro de las funciones del modelo $\mathbf{f}$ y $\boldsymbol{h}$, dividir cada parámetro por el mismo valor. Esto mejora la robustez de las operaciones matriciales del UKF y ha mejorado los resultados.

Tras realizar la identificación, se calcula la media de las covarianzas y estimaciones en las últimas 400 muestras de la simulación (últimos 20 segundos). Con estos valores, se rellena la tabla 12.2.

La raíz cuadrada de la diagonal de la matriz de covarianzas es la desviación típica σ de cada variable o parámetro. La incertidumbre de las estimaciones se define como el intervalo

Tabla 12.2: Estimaciones de las derivadas longitudinales.

Parám.	Valor real	Estimación	Incert. abs. (2σ)	Incert. rel. [%]	Error rel. [%]
C_{D0}	0,0334	0,0324	0,0029	9,01	2,94
$C_{D\alpha}$	0,0641	0,0683	0,0326	47,72	6,54
$C_{D\alpha^2}$	1,1023	1,1557	0,4034	34,90	4,85
C_{DV}	−0,0228	−0,0148	0,0086	58,11	35,09
C_{L0}	0,3811	0,3613	0,0490	13,57	5,20
$C_{L\alpha}$	5,3057	5,5706	0,6549	11,76	4,99
$C_{L\dot{\alpha}}$	0,9559	0,9718	0,0396	4,07	1,66
C_{Lq}	7,2904	8,3043	3,7881	45,62	13,91
C_{LV}	0,1595	0,1462	0,0608	41,56	8,31
C_{m0}	0,0348	0,0344	0,0030	8,82	1,03
$C_{m\alpha}$	−0,3518	−0,3473	0,0402	11,58	1,28
$C_{m\dot{\alpha}}$	−3,4858	−3,5248	1,0408	29,53	1,12
C_{mq}	−14,2010	−13,6331	0,7893	5,79	4,00
$C_{m\delta_e}$	−1,5047	−1,4576	0,0675	4,63	3,13

de confianza en el que la probabilidad de que el valor real esté incluido sea del 95 %, lo que se consigue con una banda de $\pm 2\sigma$ alrededor del valor estimado (véase [95]). Por ello, la columna con la incertidumbre absoluta contiene este valor para cada parámetro. La incertidumbre relativa se calcula como el cociente de la incertidumbre absoluta y el valor estimado del parámetro y permite cuantificar la precisión del filtro. Por último, el error relativo porcentual de cada parámetro se calcula mediante la expresión

$$e_i = 100 \cdot \frac{\hat{\bar{\theta}}_i - \theta_i}{\theta_i} \,,$$

donde θ_i denota la i-ésima componente del vector de parámetros y $\hat{\bar{\theta}}_i$ denota la media de las últimas estimaciones del parámetro θ_i. Esta métrica permite determinar la exactitud de los valores estimados y encontrar divergencias o convergencias a valores incorrectos.

Examinando la tabla 12.2, se puede comprobar que, en general, las derivadas aerodinámicas C_L y C_m se identifican bastante bien, con errores relativos bajos—convergen a los valores correctos—e incertidumbres relativas moderadas. También se puede comprobar que, para muchos parámetros, su incertidumbre relativa es bastante mayor que su error relativo. Esto indica que, por alguna razón, el filtro no ha sido capaz de reducir la covarianza de la estimación a partir de los datos de entrada, lo que podría indicar una baja sensibilidad del parámetro o una relación cruzada con otros parámetros.

Los parámetros peor identificados son las derivadas C_{DV} y C_{Lq}. Ambas se estiman con una incertidumbre demasiado grande y no convergen al valor correcto. Este fenómeno indeseable se comenta más adelante en esta sección.

Finalmente, en las figuras 12.5 a 12.8 se muestra la evolución temporal de las estimaciones. La figura 12.5 muestra las estimaciones de algunas variables de estado, que obtienen resultados muy similares a los del capítulo 11 y, por tanto, no necesitan muchos más comentarios que los incluidos en el citado capítulo.

En el resto de figuras se muestran las estimaciones del vector de parámetros completo, ordenadas por coeficientes aerodinámicos. La franja sombreada indica la banda de confianza, que se ha representado como $\pm\sigma$ en lugar de la incertidumbre considerada anteriormente ($\pm 2\sigma$) para mejorar la visualización. El valor real de cada derivada aerodinámica se muestra a modo de referencia para resaltar la convergencia.

A la vista de estas gráficas, se puede comprobar que algunos parámetros, como C_{D0}, C_{L0} o C_{mq} convergen relativamente rápido a los valores reales, además de que reducen considerablemente sus covarianzas o bandas de confianza, mientras que otros como C_{Lq} no se acercan al valor correcto ni modifican su banda de confianza, indicando que no son identificables con el experimento realizado. Por otro lado, C_{DV} parece converger a un valor equivocado. Este comportamiento está documentado para el filtro EKF en [188] y se puede atribuir a dos razones.

La primera es un ajuste incorrecto de las matrices de ruido, en particular la de ruido de proceso $\mathbf{Q}$, que puede llevar a estimaciones con grandes covarianzas o a convergencias a valores equivocados (*biased estimates*). El ajuste de $\mathbf{Q}$ es uno de los métodos para sintonizar adecuadamente el filtro y en la práctica siempre se hace, como se indica en la anterior referencia. No obstante, no es la única fuente de error.

La segunda razón documentada en el anterior artículo radica en el propio filtro y en las operaciones que hace. Mediante un desarrollo matemático complejo, en la citada fuente se expone cómo el EKF pueden obtener estimaciones sesgadas (*biased*) para algunos parámetros si se dan ciertas condiciones de ruidos o valores iniciales muy alejados del valor real. La solución propuesta consiste en corregir o modificar algunas ecuaciones del filtro para que este se parezca más a otros algoritmos de estimación de máxima verosimilitud, aunque su aplicación al UKF resulta algo más complicada que la comentada en la citada obra, que está centrada en el EKF.

En cualquier caso, las estimaciones de parámetros son aceptables, especialmente en el caso de las derivadas aerodinámicas estáticas de C_L y C_m. Además, las incertidumbres y errores obtenidos son similares a los que se presentan en [189], donde se identifican parámetros de aeronaves con un EKF.

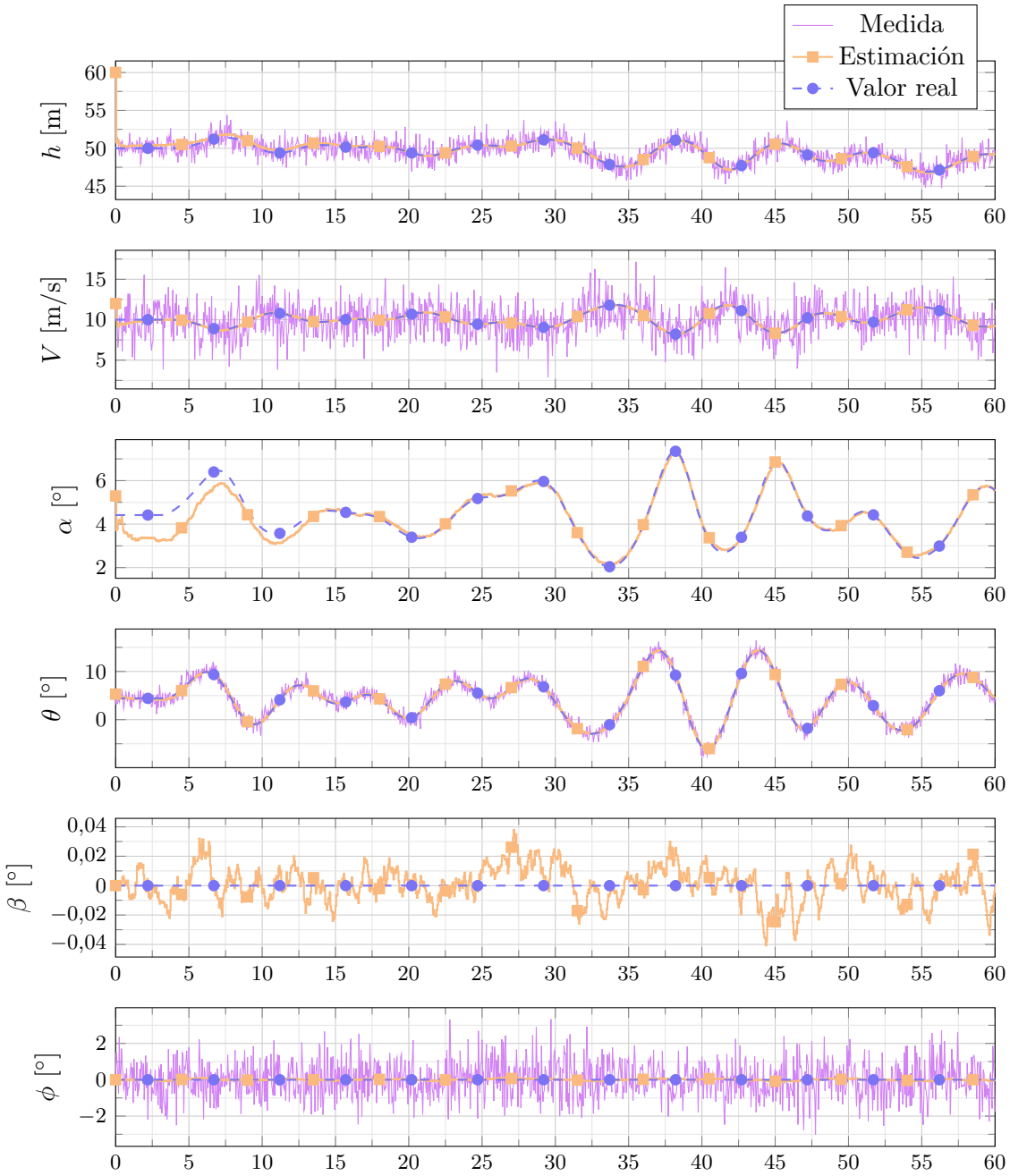


Figura 12.5: Estimaciones de las variables de estado durante la identificación de parámetros.

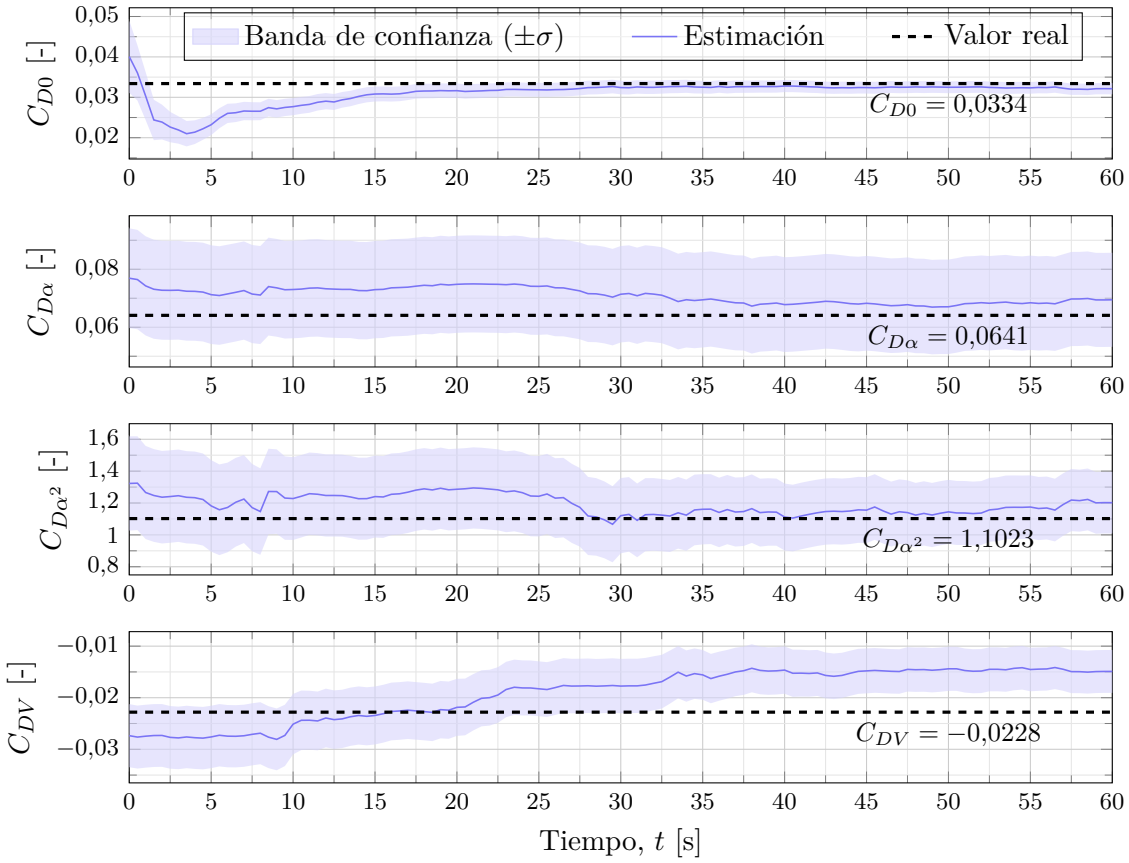


Figura 12.6: Valores estimados de las derivadas aerodinámicas C_D durante la identificación de parámetros longitudinales.

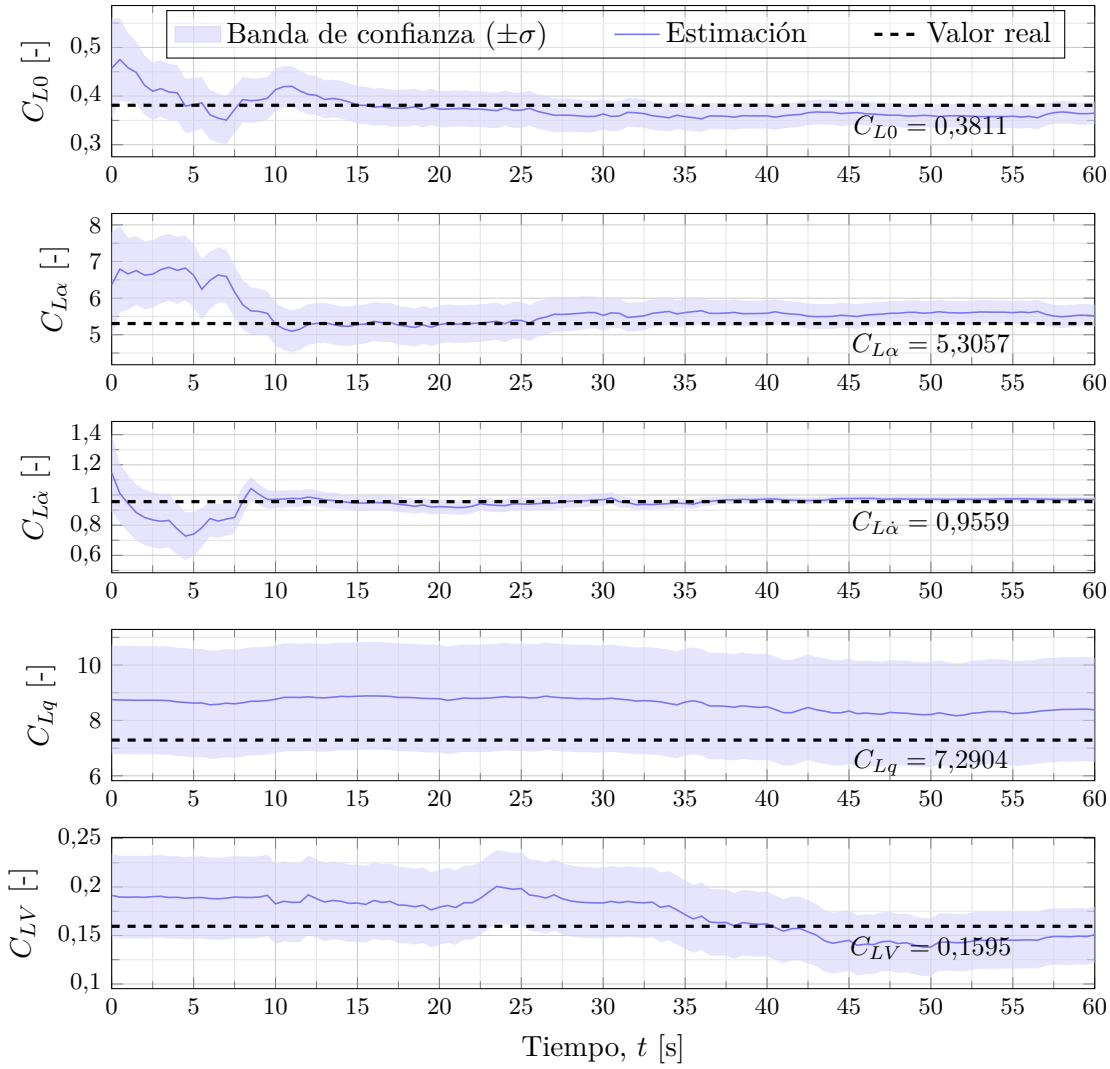


Figura 12.7: Valores estimados de las derivadas aerodinámicas C_L durante la identificación de parámetros longitudinales.

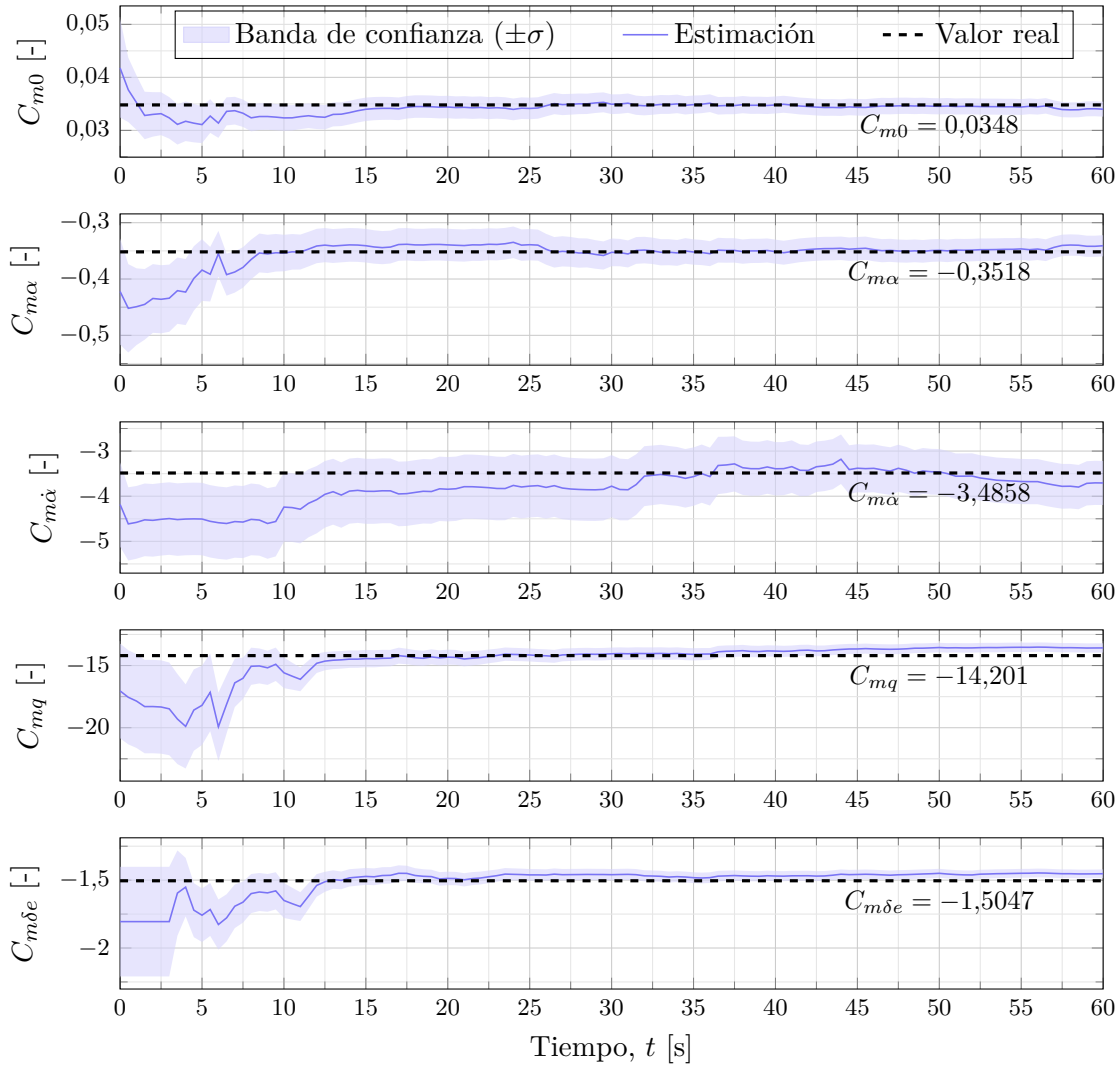


Figura 12.8: Valores estimados de las derivadas aerodinámicas C_m durante la identificación de parámetros longitudinales.

13. CONCLUSIONES

En este TFG se han tratado, de una u otra forma, todos los elementos que componen un sistema de control moderno, no lineal y multivariable. Además de estudiarlos teóricamente, se han diseñado y aplicado con éxito al control de una aeronave no tripulada de ala fija, que se asemeja a un avión en miniatura. Después de exponer su funcionamiento y mostrar su comportamiento en un entorno de simulación, se pasa a recopilar las conclusiones de la labor realizada.

En el trabajo se han desarrollado dos bloques importantes de los sistemas de control multivariable: el observador y el controlador. Si bien se han propuesto algoritmos concretos para estos bloques, cualquiera de ellos podría ser reemplazado por otro sin alterar demasiado el comportamiento del sistema conjunto. Además, para poder probar estos sistemas se ha creado un pequeño entorno de simulación basado en un modelo matemático tomado de la literatura. Por ello, aunque el proyecto trata del lazo de control completo, en realidad engloba dos estudios independientes, cada uno sobre un bloque del lazo de control. De esta forma, se han analizado sus capacidades y resultados en conjunto y por separado.

En su conjunto, el lazo de control funciona bien. Permite un control del avión configurable, dependiendo de aquellas variables que se quieran controlar de manera directa. El lazo de realimentación, compuesto por el estimador de estados UKF, permite reconstruir los valores de aquellas variables para las que no se dispone de sensores y las estimaciones que obtiene son correctas. Por otra parte, el controlador, recibiendo estas estimaciones del estado completo del sistema, es capaz de generar señales de control óptimas que lleven al avión a un nuevo estado igual o muy parecido al indicado por la señal de referencia.

Limitándose al bloque observador, el UKF ha demostrado ser un algoritmo multipropósito de gran utilidad. Además de estimar el estado de la aeronave con gran precisión, como demuestra el capítulo 11, también permite identificar algunos parámetros del modelo en tiempo real mediante los mismos datos que para estimar el estado, como se comprueba en el capítulo 12. Con estos parámetros identificados se podría actualizar el modelo del avión e implementar sistemas de control adaptativo o de detección de fallos, como también se ha visto.

Por la parte del controlador, el MPC permite un control del avión completo mediante una referencia simple, dada por unas 3 variables, mientras se cumplen las restricciones impuestas para no sobrecargar la estructura de la aeronave o que entre en pérdida. La maniobras generadas por el MPC, más allá de ser rápidas, son coordinadas y suaves, puesto que esta técnica de control trabaja con el sistema completo y es capaz de utilizar de forma simultánea todos los mandos del avión.

No obstante, no todo son éxitos. En lo referente a la identificación, no se ha podido estimar el valor de todos los parámetros. Esto se debe a su baja sensibilidad o, lo que es lo mismo, a la pequeña influencia que tienen sobre la respuesta de la aeronave para las entradas aplicadas durante el experimento. Sería necesario estudiar otras entradas, quizás con otro espectro en frecuencia, para tratar de mejorar estos resultados.

Sobre el controlador, a pesar de que ha funcionado bastante bien, los recursos computacionales que requiere son excesivos. Para un portátil moderno, las simulaciones se ejecutaban entre 5 y 10 veces más rápido que en tiempo real. Esta relación sugiere que su implementación en un sistema

embebido como una Raspberry Pi o una controladora de vuelo comercial no sería posible, ya que estos sistemas tienen mucha menos potencia que un ordenador personal. Afortunadamente, dada la modularidad del sistema de control, se puede reemplazar este controlador por otro más sencillo y seguir usando el UKF con todas sus capacidades. Esto permitiría implementar otro tipo de control adaptativo en las controladoras de vuelo comerciales.

El algoritmo optimizador que es la base del MPC permite resolver de forma sistemática problemas de optimización. En el TFG de José Morcillo [3], de hace ya 4 años, se hablaba de la utilidad que tendría desarrollar un simulador de vuelo para optimizar la trayectoria del avión en la competición ACC. Esto no se ha hecho en este trabajo por todo el tiempo y esfuerzo de investigación que han requerido los bloques desarrollados. No obstante, el MPC es un primer paso en esta dirección. Por ejemplo, con una función de coste no lineal adecuada sería posible optimizar el consumo de energía del motor principal del UAV en un recorrido dado. Esta clase de técnicas se proponen en [190] y se deja como ampliación o continuación futura del proyecto.

Aunque no se ha dado mucha importancia en la memoria a las pruebas experimentales, estas deberían hacerse para validar definitivamente cualquiera de estos algoritmos. De hecho, la mayoría de trabajos desarrollados en el equipo Xtra2 UPV tienen algún tipo de verificación experimental. Esta no fue posible para este trabajo por el cuantioso tiempo que ha llevado la labor investigadora y la gran complejidad de los temas tratados, especialmente del UKF y lo relativo a la identificación de parámetros, que era uno de los objetivos de mayor interés del trabajo. De este modo, quedan pendientes de realizar, por un lado, pruebas experimentales del UKF y del MPC y, por otro, simulaciones más complejas que consideren viento y otras perturbaciones que permitan evaluar la robustez del sistema de control.

Finalmente, la temática del trabajo debe ser comprendida en el contexto de un proyecto de interés personal. El objetivo principal del trabajo, aparte de aquellos expuestos en la introducción, era aprender fuera del aula y de los contenidos previstos en el plan de estudios. Ni el modelo de la aeronave ni las técnicas de control no lineal aquí expuestas forman parte de las asignaturas impartidas durante el grado y, sin embargo, han podido estudiarse principalmente mediante libros, artículos y otras fuentes publicadas en Internet. Además, formar parte de Xtra2 durante un curso entero y asistir a sus sesiones teórico-prácticas contribuyó enormemente al aprendizaje. Por todo ello, considerando que se han cumplido casi todos los objetivos propuestos al inicio de esta memoria y el conocimiento adquirido en el proceso, el resultado del trabajo ha satisfecho las pretensiones del autor.

14. BIBLIOGRAFÍA

[1] Reg Austin. *Unmanned Aircraft Systems: UAVS Design, Development and Deployment*. 1.ª edición. Wiley, 16 de abr. de 2010. ISBN: 978-0-470-05819-0 978-0-470-66479-7. DOI: 10.1002/9780470664797. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470664797 (visitado 26-04-2025) (véanse páginas 14, 28).

[2] *Air Cargo Challenge*. En: *Wikipedia*. Page Version ID: 1257335793. 14 de nov. de 2024. URL: https://en.wikipedia.org/w/index.php?title=Air_Cargo_Challenge&oldid=1257335793 (visitado 27-02-2025) (véase página 15).

[3] José Carlos Morcillo Morcillo. «Análisis, diseño y fabricación de un aeromodelo orientado a la maximización de la carga de pago para la competición Air Cargo Challenge». TFG. Valencia: Universitat Politècnica de València, 18 de oct. de 2021. URL: https://riunet.upv.es/handle/10251/174885 (visitado 30-08-2025) (véanse páginas 16, 25, 48, 63, 68, 126, 137).

[4] Jorge Castillo Torres. «Autopilot Design and Simulation for a Fixed-Wing UAV». TFG. Valencia: Universitat Politècnica de València, 7 de sep. de 2022. URL: https://riunet.upv.es/handle/10251/188949 (visitado 15-03-2024) (véanse páginas 17, 31, 32, 34, 43, 44, 45, 46, 47, 48, 49, 50, 51, 55, 56, 63, 91, 101, 126).

[5] Katsuhiko Ogata. *Modern Control Engineering*. 5.ª edición. Prentice Hall, 2010. 913 páginas. ISBN: 978-0-13-615673-4 (véanse páginas 22, 26, 125).

[6] Lennart Ljung. *System Identification: Theory for the User*. 2.ª edición. Google-Books-ID: fYSrk4wDKPsC. Pearson Education, 29 de dic. de 1998. 875 páginas. ISBN: 978-0-13-244053-0 (véanse páginas 22, 25, 26).

[7] Benjamin C. Kuo y Guillermo Aranda Pérez. *Sistemas de control automático*. 7"ª edición. México: Prentice Hall Hispanoamericana, 1996. ISBN: 968-880-723-0. URL: https://go.exlibris.link/BTH57sq5 (véase página 22).

[8] Ángel Valera Fernández. *Modelado y control en el espacio de estados*. Valencia: Editorial UPV, 2002. ISBN: 978-84-9705-171-2 (véanse páginas 22, 23, 26, 27, 76, 125).

[9] Julián J. Salt Llobregat et al. *Control automático : tiempo continuo y tiempo discreto*. 2.ª edición. Textos académicos universitarios ; 1. Place: Barcelona. Editorial Reverté, 2015. ISBN: 978-84-291-4753-7 (véanse páginas 22, 23).

[10] Vladislav Klein y Eugene A. Morelli. *Aircraft system identification: theory and practice*. 1.ª edición. AIAA education series. OCLC: ocm68263458. Reston, VA: American Institute of Aeronautics y Astronautics, 2006. 484 páginas. ISBN: 978-1-56347-832-1 (véanse páginas 23, 25, 26, 29, 30, 31, 32, 34, 35, 37, 38, 39, 40, 41, 42, 43, 61, 67, 70, 87, 117).

[11] *Create unscented Kalman filter object for online state estimation - MATLAB - MathWorks España*. URL: https://es.mathworks.com/help/ident/ref/unscentedkalmanfilter.html?s_tid=doc_ta (visitado 05-04-2024) (véanse páginas 23, 77, 81).

[12] Max Schwenzer et al. «Review on model predictive control: an engineering perspective». En: *The International Journal of Advanced Manufacturing Technology* 117.5 (nov. de 2021), páginas 1327-1349. ISSN: 0268-3768, 1433-3015. DOI: 10.1007/s00170-021-07682-3. URL: https://link.springer.com/10.1007/s00170-021-07682-3 (visitado 27-02-2025) (véanse páginas 24, 91, 94).

[13] R. Matousek e I. Svarc. «Simple methods for stability analysis of nonlinear control systems». En: International Conference on Systems. 22 de jul. de 2009. URL: https://www.semanticscholar.org/paper/Simple-methods-for-stability-analysis-of-nonlinear-Matousek-Svarc/31cef31b0f578eef1808b25a2bd8f51e0089ed30 (visitado 13-01-2025) (véanse páginas 24, 25).

[14] Guanrong Chen. «Stability of Nonlinear Systems». En: *Encyclopedia of RF and Microwave Engineering*. New York, NY: Wiley, 15 de abr. de 2005. ISBN: 978-0-471-65450-6. URL: https://www.researchgate.net/publication/227980165_Stability_of_Nonlinear_Systems (véanse páginas 24, 25).

[15] M. V. Cook. *Flight dynamics principles: a linear systems approach to aircraft stability and control*. 3.ª edición. Elsevier aerospace engineering series. Waltham, MA: Butterworth-Heinemann, 2013. 575 páginas. ISBN: 978-0-08-098242-7 (véanse páginas 24, 30, 32, 34, 35, 36, 37, 39).

[16] Brian L. Stevens, Frank L. Lewis y Eric N. Johnson. *Aircraft Control and Simulation*. 3.ª edición. Hoboken, N. J.: Wiley, 2016. ISBN: 978-1-118-87098-3 (véanse páginas 24, 34, 50, 56, 58, 59, 63, 73).

[17] Anastasios P. Kovanis, Vangelis Skaperdas y John A. Ekaterinaris. «Design and analysis of a light cargo UAV prototype». En: *Journal of Aerospace Engineering* 25.2 (1 de abr. de 2012), páginas 228-237. ISSN: 1943-5525. DOI: 10.1061/(ASCE)AS.1943-5525.0000120. URL: https://www.beta-cae.com/events/c4pdf/2B_4_kovanis.pdf (visitado 01-01-2025) (véase página 25).

[18] Alexandra Almeida Gomes. «Development of an UAV for the Air Cargo Challenge 2017 Competition». TFG. Covilhã: Universidade da Beira Interior, dic. de 2017. URL: https://ubibliorum.ubi.pt/bitstream/10400.6/7950/1/5859_12391.pdf (visitado 01-01-2025) (véase página 25).

[19] Christian Cruz-Sanchez et al. *Aircraft Design: Final report. Team 2*. 24 de abr. de 2015. URL: https://github.com/will214/AircraftDesign/blob/master/FINAL_REPORT_AER4855_TEAM2.pdf (visitado 21-12-2024) (véase página 25).

[20] V. Klein. «Parameter identification applied to aircraft». Tesis doctoral. Cranfield College of Aeronautics, oct. de 1973. URL: http://dspace.lib.cranfield.ac.uk/handle/1826/10513 (visitado 17-03-2025) (véase página 26).

[21] Ajoy Kumar Kundu, Mark A. Price y David Riordan. *Theory and Practice of Aircraft Performance*. 1.ª edición. Wiley, 2016. ISBN: 978-1-119-07417-5 (véanse páginas 28, 32, 62).

[22] Antonio Filippone. *Advanced Aircraft Flight Performance*. 1.ª edición. Cambridge aerospace series. Cambridge University Press, 17 de dic. de 2012. ISBN: 978-1-107-02400-7. DOI: 10.1017/CBO9781139161893. URL: https://www.cambridge.org/core/product/identifier/9781139161893/type/book (visitado 10-09-2024) (véanse páginas 28, 30, 32).

[23] Harry Smith. *Assumed knowledge*. Aircraft Flight Mechanics. URL: https://www.aircraftflightmechanics.com/Prerequisites/Prerequisites.html (visitado 27-09-2024) (véase página 28).

[24] Randal W. Beard y Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. 2.ª edición. Princeton, NJ: Princeton University Press, 26 de feb. de 2012. ISBN: 978-0-691-14921-9. URL: https://drive.google.com/file/d/15RGFf29_lMZ_I2HknxMJvUds78fFdX-/view?usp=sharing (visitado 23-02-2025) (véanse páginas 31, 34, 40, 57, 58, 59, 60, 91).

[25] Robert F. Stengel. *Flight dynamics*. Princeton, NJ: Princeton University Press, 2004. 845 páginas. ISBN: 978-0-691-11407-1 (véanse páginas 31, 32, 40, 41, 42).

[26] Warren F. Phillips. *Mechanics of flight*. 1.ª edición. Hoboken, N. J.: Wiley, 2004. 966 páginas. ISBN: 978-0-471-33458-3 (véanse páginas 32, 34, 36, 38, 39).

[27] Bernard Etkin y Lloyd Duff Reid. *Dynamics of Flight: Stability and Control*. 3.ª edición. Volumen 12. John Wiley & Sons, 1996. ISBN: 0-471-03418-5. URL: https://pubs.aip.org/physicstoday/article/12/9/54/420431/Dynamics-of-Flight-Stability-and-Control (visitado 06-04-2025) (véanse páginas 32, 34).

[28] Vladimir Dobrokhodov. «14. Kinematics and Dynamics of Fixed-Wing UAVs». En: *Handbook of Unmanned Aerial Vehicles*. Editado por Kimon P. Valavanis y George J. Vachtsevanos. Dordrecht: Springer Netherlands, 2015, páginas 243-277. ISBN: 978-90-481-9707-1. DOI: 10.1007/978-90-481-9707-1. URL: https://link.springer.com/10.1007/978-90-481-9707-1 (visitado 02-08-2024) (véase página 34).

[29] Girish Chowdhary y Ravindra Jategaonkar. «Aerodynamic parameter estimation from flight data applying extended and unscented Kalman filter». En: *Aerospace Science and Technology* 14.2 (27 de oct. de 2009), páginas 106-117. ISSN: 1270-9638. DOI: 10.1016/j.ast.2009.10.003. URL: https://www.sciencedirect.com/science/article/pii/S1270963809000650 (visitado 16-07-2024) (véanse páginas 42, 80).

[30] Caterina Grillo y Fernando Montano. «An Algorithm for Parameter Identification of UAS from Flight Data». En: *Journal of Mechanics Engineering and Automation* (25 de oct. de 2014). URL: https://www.researchgate.net/publication/282130885_An_Algorithm_for_Parameter_Identification_of_UAS_from_Flight_Data (véanse páginas 42, 43, 123).

[31] M. Majeed e Indra Narayan Kar. «Aerodynamic parameter estimation using adaptive unscented Kalman filter». En: *Aircraft Engineering and Aerospace Technology* 85.4 (28 de jun. de 2013), páginas 267-279. ISSN: 0002-2667. DOI: 10.1108/AEAT-Mar-2011-0038. URL: https://www.emerald.com/insight/content/doi/10.1108/AEAT-Mar-2011-0038/full/html (visitado 21-07-2024) (véanse páginas 42, 43, 77, 80).

[32] Lucía Villanueva Chulvi. «Development of an academic flight simulator for the study of the flight dynamics of conventional configuration aircraft». TFG. Universitat Politècnica de València, 19 de oct. de 2023. URL: https://riunet.upv.es/handle/10251/198356 (visitado 02-01-2025) (véanse páginas 43, 72, 73).

[33] Or Dantsker et al. «Performance Testing of Aero-Naut CAM Folding Propellers». En: *AIAA AVIATION 2020 FORUM*. AIAA AVIATION 2020 FORUM. VIRTUAL EVENT: American Institute of Aeronautics y Astronautics, 15 de jun. de 2020. ISBN: 978-1-62410-598-2. DOI: 10.2514/6.2020-2762. URL: https://arc.aiaa.org/doi/10.2514/6.2020-2762 (visitado 21-04-2025) (véanse páginas 44, 57, 58).

[34] T-motor. *T-motor X-Carbon 13x6.5"Propeller*. URL: https://www.rc-innovations.com/web/image/product.product/16796/image_1024/%5BT1365%5D%20Tmotor%20X-Carbon%2013x6.5%22%20Propeller?unique=2423b2b (visitado 18-04-2025) (véase página 44).

[35] *xflr5*. URL: http://www.xflr5.tech/xflr5.htm (visitado 23-01-2025) (véase página 47).

[36] André Deperrois. «Xflr5 Part IV- Theoretical limitations and shortcomings». Xflr5, jun. de 2019. URL: http://www.xflr5.tech/docs/Part%20IV:%20Limitations.pdf (visitado 24-01-2025) (véase página 47).

[37] Xtra2 UPV. *Air Cargo Challenge 2022 Technical Report Team 19*. Valencia, 2022. URL: https://akamodell-muenchen.de/wp-content/uploads/2022/05/ACC2022_technical_report_team_19.pdf (visitado 07-08-2024) (véase página 48).

[38] Marc Aragó Cebolla. «Análisis, diseño y fabricación de un flap fowler para un aeromodelo de competición de Xtra2 UPV». TFG. Valencia: Universitat Politècnica de València, jul. de 2023. URL: https://riunet.upv.es/handle/10251/197174 (visitado 28-08-2024) (véase página 48).

[39] MathWorks. *fmincon - Matlab documentation*. URL: https://es.mathworks.com/help/optim/ug/fmincon.html (visitado 22-12-2024) (véase página 51).

[40] David Silla Llopis. «Análisis numérico y optimización aerodinámica de la punta de ala de un UAV de competición». Proyecto/Trabajo fin de carrera/grado. Universitat Politècnica de València, 14 de nov. de 2024. URL: https://riunet.upv.es/handle/10251/211768 (visitado 02-01-2025) (véase página 51).

[41] Adrián Ortega Marzal. «Diseño estructural y validación de un vehículo aéreo no tripulado para el concurso internacional Air Cargo Challenge 2024». Proyecto/Trabajo fin de carrera/grado. Valencia, España: Universitat Politècnica de València, 25 de nov. de 2024. URL: https://riunet.upv.es/handle/10251/212257 (visitado 13-03-2025) (véase página 51).

[42] Xtra2 UPV. *Air Cargo Challenge 2024 Technical Report Team 19*. Valencia, 30 de abr. de 2024, página ix (véase página 52).

[43] KST. *DS589MG V8.0 Mini Digital HV Metal Gear Servo 9.2Kgf.cm 0.08sec for RC Helicopter & Fixed Wing*. URL: https://kstservos.com/products/ds589mg-digital-hv-metal-gear-servo-9-2kg-cm-0-08sec-for-rc-helicopter (visitado 17-05-2025) (véase página 54).

[44] *Servos Explained*. Sparkfun Electronics. URL: https://www.sparkfun.com/servos (visitado 10-01-2025) (véase página 54).

[45] *Digital Vs Analog Servo [A Complete Comparison] - RC Fact*. Section: Batteries. 21 de nov. de 2023. URL: https://rcfact.com/digital-vs-analog-servo/ (visitado 10-01-2025) (véase página 55).

[46] Lars Wanhammar y Yajun Yu. «Digital filter structures and their implementation». En: *Signal Processing and Machine Learning Theory*. Elsevier, 2024, páginas 267-401. ISBN: 978-0-323-91772-8. DOI: 10.1016/B978-0-32-391772-8.00012-0. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780323917728000120 (visitado 05-10-2024) (véase página 55).

[47] Tower Pro. *SG90 Servo Motor Datasheet*. URL: https://components101.com/sites/default/files/component_datasheet/SG90%20Servo%20Motor%20Datasheet.pdf (visitado 07-04-2025) (véase página 55).

[48] Mikiya Hayashi et al. «Adaptive modeling and compliance control for RC servo motor». En: *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE). Sep. de 2017, páginas 664-667. DOI: 10.23919/SICE.2017.8105557. URL: https://ieeexplore.ieee.org/document/8105557/?arnumber=8105557 (visitado 10-03-2025) (véase página 55).

[49] Takashi Wada et al. «Practical modeling and system identification of R/C servo motors». En: *2009 IEEE Control Applications, (CCA) & Intelligent Control, (ISIC)*. 2009 IEEE Control Applications, (CCA) & Intelligent Control, (ISIC). ISSN: 1085-1992. Jul. de 2009, páginas 1378-1383. DOI: 10.1109/CCA.2009.5280987. URL: https://ieeexplore.ieee.org/document/5280987/?arnumber=5280987 (visitado 10-03-2025) (véase página 55).

[50] Yoonkyu Hwang, Yuki Minami y Masato Ishikawa. «Virtual Torque Sensor for Low-Cost RC Servo Motors Based on Dynamic System Identification Utilizing Parametric Constraints». En: *Sensors* 18.11 (nov. de 2018). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, página 3856. ISSN: 1424-8220. DOI: 10.3390/s18113856. URL: https://www.mdpi.com/1424-8220/18/11/3856 (visitado 10-03-2025) (véase página 55).

[51] P. Dondon y C.A. Bulucea. «An Enhanced Spice and 3D Modelling for Standard Servomotors». En: *WSEAS Transactions on Electronics* 12 (2021), páginas 9-18. DOI: 10.37394/232017.2021.12.2. URL: https://www-scopus-com.unican.idm.oclc.org/record/display.uri?eid=2-s2.0-85126087408&origin=resultslist&sort=plf-f&src=s&sot=b&sdt=b&s=TITLE-ABS-KEY%28radio-controlled+servo+motor%29&relpos=2 (visitado 05-04-2025) (véase página 56).

[52] T-motor. *AM66A / AM Link 3D*. URL: https://www.tmotorhobby.com/goods-1140-T-MOTORHOBBY+AM66A++AM+Link+3D.html (visitado 18-04-2025) (véase página 59).

[53] Padmaraja Yedamale. *AN885: Brushless DC (BLDC) Motor Fundamentals*. 2003. URL: https://ww1.microchip.com/downloads/en/appnotes/00885a.pdf (visitado 26-01-2025) (véase página 59).

[54] T-motor. *AT2826 3D F3A Fixed Wing Long Shaft Motor | T-MOTOR®*. URL: https://store.tmotor.com/product/at2826-long-shaft-fixed-wing-motor.html (visitado 24-04-2025) (véase página 60).

[55] Yunus A. Çengel y John M. Cimbala. *Mecánica de fluidos: fundamentos y aplicaciones*. 4.ª edición. Ciudad de México: McGraw-Hill Interamericana, 2018. ISBN: 978-1-4562-6228-0 (véanse páginas 62, 63).

[56] Or D. Dantsker, Saym Imtiaz y Marco Caccamo. «Electric Propulsion System Optimization for Long-Endurance and Solar-Powered Unmanned Aircraft». En: *AIAA Propulsion and Energy 2019 Forum*. AIAA Propulsion and Energy Forum. American Institute of Aeronautics y Astronautics, 16 de ago. de 2019. DOI: 10.2514/6.2019-4486. URL: https://www.researchgate.net/publication/343706468_Propulsion_System_Design_Optimization_Simulation_and_Testing_for_a_Long-Endurance_Solar-Powered_Unmanned_Aircraft (visitado 22-02-2025) (véase página 63).

[57] RCInnovations. *Airspeed sensor kit*. URL: https://rc-innovations.es/en/shop/fa-003-pixhawk-digital-airspeed-sensor-kit-pitot-tube-11596?search=pitot&order=name+asc#attr=18886 (visitado 09-04-2025) (véase página 63).

[58] TE Connectivity. *MS4525DO datasheet*. Oct. de 2013. URL: https://www.mouser.com/datasheet/2/418/MS4525DO-710170.pdf (visitado 08-04-2025) (véase página 63).

[59] L. Sankaralingam y C. Ramprasadh. «A comprehensive survey on the methods of angle of attack measurement and estimation in UAVs». En: *Chinese Journal of Aeronautics* 33.3 (1 de mar. de 2020), páginas 749-770. ISSN: 1000-9361. DOI: 10.1016/j.cja.2019.11.003. URL: https://www.sciencedirect.com/science/article/pii/S1000936119304078 (visitado 28-02-2025) (véase página 66).

[60] Yunxiao Liu et al. «Development of a Low-Cost AOA/AOS Sensor and Data Processing System for Small-Sized UAVs». En: *2024 IEEE 19th Conference on Industrial Electronics and Applications (ICIEA)*. 2024 IEEE 19th Conference on Industrial Electronics and Applications (ICIEA). ISSN: 2158-2297. Ago. de 2024, páginas 1-6. DOI: 10.1109/ICIEA61579.2024.10664757. URL: https://ieeexplore.ieee.org/document/10664757/?arnumber=10664757 (visitado 28-02-2025) (véase página 66).

[61] Kriangkrai Wanngoen et al. «Angle of Attack Sensor for Small Fixed-Wing Unmanned Aerial Vehicles». En: *Proceedings* 39.1 (2020). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, página 19. ISSN: 2504-3900. DOI: 10.3390/proceedings2019039019. URL: https://www.mdpi.com/2504-3900/39/1/19 (visitado 28-02-2025) (véase página 66).

[62] ams OSRAM. *AS5048A/AS5048B Magnetic Rotary Encoder*. URL: https://media.digikey.com/pdf/Data%20Sheets/Austriamicrosystems%20PDFs/AS5048A,B.pdf (visitado 28-02-2025) (véase página 66).

[63] Jean-Philippe Condomines et al. «Experimental Wind Field Estimation and Aircraft Identification». En: IMAV 2015: International Micro Air Vehicles Conference and Flight Competition. Aachen, Germany, sep. de 2015. DOI: hal-01204624. URL: https://enac.hal.science/hal-01204624/document (visitado 09-04-2025) (véase página 66).

[64] Tor A. Johansen et al. «On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors». En: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2015 International Conference on Unmanned Aircraft Systems (ICUAS). Jun. de 2015, páginas 510-519. DOI: 10.1109/ICUAS.2015.7152330. URL: https://ieeexplore.ieee.org/document/7152330/?arnumber=7152330 (visitado 28-02-2025) (véase página 66).

[65] Paul D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. 2.ª edición. Boston, MA, USA: Artech House, 2013. 800 páginas. ISBN: 978-1-60807-005-3. URL: https://ieeexplore.ieee.org/document/9106151 (visitado 21-05-2025) (véanse páginas 67, 68).

[66] Mouser Electronics. *Electronic Components Distributor - Mouser Electronics Europe*. URL: https://eu.mouser.com/ (visitado 21-05-2025) (véanse páginas 67, 68).

[67] TDK InvenSense. *MPU-6000 and MPU-6050 Product Specification Revision 3.4*. 19 de ago. de 2013. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf (visitado 28-02-2025) (véase página 67).

[68] Ardupilot. *Pixhawk Overview*. ArduPilot.org. URL: https://ardupilot.org/copter/docs/common-pixhawk-overview.html (visitado 21-05-2025) (véase página 67).

[69] TDK InvenSense. *ICM-42688-P datasheet*. 27 de jul. de 2023. URL: https://invensense.tdk.com/products/motion-tracking/6-axis/icm-42688-p/ (visitado 18-04-2025) (véase página 67).

[70] Matek Systems. *F405-Wing-V2*. URL: https://www.mateksys.com/?portfolio=f405-wing-v2#tab-id-2 (visitado 18-04-2025) (véanse páginas 67, 68, 93).

[71] TDK InvenSense. *ICM-20689 datasheet*. 14 de mar. de 2018. URL: https://cdn.sparkfun.com/assets/7/5/4/3/1/ICM-20689-v2.2-002.pdf (visitado 21-05-2025) (véase página 67).

[72] Pixhawk. *DS-011 Pixhawk Autopilot v5X Standard*. Versión 0.8.0. 7 de dic. de 2022. URL: https://github.com/pixhawk/Pixhawk-Standards/blob/master/DS-011%20Pixhawk%20Autopilot%20v5X%20Standard.pdf (visitado 18-04-2025) (véase página 67).

[73] PX4. *Holybro Pixhawk 4*. PX4 Guid. URL: https://docs.px4.io/main/en/flight_controller/pixhawk4.html (visitado 21-05-2025) (véase página 67).

[74] TDK InvenSense. *MPU-9250 Product Specification Revision 1.1*. 20 de jun. de 2016. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf (visitado 28-02-2025) (véase página 67).

[75] TDK InvenSense. *ICM-20948 datasheet*. 12 de mar. de 2024. URL: https://invensense.tdk.com/wp-content/uploads/2024/03/DS-000189-ICM-20948-v1.6.pdf (visitado 21-05-2025) (véase página 67).

[76] Holybro. *Pixhawk 6X (ICM-45686)*. Holybro Store. URL: https://holybro.com/products/pixhawk-6x (visitado 21-05-2025) (véanse páginas 67, 68).

[77] David Titterton y John L. Weston. *Strapdown Inertial Navigation Technology*. Google-Books-ID: WwrCrn54n5cC. IET, 2004. 578 páginas. ISBN: 978-0-86341-358-2 (véase página 68).

[78] Oliver J. Woodman. *An introduction to inertial navigation*. UCAM-CL-TR-696. Cambridge, Reino Unido: University of Cambridge, ago. de 2017. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf (visitado 21-05-2025) (véase página 68).

[79] Bosch Sensortec. *BMP180 datasheet*. 7 de mayo de 2015. URL: https://eu.mouser.com/datasheet/2/783/BST-BMP180-DS000-1509579.pdf (visitado 23-05-2025) (véase página 68).

[80] Bosch Sensortec. *BMP388 datasheet*. Mar. de 2018. URL: https://eu.mouser.com/datasheet/2/783/BST_BMP388_DS001-1509617.pdf (visitado 23-05-2025) (véase página 68).

[81] TDK InvenSense. *ICP-10111 datasheet*. 3 de mayo de 2021. URL: https://product.tdk.com/system/files/dam/doc/product/sensor/pressure/capacitive-pressure/data_sheet/ds-000177-icp-10111-v1.3.pdf (visitado 23-05-2025) (véase página 68).

[82] STMicroelectronics. *LPS22HB datasheet*. Jun. de 2017. URL: https://eu.mouser.com/datasheet/2/389/lps22hb-1849683.pdf (visitado 23-05-2025) (véase página 68).

[83] Infineon. *DPS310 datasheet*. 15 de oct. de 2020. URL: https://eu.mouser.com/datasheet/2/196/Infineon_DPS310_DataSheet_v01_02_EN-3361265.pdf (visitado 23-05-2025) (véase página 68).

[84] Mamoun F. Abdel-Hafez. «On the GPS/IMU sensors' noise estimation for enhanced navigation integrity». En: *Mathematics and Computers in Simulation* 86 (dic. de 2012), páginas 101-117. ISSN: 03784754. DOI: 10.1016/j.matcom.2010.03.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0378475410000765 (visitado 25-05-2025) (véase página 69).

[85] Jorge Monteagudo Sáez. «Desarrollo de una interfaz gráfica dinámica para un simulador de vuelo docente en Matlab». TFG. Valencia: Universitat Politècnica de València, 2024. URL: https://riunet.upv.es/handle/10251/206756 (visitado 02-01-2025) (véanse páginas 72, 73).

[86] David Allerton. *Principles of Flight Simulation*. 1.ª edición. John Wiley & Sons, 9 de oct. de 2009. ISBN: 978-0-470-68566-2. URL: https://doi.org/10.1002/9780470685662.ch2 (visitado 01-01-2025) (véanse páginas 72, 73).

[87] MathWorks. *ode113 - Matlab documentation*. URL: https://www.mathworks.com/help/matlab/ref/ode113.html (visitado 02-03-2025) (véase página 72).

[88] MathWorks. *ode89 - Matlab documentation*. URL: https://www.mathworks.com/help/matlab/ref/ode89.html (visitado 02-03-2025) (véase página 72).

[89] Mona Strauss. *monajos/Master_thesis*. original-date: 2023-06-27T10:02:18Z. 5 de mar. de 2024. URL: https://github.com/monajos/Master_thesis (visitado 06-09-2024) (véase página 72).

[90] Jouni Hartikainen, A. Solin y Simo Särkkä. «Optimal Filtering with Kalman Filters and Smoothers». En: 2011. URL: https://www.semanticscholar.org/paper/Optimal-Filtering-with-Kalman-Filters-and-Smoothers-Hartikainen-Solin/676c815ca91da00575d6401b72347421f592adda (visitado 18-02-2025) (véase página 72).

[91] MathWorks. *ode45 - Matlab documentation*. URL: https://www.mathworks.com/help/matlab/ref/ode45.html (visitado 02-03-2025) (véase página 73).

[92] MathWorks. *Aerospace Toolbox*. MathWorks Products. URL: https://es.mathworks.com/products/aerospace-toolbox.html (visitado 02-03-2025) (véase página 74).

[93] E.A. Wan y R. Van Der Merwe. «The unscented Kalman filter for nonlinear estimation». En: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*. Symposium on Adaptive Systems for Signal Processing Communications and Control. Lake Louise, Alta., Canada: IEEE, 2000, páginas 153-158. ISBN: 978-0-7803-5800-3. DOI: 10.1109/ASSPCC.2000.882463. URL: http://ieeexplore.ieee.org/document/882463/ (visitado 12-08-2024) (véanse páginas 76, 77, 79, 80).

[94] Mohinder S. Grewal y Angus P. Andrews. *Kalman filtering: theory and practice using MATLAB*. 3.ª edición. OCLC: ocn191758403. Hoboken, N. J.: Wiley, 2008. 575 páginas. ISBN: 978-0-470-17366-4 (véanse páginas 76, 77, 79).

[95] Alex Becker. *Kalman filter from the ground up*. 1.ª edición. OCLC: 1393909496. Kalman-Filter.net, 2023. ISBN: 978-965-598-439-2 (véanse páginas 77, 78, 81, 130).

[96] Matthew Rhudy y Yu Gu. *Understanding Nonlinear Kalman Filters, Part II: An Implementation Guide*. 2013. URL: https://yugu.faculty.wvu.edu/research/interactive-robotics-letters/understanding-nonlinear-kalman-filters-part-ii (véanse páginas 77, 157).

[97] Binqi Zheng et al. «A Robust Adaptive Unscented Kalman Filter for Nonlinear Estimation with Uncertain Noise Covariance». En: *Sensors* 18.3 (7 de mar. de 2018), página 808. ISSN: 1424-8220. DOI: 10.3390/s18030808. URL: https://www.mdpi.com/1424-8220/18/3/808 (visitado 01-08-2024) (véanse páginas 77, 84, 85, 90).

[98] Ivan Popov et al. «Adaptive Kalman Filtering for Dynamic Positioning of Marine Vessels». En: *IFAC-PapersOnLine* 50.1 (jul. de 2017), páginas 1121-1126. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.394. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896317307413 (visitado 28-07-2024) (véanse páginas 77, 79, 83, 84).

[99] Chingiz Hajiyev. «An Innovation Approach Based Sensor Fault Detection and Isolation». En: *IFAC-PapersOnLine* 49.17 (2016), páginas 420-425. ISSN: 24058963. DOI: 10.1016/j.ifacol.2016.09.072. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896316315452 (visitado 30-05-2025) (véase página 77).

[100] R. E. Kalman. «A New Approach to Linear Filtering and Prediction Problems». En: *Journal of Basic Engineering* 82.1 (1 de mar. de 1960), páginas 35-45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. URL: https://doi.org/10.1115/1.3662552 (visitado 30-03-2025) (véase página 79).

[101] R. Van der Merwe y E.A. Wan. «The square-root unscented Kalman filter for state and parameter-estimation». En: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Volumen 6. ISSN: 1520-6149. Mayo de 2001, 3461-3464 vol.6. DOI: 10.1109/ICASSP.2001.940586. URL: https://ieeexplore.ieee.org/document/940586/?arnumber=940586 (visitado 13-08-2024) (véanse páginas 79, 80, 81).

[102] Baoshuang Ge et al. «Adaptive Unscented Kalman Filter for Target Tracking with Unknown Time-Varying Noise Covariance». En: *Sensors* 19.6 (19 de mar. de 2019), página 1371. ISSN: 1424-8220. DOI: 10.3390/s19061371. URL: https://www.mdpi.com/1424-8220/19/6/1371 (visitado 25-07-2024) (véase página 79).

[103] Juan P. Muñoz, Mario E. Magaña y Eduardo Cotilla-Sanchez. «Adaptive master–slave unscented Kalman filter for grid voltage frequency estimation». En: *IET Signal Processing* 12.4 (jun. de 2018), páginas 496-505. ISSN: 1751-9675, 1751-9683. DOI: 10.1049/iet-spr.2016.0199. URL: https://onlinelibrary.wiley.com/doi/10.1049/iet-spr.2016.0199 (visitado 26-07-2024) (véase página 79).

[104] M. Strauss. «Parameter Identification via Kalman Filter on a Model Motor Glider». En: (5 de mar. de 2025). Publisher: Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., Bonn. DOI: 10.25967/630541. URL: https://publikationen.dglr.de/?tx_dglrpublications_pi1%5Bdocument_id%5D=630541 (visitado 21-03-2025) (véase página 79).

[105] Natássya B. F. da Silva, Daniel B. Wilson y Kalinka R. L. J. Branco. «Performance evaluation of the Extended Kalman Filter and Unscented Kalman Filter». En: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2015 International Conference on Unmanned Aircraft Systems (ICUAS). Jun. de 2015, páginas 733-741. DOI: 10.1109/ICUAS.2015.7152356. URL: https://ieeexplore.ieee.org/document/7152356/?arnumber=7152356 (visitado 20-03-2025) (véase página 79).

[106] K. Reif et al. «Stochastic stability of the discrete-time extended Kalman filter». En: *IEEE Transactions on Automatic Control* 44.4 (abr. de 1999), páginas 714-728. ISSN: 1558-2523. DOI: 10.1109/9.754809. URL: https://ieeexplore.ieee.org/document/754809/ (visitado 03-02-2025) (véase página 79).

[107] Simon J. Julier y Jeffrey K. Uhlmann. «A New Extension of the Kalman Filter to Nonlinear Systems». En: *Signal Processing, Sensor Fusion, and Target Recognition VI*. Volumen 3068. SPIE, 28 de jul. de 1997, páginas 182-193. DOI: 10.1117/12.280797. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/3068/0000/New-extension-of-the-Kalman-filter-to-nonlinear-systems/10.1117/12.280797.full (visitado 28-03-2025) (véase página 79).

[108] Ienkaran Arasaratnam y Simon Haykin. «Cubature Kalman Filters». En: *IEEE Transactions on Automatic Control* 54.6 (jun. de 2009), páginas 1254-1269. ISSN: 1558-2523. DOI: 10.1109/TAC.2009.2019800. URL: https://ieeexplore.ieee.org/document/4982682/?arnumber=4982682 (visitado 19-02-2025) (véase página 79).

[109] Joachim Clemens y Constantin Wellhausen. «The Square-Root Unscented and the Square-Root Cubature Kalman Filters on Manifolds». En: *Sensors* 24.20 (ene. de 2024), página 6622. ISSN: 1424-8220. DOI: 10.3390/s24206622. URL: https://www.mdpi.com/1424-8220/24/20/6622 (visitado 18-02-2025) (véase página 79).

[110] R.V. Garcia et al. «Nonlinear filtering for sequential spacecraft attitude estimation with real data: Cubature Kalman Filter, Unscented Kalman Filter and Extended Kalman Filter». En: *Advances in Space Research* 63.2 (ene. de 2019), páginas 1038-1050. ISSN: 02731177. DOI: 10.1016/j.asr.2018.10.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117718307725 (visitado 28-07-2024) (véase página 79).

[111] Alireza Roudbari y Fariborz Saghafi. «Intelligent modeling and identification of aircraft nonlinear flight». En: *Chinese Journal of Aeronautics* 27.4 (1 de ago. de 2014), páginas 759-771. ISSN: 1000-9361. DOI: 10.1016/j.cja.2014.03.017. URL: https://www.sciencedirect.com/science/article/pii/S1000936114000491 (visitado 04-04-2024) (véanse páginas 79, 123).

[112] Lennart Ljung, Ahmet Arda Ozdemir y Rajiv Singh. «Online Features in the MATLAB® System Identification ToolboxTM». En: *IFAC-PapersOnLine*. 18th IFAC Symposium on System Identification SYSID 2018 51.15 (1 de ene. de 2018), páginas 700-705. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2018.09.201. URL: https://www.sciencedirect.com/science/article/pii/S2405896318318731 (visitado 04-04-2024) (véanse páginas 80, 163).

[113] MathWorks. *System Identification Toolbox*. MathWorks Products. URL: https://es.mathworks.com/products/sysid.html (visitado 14-03-2025) (véanse páginas 80, 125, 157).

[114] Jared A. Grauer y Eugene A. Morelli. «Aircraft Parameter Estimation Considering Process and Measurement Noise». En: *Journal of Aircraft* (13 de nov. de 2024), páginas 1-12. ISSN: 0021-8669, 1533-3868. DOI: 10.2514/1.C038080. URL: https://arc.aiaa.org/doi/10.2514/1.C038080 (visitado 10-03-2025) (véase página 80).

[115] Richard E. Maine y Kenneth W. Iliff. «Formulation and Implementation of a Practical Algorithm for Parameter Estimation with Process and Measurement Noise». En: *SIAM Journal on Applied Mathematics* 41.3 (dic. de 1981), páginas 558-579. ISSN: 0036-1399. DOI: 10.1137/0141045. URL: https://epubs.siam.org/doi/10.1137/0141045 (visitado 04-04-2025) (véase página 80).

[116] Jinchao Zhao et al. «An Improved Unscented Kalman Filter Applied to Positioning and Navigation of Autonomous Underwater Vehicles». En: *Sensors* 25.2 (ene. de 2025), página 551. ISSN: 1424-8220. DOI: 10.3390/s25020551. URL: https://www.mdpi.com/1424-8220/25/2/551 (visitado 21-03-2025) (véase página 80).

[117] Junting Wang, Tianhe Xu y Zhenjie Wang. «Adaptive Robust Unscented Kalman Filter for AUV Acoustic Navigation». En: *Sensors* 20.1 (ene. de 2020), página 60. ISSN: 1424-8220. DOI: 10.3390/s20010060. URL: https://www.mdpi.com/1424-8220/20/1/60 (visitado 21-03-2025) (véase página 80).

[118] Han Shen et al. «USV Parameter Estimation: Adaptive Unscented Kalman Filter-Based Approach». En: *IEEE Transactions on Industrial Informatics* 19.6 (jun. de 2023), páginas 7751-7761. ISSN: 1941-0050. DOI: 10.1109/TII.2022.3202521. URL: https://ieeexplore.ieee.org/document/9869691/?arnumber=9869691 (visitado 21-03-2025) (véase página 80).

[119] Qi Song y Jian-Da Han. «An Adaptive UKF Algorithm for the State and Parameter Estimations of a Mobile Robot». En: *Acta Automatica Sinica* 34.1 (1 de ene. de 2008), páginas 72-79. ISSN: 1874-1029. DOI: 10.3724/SP.J.1004.2008.00072. URL: https://www.sciencedirect.com/science/article/pii/S1874102908600026 (visitado 21-03-2025) (véase página 80).

[120] Wei Li et al. «Online Parameters Identification and State of Charge Estimation for Lithium-Ion Battery Using Adaptive Cubature Kalman Filter». En: *World Electric Vehicle Journal* 12.3 (sep. de 2021), página 123. ISSN: 2032-6653. DOI: 10.3390/wevj12030123. URL: https://www.mdpi.com/2032-6653/12/3/123 (visitado 21-03-2025) (véase página 80).

[121] Hamed Danandeh Hesar y Amin Danandeh Hesar. «Adaptive dual augmented extended Kalman filtering of ECG signals». En: *Measurement* 239 (ago. de 2024), página 115457. ISSN: 02632241. DOI: 10.1016/j.measurement.2024.115457. URL: https://linkinghub.elsevier.com/retrieve/pii/S0263224124013423 (visitado 01-10-2024) (véase página 80).

[122] Branko Ristic, Sanjeev Arulampalam y Neil Gordon. *Beyond the Kalman filter. Particle filters for tracking applications*. Artech House, 2004. ISBN: 978-1-58053-631-X (véase página 81).

[123] Wenyan Bai et al. «On extended state Kalman filter-based identification algorithm for aerodynamic parameters». En: *Control Theory and Technology* 22.2 (1 de mayo de 2024), páginas 235-243. ISSN: 2198-0942. DOI: 10.1007/s11768-023-00192-5. URL: https://doi.org/10.1007/s11768-023-00192-5 (visitado 15-02-2025) (véase página 81).

[124] Henrique M. T. Menegaz et al. «A Systematization of the Unscented Kalman Filter Theory». En: *IEEE Transactions on Automatic Control* 60.10 (oct. de 2015), páginas 2583-2598. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2015.2404511. URL: http://ieeexplore.ieee.org/document/7042740/ (visitado 13-08-2024) (véase página 81).

[125] Raman K. Mehra. «Approaches to adaptive filtering». En: *IEEE Transactions on Automatic Control* 17.5 (oct. de 1972), páginas 693-698. ISSN: 1558-2523. DOI: 10.1109/TAC.1972.1100100. URL: https://ieeexplore.ieee.org/document/1100100/?arnumber=1100100 (visitado 01-10-2024) (véase página 84).

[126] Ali Almagbile, Jinling Wang y Weidong Ding. «Evaluating the Performances of Adaptive Kalman Filter Methods in GPS/INS Integration». En: *Journal of Global Positioning Systems* 9.1 (20 de jun. de 2010), páginas 33-40. ISSN: 14463156, 14463164. DOI: 10.5081/jgps.9.1.33. URL: http://www.gnss.com.au/JoGPS/v9n1/JoGPS_v9n1p33-40.pdf (visitado 01-10-2024) (véase página 84).

[127] Eugene A. Morelli. «Flight test validation of optimal input design and comparison to conventional inputs». En: *22nd Atmospheric Flight Mechanics Conference*. 22nd Atmospheric Flight Mechanics Conference. New Orleans,LA,U.S.A.: American Institute of Aeronautics y Astronautics, 11 de ago. de 1997. DOI: 10.2514/6.1997-3711. URL: https://arc.aiaa.org/doi/10.2514/6.1997-3711 (visitado 15-09-2024) (véanse páginas 85, 87, 123).

[128] Nicholas N. Lam, Paul D. Docherty y Rua Murray. «Practical identifiability of parametrised models: A review of benefits and limitations of various approaches». En: *Mathematics and Computers in Simulation* 199 (1 de sep. de 2022), páginas 202-216. ISSN: 0378-4754. DOI: 10.1016/j.matcom.2022.03.020. URL: https://www.sciencedirect.com/science/article/pii/S0378475422001227 (visitado 02-02-2025) (véase página 86).

[129] César de Prada. «Estimación de parámetros». Universidad de Valladolid, 2015. URL: https://www.eii.uva.es/~prada/Estimacionparam.pdf (visitado 02-11-2025) (véase página 86).

[130] Simon Mayr, Gernot Grabmair y Johann Reger. «Input design and parameter estimation with open source tools». En: *IFAC-PapersOnLine* 48.1 (2015), páginas 286-291. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.05.107. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896315001081 (visitado 03-02-2025) (véanse páginas 86, 87, 97).

[131] Narendra K. Gupta y W. Earl Hall Jr. *Input design for identification of aircraft stability and control derivatives*. Feb. de 1975 (véase página 87).

[132] Eugene A. Morelli y Vladislav Klein. «Optimal input design for aircraft parameter estimation using dynamic programming principles». En: *17th Atmospheric Flight Mechanics Conference*. 17th Atmospheric Flight Mechanics Conference. Portland, OR, U.S.A.: American Institute of Aeronautics y Astronautics, 20 de ago. de 1990. DOI: 10.2514/6.1990-2801. URL: https://arc.aiaa.org/doi/10.2514/6.1990-2801 (visitado 14-03-2025) (véase página 87).

[133] Eugene A. Morelli. «Practical input optimization for aircraft parameter estimation experiments». Tesis doctoral. Hampton, Virginia: George Washington University, 1 de mayo de 1993. URL: https://ntrs.nasa.gov/citations/19930018075 (visitado 14-03-2025) (véase página 87).

[134] Eugenio Hernández Rodríguez, María Jesús Vázquez Gallo y María Ángeles Zurro Moro. *Álgebra lineal y geometría*. 3.ª edición. Madrid (España): Pearson Educación, S.A, 2012. ISBN: 978-84-7829-129-8. URL: https://go.exlibris.link/BJZg60D7 (visitado 26-02-2025) (véase página 87).

[135] Morten Knudsen. «A Sensitivity Approach for Estimation of Physical Parameters». En: *IFAC Proceedings Volumes* 27.8 (jul. de 1994), páginas 533-538. ISSN: 14746670. DOI: 10.1016/S1474-6670(17)47763-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667017477639 (visitado 09-09-2024) (véanse páginas 87, 88).

[136] Elías Revestido Herrero et al. «Experiment design for model basin tests with a remotely operated vehicle». En: *Ocean Engineering* 307 (1 de sep. de 2024), página 118215. ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2024.118215. URL: https://www.sciencedirect.com/science/article/pii/S0029801824015531 (visitado 10-02-2025) (véanse páginas 87, 88).

[137] Morten Knudsen. «Direct Estimation of Physical Parameters in Nonlinear Loudspeaker Models». En: *IFAC Proceedings Volumes*. IFAC Symposium on System Identification (SYSID'94), Copenhagen, Denmark, 4-6 July 27.8 (1 de jul. de 1994), páginas 551-556. ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)47766-4. URL: https://www.sciencedirect.com/science/article/pii/S1474667017477664 (visitado 17-03-2025) (véase página 87).

[138] M. Eslamdoust, A.R. Toloei y A.R. Vali. «Fault Tolerant Control (FTC) Scheme for Unmanned Aerial Vehicles (UAV)». En: *International Journal of Computer Applications* 126.3 (17 de sep. de 2015), páginas 10-14. ISSN: 09758887. DOI: 10.5120/ijca2015906003. URL: https://www.researchgate.net/publication/283201245_Fault_Tolerant_Control_FTC_Scheme_for_Unmanned_Aerial_Vehicles_UAV (visitado 22-02-2025) (véase página 89).

[139] Yan Zhou et al. «L1 Adaptive Fault-Tolerant Attitude Tracking Control of UAV Systems Subject to Faults and Input Saturation». En: *International Journal of Aerospace Engineering* 2024.1 (22 de oct. de 2024). ISSN: 1687-5974. DOI: 10.1155/2024/7955020. URL: https://www.researchgate.net/publication/382212643_L1_Adaptive_Fault-Tolerant_Attitude_Tracking_Control_of_UAV_Systems_Subject_to_Faults_and_Input_Saturation (visitado 22-02-2025) (véanse páginas 89, 90).

[140] Radosław Puchalski y Wojciech Giernacki. «UAV Fault Detection Methods, State-of-the-Art». En: *Drones* 6.11 (29 de oct. de 2022), página 330. ISSN: 2504-446X. DOI: 10.3390/drones6110330. URL: https://www.mdpi.com/2504-446X/6/11/330 (visitado 06-10-2024) (véase página 89).

[141] Benkuan Wang et al. «Multivariate Regression-Based Fault Detection and Recovery of UAV Flight Data». En: *IEEE Transactions on Instrumentation and Measurement* 69.6 (jun. de 2020), páginas 3527-3537. ISSN: 1557-9662. DOI: 10.1109/TIM.2019.2935576. URL: https://ieeexplore.ieee.org/document/8801927 (visitado 22-02-2025) (véase página 89).

[142] Qibao Shu et al. «Fault Tolerant Predictive Control Based on Discrete-Time Sliding Mode Observer for Quadrotor UAV». En: *Journal of Advanced Computational Intelligence and Intelligent Informatics* 22.4 (20 de jul. de 2018). Publisher: Fuji Technology Press Ltd., páginas 498-505. DOI: 10.20965/jaciii.2018.p0498. URL: https://www.fujipress.jp/jaciii/jc/jacii002200040498/ (visitado 22-02-2025) (véase página 90).

[143] «Fault diagnosis in dynamic systems via Kalman Filter innovation sequence». En: editado por International Federation of Automatic Control. Medium: electronic resource. Oxford, England: Pergamon, 1999. ISBN: 978-0-08-043248-9. URL: http://www.loc.gov/catdir/enhancements/fy0616/2003557803-d.html (visitado 30-09-2024) (véase página 90).

[144] Chingiz Hajiyev y Halil Ersin Soken. «Robust Adaptive Kalman Filter for estimation of UAV dynamics in the presence of sensor/actuator faults». En: *Aerospace Science and Technology* 28.1 (jul. de 2013), páginas 376-383. ISSN: 12709638. DOI: 10.1016/j.ast.2012.12.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S1270963812002027 (visitado 30-09-2024) (véase página 90).

[145] Tor A. Johansen. «Toward Dependable Embedded Model Predictive Control». En: *IEEE Systems Journal* 11.2 (jun. de 2017), páginas 1208-1219. ISSN: 1937-9234. DOI: 10.1109/JSYST.2014.2368129. URL: https://ieeexplore.ieee.org/abstract/document/6971061 (visitado 13-01-2025) (véase página 91).

[146] James B. Rawlings, David Q. Mayne y Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2.ª edición. Santa Barbara, California: Nob Hill Publishing, oct. de 2017. ISBN: 978-0-9759377-8-5. URL: https://sites.engineering.ucsb.edu/~jbraw/mpc/MPC-book-2nd-edition-5th-printing.pdf (visitado 13-01-2025) (véanse páginas 91, 93).

[147] Rolf Findeisen y Frank Allgower. «An introduction to nonlinear model predictive control». En: 21st Benelux meeting on systems and control. Volumen 11. Veldhoven, Netherlands, 2002, páginas 119-141. URL: https://pure.tue.nl/ws/files/3079152/555518.pdf#page=120 (véase página 91).

[148] Yutao Chen et al. «MATMPC - A MATLAB Based Toolbox for Real-time Nonlinear Model Predictive Control». En: *2019 18th European Control Conference (ECC)*. 2019 18th European Control Conference (ECC). Jun. de 2019, páginas 3365-3370. DOI: 10.23919/ECC.2019.8795788. URL: https://ieeexplore.ieee.org/document/8795788/?arnumber=8795788 (visitado 23-12-2024) (véanse páginas 91, 96).

[149] Robin Verschueren et al. «acados—a modular open-source framework for fast embedded optimal control». En: *Mathematical Programming Computation* 14.1 (mar. de 2022), páginas 147-183. ISSN: 1867-2949, 1867-2957. DOI: 10.1007/s12532-021-00208-8. URL: https://link.springer.com/10.1007/s12532-021-00208-8 (visitado 23-12-2024) (véanse páginas 91, 94, 95, 96).

[150] Álvaro Ortiz Moya. «Design and Simulation of the guidance and control system for an F-86 Sabre». TFG. Valencia: Universitat Politècnica de València, 27 de oct. de 2020. URL: https://riunet.upv.es/handle/10251/153251 (visitado 02-01-2025) (véase página 91).

[151] Antonio J. Gallego et al. «Adaptive UKF-based model predictive control of a Fresnel collector field». En: *Journal of Process Control* 85 (ene. de 2020), páginas 76-90. ISSN: 09591524. DOI: 10.1016/j.jprocont.2019.09.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S0959152419300691 (visitado 13-12-2024) (véase página 91).

[152] Jiwon Lee y Youdan Kim. «MPC-Based Longitudinal Control Design for a Fixed-Wing VTOL UAV in Transition Flight». En: *AIAA SCITECH 2024 Forum*. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2024-0141. American Institute of Aeronautics y Astronautics. DOI: 10.2514/6.2024-0141. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2024-0141 (visitado 01-03-2025) (véase página 91).

[153] Pengkai Ru y Kamesh Subbarao. «Nonlinear Model Predictive Control for Unmanned Aerial Vehicles». En: *Aerospace* 4.2 (17 de jun. de 2017), página 31. ISSN: 2226-4310. DOI: 10.3390/aerospace4020031. URL: https://www.mdpi.com/2226-4310/4/2/31 (visitado 23-02-2025) (véanse páginas 91, 92).

[154] Gonzalo Garcia y Shahriar Keshmiri. «Nonlinear Model Predictive Controller for Navigation, Guidance and Control of a Fixed-Wing UAV». En: *AIAA Guidance, Navigation, and Control Conference*. AIAA Guidance, Navigation, and Control Conference. Portland, Oregon: American Institute of Aeronautics y Astronautics, 8 de ago. de 2011. ISBN: 978-1-60086-952-5. DOI: 10.2514/6.2011-6310. URL: https://arc.aiaa.org/doi/10.2514/6.2011-6310 (visitado 23-02-2025) (véase página 91).

[155] Vinayak Deshpande y Youmin Zhang. «Fault-Tolerant Model Predictive Control of a Fixed-Wing UAV with Actuator Fault Estimation». En: *Guidance, Navigation and Control* 01 (1 de dic. de 2021). DOI: 10.1142/S2737480721400069. URL: https://www.researchgate.net/publication/358096085_Fault-Tolerant_Model_Predictive_Control_of_a_Fixed-Wing_UAV_with_Actuator_Fault_Estimation (véase página 91).

[156] Erlend Magnus Lervik Coates. «Nonlinear Attitude and Path-Following Control of Fixed-Wing Aircraft». Tesis doctoral. Trondheim: Norwegian University of Sciencie y Technology, sep. de 2023 (véase página 92).

[157] Erlend M. Coates y Thor I. Fossen. «Geometric Reduced-Attitude Control of Fixed-Wing UAVs». En: *Applied Sciences* 11.7 (1 de abr. de 2021), página 3147. ISSN: 2076-3417. DOI: 10.3390/app11073147. URL: https://www.mdpi.com/2076-3417/11/7/3147 (visitado 20-01-2025) (véase página 92).

[158] Dirk Reinhardt y Tor Arne Johansen. «Nonlinear Model Predictive Attitude Control for Fixed-Wing Unmanned Aerial Vehicle based on a Wind Frame Formulation». En: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2019 International Conference on Unmanned Aircraft Systems (ICUAS). Atlanta, GA, USA: IEEE, jun. de 2019, páginas 503-512. ISBN: 978-1-7281-0333-4. DOI: 10.1109/ICUAS.2019.8798229. URL: https://ieeexplore.ieee.org/document/8798229/ (visitado 23-02-2025) (véase página 92).

[159] Dirk Peter Reinhardt. «On Nonlinear and Optimization-based Control of Fixed-Wing Unmanned Aerial Vehicles». Accepted: 2022-06-14T12:53:56Z ISBN: 9788232664795 ISSN: 2703-8084. Doctoral thesis. NTNU, 2022. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2998704 (visitado 01-03-2025) (véase página 92).

[160] Dirk Reinhardt, Sebastien Gros y Tor Arne Johansen. «Fixed-Wing UAV Path-Following Control via NMPC on the Lowest Level». En: *2023 IEEE Conference on Control Technology and Applications (CCTA)*. 2023 IEEE Conference on Control Technology and Applications (CCTA). ISSN: 2768-0770. Ago. de 2023, páginas 451-458. DOI: 10.1109/CCTA54093.2023.10253315. URL: https://ieeexplore.ieee.org/document/10253315/?arnumber=10253315 (visitado 23-02-2025) (véase página 92).

[161] Thomas J. Stastny, Adyasha Dash y Roland Siegwart. «Nonlinear MPC for Fixed-wing UAV Trajectory Tracking: Implementation and Flight Experiments». En: *AIAA Guidance, Navigation, and Control Conference*. AIAA Guidance, Navigation, and Control Conference. Grapevine, Texas: American Institute of Aeronautics y Astronautics, 9 de ene. de 2017. ISBN: 978-1-62410-450-3. DOI: 10.2514/6.2017-1512. URL: https://arc.aiaa.org/doi/10.2514/6.2017-1512 (visitado 23-02-2025) (véase página 92).

[162] Erlend M. Coates et al. «Toward Nonlinear Flight Control for Fixed-Wing UAVs: System Architecture, Field Experiments, and Lessons Learned». En: *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2022 International Conference on

Unmanned Aircraft Systems (ICUAS). Dubrovnik, Croatia: IEEE, 21 de jun. de 2022, páginas 724-734. ISBN: 978-1-6654-0593-5. DOI: 10.1109/ICUAS54217.2022.9836064. URL: https://ieeexplore.ieee.org/document/9836064/ (visitado 20-01-2025) (véanse páginas 92, 97).

[163] Khadas. *VIM3*. Khadas. URL: https://www.khadas.com/vim3https://www.khadas.com/shop?Collection=All&sort=price_descending (visitado 25-03-2025) (véase página 92).

[164] Per Anders Stadheim. «Robustness Analysis of a Nonlinear Model Predictive Controller for Fixed-Wing UAVs». Accepted: 2021-09-23T18:13:00Z. Master thesis. NTNU, 2020. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2780956 (visitado 24-02-2025) (véanse páginas 92, 97).

[165] Ahmed Samir, Horacio M Calderon y Herbert Werner. «Predictive Path-Following with Stability Guarantee for Fixed-Wing UAVs Using the qLMPC Framework». En: (). URL: https://paperhost.org/proceedings/controls/ECC24/files/0690.pdf (visitado 01-03-2025) (véase página 92).

[166] Victor M. Zavala y Lorenz T. Biegler. «The advanced-step NMPC controller: Optimality, stability and robustness». En: *Automatica* 45.1 (ene. de 2009), páginas 86-93. ISSN: 00051098. DOI: 10.1016/j.automatica.2008.06.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109808004196 (visitado 06-03-2025) (véase página 92).

[167] *ArduPilot*. ArduPilot.org. URL: https://ardupilot.org (visitado 25-03-2025) (véase página 92).

[168] *PX4 - Open Source Autopilot for Drones*. PX4 Autopilot. URL: https://px4.io/ (visitado 25-03-2025) (véase página 93).

[169] Yutao Chen et al. «An Adaptive Partial Sensitivity Updating Scheme for Fast Nonlinear Model Predictive Control». En: *IEEE Transactions on Automatic Control* 64.7 (jul. de 2019), páginas 2712-2726. ISSN: 0018-9286, 1558-2523, 2334-3303. DOI: 10.1109/TAC.2018.2867916. arXiv: 1808.00877[cs]. URL: http://arxiv.org/abs/1808.00877 (visitado 21-01-2025) (véase página 94).

[170] *acados/docs/problem_formulation/problem_formulation_ocp_mex.pdf at master · acados/acados · GitHub*. URL: https://github.com/acados/acados/blob/master/docs/problem_formulation/problem_formulation_ocp_mex.pdf (visitado 24-12-2024) (véanse páginas 94, 95).

[171] José Joaquín Sainz et al. «Auto-tuning of Non-Linear Model Predictive Controllers for a Remote Operated Vehicle». En: *OCEANS 2023 - Limerick*. OCEANS 2023 - Limerick. Jun. de 2023, páginas 1-5. DOI: 10.1109/OCEANSLimerick52467.2023.10244375. URL: https://ieeexplore.ieee.org/document/10244375/?arnumber=10244375 (visitado 21-01-2025) (véase página 96).

[172] MathWorks. *Model Predictive Control Toolbox*. MathWorks Products. URL: https://es.mathworks.com/help/mpc/ (visitado 12-01-2025) (véase página 97).

[173] Joel A. E. Andersson et al. «CasADi: a software framework for nonlinear optimization and optimal control». En: *Mathematical Programming Computation* 11.1 (mar. de 2019), páginas 1-36. ISSN: 1867-2949, 1867-2957. DOI: 10.1007/s12532-018-0139-4. URL: http://link.springer.com/10.1007/s12532-018-0139-4 (visitado 17-12-2024) (véase página 97).

[174] Dirk Reinhardt y Tor Arne Johansen. «Control of Fixed-Wing UAV Attitude and Speed based on Embedded Nonlinear Model Predictive Control». En: *IFAC-PapersOnLine*. 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021 54.6 (1 de ene. de 2021), páginas 91-98. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2021.08.529. URL: https://www.sciencedirect.com/science/article/pii/S2405896321013045 (visitado 12-01-2025) (véase página 97).

[175] Dirk Reinhardt y Tor Arne Johansen. «Nonlinear Model Predictive Control combined with Geometric Attitude and Speed Control for Fixed-Wing UAVs». En: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2021 International Conference on Unmanned Aircraft Systems (ICUAS). Athens, Greece: IEEE, 15 de jun. de 2021, páginas 465-475. ISBN: 978-1-6654-1535-4. DOI: 10.1109/ICUAS51884.2021.9476855. URL: https://ieeexplore.ieee.org/document/9476855/ (visitado 23-02-2025) (véase página 97).

[176] Marco Concetto Bonazza. «Implementation of adaptive nonlinear model predictive control on a PX4-enabled quad-rotor platform». Tesis de máster. University of Padova, 2022. URL: https://thesis.unipd.it/retrieve/fe67d9a9-7ef1-4a65-83d1-3416b4111d7a/Bonazza_MarcoConcetto.pdf (visitado 24-02-2025) (véase página 97).

[177] Mohamed Elhesasy et al. «Non-Linear Model Predictive Control Using CasADi Package for Trajectory Tracking of Quadrotor». En: *Energies* 16.5 (ene. de 2023). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, página 2143. ISSN: 1996-1073. DOI: 10.3390/en16052143. URL: https://www.mdpi.com/1996-1073/16/5/2143 (visitado 24-02-2025) (véase página 97).

[178] Siri Holthe Mathisen, Thor I. Fossen y Tor A. Johansen. «Non-linear model predictive control for guidance of a fixed-wing UAV in precision deep stall landing». En: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2015 International Conference on Unmanned Aircraft Systems (ICUAS). Jun. de 2015, páginas 356-365. DOI: 10.1109/ICUAS.2015.7152310. URL: https://ieeexplore.ieee.org/abstract/document/7152310 (visitado 24-02-2025) (véase página 97).

[179] Eugene A. Morelli. «Practical Aspects of Multiple-Input Design for Aircraft System Identification Flight Tests». En: AIAA AVIATION 2021 FORUM. AIAA AVIATION Forum. American Institute of Aeronautics y Astronautics, 28 de jul. de 2021. DOI: 10.2514/6.2021-2795. URL: https://ntrs.nasa.gov/api/citations/20210018115/downloads/Morelli_Practical_FTI_Paper_v1%200706.pdf (visitado 25-09-2024) (véanse páginas 123, 124).

[180] Eugene A. Morelli. «Flight Test Maneuvers for Efficient Aerodynamic Modeling». En: *Journal of Aircraft* 49.6 (nov. de 2012), páginas 1857-1867. ISSN: 0021-8669, 1533-3868. DOI: 10.2514/1.C031699. URL: https://arc.aiaa.org/doi/10.2514/1.C031699 (visitado 09-03-2025) (véase página 123).

[181] André Fialho Coelho. «System Identification and Parameter Space Control Design for a Small Unmanned Aircraft». TFG. Brasil: Instituto Federal Fluminense, 18 de sep. de 2017. URL: https://www.researchgate.net/publication/321669893_System_Identification_and_Parameter_Space_Control_Design_for_a_Small_Unmanned_Aircraft (véase página 123).

[182] Eugene A. Morelli. «System Identification Programs for AirCraft (SIDPAC)». En: *AIAA Atmospheric Flight Mechanics Conference and Exhibit*. AIAA Atmospheric Flight Mechanics Conference and Exhibit. Monterey, California: American Institute of Aeronautics y Astronautics, 5 de ago. de 2002. ISBN: 978-1-62410-107-6. DOI: 10.2514/6.2002-4704. URL: https://arc.aiaa.org/doi/10.2514/6.2002-4704 (visitado 08-03-2025) (véase página 124).

[183] Steve Derry y Eugene A. Morelli. «System Identification Methods for Aerodynamic Modeling and Validation using Flight Data». URL: https://nescacademy.nasa.gov/review/downloadfile.php?file=0305_System_ID_Methods_for_Aero_Modeling_and_Validation__Morelli_17Nov11.pdf&id=166&distr=Public (visitado 15-02-2025) (véase página 124).

[184] Eugene A. Morelli y Jared A. Grauer. «Advances in Aircraft System Identification at NASA Langley Research Center». En: *Journal of Aircraft* 60.5 (sep. de 2023), páginas 1354-1370. ISSN: 0021-8669, 1533-3868. DOI: 10.2514/1.C037274. URL: https://arc.aiaa.org/doi/10.2514/1.C037274 (visitado 07-03-2025) (véase página 124).

[185] MathWorks. *idinput - Generate input signals to support system identification*. Documentación de MATLAB. URL: https://es.mathworks.com/help/ident/ref/idinput.html (visitado 25-09-2024) (véase página 125).

[186] commodos. *MUMI*. original-date: 2021-11-19T13:13:23Z. 4 de jul. de 2022. URL: https://github.com/commodos/-MUMI (visitado 28-09-2024) (véase página 125).

[187] MathWorks. *damp - Matlab documentation*. URL: https://es.mathworks.com/help/control/ref/dynamicsystem.damp.html (visitado 08-03-2025) (véase página 125).

[188] L. Ljung. «Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems». En: *IEEE Transactions on Automatic Control* 24.1 (feb. de 1979), páginas 36-50. ISSN: 1558-2523. DOI: 10.1109/TAC.1979.1101943. URL: https://ieeexplore.ieee.org/document/1101943/?arnumber=1101943 (visitado 20-03-2025) (véase página 131).

[189] R. V. Jategaonkar y E. Plaetschke. «Algorithms for aircraft parameter estimation accounting for process and measurement noise». En: *Journal of Aircraft* 26.4 (abr. de 1989), páginas 360-372. ISSN: 0021-8669. DOI: 10.2514/3.45769. URL: https://arc.aiaa.org/doi/10.2514/3.45769 (visitado 04-04-2025) (véase página 131).

[190] Jochem De Schutter, Mario Zanon y Moritz Diehl. «TuneMPC—A Tool for Economic Tuning of Tracking (N)MPC Problems». En: *IEEE Control Systems Letters* 4.4 (oct. de 2020), páginas 910-915. ISSN: 2475-1456. DOI: 10.1109/LCSYS.2020.2996019. URL: https://ieeexplore.ieee.org/document/9097289/?arnumber=9097289 (visitado 02-03-2025) (véase página 137).

Parte V

APÉNDICES

A. PROGRAMAS DE MATLAB

Para realizar las simulaciones de este trabajo fue necesario desarrollar una serie de programas de MATLAB. En este capítulo se incluyen los programas completos en forma de texto (*listings*).

A.1. FILTRO DE KALMAN

El UKF y CKF están disponibles directamente en MATLAB como parte de la toolbox de identificación [113]. No obstante, como ejercicio académico, se ha programado para este trabajo el UKF en forma de clase de MATLAB, de forma que se pueden crear múltiples objetos a partir de ella y que estos desempeñen diversas tareas. Esta forma de programación facilitará reutilizar los algoritmos en otros proyectos. El código básico del UKF se obtuvo de [96] y se adaptó a una clase de MATLAB, obteniendo el código presentado a continuación.

Listing A.1: Clase del UKF

```
1  classdef AdaptiveUnscentedKalmanFilter < handle
2      % Class for implementing an Adaptive Unscented Kalman Filter
3      % The functions predict and correct have been adapted from the
4      % paper "Understanding Nonlinear Kalman Filters, Part II: An
5      % Implementation Guide" by Matthew Rhudy and Yu Gu
6
7      properties (Access=public)
8          % General or basic
9          L % Number of states
10         O % Dimension of output vector
11         Q % Process noise covariance
12         Qlambda % Weighting factor for updating the Q matrix
13         R % Measurement noise covariance
14         Rdelta % Weighting factor for updating the R matrix
15         x_m % Last predicted state
16         P_m % Last prediction covariance
17         xhat % Last estimated state
18         Phat % Last estimation covariance
19         Pzz
20         alpha
21         beta
22         kappa
23         lambda
24         K % Last Kalman gain matrix
25
26         % Functions and discretizing
27         Ts % Sample time of discrete system
28         disc_method % Discretization method
29         f % State update function, either Xdot(x,u) or x[k+1](x,u)
30         h % Measurement function
31         f_continous % Continous time system model: Xdot(x,u)
32
33         % Adaptive
34         enabled_adaptive % Flag to execute the adaptive methods
35         lambda_Q0
36         delta_R0
```

```
37          chi2_sigma % Fault detection threshold
38          a
39          b
40          phi
41
42          % Weights and sigma points
43          Wm
44          Wc
45          chi_m % Sigma points progapated through f
46          % psi_m % Sigma points progapated through h
47
48      end
49
50      %% Dynamic methods
51      methods
52          function obj = AdaptiveUnscentedKalmanFilter(f_fun, h_fun, ...
53                  Xinit, Pinit, Q, alpha, beta, kappa, options)
54              % Class constructor
55              arguments
56                  f_fun function_handle
57                  h_fun function_handle
58                  Xinit (:,1) double {mustBeVector}
59                  Pinit double
60                  Q (1,1) double
61                  alpha (1,1) double  = 1e-3
62                  beta (1,1) double = 2
63                  kappa (1,1) double = 0
64                  options.model_type string {mustBeMember(options.model_type,
                        ["discrete", "continous"])} = "discrete"
65                  options.sampling_period double = 1
66                  options.disc_method string {mustBeMember(options.disc_method,
                        ["euler", "modified euler"])} = "modified euler"
67                  options.lambda_min (1,1) double = 0.1
68                  options.a (1,1) double = 5
69                  options.chi2_sigma_confidence (1,1) double = 0.9
70                  options.enabled_adaptive (1,1) logical = false
71              end
72
73              obj.L = size(Xinit,1);
74              obj.Q = Q;
75              obj.xhat = Xinit;
76              obj.Phat = Pinit;
77
78              obj.alpha = alpha;
79              obj.beta = beta;
80              obj.kappa = kappa;
81
82              obj.enabled_adaptive = options.enabled_adaptive;
83
84              if options.model_type == "continous"
85                  obj.Ts = options.sampling_period;
86                  obj.disc_method = options.disc_method;
87                  obj.f_continous = f_fun;
88                  obj.f = @obj.discretize_model;
89              elseif options.model_type == "discrete"
90                  obj.f = f_fun;
91              end
92
93              obj.h = h_fun;
```

```
94
95              obj.chi2_sigma = chi2inv(options.chi2_sigma_confidence, obj.O);
96              obj.lambda_Q0 = options.lambda_min;
97              obj.delta_R0 = obj.lambda_Q0;
98              obj.a = options.a;
99              obj.b = obj.a;
100             obj.phi = 0;
101
102             % Calculate weights for sigma points
103             obj.ut_weights();
104         end
105
106
107         function [x_m, P_m] = predict(obj, varargin)
108             % Predict the next state x_{k+1} and its covariance P given
109             % the actual estimate x_k and input u_k using the state
110             % transition function.
111
112             % x_p = "x previous" = x(k-1)
113             x_p = obj.xhat;
114
115             % Step 1: Generate sigma points
116             % chi_p = "chi previous" = chi(k-1|k-1)
117
118             chi_p = obj.generate_sigma_points(x_p, obj.Phat) ;
119
120             % Step 2: Prediction Transformation
121             % Propagate each sigma-point through prediction
122             % chi_m = "chi minus" = chi(k|k-1)
123             chi_m = zeros(obj.L, 2*obj.L+1); % Allocate memory
124             for i = 1:size(chi_m,2)
125                 chi_m(:,i) = feval(obj.f, chi_p(:,i), varargin{:});
126             end
127             % Calculate mean of predicted state
128             x_m = chi_m*obj.Wm;
129             % Calculate covariance of predicted state
130             P_m = obj.Q;
131             for i = 1:2*obj.L+1
132                 P_m = P_m + obj.Wc(i)*(chi_m(:,i) - x_m)*(chi_m(:,i) - x_m)';
133             end
134
135             obj.P_m = P_m;
136             obj.x_m = x_m;
137             obj.chi_m = chi_m;
138         end
139
140         function [xhatnew, Phatnew] = correct(obj, Y, varargin)
141             % Correct the a priori predictions with new measurments
142
143             % Step 3: Observation Transformation
144             % Propagate each sigma-point through observation
145             psi_m = zeros(obj.O, 2*obj.L+1);
146             for i = 1:size(obj.chi_m,2)
147                 psi_m(:,i) = heval(obj, obj.chi_m(:,i), varargin{:});
148             end
149             % Calculate mean of predicted output
150             y_m = psi_m*obj.Wm;
151             % Calculate covariance of predicted output
152             % and cross-covariance between state and output
```

```
153             Pyy = obj.R;
154             Pxy = zeros(obj.L,obj.O);
155             for i = 1:2*obj.L+1
156                 Pyy = Pyy + obj.Wc(i)*(psi_m(:,i) - y_m)*(psi_m(:,i) - y_m)';
157                 Pxy = Pxy + obj.Wc(i)*(obj.chi_m(:,i) - obj.x_m)*(psi_m(:,i) -
                        y_m)';
158             end
159
160             % Step 4: Measurement Update
161             obj.K = Pxy/(Pyy+eps(10)*ones(size(Pyy))); % Calculate Kalman gain
                    matrix
162             if any(isnan(obj.K(:)))
163                 xx=1;
164             end
165             x_hat = obj.x_m + obj.K*(Y - y_m); % Update state estimate
166             P_hat = obj.P_m - obj.K*Pyy*obj.K'; % Update covariance estimate
167
168             obj.xhat = x_hat;
169             obj.Phat = P_hat;
170             obj.Pzz = Pyy;
171
172             if obj.enabled_adaptive == true
173                 [xhatnew, Phatnew] = obj.fault_detection(Y, varargin{:});
174             else
175                 xhatnew = x_hat;
176                 Phatnew = P_hat;
177             end
178         end
179
180         function ut_weights(obj)
181             % Calculate the weights for the Unscented Transform
182
183             obj.lambda = obj.alpha^2 * (obj.L + obj.kappa) - obj.L;
184
185             obj.Wm = ones(2*obj.L + 1,1)*1/(2*(obj.L+obj.lambda));
186             obj.Wc = obj.Wm;
187
188             obj.Wm(1) = obj.lambda/(obj.lambda+obj.L);
189             obj.Wc(1) = obj.lambda/(obj.lambda+obj.L) + 1 - obj.alpha^2 +
                    obj.beta;
190         end
191
192         function [epsilon] = residual(obj, Z, varargin)
193             % RESIDUAL  Calculate the residual using the provided measurement
                    and the
194             % last estimated output
195                 epsilon = Z - obj.heval(obj.xhat, varargin{:});
196         end
197
198         function [mu] = innovation(obj, Z, varargin)
199             % Calculate the innovation using the provided measurement and
200             % the last predicted state
201
202             mu = Z - obj.heval(obj.x_m, varargin{:});
203         end
204
205         function [xhatnew, Phatnew] = fault_detection(obj, Z, varargin)
206             % Detect a fault in the estimation due to errors in Q and R
207             % matrices. If there is one, both matrices are corrected
```

```
208
209             mu = obj.innovation(Z, varargin{:});
210             epsilon = obj.residual(Z, varargin{:});
211
212             obj.phi = mu'/obj.Pzz*mu;
213
214             if obj.phi > obj.chi2_sigma
215                 % Update Q matrix
216                 lambda_Q = max(obj.lambda_Q0,
                        (obj.phi-obj.b*obj.chi2_sigma)/obj.phi);
217                 obj.updateQmatrix(mu, lambda_Q)
218
219                 % Update R matrix
220                 delta_R = max(obj.delta_R0,
                        (obj.phi-obj.a*obj.chi2_sigma)/obj.phi);
221                 [y_kk, chi_kk, psi_kk, S_zz] = obj.updateRmatrix(epsilon,
                        delta_R, varargin{:});
222
223                 % Correct estimations
224                 [obj.xhat, obj.Phat] = correct_estimations(obj, obj.xhat,
                        y_kk, Z, chi_kk, psi_kk, S_zz);
225
226             end
227
228             xhatnew = obj.xhat;
229             Phatnew = obj.Phat;
230         end
231
232         function updateQmatrix(obj, mu, lambda)
233             % Correct the Q matrix given the weighting factor lambda and
234             % the innovation mu
235
236             obj.Q = (1-lambda)*obj.Q + lambda*(obj.K*(mu*mu')*obj.K');
237         end
238
239         function [y_kk, chi_kk, psi_kk, S_zz] = updateRmatrix(obj, epsilon,
                delta, varargin)
240             % Correct the R matrix given the weighting factor delta and
241             % the residual epsilon
242
243             % Step 1: Recalculate sigma points
244             % chi_kk = "chi for current estimate" = chi(k|k)
245             chi_kk = obj.generate_sigma_points(obj.xhat, obj.Phat);
246
247             % Propagate each sigma-point through observation
248             psi_kk = zeros(obj.O, 2*obj.L+1);
249             for i = 1:size(chi_kk,2)
250                 psi_kk(:,i) = obj.heval(chi_kk(:,i), varargin{:});
251             end
252             % Calculate mean of current estimated output
253             y_kk = psi_kk*obj.Wm;
254
255             % Calculate S matrix
256             % S_zz = S_zz(k|k)
257             S_zz = zeros(obj.O);
258             for i = 1:size(chi_kk,2)
259                 S_zz = S_zz + obj.Wc(i)*(obj.heval(chi_kk(:,i),varargin{:}) -
                        ...
260                     y_kk)*(obj.heval(chi_kk(:,i),varargin{:}) - y_kk)';
```

```
261             end
262
263             % Update R matrix
264             obj.R = (1-delta)*obj.R + delta*(epsilon*epsilon' + S_zz);
265
266         end
267
268         function [sigma_points] = generate_sigma_points(obj, x, P)
269             [V,D] = eig(P);
270             d=diag(D);
271             if not(all(d > 0)) %length(d)*eps(max(d)))
272                 d(d<0)=eps;
273                 P=V*D/V;
274             end
275             S_P = schol(P); % Calculate square root of error covariance
276
277             sigma_points = [x, x*ones(1,obj.L)+sqrt(obj.L+obj.lambda)*S_P, ...
278                 x*ones(1,obj.L)-sqrt(obj.L+obj.lambda)*S_P];
279
280         end
281
282         function [x_kk_new, P_kk_new] = correct_estimations(obj, x_kk, z_kk,
                Z, chi_kk, psi_kk, S_kk)
283             % x_kk = obj.xhat;
284
285             % Compute new predicted error covariance matrix
286             P_xx_kk_pred = obj.Q;
287             for i = 1:2*obj.L+1
288                 P_xx_kk_pred = P_xx_kk_pred + obj.Wc(i)*(chi_kk(:,i) -
                        x_kk)*(chi_kk(:,i) - x_kk)';
289             end
290
291             % Calculate new cross-covariance matrix
292             P_xy_kk = zeros(obj.L,obj.O); % Allocate memory
293             for i = 1:2*obj.L+1
294                 P_xy_kk = P_xy_kk + obj.Wc(i)*(chi_kk(:,i) -
                        x_kk)*(psi_kk(:,i) - z_kk)';
295             end
296
297             P_zz_kk = S_kk + obj.R;
298             obj.K = P_xy_kk/P_zz_kk;
299
300             if isnan(P_xy_kk)
301                 w = 1;
302             end
303             x_kk_new = x_kk + obj.K*(Z - z_kk);
304             P_kk_new = P_xx_kk_pred - obj.K*P_zz_kk*obj.K';
305         end
306
307         function y = heval(obj, varargin)
308             y = feval(obj.h, varargin{:});
309         end
310
311         function x_next = discretize_model(obj, x, varargin)
312             if obj.disc_method == "euler"
313                 x_next = x + obj.f_continous(x, varargin{:})*obj.Ts;
314             elseif obj.disc_method == "modified euler"
315                 x_euler = x + obj.f_continous(x, varargin{:})*obj.Ts;
316                 f_avg = (obj.f_continous(x_euler,varargin{:}) + ...
```

```
317                     obj.f_continous(x,varargin{:}))/2;
318                 x_next = x + f_avg*obj.Ts;
319             end
320         end
321     end
322 end
```

El filtro debe recibir un modelo en tiempo discreto de la aeronave. Este debe calcular el estado siguiente partiendo del estado actual y la entrada actual. Para discretizar el modelo continuo del UAV, se utiliza el método de Runge-Kutta 4 (RK4), cuyo código MATLAB se presenta a continuación.

Listing A.2: Función para discretizar mediante RK4

```
1  function [y_next] = rk4(f, h, y, varargin)
2  % rk4 performs a single Runge-Kutta 4th order integration step.
3  %
4  %   [y_next, t_next] = rk4(f, y, h, varargin) where:
5  %       f         : function describing the ODE, f(t, y, ...)
6  %       h         : integration step
7  %       y         : current solution value (vector/column matrix)
8  %       varargin : additional arguments passed to f
9  %
10 %   Outputs:
11 %       y_next    : solution value at t + h
12
13     k1 = f(y,             varargin{:});
14     k2 = f(y + (h/2)*k1, varargin{:});
15     k3 = f(y + (h/2)*k2, varargin{:});
16     k4 = f(y + h*k3,     varargin{:});
17
18     y_next = y + (h/6)*(k1 + 2*k2 + 2*k3 + k4);
19 end
```

A.1.1. Ejemplo de uso del filtro

En el programa siguiente se presenta un ejemplo de uso del UKF para identificar los parámetros de un sistema de primer orden. El ejemplo está sacado de [112].

Listing A.3: Ejemplo de identificación con KF

```
1  %% Example of system identification using filters
2  % This example shows the capabilites of the unscented Kalman Filter (UKF)
3  % for simultaneus state and parameter estimation.
4  % This example was taken from the 2018 paper "Online Features in the MATLAB
5  % System Identification Toolbox" by Lennart Ljung et al.
6  % DOI: 10.1016/j.ifacol.2018.09.201
7  %
8  % This example considers a first order discrete-time system given by
9  % y(z) = 25*z^-1 / (1-0.6*z^-1) u(z) + e(z)
10 % where y(z) is the output, u(z) is the input and e(z) is the measurement
11 % noise.
12 %
13 % The filter has to estimate the current state of the system (its
14 % output) and identify the 2 system parameters: gain (b = 25) and its
15 % discrete pole (f = -0.6). For this purpose, the filter will process an
16 % augmented state vector given by
17 % xAug = [z, f, b]'
```

```
18 % where z is the true system output without noise
19
20 clc
21 clear
22 close
23
24 %% Create the system
25 m = tf([0 25], [1 -0.6], 1);
26
27 %% Create a random input
28 nSamples = 10;
29 % u = idinput(nSamples, 'PRBS', [0, 0.5], [-1,1]);
30 u = randn(nSamples,1);
31
32 %% Simulate the system responde adding measurement noise
33 e1 = 1; % Measurement noise covariance
34 y = lsim(m,u,1:nSamples)';
35 y_noise = y + sqrt(e1)*randn(1,length(u));
36
37 %% Define the state transtion function with parameters as states
38 f = @(x,u) [-x(1)*x(2) + x(3)*u; ... % System dynamics as difference eq.
39             x(2); ... % Pole (constant)
40             x(3)]; % Gain (constant)
41
42 %% Create the measurement function handle
43 h = @(x) x(1); % We measure the first state component (the true state)
44
45 %% Initial state guess and covariance
46 initialState = [0;0;0];
47 initialCov = diag([1e1 5 1e3]);
48
49 %% Create the filter object
50 O = 1; % Measurement vector length
51
52 myFilter = AdaptiveUnscentedKalmanFilter(f, h, ...
53     initialState,initialCov,O);
54
55 myFilter.Q = diag([1e-6,0,0]); % Process noise covariance
56 myFilter.R = 1; % Measurement noise covariance guess
57
58 %% Initialize matrices for the simulation
59 xHistory = zeros(3,length(y_noise));
60 xHistory(:,1) = initialState;
61 xCovariance = zeros(3,3, length(y_noise));
62 xCovariance(:,:,1) = initialCov;
63
64 %% Run the UKF loop
65 for kk=1:length(y_noise)-1
66     myFilter.predict(u(kk));
67     [xHistory(:,kk+1), xCovariance(:,:,kk+1)] =
           myFilter.correct(y_noise(kk+1));
68     % xHistory(:,kk+1); xCovariance(:,:,kk+1);
69 end
70
71 %% Comparison of real and estimated values
72 figure
73 subplot(3,1,1)
74 plot(xHistory(1,:))
75 hold on
```

```
76 plot(y, '--')
77 plot(y_noise, ':')
78 ylabel('x_1(k) = y')
79 grid on
80 legend("UKF estimate", "True value", "Measured data")
81
82 subplot(3,1,2)
83 plot(xHistory(2,:))
84 hold on
85 yline(-0.6, Label='f')
86 ylabel('x_2(k) = f')
87 grid on
88
89 subplot(3,1,3)
90 plot(xHistory(3,:))
91 hold on
92 yline(25, Label='b')
93 ylabel('x_3(k) = b')
94 grid on
95
96 %% Evaluate the quality of the estimate
97 ARMSE(xHistory, [y;-0.6*ones(1,length(y));25*ones(1,length(y))], ...
98     [1;-0.6;25])
99 % ARMSE(xHistory, [y;-0.6*ones(1,length(y));25*ones(1,length(y))], [1;1;1])
```

A.2. GENERACIÓN DE ENTRADAS ÓPTIMAS

Para generar las entradas óptimas para identificación expuestas en el apartado 12.1, se desarrolló una función de MATLAB que aplica las expresiones comentadas en la anterior sección de forma sistemática. El código de esta función se incluye en el listado A.4.

Listing A.4: Función para generar entradas multiseno para identificación

```
1  function [u, selectedFrequencies, rpfOpt] = generateMultisineInput(T, ...
2                                       Ts, freqRange, numSinusoids, amplitude)
3  % generateMultisineInput generates a multisine input signal
4  % for system identification. All the equations are taken from
5  % the paper "Practical Aspects of Multiple-Input Design for
6  % Aircraft System Identification Flight Tests" by Eugene A. Morelli,
7  % DOI: 10.2514/6.2021-2795
8  %
9  % Inputs:
10 %   T                  - Total time duration
11 %   Ts                 - Sampling time
12 %   freqRange          - Frequency range [minFreq, maxFreq]
13 %   numSinusoids       - Number of sinusoids to use
14 %   amplitude          - Desired amplitude of the signal
15 %
16 % Outputs:
17 %   u                  - Generated multisine input signal
18 %   selectedFrequencies - Frequencies selected within the specified range
19 %   rpfOpt             - Optimized relative peak factor of the signal
20
21 channels = size(freqRange,1);
22 timeVector = 0:Ts:T;
23 numSamples = length(timeVector) - 1;
24
```

```
25 % Frequency vector for FFT
26 freqVector = (0:numSamples-1) / (Ts * numSamples);
27
28 %% Select frequencies within the desired range
29 validIndices = find(freqVector >= freqRange(:,1) ...
30                     & freqVector <= freqRange(:,2));
31 validFrequencies = freqVector(validIndices);
32
33 if length(validFrequencies) < numSinusoids
34     error('Not enough frequencies within the specified range.');
35 end
36
37 indices = round(linspace(1, length(validFrequencies), numSinusoids));
38
39 % Select the frequencies corresponding to these indices
40 selectedFrequencies = validFrequencies(indices);
41
42 %% Optimize Relative Peak Factor by tuning the phases
43 if exist('rpfOptim.mat', 'file')
44     freqFile = load('rpfOptim.mat','selectedFrequencies');
45 else
46     freqFile.selectedFrequencies = 0;
47 end
48
49 if isequal(freqFile.selectedFrequencies,selectedFrequencies)
50     load('rpfOptim.mat', 'rpfOpt', 'phasesOpt');
51 else
52     warning('Running phase-optimization process')
53     minFun = @(phases) rpf(phases, selectedFrequencies, Ts, T);
54
55     options = optimoptions("ga","FunctionTolerance", eps(10), ...
56         "PopulationSize", 1000, "Display", "none", ...
57         "MaxGenerations", 150, "MaxStallGenerations", 12);
58     [phasesOpt, rpfOpt] = ga(minFun, numSinusoids, [], [], [], [], ...
59         -pi*ones(numSinusoids,1)', pi*ones(numSinusoids,1)',[],[], ...
60         options);
61     save('rpfOptim.mat',"rpfOpt","phasesOpt","selectedFrequencies");
62 end
63
64 % Generate initial input
65 u = zeros(size(timeVector));
66 for i = 1:length(phasesOpt)
67     u = u + sin(2*pi*selectedFrequencies(i)*timeVector + phasesOpt(i));
68 end
69
70 %% Time shift so that the input starts a 0 value
71 % Find first zero crossing with interpolation
72 idx = find(u(1:end-1).*u(2:end) <= 0, 1, 'first');
73 if isempty(idx)
74     error('No zero crossing found in the generated signal.');
75 end
76 tZero = timeVector(idx) - ...
77     u(idx)*(timeVector(idx+1)-timeVector(idx))/(u(idx+1)-u(idx));
78
79 % Compute phase shift corresponding to time shift
80 timeShift = tZero;
81 phaseShift = 2*pi*selectedFrequencies*timeShift;
82
83 % Recalculate input with phase shift
```

```
84  u = zeros(size(timeVector));
85  for i = 1:length(phasesOpt)
86      u = u + sin(2*pi*selectedFrequencies(i)*timeVector ...
87          + phasesOpt(i) + phaseShift(i));
88  end
89
90  %% Adjust amplitude
91  u = u/max(abs(u))*amplitude;
92
93  end
94
95  %% Function to calculate the RPF
96  function rpfVal = rpf(phases, freq, Ts, T)
97  u = 0;
98  timeVector = 0:Ts:T;
99
100 for i = 1:length(phases)
101     u = u + sin(2*pi*freq(i)*timeVector + phases(i));
102 end
103
104 rpfVal = (max(u)-min(u))/(2*sqrt(2)*rms(u));
105 end
```

A.3. MODELO DEL AVIÓN

El modelo del avión se programó en forma de función `xdot`, que calcula la derivada del vector de estado respecto del tiempo, y la función de salida o medida. Además, se creó una función para generar una estructura con los parámetros del modelo, así como funciones auxiliares para el cálculo de coeficientes aerodinámicos. Los códigos de estas funciones se muestran a continuación.

Listing A.5: Función para inicializar el vector de estado ($\dot{\mathbf{x}}$, `xDotVector`) de 6 grados de libertad.

```
1   function [params, state] = loadStateAndParameters_6dof(xAugmented)
2   arguments (Input)
3       xAugmented (:,:) double = []
4   end
5
6   arguments (Output)
7       params (1,1) struct
8       state (1,1) struct
9   end
10
11  %% Load Aircraft parameters (slows down the execution)
12  % run("AircraftParameters.m"); % Very slow
13  % load("AircraftParameters.mat","params"); % Too slow
14
15  %%
16  params.Ts = 5e-2; % [s] Sample time
17
18  %% Geometrical parameters
19  params.b = 2.02; % [m] Wing span
20  params.c = 0.207; % [m] Mean aerodynamic chord
21  params.S = 0.4242; % [m^2] Reference wing surface
22  params.zT = -0.009; % [m] Vertical distance of the thrust line of action from
        the CG
23  params.alphaT = 0; % [rad] Angle of Thrust line of action and the logitudinal
        body axis
```

```
24
25 %% Inertial parameters
26 params.m = 2.107; % [kg] Mass
27 params.Ix = 0.2770; % [kg.m^2] Moment of inertia around X axis
28 params.Iy = 0.2387; % [kg.m^2] Moment of inertia around Y axis
29 params.Iz = 0.5121; % [kg.m^2] Moment of inertia around Z axis
30 params.Ixz = -0.0051; % [kg.m^2] Product of inertia
31 params.g = 9.80665; % Gravity acceleration
32
33 %% Atmospheric conditions
34 params.rho = 1.225; % [kg/m3]
35
36 %% Thrust model
37 params.T0 = 11.8366;
38 params.Ta = -0.05; % a*V
39 params.Tb = -0.014; % b*V^2
40
41 %% Trim conditions
42 params.Vtrim = 10; % [m/s] Reference airspeed
43 params.alpha0 = 0.076992979948935;
44 params.beta0 = 0;
45 params.p0 = 0;
46 params.q0 = 0;
47 params.r0 = 0;
48 params.phi0 = 0;
49 params.theta0 = 0.076992979948935;
50 params.psi0 = 0;
51 params.x0 = 0;
52 params.y0 = 0;
53 params.h0 = 50;
54
55 params.xState0 = [params.Vtrim;params.alpha0;params.beta0; ...
56     params.p0;params.q0;params.r0;...
57     params.phi0;params.theta0;params.psi0;...
58     params.x0;params.y0;params.h0];
59
60 params.de0 = 0;
61 params.dp0 = 0.312439167994252;
62 params.df0 = 0;
63 params.it0 = 0.003263297103875;
64 params.dr0 = 0;
65 params.da0 = 0;
66
67 params.u0 = [params.de0; params.dp0; params.df0; ...
68     params.it0; params.dr0; params.da0];
69
70 %% Sensor noise
71 params.measurementNoise = 1*diag([2, ... % [m/s]
72     deg2rad(0.3), ... % [rad/s]
73     deg2rad(0.3), ... % [rad/s]
74     deg2rad(0.3), ... % [rad/s]
75     deg2rad(1), ... % [rad]
76     deg2rad(1), ... % [rad]
77     deg2rad(1), ... % [rad]
78     1, ... % [m]
79     1, ... % [m]
80     1, ... % [m]
81     0.08, ... % [m/s2]
82     0.08, ... % [m/s2]
```

```
83      0.08].^2); % [m/s2] Measurement noise covariance matrix
84
85  % params.measurementNoise = [0, ... % [m/s]
86  %     deg2rad(0), ... % [rad]
87  %     deg2rad(0), ... % [rad]
88  %     deg2rad(0), ... % [rad/s]
89  %     deg2rad(0), ... % [rad/s]
90  %     deg2rad(0), ... % [rad/s]
91  %     deg2rad(0), ... % [rad]
92  %     deg2rad(0), ... % [rad]
93  %     deg2rad(0), ... % [rad]
94  %     0, ... % [m]
95  %     0, ... % [m]
96  %     0, ... % [m]
97  %     0, ... % [m/s2]
98  %     0, ... % [m/s2]
99  %     0]; % [m/s2] Std dev of measurement noise
100
101 %% Process noise
102 params.processNoise = diag([0.015, ... % [m/s]
103     deg2rad(0.0025), ... % [rad]
104     deg2rad(0.0025), ... % [rad]
105     deg2rad(0.0025), ... % [rad/s]
106     deg2rad(0.0025), ... % [rad/s]
107     deg2rad(0.0025), ... % [rad/s]
108     deg2rad(0.0025), ... % [rad]
109     deg2rad(0.0025), ... % [rad]
110     deg2rad(0.0025), ... % [rad]
111     0.0005, ... % [m]
112     0.0005, ... % [m]
113     0.0005].^2); % [m] Process noise covariance
114
115 %%
116 params.kp_long = [1000,1000,100,1000,...
117     100,100,10,10,100,...
118     1000,100,10,10,10];
119 % params.kp_long = [1,1,1,1,...
120 %     1,1,1,1,1,...
121 %     1,1,1,1,1];
122
123 %% Aerodynamic derivatives in stability axes
124 % Drag coefficient derivatives
125 params.CD0 = 0.0334*params.kp_long(1);
126 params.CDAlpha = 0.0641*params.kp_long(2);
127 params.CDAlpha2 = 1.1023*params.kp_long(3);
128 params.CDAlphaDot = 0;
129 params.CDq = 0;
130 params.CDV = -0.0228*params.kp_long(4);
131 params.CDde = 0;
132 params.CDdf = 0;
133 params.CDit = 0;
134
135 % Lift coefficient derivatives
136 params.CL0 = 0.3811*params.kp_long(5);
137 params.CLAlpha = 5.3057*params.kp_long(6);
138 params.CLAlphaDot = 0.9559*params.kp_long(7);
139 params.CLq = 7.2904*params.kp_long(8);
140 params.CLV = 0.1595*params.kp_long(9);
141 params.CLde = 0;
```

```
142 params.CLdf = 0;
143 params.CLit = 0;
144
145 % Side force coefficient derivatives
146 params.CYbeta = -0.2739;
147 params.CYbetaDot = 0;
148 params.CYp = -0.0742;
149 params.CYr = 0.1883;
150
151 % Pitch moment coefficient derivatives
152 params.Cm0 = 0.0348*params.kp_long(10);
153 params.CmAlpha = -0.3518*params.kp_long(11);
154 params.CmAlphaDot = -3.4858*params.kp_long(12);
155 params.Cmq = -14.2010*params.kp_long(13);
156 params.CmV = 0;
157 params.Cmde = 0; % -1.5047;
158 params.Cmdf = 0; % 0.1195;
159 params.Cmit = 0; % -1.7496;
160
161 % Roll moment coefficient derivatives
162 params.Clbeta = -0.0644;
163 params.ClbetaDot = 0;
164 params.Clp = -0.6244;
165 params.Clr = 0.2516;
166 params.Clda = 0; % -0.3447;
167 params.Cldr = 0; % 0.0005;
168
169 % Yaw moment coefficient derivatives
170 params.Cnbeta = 0.0475;
171 params.CnbetaDot = 0;
172 params.Cnp = -0.1484;
173 params.Cnr = -0.0618;
174 params.Cnda = 0; % -0.0031;
175 params.Cndr = 0; % -0.0535;
176
177 %% Control derivatives in body-fixed axes
178 params.CXde = -0.0231;
179 params.CXdf = -0.0590;
180 params.CXit = -0.0277;
181
182 params.CYda = -0.0418;
183 params.CYdr =  0.1100;
184
185 params.CZde = -0.4153;
186 params.CZdf = -1.3365;
187 params.CZit = -0.4974;
188
189 params.Clda_b = -0.3447;
190 params.Cldr_b =  0.0005;
191
192 params.Cmde_b = -1.5047*params.kp_long(14);
193 params.Cmdf_b =  0.1195;
194 params.Cmit_b = -1.7496;
195
196 params.Cnda_b = -0.0031;
197 params.Cndr_b = -0.0535;
198
199 %% Model limits
200 %% Limits
```

```
201 % Input limits
202 params.uMin = [deg2rad(-8), 0, ...
203     deg2rad(0)-1e-7, params.it0-1e-7, ...
204     deg2rad(-8), deg2rad(-8)]';
205 params.uMax = [deg2rad( 8), 1, ...
206     deg2rad(0)+1e-7, params.it0+1e-7, ...
207     deg2rad( 8), deg2rad( 8)]';
208
209 % State limits
210 params.xMin = [8, deg2rad(-4), deg2rad(-1), ...
211        deg2rad(-40),deg2rad(-30),deg2rad(-40),...
212        deg2rad(-40),deg2rad(-30),deg2rad(-360),...
213        -10000,-10000,10];
214 params.xMax = [25, deg2rad(8), deg2rad(1), ...
215        deg2rad(40),deg2rad(30),deg2rad(40),...
216        deg2rad(40),deg2rad(30),deg2rad(360),...
217        10000,10000,150];
218
219 if ~isempty(xAugmented)
220     %% State variables
221     state.V = xAugmented(1,:);
222     state.alpha = xAugmented(2,:);
223     state.beta = xAugmented(3,:);
224     state.p = xAugmented(4,:);
225     state.q = xAugmented(5,:);
226     state.r = xAugmented(6,:);
227     state.phi = xAugmented(7,:);
228     state.theta = xAugmented(8,:);
229     state.psi = xAugmented(9,:);
230     state.x = xAugmented(10,:);
231     state.y = xAugmented(11,:);
232     state.h = xAugmented(12,:);
233
234     %% Aerodynamic derivatives
235     %% Drag coefficient
236     params.CD0 = xAugmented(13,:);
237     params.CDAlpha = xAugmented(14,:);
238     params.CDAlpha2 = xAugmented(15,:); % 0;
239     params.CDV = xAugmented(16,:);
240     %% Lift coefficient
241     params.CL0 = xAugmented(17,:);
242     params.CLAlpha = xAugmented(18,:);
243     % params.CLAlphaDot = 0;
244     params.CLq = xAugmented(19,:);
245     params.CLV = xAugmented(20,:);
246
247     %% Pitching moment coefficient
248     params.Cm0 = xAugmented(21,:);
249     params.CmAlpha = xAugmented(22,:);
250     params.CmAlphaDot = xAugmented(23,:);
251     % params.CmV = 0; %x(15,:);
252     params.Cmq = xAugmented(24,:);
253     params.Cmde_b = xAugmented(25,:);
254
255 else
256     state.V = params.Vtrim;
257     state.alpha = params.alpha0;
258     state.beta = params.beta0;
259     state.p = params.p0;
```

```
260      state.q = params.q0;
261      state.r = params.r0;
262      state.phi = params.phi0;
263      state.theta = params.theta0;
264      state.psi = params.psi0;
265      state.x = params.x0;
266      state.y = params.y0;
267      state.h = params.h0;
268  end
269  end
```

Listing A.6: Función `xdot` de 6 grados de libertad.

```
1  function [xDotVector, Cl, Cm, Cn, CL, CD, CY] = xdot_6dof(xVector, u, prms)
2  % xdot_6dof Calculate the time derivative of the aircraft's state vector
3  %
4  % This functions returns the derivative of the state vector xDot given the
5  % current state vector xVector, the current input vector u and the aircraft
6  % parameter structure
7  %
8  % Inputs:
9  %   xVector     - Current state vector of the aircraft (12x1)
10 %   u           - Current input vector (current actuator positions) (6x1)
11 %   prms        - Structure containing the aircraft parameters (1x1)
12 %
13 % Output:
14 %   xDotVector  - State derivative vector (12x1)
15 %
16
17 % Juan Villar Fernandez
18 % 06/11/2024
19
20 %% Convert state vector to variables
21 V       = xVector(1);
22 alpha   = xVector(2);
23 beta    = xVector(3);
24 p       = xVector(4);
25 q       = xVector(5);
26 r       = xVector(6);
27 phi     = xVector(7);
28 theta   = xVector(8);
29 psi     = xVector(9);
30 x       = xVector(10);
31 y       = xVector(11);
32 h       = xVector(12);
33
34 % Convert input vector to variables
35 de = u(1);
36 dp = u(2);
37 df = u(3);
38 it = u(4);
39 da = u(5);
40 dr = u(6);
41
42 %% Aerodynamic force coefficients
43 [CY,CD,CL,CDw,CYw] = calculateForceCoefficients(xVector, u, prms);
44
45 %% Dynamic pressure
46 qbar = 0.5*prms.rho*V^2;
```

```
47
48  %% Thrust model
49  TS=12.089/(1+exp(-9.721*dp+4.022))-0.212;
50  % T=TS+1-exp(0.146*V);
51  % TS = prms.T0*dp;
52  T = TS + prms.Ta*V + prms.Tb*V^2;
53
54  if (isa(V,'double') && T < 0)
55      T = 0;
56  end
57
58  Mt = T*prms.zT;
59
60  %% Equations of motion (6 DoF)
61  % Aerodynamic variables
62  Vdot = - qbar*prms.S/prms.m*CDw + T/prms.m*cos(alpha)*cos(beta) ...
63          + prms.g*(cos(phi)*cos(theta)*sin(alpha)*cos(beta) ...
64          + sin(phi)*cos(theta)*sin(beta) - sin(theta)*cos(alpha)*cos(beta));
65
66  K1 = - qbar*prms.S/(prms.m*V*cos(beta));
67  alphaDot = 1/(1 - K1*prms.CLAlphaDot/prms.kp_long(7)*prms.c/(2*prms.Vtrim)) *
        (K1*CL ...
68              + q - tan(beta)*(p*cos(alpha)+r*sin(alpha)) ...
69              - T/(prms.m*V*cos(beta))*sin(alpha) + prms.g/(V*cos(beta)) ...
70              *(cos(phi)*cos(theta)*cos(alpha) + sin(theta)*sin(alpha)));
71
72  betaDot = qbar*prms.S/(prms.m*V)*CYw + p*sin(alpha) - r*cos(alpha) ...
73            + prms.g/V*cos(beta)*sin(phi)*cos(theta) ...
74            + sin(beta)/V*(prms.g*cos(alpha)*sin(theta) -
                  prms.g*sin(alpha)*cos(phi)*cos(theta) ...
75            + T*cos(alpha)/prms.m);
76
77  %% Aerodynamic moment coefficients
78  [Cl,Cm,Cn] = calculateMomentCoefficients(xVector, u, prms, alphaDot, betaDot);
79
80  %% Moment equations
81  Ix = prms.Ix;
82  Iy = prms.Iy;
83  Iz = prms.Iz;
84  Ixz = prms.Ixz;
85
86  Gamma=Ix*Iz - Ixz^2;
87  c1 = ((Iy - Iz)*Iz - Ixz^2)/Gamma;
88  c2 = ((Ix - Iy + Iz)*Ixz)/Gamma;
89  c3 = Iz/Gamma;
90  c4 = Ixz/Gamma;
91  c5 = (Iz - Ix)/Iy;
92  c6 = Ixz/Iy;
93  c7 = 1/Iy;
94  c8 = ((Ix - Iy)*Ix - Ixz^2)/Gamma;
95  c9 = Ix/Gamma;
96
97  pDot = (c1*r + c2*p)*q + qbar*prms.S*prms.b*(c3*Cl + c4*Cn);
98  qDot = c5*p*r - c6*(p^2 - r^2) + c7*(qbar*prms.S*prms.c*Cm+Mt);
99  rDot = (c8*p - c2*r)*q + qbar*prms.S*prms.b*(c9*Cn + c4*Cl);
100
101 %% Angular rate equations
102 phiDot = p + tan(theta)*(q*sin(phi) + r*cos(phi));
103 thetaDot = q*cos(phi) - r*sin(phi);
```

```
104 psiDot = (q*sin(phi) + r*cos(phi))/cos(theta);
105
106 %% Navigation equations
107 xDot = V*cos(alpha)*cos(beta)*cos(psi)*cos(theta) + V*sin(beta)*( ...
108         cos(psi)*sin(theta)*sin(phi) - sin(psi)*cos(phi)) ...
109         + V*sin(alpha)*cos(beta)*(cos(psi)*sin(theta)*cos(phi) +
                sin(psi)*sin(phi));
110
111 yDot = V*cos(alpha)*cos(beta)*sin(psi)*cos(theta) + V*sin(beta)*( ...
112         sin(psi)*sin(theta)*sin(phi) - cos(psi)*cos(phi)) ...
113         + V*sin(alpha)*cos(beta)*(sin(psi)*sin(theta)*sin(phi) -
                cos(psi)*sin(phi));
114
115 hDot = V*cos(alpha)*cos(beta)*sin(theta) - V*sin(beta)*cos(theta)*sin(phi) ...
116        - V*sin(alpha)*cos(beta)*cos(theta)*cos(phi);
117
118 %% Output xdot vector
119 xDotVector = [Vdot; alphaDot; betaDot; pDot; qDot; rDot; phiDot; thetaDot; ...
120               psiDot; xDot; yDot; hDot];
121
122 end
```

Listing A.7: Cálculo de coeficientes de fuerza.

```
1  function [CY,CD,CL,CDw,CYw] = calculateForceCoefficients(xVector, u, prms)
2  % calculateForceCoefficients Calculate the aerodynamic force coefficients
3
4  % This functions returns the aerodynamic coefficients of forces acting on
5  % the aircraft in the stability and wind frames given the current state
6  % and input vectors and the parameter structure
7  %
8  % Inputs:
9  %   xVector - Current state vector of the aircraft (12x1)
10 %   u       - Current input vector (current actuator positions) (6x1)
11 %   prms    - Structure containing the aircraft parameters (1x1)
12 %
13 % Outputs:
14 %   CY      - Sideforce coefficient (stability axes)
15 %   CD      - Drag force coefficient (stability axes)
16 %   CL      - Lift force coefficient (stability axes)
17 %   CDw     - Drag force coefficient (wind axes)
18 %   CYw     - Sideforce coefficient (wind axes)
19 %
20
21 % Juan Villar Fernandez
22
23 %% Convert state vector to variables
24 V       = xVector(1);
25 alpha   = xVector(2);
26 beta    = xVector(3);
27 p       = xVector(4);
28 q       = xVector(5);
29 r       = xVector(6);
30 phi     = xVector(7);
31 theta   = xVector(8);
32 psi     = xVector(9);
33 x       = xVector(10);
34 y       = xVector(11);
35 h       = xVector(12);
```

```
36
37 %% Convert input vector to variables
38 de = u(1);
39 dp = u(2);
40 df = u(3);
41 it = u(4);
42 da = u(5);
43 dr = u(6);
44
45 %% Calculate angular rates in stability axes
46 Mbs = [cos(alpha), 0, -sin(alpha);
47         0          1,           0;
48        sin(alpha), 0, cos(alpha)];
49 Msb = Mbs';
50 omega_s = Msb*[p;q;r];
51 ps = omega_s(1);
52 qs = omega_s(2);
53 rs = omega_s(3);
54
55 %% Controls in body axes
56 CX = prms.CXde*de + prms.CXdf*df + prms.CXit*it;
57 CY = prms.CYda*da + prms.CYdr*dr;
58 CZ = prms.CZde*de + prms.CZdf*df + prms.CZit*it;
59
60 % Convert to stability axes
61 CF_s = Msb*[CX;CY;CZ];
62 CD_s = -CF_s(1);
63 CY_s = CF_s(2);
64 CL_s = -CF_s(3);
65
66 % Add terms in stability axes
67 CD = prms.CD0/prms.kp_long(1) ...
68      + prms.CDAlpha/prms.kp_long(2)*alpha ...
69      + prms.CDAlpha2/prms.kp_long(3)*alpha^2 ...
70      + prms.CDV/prms.kp_long(4)*(V-prms.Vtrim)/prms.Vtrim ...
71      + prms.CDq*prms.c/(2*prms.Vtrim)*qs ...
72      + prms.CDde*de ...
73      + prms.CDdf*df ...
74      + prms.CDit*it + CD_s;
75
76 CL = prms.CL0/prms.kp_long(5) ...
77      + prms.CLAlpha/prms.kp_long(6)*alpha ...
78      + prms.CLV/prms.kp_long(9)*(V-prms.Vtrim)/prms.Vtrim ...
79      + prms.CLq/prms.kp_long(8)*prms.c/(2*prms.Vtrim)*qs ...
80      + prms.CLde*de ...
81      + prms.CLdf*df ...
82      + prms.CLit*it + CL_s;
83
84 CY = prms.CYbeta*beta ...
85      + prms.CYp*prms.b/(2*prms.Vtrim)*ps ...
86      + prms.CYr*prms.b/(2*prms.Vtrim)*rs + CY_s;
87
88 %% Convert to wind axes
89 CDw = CD*cos(beta) - CY*sin(beta);
90 CYw = CY*cos(beta) + CD*sin(beta);
91 end
```

Listing A.8: Cálculo de coeficientes de momento.

```
function [Cl,Cm,Cn] = calculateMomentCoefficients(xVector, u, prms, alphaDot,
    betaDot)
% calculateMomentCoefficients Calculate the aerodynamic moment coefficients

% This functions returns the aerodynamic coefficients of moments acting on
% the aircraft in the stability frame given the current state and input
% vectors, the parameter structure and the values of alphaDot and betaDot
%
% Inputs:
%   xVector     - Current state vector of the aircraft (12x1)
%   u           - Current input vector (current actuator positions) (6x1)
%   prms        - Structure containing the aircraft parameters (1x1)
%   alphaDot    - Time derivative of the angle of attack (1x1)
%   betaDot     - Time derivative of the sideslip angle (1x1)
%
% Outputs:
%   Cl      - Rolling coefficient
%   Cm      - Pitching coefficient
%   Cn      - Yawing coefficient
%

% Juan Villar Fernandez

%% Convert state vector to variables
V       = xVector(1);
alpha   = xVector(2);
beta    = xVector(3);
p       = xVector(4);
q       = xVector(5);
r       = xVector(6);
phi     = xVector(7);
theta   = xVector(8);
psi     = xVector(9);
x       = xVector(10);
y       = xVector(11);
h       = xVector(12);

%% Convert input vector to variables
de = u(1);
dp = u(2);
df = u(3);
it = u(4);
da = u(5);
dr = u(6);

%% Calculate angular rates in stability axes
Mbs = [cos(alpha), 0, -sin(alpha);
         0          1,           0;
        sin(alpha), 0, cos(alpha)];
Msb = Mbs';
omega_s = Msb*[p;q;r];
ps = omega_s(1);
qs = omega_s(2);
rs = omega_s(3);

%% Controls in body-fixed axes
Cl_b = prms.Clda_b*da + prms.Cldr_b*dr;
Cm_b = prms.Cmde_b/prms.kp_long(14)*de + prms.Cmdf_b*df + prms.Cmit_b*it;
Cn_b = prms.Cnda_b*da + prms.Cndr_b*dr;
```

```
59
60 %% Convert to stability axes
61 M_s = Msb*[Cl_b;Cm_b;Cn_b];
62 Cl_s = M_s(1);
63 Cm_s = M_s(2);
64 Cn_s = M_s(3);
65
66 %% Add terms in stability axes
67 Cl = prms.Clbeta*beta ...
68      + prms.ClbetaDot*prms.b/(2*V)*betaDot ...
69      + prms.Clp*prms.b/(2*prms.Vtrim)*ps ...
70      + prms.Clr*prms.b/(2*prms.Vtrim)*rs ...
71      + prms.Clda*da ...
72      + prms.Cldr*dr + Cl_s;
73
74 Cm = prms.Cm0/prms.kp_long(10) ...
75      + prms.CmV*(V-prms.Vtrim)/prms.Vtrim ...
76      + prms.CmAlpha/prms.kp_long(11)*alpha ...
77      + prms.CmAlphaDot/prms.kp_long(12)*prms.c/(2*prms.Vtrim)*alphaDot ...
78      + prms.Cmq/prms.kp_long(13)*qs*prms.c/(2*prms.Vtrim) ...
79      + prms.Cmde*de ...
80      + prms.Cmdf*df ...
81      + prms.Cmit*it + Cm_s;
82
83 Cn = prms.Cnbeta*beta ...
84      + prms.CnbetaDot*prms.b/(2*prms.Vtrim)*betaDot ...
85      + prms.Cnp*prms.b/(2*prms.Vtrim)*ps ...
86      + prms.Cnr*prms.b/(2*prms.Vtrim)*rs ...
87      + prms.Cnda*da ...
88      + prms.Cndr*dr + Cn_s;
89 end
```

Listing A.9: Función de medida.

```
1  function [yVector] = y_6dof(xVector, u, prms)
2  % y_6dof Calculate the measurement vector
3  %
4  % This functions returns the measurement vector given the current state
5  % vector xVector, the current input vector u and the aircraft
6  % parameter structure
7  %
8  % Inputs:
9  %   xVector     - Current state vector of the aircraft (12x1)
10 %   u           - Current input vector (current actuator positions) (6x1)
11 %   prms        - Structure containing the aircraft parameters (1x1)
12 %
13 % Output:
14 %   yVector  - State derivative vector (13x1)
15 %
16
17 % Juan Villar Fernandez
18 % 06/11/2024
19
20 %% Convert state vector to variables
21 V       = xVector(1);
22 alpha   = xVector(2);
23 beta    = xVector(3);
24 p       = xVector(4);
25 q       = xVector(5);
```

```
26 r       = xVector(6);
27 phi     = xVector(7);
28 theta   = xVector(8);
29 psi     = xVector(9);
30 x       = xVector(10);
31 y       = xVector(11);
32 h       = xVector(12);
33
34 % Convert input vector to variables
35 de = u(1);
36 dp = u(2);
37 df = u(3);
38 it = u(4);
39 da = u(5);
40 dr = u(6);
41
42 % Calculate angular rates in stability axes
43 Mbs = [cos(alpha), 0, -sin(alpha);
44         0          1,           0;
45        sin(alpha), 0, cos(alpha)];
46 Msb = Mbs';
47 omega_s = Msb*[p;q;r];
48 ps = omega_s(1);
49 qs = omega_s(2);
50 rs = omega_s(3);
51
52 %% Aerodynamic force coefficients
53 [CY,CD,CL,CDw,CYw] = calculateForceCoefficients(xVector, u, prms);
54
55 %% Dynamic pressure
56 qbar = 0.5*prms.rho*V^2;
57
58 %% Thrust model
59 TS=12.089/(1+exp(-9.721*dp+4.022))-0.212;
60 % T=TS+1-exp(0.146*V);
61 % TS = prms.T0*dp;
62 T = TS + prms.Ta*V + prms.Tb*V^2;
63
64 if (isa(V,'double') && T < 0)
65     T = 0;
66 end
67
68 %% Add CLalphaDot term
69 K1 = - qbar*prms.S/(prms.m*V*cos(beta));
70 alphaDot = 1/(1 - K1*prms.CLAlphaDot/prms.kp_long(7)*prms.c/(2*prms.Vtrim)) *
       (K1*CL ...
71             + q - tan(beta)*(p*cos(alpha)+r*sin(alpha)) ...
72             - T/(prms.m*V*cos(beta))*sin(alpha) + prms.g/(V*cos(beta)) ...
73             *(cos(phi)*cos(theta)*cos(alpha) + sin(theta)*sin(alpha)));
74
75 CL = CL + prms.CLAlphaDot/prms.kp_long(7)*alphaDot;
76
77 % Convert back to body-fixed axes
78 C_Fb = Mbs*[-CD;0;-CL];
79 CX = C_Fb(1);
80 CZ = C_Fb(3);
81
82 %% Accelerations in body-fixed axes
83 ax = (qbar*prms.S*CX + T)/prms.m;
```

```
84  ay = qbar*prms.S*CY/prms.m;
85  az = qbar*prms.S*CZ/prms.m;
86
87  %% Output vector
88  yVector = [V; p; q; r; phi; theta; psi; x; y; h; ax; ay; az];
89  end
```

A.4. SIMULADOR

Para la identificación del avión, primero se debe simular una maniobra en lazo abierto aplicando una entrada adecuada. Para simular obtener la respuesta se puede recurrir a la función del listado A.10, que devuelve la respuesta completa del sistema y un vector de tiempos, todo ello en forma de estructura. La estructura en realidad es un objeto de la clase AircraftFlightDataStruct, cuyo código se puede ver en el listado A.11.

Listing A.10: Simulación en lazo abierto.

```
1   function [simulationData] = simulateOpenLoopAircraftResponse2(Tssim, Tend, ...
2       uCmdHist, prms, initState)
3   %simulateOpenLoopAircraftResponse Run an open-loop aircraft simulation
4   %   This function receives as inputs the simulation timestep, the
5   %   simulation final time, a sequence of input commands, the aircraft
6   %   parameter structure and optionally the aircraft's initial state
7
8   if (nargin < 5)
9       initState = prms.xState0;
10  end
11
12  %% Prepare simulation data
13  time = 0:Tssim:Tend;
14
15  nx = length(initState);
16  xHist = zeros(nx, length(time));
17  xHist(:, 1) = initState;
18
19  nu = size(uCmdHist, 1);
20  uHist = zeros(nu, length(time));
21  uHist(:, 1) = uCmdHist(:, 1);
22
23  initOutput = y_6dof(initState, uCmdHist(:,1), prms);
24  ny = length(initOutput);
25  yHist = zeros(ny, length(time));
26  yHist(:,1) = initOutput;
27
28  zHist = yHist + sqrt(prms.measurementNoise)*randn(ny, 1);
29
30  %% Prepare sensitivity calculations
31
32  prmsVectorTrue = [prms.CD0 prms.CDAlpha prms.CDAlpha2 prms.CDV ...
33      prms.CL0 prms.CLAlpha prms.CLAlphaDot prms.CLq prms.CLV ...
34      prms.Cm0 prms.CmAlpha prms.CmAlphaDot prms.Cmq prms.Cmde_b];
35
36  % Create symbolic vectors
37  syms xVector [12 1] real
38  syms uVector [6 1] real
39  syms CD0 CDAlpha CDAlpha2 CDV ...
40      CL0 CLAlpha CLAlphaDot CLq CLV ...
```

```
41      Cm0 CmAlpha CmAlphaDot Cmq Cmde_b
42
43  prmsSym = prms;
44  prmsSym.CD0 = CD0;
45  prmsSym.CDAlpha = CDAlpha;
46  prmsSym.CDAlpha2 = CDAlpha2;
47  prmsSym.CDV = CDV;
48  prmsSym.CL0 = CL0;
49  prmsSym.CLAlpha = CLAlpha;
50  prmsSym.CLAlphaDot = CLAlphaDot;
51  prmsSym.CLq = CLq;
52  prmsSym.CLV = CLV;
53  prmsSym.Cm0 = Cm0;
54  prmsSym.CmAlpha = CmAlpha;
55  prmsSym.CmAlphaDot = CmAlphaDot;
56  prmsSym.Cmq = Cmq;
57  prmsSym.Cmde_b = Cmde_b;
58
59  prmsVector = [CD0 CDAlpha CDAlpha2 CDV ...
60      CL0 CLAlpha CLAlphaDot CLq CLV ...
61      Cm0 CmAlpha CmAlphaDot Cmq Cmde_b];
62
63  dfdx = jacobian(xdot_6dof(xVector,uVector,prms), xVector);
64  dfdthetaLong = jacobian(xdot_6dof(xVector,uVector,prmsSym), prmsVector);
65  dhdx = jacobian(y_6dof(xVector,uVector,prms), xVector);
66  dhdthetaLong = jacobian(y_6dof(xVector,uVector,prmsSym), prmsVector);
67
68  dfdx = matlabFunction(dfdx, ...
69      "Vars", {xVector,uVector});
70  dfdthetaLong = matlabFunction(dfdthetaLong, ...
71      "Vars", {xVector,uVector,prmsVector});
72  dhdx = matlabFunction(dhdx, ...
73      "Vars", {xVector,uVector});
74  dhdthetaLong = matlabFunction(dhdthetaLong, ...
75      "Vars", {xVector,uVector,prmsVector});
76
77  %% Run simulation
78  opts = odeset('Reltol',5e-5,'AbsTol',1e-7,'Stats','on');
79  % opts = odeset('Stats','off');
80  [~,sol] = ode45(@(t,x) odefun(t, x, time, uCmdHist, nx, nu, prms, ...
81      dfdx, dfdthetaLong, prmsVectorTrue), time, ...
82      [initState; uCmdHist(:, 1); zeros(nx*length(prmsVectorTrue),1)], opts);
83
84  xHist = sol(:, 1:nx)';
85  uHist = sol(:, nx+1:nx+nu)';
86  SxHist = reshape(sol(:, nx+nu+1:end)', nx, length(prmsVectorTrue), []);
87
88  for i = 1:(length(time) - 1)
89      yHist(:, i+1) = y_6dof(xHist(:,i+1), uHist(:,i+1), prms);
90      zHist(:, i+1) = yHist(:, i+1) + ...
91          sqrt(prms.measurementNoise)*randn(ny, 1);
92      SyHist(:, :, i+1) = dhdx(xHist(:,i+1), uHist(:,i+1))*SxHist(:,:,i) ...
93          + dhdthetaLong(xHist(:,i+1), uHist(:,i+1),prmsVectorTrue);
94  end
95
96  %% Save data to structure
97  simulationData = AircraftFlightDataStruct(time, zHist, uCmdHist, prms);
98
99  simDataStruct = AircraftFlightDataStruct.createSimulationStruct(time, ...
```

```
100     xHist, yHist, uHist, SxHist, SyHist);
101 simulationData.simulationData = simDataStruct;
102 %     xHist, yHist, uHist, SxHist, SyHist);
103
104 [H, Hr, S, Hdr] = simulationData.calculateHessianMatrix(prmsVectorTrue);
105 Ri = sqrt(eig(Hr)) / sqrt(min(eig(Hr)));
106 disp(max(Ri));
107 end
108
109 function [odeVector] = odefun(t, x, timeVector, uCmdHist, nx, nu, prms, ...
110     dfdx, dfdthetaLong, prmsVector)
111
112     uCmd = interp1(timeVector', uCmdHist', t, "previous")';
113
114     xVector = x(1:nx);
115     uVector = x(nx+1:nx+nu);
116     SxMatrix = reshape(x(nx+nu+1:end), nx, []);
117
118     xDotVector = xDotDouble(xVector, uVector, prms);
119     uDotVector = actuator_model(uCmd, xVector, uVector, prms);
120
121     SxDotMatrix = dfdx(xVector, uVector)*SxMatrix ...
122         + dfdthetaLong(xVector, uVector, prmsVector);
123
124     odeVector = [xDotVector; uDotVector; reshape(SxDotMatrix, [], 1)];
125 end
```

Listing A.11: Clase para datos de simulación e identificación.

```
1  classdef AircraftFlightDataStruct < handle
2      % AIRCRAFTIDENTIFICATIONDATASTRUCT Create container for aircraft sim data
3
4      properties
5          % Experimental data
6          experimentData
7          % time         % Time vector
8          % zHist        % Measurement vector history
9          % uCmdHist     % Input command vector history
10
11         % Estimation data
12         estimationData
13         % xHatHist     % Estimated augmented state vector history
14         % xCovHist     % Estimated augmented state covariance history
15         % yHatHist     % Estimated output vector history
16
17         % Simulated response data
18         simulationData
19         % xSimHist     % Simulated state vector history
20         % ySimHist     % Simulated output vector history
21         % uHist        % Simulated input vector history
22
23         % Sensitivities
24         SxHist
25         SyHist
26
27         % Identified model response data
28         yIdHist      % Simulated output vector history
29
30         aircraftParameters % Aircraft parameter structure
```

```
31      end
32
33
34      methods (Static)
35          function expStruct = createExperimentStruct(time, zHist, uCmdHist)
36              arguments
37                  time double = []
38                  zHist double = []
39                  uCmdHist double = []
40              end
41
42              zHistVarNames = {'Vnoise', 'pnoise', 'qnoise', 'rnoise', ...
43                  'phinoise', 'thetanoise', 'psinoise', ...
44                  'xnoise', 'ynoise', 'hnoise', ...
45                  'axnoise', 'aynoise', 'aznoise'}';
46              uCmdVarHistNames = {'decmd', 'dpcmd', 'dfcmd', 'itcmd', ...
47                  'dacmd', 'drcmd'}';
48
49              expStruct = struct('time', [], 'zHist', [], ...
50                  'zHistVarNames', {}, 'uCmdHist', [], ...
51                  'uCmdHistVarNames', {});
52              expStruct(1).time = time;
53              expStruct.zHist = zHist;
54              expStruct.zHistVarNames = zHistVarNames;
55              expStruct.uCmdHist = uCmdHist;
56              expStruct.uCmdHistVarNames = uCmdVarHistNames;
57          end
58
59          function simStruct = createSimulationStruct(time, xHist, yHist, ...
60                  uHist, SxHist, SyHist)
61              arguments
62                  time = []
63                  xHist = []
64                  yHist = []
65                  uHist = []
66                  SxHist = []
67                  SyHist = []
68              end
69
70              xVarNames = {'V', 'alpha', 'beta', 'p', 'q', 'r', ...
71                  'phi', 'theta', 'psi', 'x', 'y', 'h'}';
72              yVarNames = {'ax','ay','az'}';
73              uVarNames = {'de', 'dp', 'df', 'it', ...
74                  'da', 'dr'}';
75
76              simStruct = struct('time', [], ...
77                  'xHist', [], 'xSimVarNames', {}, ...
78                  'yHist', [], 'ySimVarNames', {}, ...
79                  'uHist', [], 'uSimVarNames', {}, ...
80                  'sensitivity', struct());
81              simStruct(1).time = time;
82              simStruct.xHist = xHist;
83              simStruct.xSimVarNames = xVarNames;
84              simStruct.yHist = yHist;
85              simStruct.ySimVarNames = yVarNames;
86              simStruct.uHist = uHist;
87              simStruct.uSimVarNames = uVarNames;
88              for i = 1:length(time)
89                  simStruct.sensitivity(i).SxHist = SxHist(:,:,i);
```

```
90                  simStruct.sensitivity(i).SyHist = SyHist(:,:,i);
91              end
92          end
93
94          function estStruct = createEstimationStruct(time, xHatHist, ...
95                  xCovHist, yHatHist)
96              arguments
97                  time = []
98                  xHatHist = []
99                  xCovHist = []
100                 yHatHist = []
101             end
102
103             estStruct = struct('time', time, 'xHatHist', xHatHist, ...
104                 'xCovHist', xCovHist, 'yHatHist', yHatHist);
105         end
106     end
107
108     methods (Access = public)
109         function obj = AircraftFlightDataStruct(time, zHist, uCmdHist, prms)
110             arguments
111                 time
112                 zHist
113                 uCmdHist
114                 prms
115             end
116
117             obj.experimentData = obj.createExperimentStruct(time, zHist, ...
118                 uCmdHist);
119
120             obj.aircraftParameters = prms;
121         end
122
123         function [] = save2file(obj, filepath)
124
125             expData = obj.experimentData;
126             simData = obj.simulationData;
127             estData = obj.estimationData;
128             % Undo the parameter-scaling correction
129             xHatHist = estData.xHatHist;
130             xHatHistUnscaled = xHatHist;
131             xHatHistUnscaled(13:end,:) = ...
132                 xHatHist(13:end,:) ./ obj.aircraftParameters.kp_long';
133
134             xCovHist = estData.xCovHist;
135             xCovHistUnscaled = xCovHist;
136             xCovHistUnscaled(13:end,:) = ...
137                 xCovHist(13:end,:) ./ obj.aircraftParameters.kp_long.^2';
138
139             %% Variable name cell arrays
140             timeName = {'t'};
141             zHistVarNames = expData.zHistVarNames;
142
143             xHatVarNames = {'Vest', 'alphaest', 'betaest', ...
144                 'pest', 'qest', 'rest', ...
145                 'phiest', 'thetaest', 'psiest', ...
146                 'xest', 'yest', 'hest', ...
147                 'CD0est', 'CDAlphaest', 'CDAlpha2est', 'CDVest', ...
148                 'CL0est', 'CLAlphaest', 'CLAlphaDotest', 'CLqest', 'CLVest',
```

```
                    ...
149                 'Cm0est', 'CmAlphaest', 'CmAlphaDotest', 'Cmqest',
                        'Cmde_best'};
150             xCovVarNames = {'Vcov', 'alphacov', 'betacov', ...
151                 'pcov', 'qcov', 'rcov', ...
152                 'phicov', 'thetacov', 'psicov', ...
153                 'xcov', 'ycov', 'hcov', ...
154                 'CD0cov', 'CDAlphacov', 'CDAlpha2cov', 'CDVcov', ...
155                 'CL0cov', 'CLAlphacov', 'CLAlphaDotcov', 'CLqcov', 'CLVcov',
                        ...
156                 'Cm0cov', 'CmAlphacov', 'CmAlphaDotcov', 'Cmqcov',
                        'Cmde_bcov'};
157             lastName = {'end'};
158
159             T = array2table([obj.experimentData.time', ...
160                 obj.experimentData.zHist', obj.experimentData.uCmdHist', ...
161                 obj.simulationData.xSimHist', obj.ySimHist(11:end,:)',
                        obj.uHist', ...
162                 xHatHistUnscaled', xCovHistUnscaled', ...
163                 zeros(length(obj.time), 1)], ...
164                 "VariableNames", ...
165                 [timeName, zHistVarNames, uCmdHistNames, ...
166                 xSimHistNames, ySimHistNames, uHistNames, ...
167                 xHatVarNames, xCovVarNames, lastName]);
168
169             writetable(T, filepath);
170         end
171
172         function [H, Hr, S, Hdr] = calculateHessianMatrix(obj, prmsVector)
173             H = obj.simulationData.sensitivity(1).SyHist' ...
174                 * obj.simulationData.sensitivity(1).SyHist;
175             for k = 2:length(obj.simulationData.time)
176                 H = H + obj.simulationData.sensitivity(k).SyHist' ...
177                 * obj.simulationData.sensitivity(k).SyHist;
178             end
179             H = H/k;
180
181             L = diag(prmsVector);
182             Hr = L'*H*L;
183             S = sqrt(diag(Hr));
184             [~, T] = eig(Hr);
185
186             Hdr = T'*Hr*T;
187         end
188     end
189 end
```

B. COLOFÓN

Este documento se escribió con el sistema LaTeX, fue compilado el día 11 de junio de 2025 con LuaTeX y se produjo con los siguientes paquetes:

- La clase (*document class*) `memoir` con los pies de página, encabezados y márgenes configurados.
- El paquete `nomencl` para incluir la sección de nomenclatura y gestionar sus entradas de forma automática.
- `babel` para cambiar el idioma del documento y la partición automática de palabras a final de línea.
- `listings` para incluir programas de ordenador (*scripts*) con resaltado automático de palabras clave en el lenguaje de programación.
- `siunitx` para gestionar de forma automática las unidades de cualquier magnitud física, además de alinear los separadores decimales (las comas) en todas las columnas de tablas de la memoria.
- `pdfpages` para incluir documentos PDF en el documento. En particular, para incrustar la portada del trabajo, que se hizo en Word.
- `bookmark` para incluir marcadores de índice (*bookmarks*) en el PDF generado.
- `Tikz` y `PGFPlots` para diseñar todos los diagramas y gráficas de este trabajo, adaptando los originales de las fuentes correspondientes.
- `xcolor` para poder usar colores en los diagramas y gráficas y configurar unos colores comunes en todo el documento.
- `biblatex` para incluir y maquetar de forma automática la bibliografía con todos los campos relevantes.
- La tipografía Latin Modern, basada en Computer Modern, obtenida con el paquete `lmodern`.