Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Modelado e identificación de vehículos aéreos no tripulados comerciales a partir de datos de vuelo sintéticos

Autor: Rafael Adolfo Blanco Villafañe

Tutor: Carlos Bordons Alba

Departamento de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Modelado e identificación de vehículos aéreos no tripulados comerciales a partir de datos de vuelo sintéticos

Autor:
Rafael Adolfo Blanco Villafañe

Tutor:
Carlos Bordons Alba
Catedrático de Universidad

Departamento de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado: Modelado e identificación de vehículos aéreos no tripulados comerciales a partir de datos de vuelo sintéticos

Autor: Rafael Adolfo Blanco Villafañe
Tutor: Carlos Bordons Alba

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A Dios, por permitirme nacer en el lugar indicado, en el momento justo, rodeado de la gente apropiada. Por todo lo que me ha permitido vivir, por lo bueno y por lo no tan bueno. Por la cadena infinita de eventos y casualidades que forjó y que han desembocado en lo que soy el día de hoy.

A mi familia, a los amigos que se convirtieron en familia y a todas esas buenas personas que creyeron en mi, cuando ni yo mismo lo hacía.

A la Universidad Simón Bolívar y a todas las personas que allí conocí, por enseñarme lo que significa la calidad profesional, el honor al mérito y la búsqueda de la excelencia. A a la Universidad de Sevilla, por darme la oportunidad de formarme y finalmente colocar delante de mi nombre el titulo de "ingeniero". Y a mi tutor, por su increíble disposición y su gran atención al detalle, que han permitido que este trabajo sea lo que es hoy.

A todas aquellas personas que se encuentran a más de 7000 km de distancia, en un lugar que alguna vez fue (y espero pueda volver a ser) mi hogar, que hicieron todo lo posible por que yo, Rafael, me encontrase aquí el día de hoy, escribiendo estas líneas.

Y por último, pero no menos importante, a Rafael. Por lograr lo que hace tan solo unos años veía tan lejano. Por creer en el proceso y nunca parar, incluso en los momentos más oscuros. Por resistir. Por crecer tan rápido y finalmente, por haberte ganado tu libertad.

A todos ellos, desde hoy y para siempre,

Gracias.

Resumen

El presente trabajo se centra en el desarrollo de simuladores de vuelo para vehículos aéreos no tripulados (VANTs o UAVs) de ala fija y en la creación de un método de identificación de parámetros aerodinámicos. A través de la implementación de dos modelos de resistencia aerodinámica y tres modelos propulsivos, se logró la creación exitosa de simuladores que ofrecen experiencias realistas de vuelo y permiten al piloto interactuar con la aeronave en tiempo real mediante una interfaz usuario-máquina. Además, se diseñó un algoritmo de identificación de coeficientes aerodinámicos basado en el método de mínimos cuadrados, lo que llevó a la identificación exitosa de parámetros en ciertos escenarios.

Los resultados de este trabajo tienen implicaciones significativas, ya que la utilización de modelos más precisos de resistencia y propulsión permitiría un desarrollo más efectivo de sistemas de control en velocidad y altitud. El modelo de motor BLDC (del inglés *Brushless DC*, o *DC sin escobillas*) presentado junto con el modelo de hélice basado en datos de túnel de viento, también ofrece una base sólida para el análisis del rendimiento de UAVs en diversas condiciones. A pesar de los logros, se identificaron áreas de mejora, como la necesidad de abordar el cálculo de derivadas a partir de señales ruidosas de giroscopios y la exploración de algoritmos más avanzados para la identificación de parámetros.

Este trabajo abre la puerta a futuras investigaciones y mejoras en el simulador presentado. Las posibilidades incluyen la integración con autopilotos, la navegación mediante waypoints y la aplicación en competiciones de aeromodelismo universitario. Además, se podrían explorar métodos de identificación más avanzados y técnicas para abordar las no-linealidades en la dinámica de la aeronave.

Abstract

The focus of this study is on developing a flight simulator for fixed-wing unmanned aerial vehicles (UAVs) and creating a method to identify the aerodynamic parameters of these kind of platforms. By using two different models for aerodynamic resistance and three for propulsion, it has been successfully developed simulators that offer realistic flight experiences and allow pilots to interact in real time using a user-machine interface. Additionally, it has been designed an algorithm based on the least squares method to determine aerodynamic coefficients, successfully identifying parameters in specific scenarios.

These findings are significant because improved drag and propulsion models could enhance the development of velocity and altitude control systems. The integration of the BLDC (*brushless DC*) motor model and the propeller model, based on wind tunnel data, provides a solid foundation for analyzing UAV performance under varying conditions. Despite these achievements, there is room for improvement, such as addressing the challenge of calculating derivatives from noisy gyroscope signals and exploring advanced algorithms for parameter identification.

This work paves the way for further research and enhancements to the simulator. Possibilities include the integration with autopilots, waypoint navigation, and potential use in university RC aircraft competitions. Moreover, exploring more advanced identification methods and techniques, could help to address the nonlinear behavior in aircraft dynamics.

Índice

Agradecimientos III
Resumen V
Abstract VII

Índice de Figuras XIII
Índice de Tablas XVII

1 Introducción y objetivos del trabajo 1

2 Fundamentos y modelos matemáticos a implementar 5
2.1 Modelado de la mecánica de la plataforma 5
2.1.1 Sistemas de referencia 5
Sistema inercial geocéntrico 6
Sistema de ejes tierra 6
Sistema de ejes geográficos 6
Sistema de ejes navegación 7
Sistema de ejes avión 7
Sistema de ejes viento 7
Sistema de ejes estabilidad 8
Sistema de ejes cuerpo 9
2.1.2 Ecuaciones e hipótesis fundamentales 9
Mecanizado de las ecuaciones cinemáticas en ejes cuerpo 10
Mecanizado de las ecuaciones dinámicas en ejes cuerpo 10
Resumen de las ecuaciones a implementar y variables de estado 11
2.2 Modelado de fuerzas no gravitatorias y momentos 12
2.2.1 Aclaratoria sobre la complejidad de los modelos. Las versiones del simulador. 13
2.2.2 Modelo aerodinámico 13
Coeficientes aerodinámicos de la dinámica longitudinal 14
Coeficientes aerodinámicos de la dinámica lateral 16
2.2.3 Modelo propulsivo 16
Modelo sencillo y Teoría Ideal de Froude 17
Modelo complejo de hélice 18
Modelado del motor eléctrico BLDC 19
2.3 Modelo ambiental 20
2.3.1 Modelo gravitatorio 20
2.3.2 Modelo atmosférico 20

3 Implementación de modelos en `Simulink` 23
3.1 Generalidades 23
3.1.1 Realización de una simulación simple 23
3.1.2 Carga y configuración de un UAV específico para simulación 24

3.1.3 Diagrama de árbol del modelo de Simulink 28
3.2 Buses de datos 28
3.2.1 Control BUS (CTRL BUS) 28
3.2.2 Air Data BUS (AD BUS) 28
3.2.3 Propulsion Data BUS (PD BUS) 29
3.2.4 Wind BUS (WIND BUS) 29
3.2.5 Environment BUS (ENV BUS) 29
3.2.6 Aerodynamic BUS (AERO BUS) 29
3.2.7 Aircraft BUS (AC BUS) 30
3.3 Bloques de Simulink más importantes 30
3.3.1 Condicionado de las señales de control - Control BUS Creator 30
3.3.2 Modelos de fuerzas y momentos - Forces & Moments Calculation, 6DOF (Quaternion) y AC BUS Creator 31
3.3.3 Modelo Aerodinámico - Aerodynamic Coefficients y Aerodynamic Forces and Moments 33
Calculo de los coeficientes de la dinámica longitudinal - Longitudinal Aerodynamics (Cx, Cz, Cm coefficients) 33
Cálculo de los términos aerodinámicos 34
3.3.4 Modelos propulsivos - Propulsion 36
Modelo propulsivo muy simplificado - Simulador versión V3 36
Modelo simple de motor y modelo semi-analítico de la hélice - Simulador versión V2 37
Modelo eléctrico del motor y modelo semi-analítico de la hélice. - Simulador versión V1 38
3.3.5 Modelo de gravedad - Gravity 39
3.4 Modelo linealizado de la plataforma 40
3.4.1 Trimado del UAV 40
3.4.2 Análisis de los modos de vuelo del UAV 41

4 Interfaz usuario-simulador 45
4.1 Entradas al modelo 45
4.1.1 Segmento hardware 45
4.1.2 Segmento software 46
4.2 Salida del modelo y visualización 48
4.2.1 Configuración en Simulink 48
4.2.2 Configuración en Flight Gear 49

5 Identificación de parámetros del modelo a partir de datos de vuelo sintéticos 51
5.1 Modelos usados para la identificación 51
5.1.1 Modelo de generación de datos sintéticos 52
Modelado del acelerómetro del UAV 53
Modelado del giroscopio del UAV 55
Modelado de otros sensores del UAV para su uso en la identificación del sistema 55
Otros datos almacenados tras la simulación 56
5.1.2 Modelo de validación 56
5.2 Identificación de los parámetros del UAV 58
5.2.1 Planteamiento del problema 58
5.2.2 Función lqscurvefit() 59
5.2.3 Cálculo de las derivadas $\dot{p}, \dot{q}, \dot{r}$ 60
5.2.4 Script de identificacion final 60
Sección Ajustment of vector sizes 61
Sección Build the DATA MATRICES 61
Sección nonlinear regression function 61
Sección Results 61

6 Análisis de los resultados 63
6.1 Simulador V3. Modelo Simplificado. 63

6.1.1 Identificación con sensores ideales 63
6.1.2 Identificación con sensores reales 65
6.2 Simulador V2. Modelo intermedio. 65
6.2.1 Identificación de todos los coeficientes con $3\,\mathrm{min}$ de vuelo 65
6.2.2 Identificación de todos los coeficientes con $6\,\mathrm{min}$ de vuelo 66

7 Conclusiones y perspectiva futura 71

Apéndice A Códigos de MATLAB 73
A.1 Script de configuración de una aeronave para el Simulador y los modelos de identificación - config_model.m 73
A.2 Script de configuración del modelo de validación V2 - config_model_validation.m 76
A.3 Script de configuración del modelo de validación V3 - config_model_validation.m 78
A.4 Análisis de datos de APC para el modelado de la hélice del UAV 80
A.4.1 Conversión de unidades imperiales a sistema internacional de la matriz de datos de la hélice APC - siUnits.m 80
A.4.2 Filtrado de datos de la hélice en función de las RPM de operación - extractRPM.m 80
A.4.3 Creación de las matrices de velocidad, RPM/1000, empuje y par para el modelado de la hélice mediante Look-up Tables - LOOKUP_TABLES.m 80
A.5 Análisis de modelos linealizados - LINEAR_ANALYSIS.m 81
A.6 Identificación del modelo 84
A.6.1 Cálculo de las derivadas de las señales medidas por el giroscopio $\dot{\hat{p}}, \dot{\hat{q}}, \dot{\hat{r}}$ - ratesDerivatives.m 84
A.6.2 Función que integra el modelo propulsivo del simulador V2 usado en identificación - propulsiveModel.m 88
A.6.3 Función que integra el modelo V2 al que se le desea realizar el ajuste - FUN_v2.m 88
A.6.4 Función que integra el modelo V3 al que se le desea realizar el ajuste - FUN_v3.m 90
A.6.5 Identificación del modelo no lineal V2 (calculo de coeficientes) mediante mínimos cuadrados - NONLINEAR_IDENTIFICATION_v2.m 92
A.6.6 Identificación del modelo no lineal V3 (calculo de coeficientes) mediante mínimos cuadrados - NONLINEAR_IDENTIFICATION_v3.m 96

Bibliografía 101

Índice de Figuras

1.1 Dron para inspección de infraestructuras diseñado y fabricado por FADA-CATEC 1
1.2 Sistemas E-500 X-VISION (en frente) y USOL K2B (al fondo) en el centro ATLAS de FADA-CATEC 2
1.3 Primer vuelo del Jaleo-I diseñado y construido por el equipo VANTUS AeroTeam. De esta aeronave se extraen posteriormente los modelos de resistencia aerodinámica a utilizar en el presente trabajo, así como los datos de empuje y par proporcionados por su hélice en estudios en túnel de viento 2

2.1 Representación de los sistemas de referencia E, G y A. Además, se ilustran las direcciones x_w y x_s, y los ángulos de curso (χ o *course*), guiñada o rumbo (ψ, en inglés *yaw* o *heading*) y resbalamiento (β, en inglés *sideslip angle*) para el caso particular de $V_a = V$, es decir, en ausencia de viento. Por ende, se cumple que $\chi = \chi_a, \gamma = \gamma_a, \mu = \mu_a$. 7
2.2 Sistemas de referencia A y N. Un punto Q, origen de N, con coordenadas geodésicas (λ_Q, φ_Q). Además, se ilustra la posición relativa del UAV respecto al punto Q, $\mathbf{p}$ y las posiciones absolutas del punto Q, $\mathbf{r}_Q$, y del UAV, $\mathbf{r}$ (medidas respecto al centro de la Tierra O_e) 8
2.3 Relación entre los ejes W, los ejes S y los ejes B, dada por los ángulos α y β, así como las fuerzas aerodinámicas y el momento de cabeceo 14
2.4 Polar parabólica y modelo linealizado. 16
2.5 Teoría ideal de Froude 18
2.6 Empuje proporcionado por la hélice, T, en función de las RPM del motor y la velocidad incidente del viento u_a 19
2.7 Torque requerido por la hélice, Q, en función de las RPM del motor y la velocidad incidente del viento u_a 19

3.1 Modelo de UAV de propulsión por hélice con sus entradas y salidas. El modelo presenta una serie de instrumentos en la parte superior (`COCKPIT`) para comprobar el estado del vuelo mientras transcurre la simulación 24
3.2 *Mask* del bloque `UAV` 24
3.3 Diagrama de bloques dentro del bloque `UAV`. Se tienen 5 bloques principales: `Control BUS Creator`, `AC State Creator`, `Environment Model`, `Flight Data Recorder` y `Platform`. El bloque `Platform` integra toda la dinámica del UAV asi como también modela el sistema de datos de aire (bloque `Platform/Air Data System`) para el cálculo del AOA, el *sideslip angle* y el módulo de la velocidad del avión respecto al aire (en inglés *airspeed*). El bloque `AC State Creator` es un bloque auxiliar que crear los buses de estado para un fácil manejo de estos datos 25
3.4 Diagrama de bloques del bloque `Control BUS Creator` 31
3.5 Diagrama de bloques del bloque `Aircraft Mechanics`, el cual contiene en su interior los bloques `Forces & Moments Calculation`, `6DOF (Quaternion)` y `AC BUS Creator` 32
3.6 Diagrama de bloques del bloque `Forces & Moments Calculation` para las versiones `V1` y `V2` del simulador 32
3.7 Diagrama de bloques del bloque `Forces & Moments Calculation` para la version `V3` 33
3.8 Diagrama de bloques del bloque `AC BUS Creator` 34
3.9 Diagrama de bloques del bloque `Aerodynamic Coefficients` 34
3.10 Diagrama de bloques del bloque `Longitudinal Aerodynamics (Cx, Cz, Cm coefficients)` 35

3.11 Diagrama de bloques del bloque `Stability Axis Aerodynamics` para las versiones del simulador `V1` y `V2` 35
3.12 Diagrama de bloques del bloque `Stability Axis Aerodynamics` para la versión `V3` del simulador 35
3.13 Cálculo de la matriz de rotación de S a B 36
3.14 Diagrama de bloques del bloque `alpha-beta Coefficients` 36
3.15 Diagrama de bloques del bloque `Body Rate Damping` 37
3.16 Diagrama de bloques del bloque `Actuator Increments` 37
3.17 Diagrama de bloques del bloque `Propulsion` para la versión `V3` del simulador. 38
3.18 Diagrama de bloques del bloque `Propulsion` para la versión `V2` del simulador. 38
3.19 Modelo del motor en la versión `V2` del simulador 38
3.20 Modelo de la hélice implementado en el bloque `Propeller` 39
3.21 Diagrama de bloques del bloque `Propulsion` para la versión `V1` del simulador. 39
3.22 Diagrama de bloques del bloque `Motor` 39
3.23 Diagrama de bloques del bloque `Gravity` 40
3.24 Modelo `simulador2017bTRIM.slx` 41
3.25 Menu contextual del bloque `UAV` 41
3.26 Herramienta `Linear Analysis Tool` 42
3.27 Menú contextual `Trim the model` 42
3.28 Modos de vuelo para el simulador `V3` 43

4.1 *Gamepad Controller Logitech Dual Action G-UF13A*, el nombre de los canales correspondientes a cada señal de los joysticks y los nombres de las correspondientes señales de entrada al bloque `UAV` 46
4.2 Señales de entrada al modelo de `Simulink`, sin procesamiento alguno 47
4.3 Bloques de entrada al modelo 47
4.4 Bloques de adaptación de las señales de entrada al modelo. Se observa como los valores de las constantes para la corrección del offset son iguales y opuestos a los valores reflejados en los displays en la parte superior izquierda del diagrama. Cabe destacar que en esta imagen los joysticks se encuentran libres, sin deflexión alguna 48
4.5 Transformación de las señales de posición en ejes N a coordenadas geodésicas y conexión al bloque de comunicación con `Flight Gear` 49
4.6 Configuración del bloque `FlightGear Preconfigured 6DoF Animation` 49
4.7 Configuración de `Flight Gear` 50

5.1 Procedimiento de generación de datos sintéticos, identificación de UAVs y validación de resultados mediante modelos de `Simulink` y scripts de `MATLAB`. Las flechas representan el flujo de datos, que pueden ser cargas de archivos `.mat` o lecturas de archivos de configuración `.m`. Conectan los distintos tipos de archivos con los que se trabaja (estos pueden ser modelos de `Simulink`, `.slx` o scripts de `MATLAB`, `.m`) 52
5.2 Contenido del bloque `Flight Data Recorder (FDR)` 53
5.3 Contenido del bloque `IMU Sensor Recorder` 53
5.4 Señal real de la aceleración lineal $y_{\text{accel},\,x} - v_{\text{accel},\,x}$ y su equivalente medido por el acelerómetro artificial $y_{\text{accel},\,x}$ 54
5.5 Señal real de la velocidad angular $r \equiv$ r y su equivalente medido por el giroscopio artificial $y_{\text{gyro},\,z} \equiv$ r + noise 55
5.6 Estructura interna del bloque `Other Sensors Recorder` 56
5.7 Resultado de añadir ruido a las señales medidas por los demás sensores (vuelo de modelo `V3` de $180\,\text{s}$ de duración). En rojo se muestra la señal original y en azul la señal con ruido. Como puede apreciarse, el ruido añadido a las señales, en principio, no es muy significativo en el caso de p, q y r, mientras que se aprecia mejor en las medidas de α, β y V_a 57
5.8 Configuración del `Callback/StopFcn` del bloque UAV. Es igual para todas las versiones del simulador 57
5.9 Modelo de validación 58
5.10 Señales del giroscopio a lo largo del proceso de filtrado y derivación. 60

6.1 Resultados de la identificación del sistema `V3` con sensores ideales y una simulación de $3\,\text{min}$ 65
6.2 Validación de las variables de estado para la identificación utilizando una simulación de $3\,\text{min}$ en el modelo `V3` e incorporando al modelo de validación únicamente los coeficientes de fuerzas. En línea roja continua se muestra el estado del modelo de generación de datos, mientras que la línea azul discontinua, representa el estado del modelo de validación 66

6.3 Resultados de la identificación del sistema V3 con sensores reales y una simulación de 3 min 67
6.4 Resultados de la identificación del sistema V2 con sensores reales y una simulación de 3 min 68
6.5 Resultados de la identificación del sistema V2 con sensores reales y una simulación de 6 min 69

Índice de Tablas

2.1 Descripción de las variables de estado del simulador 12

3.1 Parámetros del AerosondeUAV 26
3.2 Parámetros del Jaleo-I 26
3.3 Señales de control - `Control BUS (CTRL BUS)` 29
3.4 Datos de aire - `Air Data BUS (AD BUS)` 29
3.5 Señales de control - `Propulsion Data BUS (PD BUS)` 29
3.6 Señales del modelo de vientos - `Wind BUS (WIND BUS)` 29
3.7 Señales de datos ambientales - `Environment BUS (ENV BUS)` 30
3.8 Señales de los coeficientes aerodinámicos `Aerodynamic BUS (AERO BUS)` 30
3.9 Señales del estado y demás buses - `Aircraft BUS (AC BUS)` 30

5.1 Desviaciones típicas en las medidas de los sensores modelados 53

6.1 Coeficientes identificados en el modelo `V3`. Sensores ideales, $3\,\mathrm{min}$ de simulación 64
6.2 Coeficientes identificados en el modelo `V3`. Sensores reales, $3\,\mathrm{min}$ de simulación 67
6.3 Coeficientes identificados en el modelo `V2`. Sensores reales, $3\,\mathrm{min}$ de simulación 68
6.4 Coeficientes identificados en el modelo `V2`. Sensores reales, $6\,\mathrm{min}$ de simulación 69

1 Introducción y objetivos del trabajo

En los últimos años, los vehículos aéreos no tripulados (VANTs) han emergido como una tecnología transformadora en diversas industrias, desde la agricultura hasta la logística. Estos VANTs, también conocidos como drones o UAVs, por sus siglas en inglés (*Unmmaned Aerial Vehicles*), han demostrado su valía en aplicaciones como inspecciones de infraestructuras, mapeo topográfico y de edificaciones, y entregas de última milla (Ver Figura 1.1). Sin embargo, a medida que su adopción continúa creciendo, surge la necesidad crítica de comprender en profundidad su comportamiento en vuelo, más aún cuando se pretende trabajar con plataformas fabricadas por terceros, de las que se desconocen los modelos utilizados para diseñarlas. Así mismo, es de vital importancia en las etapas de diseño de estas aeronaves, conocer con cierta precisión las actuaciones y capacidades técnicas (o *performance*) de la plataforma con la que se trabaja, a medida que se realizan los diversos cambios en el diseño de la misma.

Figura 1.1 Dron para inspección de infraestructuras diseñado y fabricado por FADA-CATEC.

El presente trabajo se enfoca en abordar estos desafíos mediante la creación de un simulador de vuelo y el desarrollo de un algoritmo de identificación de coeficientes aerodinámicos para VANTs comerciales. El objetivo principal de esta investigación es proporcionar una herramienta que permita modelar y analizar el vuelo de UAVs de ala fija, así como identificar los coeficientes aerodinámicos clave a partir de datos de vuelo sintéticos generados en el simulador. Además, esta herramienta debe permitir la interacción con el usuario, de forma que se pueda constatar el correcto funcionamiento del simulador con pilotos de drones reales, así como permitir la generación de datos orgánicos congruentes con los obtenidos en misiones reales.

El presente proyecto se justifica por la necesidad de mejorar la comprensión de la dinámica de vuelo de diferentes UAVs disponibles en el mercado y de los que dispone FADA-CATEC en sus instalaciones (como por ejemplo el USOL K2B o el E-500 X-VISION, Ver Figura 1.2). Esto, con el fin de facilitar la labor de desarrollo de sistemas de control de vuelo para drones de ala fija implementados dentro del Autopiloto CATEC. De igual forma, existe la necesidad dentro del equipo de aeromodelismo universitario de la Escuela Técnica Superior de Ingeniería, VANTUS AeroTeam, de poder predecir el performances de las aeronaves durante la etapa de diseño (Ver Figura 1.3), tanto para conocer si el UAV será capaz de concretar la misión

Figura 1.2 Sistemas E-500 X-VISION (en frente) y USOL K2B (al fondo) en el centro ATLAS de FADA-CATEC.

para la cual esta siendo diseñado, como también para recibir *feedback* directa de los pilotos del equipo. Existe además gran interés en tener la capacidad de entrenar a nuevos pilotos de competición dentro del equipo, disponiendo de un simulador propio en el cual se pueda emular con precisión la experiencia de pilotaje de las aeronaves con las que se pretende competir, así como disminuir costes debidos a accidentes en vuelo o la adquisición de software comercial como `Real Flight`.

Figura 1.3 Primer vuelo del Jaleo-I diseñado y construido por el equipo VANTUS AeroTeam. De esta aeronave se extraen posteriormente los modelos de resistencia aerodinámica a utilizar en el presente trabajo, así como los datos de empuje y par proporcionados por su hélice en estudios en túnel de viento.

La metodología del presente trabajo se basará en la combinación de modelado y simulación en el entorno `Simulink`, con la configuración del simulador, el análisis de modos de vuelo y la implementación de un algoritmo de identificación en `MATLAB`. Se utilizarán datos de vuelo sintéticos generados en el simulador para alimentar el algoritmo de identificación, y finalmente verificar los resultados de la identificación mediante un modelo de validación que emule las entradas del usuario que generó los datos, pero sobre un modelo que integre los parámetros recién identificados.

Este documento está organizado como sigue: en el Capítulo 2, se presentarán los fundamentos teóricos relacionados con la mecánica del vuelo de la aeronave, así como los modelos de fuerzas, momentos y ambientales necesarios para la simulación. El Capítulo 3 se detallará el diseño del simulador de vuelo en el entorno `Simulink`, así como la linealización y análisis del modelo linealizado. El Capítulo 4 pretende explicar el funcionamiento de la interfaz usuario-simulador, tanto a nivel de entradas (mediante joysticks) como de salidas (mediante el software `Flight Gear`). En el Capítulo 5, se abordarán los desafíos encontrados en la identificación de coeficientes aerodinámicos a partir de datos de vuelo sintéticos, indicando los pasos seguidos para el desarrollo del algoritmo de identificación. El Capítulo 6 analizará los resultados obtenidos a través del simulador y el algoritmo de identificación. Finalmente, el Capítulo 7 ofrecerá conclusiones y recomendaciones para futuras investigaciones y trabajos relacionados con esta materia.

2 Fundamentos y modelos matemáticos a implementar

En el presente capítulo se enuncian los modelos que fueron implementados en `Simulink` para aproximar el comportamiento real de los UAVs de ala fija con los que se pretende trabajar. Cabe destacar que los modelos son apropiados para condiciones de operación normales, características de plataformas como las presentadas en este trabajo, de forma que son válidos para bajas altitudes de operación, velocidades de vuelo relativamente bajas y ángulos de ataque pequeños.

2.1 Modelado de la mecánica de la plataforma

Para el modelado de la mecánica de las aeronaves, este trabajo de apoya en los apuntes de la asignatura Mecánica del Vuelo y Operaciones Aéreas, impartida en el Grado de Ingeniería Aeroespacial de la Universidad de Sevilla, en la mención Navegación Aérea [1]. Además, se hace referencia al libro de Beard y McLain [2].

2.1.1 Sistemas de referencia

Para describir el comportamiento de la aeronave de forma integra y clara, se utilizan distintos sistemas de referencia en los cuales la formulación de las ecuaciones que describen la mecánica de la aeronave, se ve significativamente simplificada.

Para obtener las ecuaciones escalares que describen el comportamiento de la plataforma, es necesario proyectar las ecuaciones vectoriales de la mecánica en un sistema de referencia determinado. Dados dos sistemas de referencia distintos $U(O_u, x_u, y_u, z_u)$ y $V(O_v, x_v, y_v, z_v)$, un vector genérico $\mathbf{a}$ puede expresarse de forma unívoca en ambos sistemas. Si se considera la matriz de rotación $\mathbf{R}_v^u$ asociada a la transformación $V \rightarrow U$, se puede escribir

$$\mathbf{a}^u = \mathbf{R}_v^u \mathbf{a}^v \tag{2.1}$$

donde

$$\mathbf{a}^u = a_{x_u}\mathbf{i}_u + a_{y_u}\mathbf{j}_u + a_{z_u}\mathbf{k}_u \equiv \begin{bmatrix} a_{x_u} \\ a_{y_u} \\ a_{z_u} \end{bmatrix}_u \tag{2.2}$$

$$\mathbf{a}^v = a_{x_v}\mathbf{i}_v + a_{y_v}\mathbf{j}_v + a_{z_v}\mathbf{k}_v \equiv \begin{bmatrix} a_{x_v} \\ a_{y_v} \\ a_{z_v} \end{bmatrix}_v \tag{2.3}$$

$$\mathbf{R}_v^u = \begin{bmatrix} \mathbf{i}_u \cdot \mathbf{i}_v & \mathbf{i}_u \cdot \mathbf{j}_v & \mathbf{i}_u \cdot \mathbf{k}_v \\ \mathbf{j}_u \cdot \mathbf{i}_v & \mathbf{j}_u \cdot \mathbf{j}_v & \mathbf{j}_u \cdot \mathbf{k}_v \\ \mathbf{k}_u \cdot \mathbf{i}_v & \mathbf{k}_u \cdot \mathbf{j}_v & \mathbf{k}_u \cdot \mathbf{k}_v \end{bmatrix} \tag{2.4}$$

A continuación se definen los ejes utilizados a los largo de las posteriores secciones de este capítulo.

Sistema inercial geocéntrico

El sistema inercial geocéntrico $I(O_i, x_i, y_i, z_i)$ se define como sigue:

- $O_i \rightarrow$ centro de masas de la Tierra.
- $x_i \rightarrow$ dirigido hacia el primer punto de Aries.
- $z_i \rightarrow$ dirigido según el eje de rotación de la Tierra.
- $y_i \rightarrow$ completa el triedro a derechas, contenido en el plano ecuatorial.

Sistema de ejes tierra

El sistema de ejes tierra $E(O_e, x_e, y_e, z_e)$ se define como sigue:

- $O_e \rightarrow$ centro de masas de la Tierra.
- $x_e \rightarrow$ dirigido hacia la intersección entre el ecuador y el meridiano de Greenwich.
- $z_e \rightarrow$ dirigido según el eje de rotación de la Tierra.
- $y_e \rightarrow$ completa el triedro a derechas, contenido en el plano ecuatorial.

Este sistema es solidario con la Tierra, de forma que rota alrededor de $z_e \equiv z_i$ (es decir, respecto al sistema inercial I) con una velocidad angular de $|\boldsymbol{\omega}_{e/i}| = \omega_{e/i} = 7.27 \times 10^{-5}\,\text{rad/s} \approx 0.999\,697\,\text{rev/day}$.

Sistema de ejes geográficos

El sistema de ejes geográficos $G(O_g, x_g, y_g, z_g)$ se define como sigue:

- $O_g \rightarrow$ centro de masas de la Tierra.
- $x_g \rightarrow$ en la dirección y sentido del vector posición $\mathbf{r}$.
- $y_g \rightarrow$ ortogonal a x_g, en el plano ecuatorial y sentido este.
- $z_g \rightarrow$ completa el triedro a derechas.

La posición del UAV sobre la superficie terrestre ($\mathbf{r}$) queda determinada una vez se conoce la orientación del sistema de referencia geográfico, respecto al sistema de ejes tierra:

– **Latitud** φ. Ángulo del vector posición $\mathbf{r}$ con el plano ecuatorial, positivo hacia el norte.
– **Longitud** λ. Ángulo que forma la proyección de $\mathbf{r}$ sobre el plano ecuatorial con el eje x_e, positivo hacia el este.
De forma que la posición de la plataforma viene dada por las coordenadas geodésicas $(\varphi, \lambda, |\mathbf{r}| = r)$.

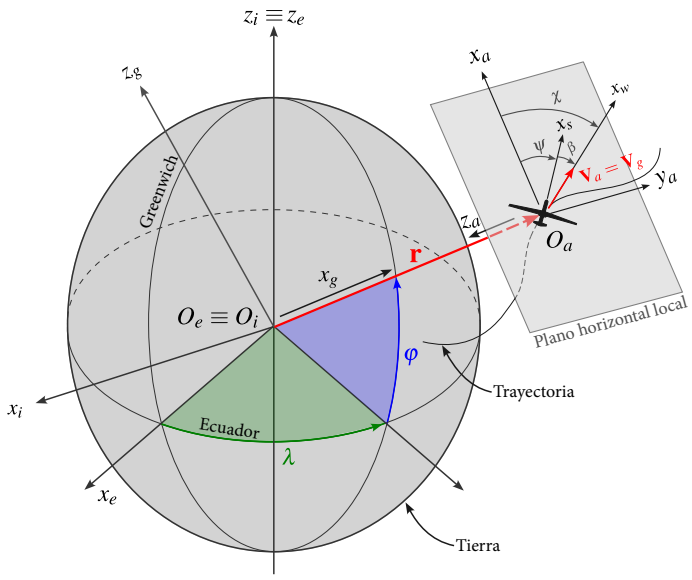


Figura 2.1 Representación de los sistemas de referencia E, G y A. Además, se ilustran las direcciones x_w y x_s, y los ángulos de curso (χ o *course*), guiñada o rumbo (ψ, en inglés *yaw* o *heading*) y resbalamiento (β, en inglés *sideslip angle*) para el caso particular de $V_a = V$, es decir, en ausencia de viento. Por ende, se cumple que $\chi = \chi_a, \gamma = \gamma_a, \mu = \mu_a$..

Sistema de ejes navegación

El sistema de ejes navegación o NED (North-East-Down) $N(O_n, x_n, y_n, z_n)$ define un plano tangente a la superficie terrestre en un punto genérico Q, elegido a conveniencia tal que $O_n \equiv Q$. Se dice que es *topocéntrico* a cualquier sistema fijo a la superficie de la Tierra. Este sistema de ejes se define como:

- $O_n \rightarrow$ punto Q ubicado sobre la superficie terrestre.
- $y_n \rightarrow$ paralelo a y_g. Naturalmente, pertenece al plano horizontal local. Define los *paralelos* en un modelo de Tierra esférica. Apunta en la dirección "*East*".
- $z_n \rightarrow$ según el eje $(-x_g)$, es decir, en dirección hacia el centro de la Tierra en un modelo esférico. Apunta en la dirección "*Down*".
- $x_n \rightarrow$ completa el triedro a derechas. Al igual que el eje y_n, este también pertenece al plano horizontal local. Define los *meridianos* en un modelo de Tierra esférica. Apunta en la dirección "*North*".

Este sistema se considera inercial en el *modelo de tierra plana*, como el que se pretende utilizar en el modelado de la dinámica de la aeronave. Se utilizará el modelo elipsoidal de la Tierra, WGS84, unicamente para el modelado de la aceleración gravitatoria y para la posterior visualización del vuelo mediante el programa `Flight Gear`.

Sistema de ejes avión

El sistema de ejes avión, $A(O_a, x_a, y_a, z_a)$ también llamado sistema horizonte local, coincide con el *plano horizontal local*, el cual se define como el plano perpendicular al vector posición absoluta $\mathbf{r}$ que pasa por el centro de masas del UAV. Este sistema de ejes difiere del sistema de ejes N únicamente en que su origen O_v se haya en el centro de masas de la aeronave, y no en un punto genérico sobre la superficie terrestre M.

Sistema de ejes viento

El sistema de ejes viento $W(O_w, x_w, y_w, z_w)$ es aquel sobre el cual se definen las fueras aerodinámicas que actúan sobre la plataforma. Se define como sigue:

- $O_w \rightarrow$ centro de masas de la aeronave.

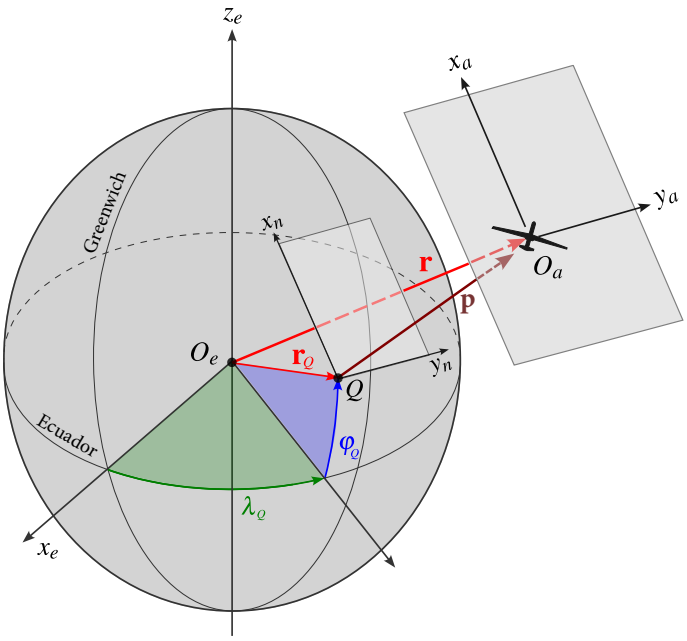


Figura 2.2 Sistemas de referencia A y N. Un punto Q, origen de N, con coordenadas geodésicas (λ_Q, φ_Q). Además, se ilustra la posición relativa del UAV respecto al punto Q, $\mathbf{p}$ y las posiciones absolutas del punto Q, $\mathbf{r}_Q$, y del UAV, $\mathbf{r}$ (medidas respecto al centro de la Tierra O_e).

- $x_w \rightarrow$ con la dirección y sentido de la velocidad aerodinámica de la aeronave $\mathbf{V}_a$. Sobre este eje se define la fuerza de resistencia aerodinámica (**D**, en inglés *drag*), con sentido opuesto al mismo (es decir, con sentido opuesto al movimiento relativo de la aeronave respecto al viento).
- $z_w \rightarrow$ contenido en el plano de simetría del avión, dirigido hacia abajo. Sobre este eje se define la fuerza de sustentación aerodinámica (**L**, en inglés *lift*), con sentido negativo.
- $y_w \rightarrow$ completa el triedro a derechas. Sobre este eje se definen las fuerzas aerodinámicas laterales (**Q**).

La orientación del vector velocidad aerodinámica $\mathbf{V}_a$ (sistema W) respecto al sistema N se define mediante los siguientes ángulos:

– **Ángulo de asiento de velocidad (*aerodynamic flight-path angle*)** γ_a. Ángulo entre la velocidad aerodinámica $\mathbf{V}_a$ y el plano horizontal local, positivo hacia arriba.
– **Ángulo de guiñada de velocidad (*aerodynamic course angle*)** χ_a. Ángulo que forma la proyección del vector velocidad aerodinámica $\mathbf{V}_a$ sobre el plano horizontal local con el eje x_n, es decir, con respecto a la dirección del meridiano local, positivo hacia el este.
– **Ángulo de balance de velocidad o alabeo (*aerodynamic bank angle*)** μ_a. Ángulo formado por y_w con la intersección entre el plano horizontal local y el plano $y_w z_w$, positivo en el sentido de bajar el ala derecha.

En ausencia de viento, la velocidad aerodinámica coincide con la velocidad del UAV o *ground speed* ($\mathbf{V}_a = \mathbf{V}$) de forma que se pueden definir los siguientes ángulos:

– **Ángulo de trayectoria (*flight-path angle*)** γ.
– **Ángulo de curso (*course angle*)** χ.
– **Ángulo de balance (*bank angle*)** μ.

Sistema de ejes estabilidad

El sistema de ejes estabilidad $S(O_s, x_s, y_s, z_s)$ es el sistema intermedio entre el sistema de ejes viento y el sistema de ejes cuerpo. Entre este sistema S y el sistema W existe una rotación única alrededor z_w de ángulo β, llamado *ángulo de resbalamiento* (en inglés *sideslip angle*). Los ejes estabilidad se definen como sigue:

- $O_s \rightarrow$ centro de masas de la aeronave.

- $x_s \rightarrow$ proyección del eje x_w sobre el plano de simetría del UAV. El ángulo entre x_s y x_w es β.
- $z_s \rightarrow$ coincide con z_w.
- $y_s \rightarrow$ completa el triedro a derechas. Con dirección perpendicular al plano de simetría de la aeronave y sentido hacia el ala derecha.

Sistema de ejes cuerpo

El sistema de ejes cuerpo $B(O_b, x_b, y_b, z_b)$ se obtiene al girar positivamente el sistema S alrededor de y_s un ángulo α o AOA, llamado *ángulo de ataque* (en inglés *angle of attack*). Ver Figura 2.3. Este sistema de referencia se define como:

- $O_b \rightarrow$ centro de masas del avión.
- $x_b \rightarrow$ contenido en el plano de simetría del avión, proyección de x_s sobre una línea longitudinal de referencia que pasa por el morro de la aeronave. El ángulo entre x_b y x_s es α.
- $y_b \rightarrow$ coincide con y_s.
- $z_b \rightarrow$ completa el triedro a derechas. contenido en el plano de simetría de la aeronave.

La orientación de la aeronave respecto al sistema N es de gran interés y se define mediante los llamados *ángulos de Euler*:

– **Ángulo de asiento (*Pitch*)** θ. Ángulo entre x_b y el plano horizontal local, positivo hacia arriba.
– **Ángulo de guiñada (*Yaw*)** ψ. Ángulo que forma la proyección de x_b sobre el plano horizontal local con el eje x_n, es decir, con respecto a la dirección del meridiano local, positivo hacia el este.
– **Ángulo de balance (*Roll*)** ψ. Ángulo formado por y_b con la intersección entre el plano horizontal local y el plano $y_b z_b$, positivo en el sentido de bajar el ala derecha.

2.1.2 Ecuaciones e hipótesis fundamentales

Las ecuaciones vectoriales que se utilizan para modelar el comportamiento del UAV, se pueden clasificar de dos maneras distintas: según si se estudia el movimiento en si mismo o sus causas (ecuaciones cinemáticas y ecuaciones dinámicas), o según la naturaleza misma del movimiento (ecuaciones traslacionales y ecuaciones rotacionales). A continuación se presenta un resumen de dichas ecuaciones, su derivación y mecanizado en el sistema B (únicamente de las ecuaciones dinámicas), el cual presenta como principal ventaja que el tensor de inercia del sólido, $\mathbf{J}$, es constante al expresarlo en dichos ejes.

Las ecuaciones de la dinámica newtoniana solo tienen validez en un sistema de referencia inercial, y por esta razón se parte de las mismas representadas en el sistema de referencia N. Esta aproximación es apropiada ya que a pesar de la existencia de aceleraciones no inerciales debidas a la rotación de la tierra (la aceleración centrifuga, $\boldsymbol{\omega}_{e/i} \times [\boldsymbol{\omega}_{e/i} \times \mathbf{r}]$ y la aceleración de Coriolis, $2\boldsymbol{\omega}_{e/i} \times \mathbf{V}_g$), estas son 3 ordenes de magnitud inferiores a la aceleración gravitatoria cuando se realizan vuelos a velocidades y altitudes relativamente bajas, por lo que pueden ser despreciados sus efectos.

Por último, es necesario conocer un método para determinar el valor de las derivadas temporales de ciertas cantidades respecto al sistema inercial I, pero expresando estas derivadas en el sistema B. El calculo de una derivada temporal medida por un observador desde un sistema genérico V (supuesto inercial), de un vector cualquiera $\mathbf{a} \equiv \mathbf{a}^u$ expresado en otro sistema U (el cual rota respecto a los ejes inercial con velocidad angular $\boldsymbol{\omega}_{u/v}$), se puede denotar como:

$$\left.\frac{d\mathbf{a}^u}{dt}\right|_v \equiv \frac{d\mathbf{a}^u}{dt_v} \tag{2.5}$$

Finalmente, por el *Teorema de Poisson* o *Teorema del Transporte de la Dinámica*, se cumple que:

$$\frac{d\mathbf{a}^u}{dt_v} = \frac{d\mathbf{a}^u}{dt_u} + \boldsymbol{\omega}_{u/v} \times \mathbf{a}^u = \left(\dot{a}_{x_u}\mathbf{i}_u + \dot{a}_{y_u}\mathbf{j}_u + \dot{a}_{z_u}\mathbf{k}_u\right) + \boldsymbol{\omega}_{u/v} \times \mathbf{a}^u \tag{2.6}$$

Mecanizado de las ecuaciones cinemáticas en ejes cuerpo

Dado el vector posición medido desde en el sistema E, es decir $\mathbf{r} \equiv \overline{O_e O_a}$ y el vector posición medido desde el sistema N, $\mathbf{p} \equiv \overline{QO_a}$, se define el vector velocidad respecto a tierra de la plataforma $\mathbf{V}$ o *ground speed* como

$$\mathbf{V} \triangleq \frac{d\mathbf{r}}{dt_e} = \frac{d}{dt_e}\left(\mathbf{p} + \mathbf{r}_Q\right) = \frac{d\mathbf{p}}{dt_e} = \frac{d\mathbf{p}}{dt_a} \tag{2.7}$$

Donde $\boldsymbol{\omega}_{e/a} = 0 \Rightarrow d/dt_e = d/dt_a$. Definiendo $\mathbf{V}^b \triangleq [u,v,w]^T, \mathbf{p}^a \triangleq [x,y,z]^T$, se puede mecanizar la *ecuación vectorial de la cinemática traslacional* en ejes A como sigue:

$$\mathbf{V}^a = \mathbf{R}_b^a(\phi,\theta,\psi)\mathbf{V}^b = \frac{d\mathbf{p}^a}{dt_a} \equiv \boxed{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \mathbf{R}_b^a \begin{bmatrix}u\\ v\\ w\end{bmatrix}} \tag{2.8}$$

donde

$$\mathbf{R}_a^b = (\mathbf{R}_b^a)^T = \underbrace{\begin{bmatrix}\cos\phi & \sin\phi & 0\\ -\sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{bmatrix}}_{\text{Roll - }\phi} \underbrace{\begin{bmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{bmatrix}}_{\text{Pitch- }\theta} \underbrace{\begin{bmatrix}\cos\psi & \sin\psi & 0\\ -\sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}}_{\text{Yaw - }\psi} \tag{2.9}$$

A la matriz $\mathbf{R}_a^b$ se suele escribir también como **DCM** (*Director Cosines Matrix* o Matriz de Cosenos Directores).

Luego, la *ecuación vectorial de la cinemática rotacional* relaciona los ángulos de euler con la velocidad angular del UAV respecto de tierra (es decir $\boldsymbol{\omega}_{b/n}$). Se tiene que

$$\boldsymbol{\omega}_{b/n} = \boldsymbol{\omega}_{b/a} = \dot{\psi}\mathbf{k}_a + \dot{\theta}\mathbf{j}_X + \dot{\phi}\mathbf{i}_b \tag{2.10}$$

donde se puede definir $\boldsymbol{\omega}_{b/a}^b \triangleq [p,q,r]^T$ y donde $\mathbf{j}_X$ esta definido por el eje y_a inmediatamente después de realizar un giro $d\psi$ alrededor del eje z_a. Se puede comprobar que, al escribir los versores $\mathbf{k}_a$ y $\mathbf{j}_X$ en sus propias bases y aplicar las rotaciones correspondientes para expresarlos en el sistema B, se llega a la siguiente expresión:

$$\boxed{\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}} \tag{2.11}$$

Mecanizado de las ecuaciones dinámicas en ejes cuerpo

La Segunda Ley de Newton aplicada a un sólido en un movimiento traslacional se enuncia como

$$\mathbf{f} = m\frac{d\mathbf{V}}{dt_i} \tag{2.12}$$

Si expresamos la velocidad en ejes B, definimos el sumatorio de fuerzas externas sobre la aeronave en ejes B como $\mathbf{f}^b \triangleq [X,Y,Z]^T$ y aplicamos el Teorema de Poisson (2.6) para expresar la derivada respecto al

sistema inercial en el sistema B, se obtiene

$$\mathbf{f}^{\,b} = m\left(\frac{d\mathbf{V}^b}{dt_a} + \boldsymbol{\omega}^a_{b/a} \times \mathbf{V}^b\right)$$

y finalmente, reordenando se llega a la *ecuación vectorial de la dinámica traslacional* del UAV en ejes B:

$$\boxed{\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} = \begin{bmatrix}rv-qw\\ pq-ru\\ qu-pv\end{bmatrix} + \frac{1}{m}\begin{bmatrix}X\\ Y\\ Z\end{bmatrix}} \tag{2.13}$$

Luego, para el estudio de la dinámica rotacional se parte igualmente de la Segunda Ley de Newton

$$\boldsymbol{\tau} = \frac{d\mathbf{h}}{dt_i}$$

donde la suma de todos los momentos externos sobre la aeronave en ejes B se define como $\boldsymbol{\tau}^b \triangleq [l,m,n]^T$, el momento angular del sólido en ejes B cumple que $\mathbf{h}^b \triangleq \mathbf{J}\boldsymbol{\omega}^b_{b/n}$ y donde el tensor de inercia de la aeronave calculado desde los ejes B se define como

$$\mathbf{J} = \begin{bmatrix}\int (y_b^2+z_b^2)\,dm & \int -x_b y_b\,dm & \int -x_b z_b\,dm\\ \int -x_b y_b\,dm & \int (x_b^2+z_b^2)\,dm & \int -y_b z_b\,dm\\ \int -x_b z_b\,dm & \int -y_b z_b\,dm & \int (x_b^2+y_b^2)\,dm\end{bmatrix} \triangleq \begin{bmatrix}J_x & -J_{xy} & -J_{xz}\\ -J_{xy} & J_y & -J_{yz}\\ -J_{xz} & -J_{yz} & J_z\end{bmatrix} \approx \begin{bmatrix}J_x & 0 & -J_{xz}\\ 0 & J_y & 0\\ -J_{xz} & 0 & J_z\end{bmatrix} \tag{2.14}$$

Finalmente, desarrollando esta ecuación y reordenando se llega a la *ecuación vectorial de la dinámica rotacional* del UAV en ejes B:

$$\boxed{\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\Gamma_1 pq-\Gamma_2 qr\\ \Gamma_5 pr-\Gamma_6(p^2-r^2)\\ \Gamma_7 pq-\Gamma_1 qr\end{bmatrix} + \begin{bmatrix}\Gamma_3 l+\Gamma_4 n\\ m/J_y\\ \Gamma_4 l+\Gamma_8 n\end{bmatrix}} \tag{2.15}$$

donde

$$\begin{aligned}
\Gamma &\triangleq J_x J_z - J_{xz}^2\\
\Gamma_1 &\triangleq \frac{J_{xz}(J_x-J_y+J_z)}{\Gamma}\\
\Gamma_2 &\triangleq \frac{J_z(J_z-J_y)+J_{xz}^2}{\Gamma}\\
\Gamma_3 &\triangleq J_z/\Gamma\\
\Gamma_4 &\triangleq J_{xz}/\Gamma\\
\Gamma_5 &\triangleq (J_z-J_x)/J_y\\
\Gamma_6 &\triangleq J_{xz}/J_y\\
\Gamma_7 &\triangleq \frac{J_x(J_x-J_y)+J_{xz}^2}{\Gamma}\\
\Gamma_8 &\triangleq J_x/\Gamma
\end{aligned}$$

Resumen de las ecuaciones a implementar y variables de estado

Tras el desarrollo anterior, se obtiene el siguiente sistema de ecuaciones diferenciales:

$$
\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pq - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}
$$

$$
\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6 (p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{bmatrix} + \begin{bmatrix} \Gamma_3 l + \Gamma_4 n \\ m/J_y \\ \Gamma_4 l + \Gamma_8 n \end{bmatrix}
$$

$$
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \mathbf{R}_b^a(\phi, \theta, \psi) \begin{bmatrix} u \\ v \\ w \end{bmatrix}
$$

$$
\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi \sec\theta & \cos\phi \sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}
$$

De forma que es claramente identificable un vector de estados con sus 12 componentes. Definiendo la altitud sobre el punto Q como $h \triangleq -z$ tal que $\dot{z} = -\dot{h}$, podemos definir el estado de la aeronave como:

$$
\mathbf{x} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \\ x \\ y \\ h \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{2.16}
$$

En la Tabla (2.1) se presenta un breve resumen de las variables de estado y su interpretación. Esta es la base del desarrollo del simulador, donde estas variables se integrarán mediante el programa `Simulink` y se logrará una descripción completa de la dinámica del UAV en todo instante de tiempo.

Tabla 2.1 Descripción de las variables de estado del simulador.

Variables de Estado	Descripción
u	Velocidad absoluta proyectada en el eje $\mathbf{i}_b$
v	Velocidad absoluta proyectada en el eje $\mathbf{j}_b$
w	Velocidad absoluta proyectada en el eje $\mathbf{k}_b$
p	Velocidad de balanceo (roll rate) medida alrededor del eje $\mathbf{i}_b$
q	Velocidad de cabeceo (pitch rate) medida alrededor del eje $\mathbf{j}_b$
r	Velocidad de guiñada (yaw rate) medida alrededor del eje $\mathbf{k}_b$
x	Posición norte del UAV respecto al punto Q a lo largo del eje $\mathbf{i}_n$
y	Posición este del UAV respecto al punto Q a lo largo del eje $\mathbf{j}_n$
h	Altitud del UAV respecto al punto Q a lo largo del eje $-\mathbf{k}_n$
ϕ	Ángulo de balance (roll angle)
θ	Ángulo de cabeceo (pitch angle)
ψ	Ángulo de guiñada (yaw angle)

2.2 Modelado de fuerzas no gravitatorias y momentos

Las fuerzas y momentos que actúan sobre la plataforma son fundamentalmente de naturaleza: gravitatoria, aerodinámica y propulsiva. A continuación, se describen los modelos que permiten determinar las compo-

nentes en ejes B, de los vectores

$$\mathbf{f}^b = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{f}_g^b + \mathbf{f}_a^b + \mathbf{f}_p^b$$
$$\boldsymbol{\tau}^b = \begin{bmatrix} l \\ m \\ n \end{bmatrix} = \boldsymbol{\tau}_a^b + \boldsymbol{\tau}_p^b$$

2.2.1 Aclaratoria sobre la complejidad de los modelos. Las versiones del simulador.

El Simulador de UAVs de ala fija que se pretende desarrollar en el presente trabajo (a partir de ahora, *Fixed Wing Simulator, FWS*) posee tres versiones con distintos grados de complejidad y detalle en sus modelos aerodinámicos y propulsivos.

- **FWS V1. Versión Compleja.**
  - Modelo eléctrico del motor y modelo semi-analítico de la hélice.
  - Modelo aerodinámico completo de polar parabólica extendida: $C_D(C_L) = C_{D_p} + k_2 C_L + k_1 C_L^2$
- **FWS V2. Versión Intermedia.**
  - Modelo simple del motor (RPM $= f(\delta_t)$) y modelo semi-analítico de la hélice.
  - Modelo aerodinámico completo de polar parabólica extendida.
- **FWS V3. Versión Simplificada.**
  - Modelo propulsivo muy simplificado. $T = \frac{1}{2}\rho S_{\text{prop}} C_{\text{prop}} \left[(k_{\text{motor}}\delta_t)^2 - V_a^2\right]$, $Q = 0$
  - Modelo aerodinámico simplificado. $C_D(\alpha) = C_{D_0} + C_{D_\alpha}\alpha$

De forma que a continuación, se presentan los detalles de cada uno de los modelos mencionados anteriormente.

2.2.2 Modelo aerodinámico

Como Beard y McLain [2] hacen constar en su libro, las fuerza aerodinámica que experimenta una aeronave en un instante determinado puede descomponerse en un par de componentes sobre el plano de simetría del avión $X^bZ^b \equiv X^sZ^s$, es decir, las fuerzas longitudinales (L y D) y una componente normal al mismo, llamada fuerza lateral Y. De igual forma, el momento neto sobre el UAV debido a la interacción con el viento incidente, puede descomponerse, pero esta vez sobre los ejes B, adquiriendo los nombres de: momento de balance (en inglés *rolling moment*) l, momento de cabeceo (en inglés *pitching moment*) m y momento de guiñada (en inglés *yawing moment*) n. Ver Figura 2.3.

Más formalmente

$$\mathbf{L} = L\,(-\mathbf{k}_s) \tag{2.17}$$
$$\mathbf{Y} = Y\,\mathbf{j}_s \tag{2.18}$$
$$\mathbf{D} = D\,(-\mathbf{i}_s) \tag{2.19}$$
$$\mathbf{l} = l\,\mathbf{i}_b \tag{2.20}$$
$$\mathbf{m} = m\,\mathbf{j}_b \tag{2.21}$$
$$\mathbf{n} = n\,\mathbf{k}_b \tag{2.22}$$

Es importante destacar que, a diferencia de la mayoría de las literaturas, en este caso no se define la resistencia aerodinámica en dirección igual y sentido contrario al vector de velocidad aerodinámica, es decir

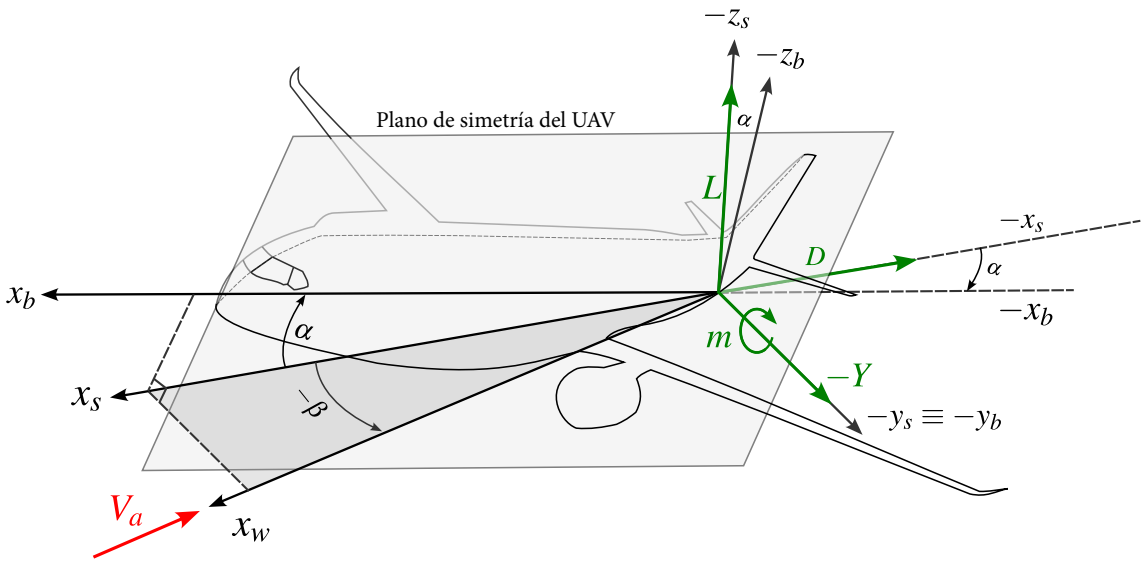


Figura 2.3 Relación entre los ejes W, los ejes S y los ejes B, dada por los ángulos α y β, así como las fuerzas aerodinámicas y el momento de cabeceo.

$\mathbf{D} = D\ (-\mathbf{i}_w)$, y por tanto $\mathbf{Y} = Y\ \mathbf{j}_w$. Esto es así debido a que los modelos del coeficiente de resistencia aerodinámica C_D han sido calculados para viento con $\beta = 0$, por tanto $x_s \equiv x_w$, tal y como se muestra en [3]. Además, podemos definir las fuerzas aerodinámicas sobre el UAV en ejes B como:

$$\mathbf{f}_{a,\text{long}}^{b} \triangleq \begin{bmatrix} X \\ 0 \\ Z \end{bmatrix} = \bar{q}S \begin{bmatrix} C_X(\alpha,q,\delta_e) \\ 0 \\ C_Z(\alpha,q,\delta_e) \end{bmatrix} \tag{2.23}$$

$$\mathbf{f}_{a,\text{lat}}^{b} \triangleq \begin{bmatrix} 0 \\ Y \\ 0 \end{bmatrix} = \bar{q}S \begin{bmatrix} 0 \\ C_Y(\beta,p,r,\delta_a,\delta_r) \\ 0 \end{bmatrix} \tag{2.24}$$

$$\mathbf{f}_a \triangleq \mathbf{f}_{a,\text{long}} + \mathbf{f}_{a,\text{lat}} \tag{2.25}$$

donde $\bar{q} \triangleq 0.5\rho(h)V_a^2$ es la presión dinámica del aire, ρ es la densidad del fluido y S es la superficie de referencia de la aeronave.

Coeficientes aerodinámicos de la dinámica longitudinal

Para relacionar las fuerzas aerodinámicas expresadas en ejes S con sus respectivas expresiones en ejes B, se realiza la siguiente transformación

$$\begin{bmatrix} X \\ 0 \\ Z \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} -L \\ 0 \\ -D \end{bmatrix} \tag{2.26}$$

Luego, podemos realizar las siguientes definiciones:

$$L \triangleq \bar{q}SC_L(\alpha,q,\delta_e) \tag{2.27}$$
$$D \triangleq \bar{q}SC_D(\alpha,q,\delta_e) \tag{2.28}$$
$$m \triangleq \bar{q}ScC_m(\alpha,q,\delta_e) \tag{2.29}$$

Los coeficientes aerodinámicos de la sustentación y del momento de cabeceo, no cambian en función de modelo del simulador a implementar, y se definen de la siguiente forma:

$$C_D(\alpha,q,\delta_e) \triangleq C_D(\alpha) + C_{D_q}\left(\frac{c}{2V_a}q\right) + C_{D_{\delta_e}}\delta_e \tag{2.30}$$
$$C_L(\alpha,q,\delta_e) \triangleq C_L(\alpha) + C_{L_q}\left(\frac{c}{2V_a}q\right) + C_{L_{\delta_e}}\delta_e \tag{2.31}$$
$$C_m(\alpha,q,\delta_e) \triangleq C_m(\alpha) + C_{m_q}\left(\frac{c}{2V_a}q\right) + C_{m_{\delta_e}}\delta_e \tag{2.32}$$

donde

$$C_L(\alpha) = C_{L_0} + C_{L_\alpha}\alpha \tag{2.33}$$
$$C_m(\alpha) = C_{m_0} + C_{m_\alpha}\alpha \tag{2.34}$$

Se puede apreciar que en el modelo elegido de $C_L(\alpha)$ no se han tomando en cuenta los efectos de la entrada en pérdida de la aeronave. Las dos versiones diferentes del modelo aerodinámica del UAV, se diferencian por la definición de el término $C_D(\alpha)$, es decir, difieren en la dependencia de la resistencia aerodinámica del ángulo de ataque y del coeficiente de sustentación.

Para el *modelo aerodinámico simplificado* se asume que existe una dependencia lineal entre el ángulo de ataque α y el coeficiente de resistencia aerodinámica, es decir

$$C_D(\alpha) = C_{D_0} + C_{D_\alpha}\alpha \tag{2.35}$$

mientras que para el *modelo aerodinámico completo de polar parabólica extendida* se tiene una dependencia polinómica y de segundo orden entre C_L y C_D, de allí que a esta curva en particular se le llame *polar parabólica*.

$$C_D(\alpha) = C_{D_p} + k_2 C_L + k_1 C_L^2 \tag{2.36}$$

Al igual que se han definido los coeficientes aerodinámicos en ejes S es necesario definirlos en ejes B. Para el coeficiente del momento de cabeceo, no existe necesidad de definirlo en otros ejes ya que $y_s \equiv y_b$.

$$C_X(\alpha,q,\delta_e) \triangleq C_X(\alpha) + C_{X_q}(\alpha)(c/2V_a)\, q + C_{X_{\delta_e}}(\alpha)\delta_e \tag{2.37}$$
$$C_Z(\alpha,q,\delta_e) \triangleq C_Z(\alpha) + C_{Z_q}(\alpha)(c/2V_a)\, q + C_{Z_{\delta_e}}(\alpha)\delta_e \tag{2.38}$$

Se puede comprobar que, finalmente, tras sustituir las definiciones (2.30),(2.31),(2.37) Y (2.38) en la ecuación (2.26), se obtiene una expresión que nos permite calcular los coeficientes aerodinámicos en ejes S (es decir, los coeficientes que deseamos identificar) en función de las fuerzas en ejes cuerpo:

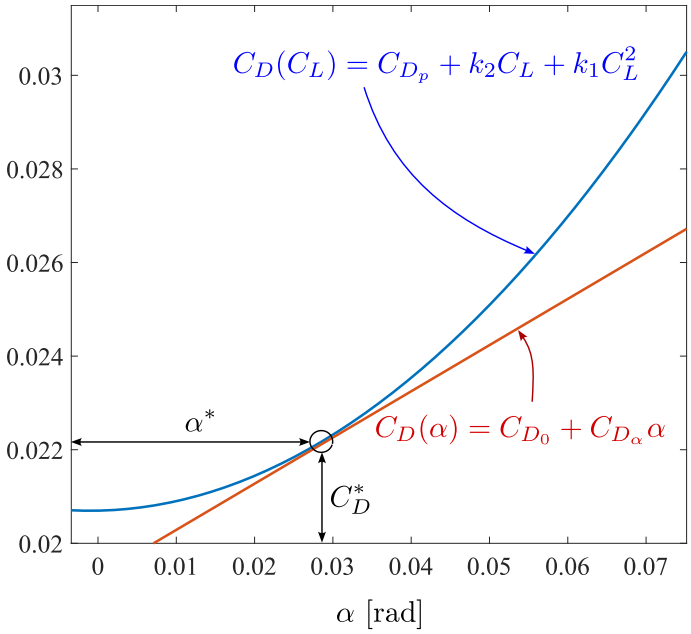


Figura 2.4 Polar parabólica y modelo linealizado..

$$\begin{bmatrix} C_X(\alpha) & C_{X_q} & C_{X_{\delta_e}} \\ C_Z(\alpha) & C_{Z_q} & C_{Z_{\delta_e}} \end{bmatrix}_{2\times 3} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}_{2\times 2} \begin{bmatrix} -C_L(\alpha) & -C_{L_q} & -C_{L_{\delta_e}} \\ -C_D(\alpha) & -C_{D_q} & -C_{D_{\delta_e}} \end{bmatrix}_{2\times 3} \tag{2.39}$$

Coeficientes aerodinámicos de la dinámica lateral

En la dinámica lateral de la aeronave se pueden definir

$$Y \triangleq \bar{q}SC_Y(\beta,p,r,\delta_a,\delta_r) \tag{2.40}$$
$$l \triangleq \bar{q}SbC_l(\beta,p,r,\delta_a,\delta_r) \tag{2.41}$$
$$n \triangleq \bar{q}SbC_n(\beta,p,r,\delta_a,\delta_r) \tag{2.42}$$

donde

$$C_Y(\beta,p,r,\delta_a,\delta_r) \triangleq C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}(b/2V_a)\ p + C_{Y_r}(b/2V_a)\ r + C_{Y_{\delta_r}}\delta_r + C_{Y_{\delta_a}}\delta_a \tag{2.43}$$
$$C_l(\beta,p,r,\delta_a,\delta_r) \triangleq C_{l_0} + C_{l_\beta}\beta + C_{l_p}(b/2V_a)\ p + C_{l_r}(b/2V_a)\ r + C_{l_{\delta_r}}\delta_r + C_{l_{\delta_a}}\delta_a \tag{2.44}$$
$$C_n(\beta,p,r,\delta_a,\delta_r) \triangleq C_{n_0} + C_{n_\beta}\beta + C_{n_p}(b/2V_a)\ p + C_{n_r}(b/2V_a)\ r + C_{n_{\delta_r}}\delta_r + C_{n_{\delta_a}}\delta_a \tag{2.45}$$

Debido a la simetría de la mayoría de las aeronaves de ala fija, se suele considerar $C_{Y_0} = C_{l_0} = C_{n_0} \approx 0$.

2.2.3 Modelo propulsivo

Como se había comentado anteriormente, existen dos modelos propulsivos con distinto grado de detalle, que se implementan en los diferentes simuladores en función de la complejidad que se desee en cada modelo. A continuación se detalla cada uno de ellos.

Una hipótesis de gran importancia en el modelo es que el eje de la hélice del UAV se encuentra perfectamente alineado con el eje x_b, de forma que la fuerza de empuje cumple que $\mathbf{T} = T\mathbf{i}_b$ y el momento generado por el motor viene dado por $\mathbf{Q} = Q\mathbf{i}_b$, evitando así la aparición de momentos adicionales generados cuando

el eje de la fuerza de empuje no atraviesa el centro de masas de la aeronave.

Modelo sencillo y Teoría Ideal de Froude

El modelo propulsivo se basa en la *Teoría Ideal de Froude* descrita en [4], pero aun más simplificada. Se considera la hélice un disco infinitamente delgado de superficie S_{prop} a través del cual existe una diferencia de presiones. Esta diferencia de presiones define la tracción de la hélice, y debe de suministrarse una cierta potencia a la misma para mantener esta diferencia de presiones. Ver Figura 2.5. Se realizan las siguientes hipótesis:

- Se asume que toda la energía suministrada a la hélice se transfiere al aire (hélice sin masa, indeformable).
- Se asume que la velocidad no presenta discontinuidad a través del disco, se asume que la hélice no ofrece resistencia al paso del aire.
- Hipótesis del principio de Bernoulli (fundamental en la aerodinámica potencial):
  - Flujo unidimensional a través de la vena fluida.
  - Flujo estacionario ($\frac{\partial}{\partial t} = 0$).
  - Flujo incompresible ($\rho(x,y,z,t) = \text{const}$).
  - Flujo no viscoso ($\mu \approx 0$). No se tiene en cuenta la resistencia aerodinámica de las palas de la hélice real.

Este modelo predice que el empuje es igual a la variación de la cantidad de movimiento del flujo de aire, y posee la siguiente expresión:

$$T = \dot{m}_{\text{air}}(V_{\text{exit}} - V_a) = \left(P_{\text{downstream}} - P_{\text{upstream}}\right) \cdot S_{\text{prop}} \tag{2.46}$$

donde:

- V_{exit}: velocidad del flujo unidimensional aguas abajo cuando el fluido se expande y recupera la presión de remanso P_0, de forma que posee una presión $P_{\text{downstream}} = P_0 + \frac{1}{2}\rho V_{\text{exit}}^2$.
- V_a: velocidad del flujo sin perturbar aguas arriba. Si la atmósfera se encuentra estática (en ausencia de viento), la presión del fluido es $P_{\text{upstream}} = P_0 + \frac{1}{2}\rho V_a^2$.
- ρ : densidad del aire.

Para simplificar aún más el modelo presentado en la ecuación (2.46):

- Se asume que la velocidad del fluido inmediatamente después de la hélice es directamente proporcional a la señal de control de la palanca de potencia (comúnmente llamada *palanca de gases*, indiferentemente de si el motor de la aeronave es eléctrico o de combustión). Es decir: $V_{\text{exit}} = k_{\text{motor}}\delta_t$. La constante de proporcionalidad se puede determinar mediante experimentación.
- Se añade un factor de corrección C_{prop} que permite capturar de forma aproximada los efectos de las irreversibilidades despreciadas en las hipótesis anteriores.

Finalmente el modelo de empuje utilizado queda como:

$$T = \frac{1}{2}\rho S_{\text{prop}} C_{\text{prop}} \left[(k_{\text{motor}}\delta_t)^2 - V_a^2\right] \tag{2.47}$$

Debido a que no es posible estimar las RPM del motor, se toma como hipótesis que el torque del mismo es despreciable frente a los demás torques actuando sobre el eje x_b, es decir $Q \approx 0$.

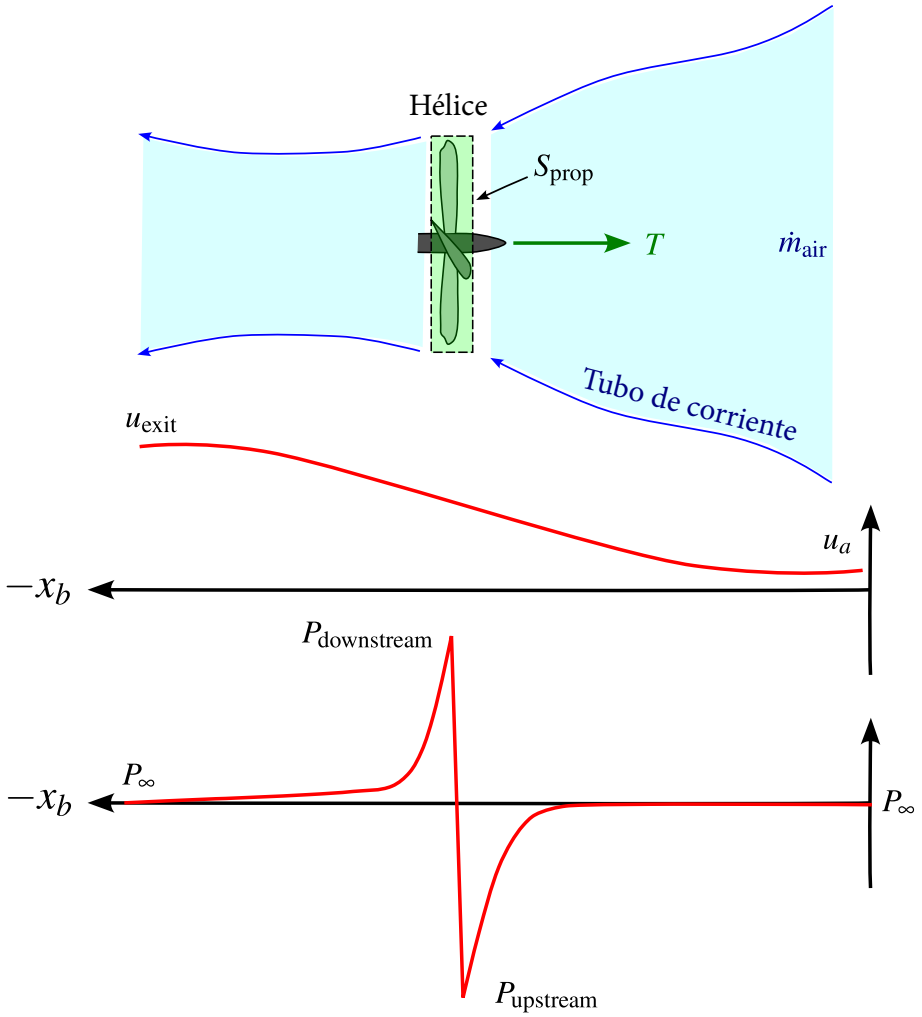


Figura 2.5 Teoría ideal de Froude.

Modelo complejo de hélice

Las RPM del motor se modelan como una función lineal de la palanca de gases δ_t de forma que se puedan expresar se la siguiente forma:

$$\text{RPM} = 9000 \times \delta_t \tag{2.48}$$

donde se asumen que el motor posee un máximo de 9000 RPM. De igual manera, este sencillo modelo aproxima apropiadamente el comportamiento de cualquier tipo de motor, eléctricos como de combustión, proporcionando un grado apropiado de flexibilidad al simulador para implementar modelos de aeronaves con diferentes tipos de motores.

Posteriormente, se procede a realizar el modelado de la hélice del UAV mediante los datos proporcionados por APC. Tomando en cuenta que la hélice utilizada por la aeronave es una APC 10×6, se pueden utilizar los datos de performance en túnel de viento, disponibles a través de un archivo `.csv` disponible en el sitio web de la empresa. De forma que conocidos los valores del empuje T y el torque Q que proporciona la hélice dadas unas RPM y una velocidad aerodinámica incidente u, se puede realizar una interpolación lineal de valores en dos dimensiones (RPM,u), generando así sendas superficies para el empuje y el torque del motor, como se muestra en la Figuras 2.6 y 2.7.

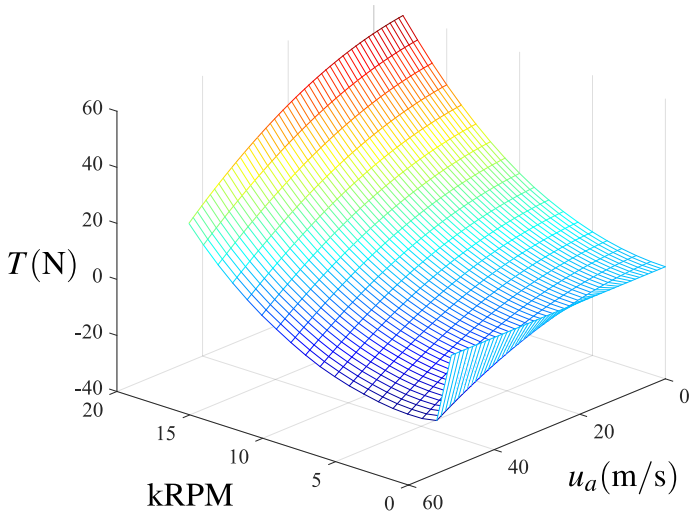


Figura 2.6 Empuje proporcionado por la hélice, T, en función de las RPM del motor y la velocidad incidente del viento u_a.

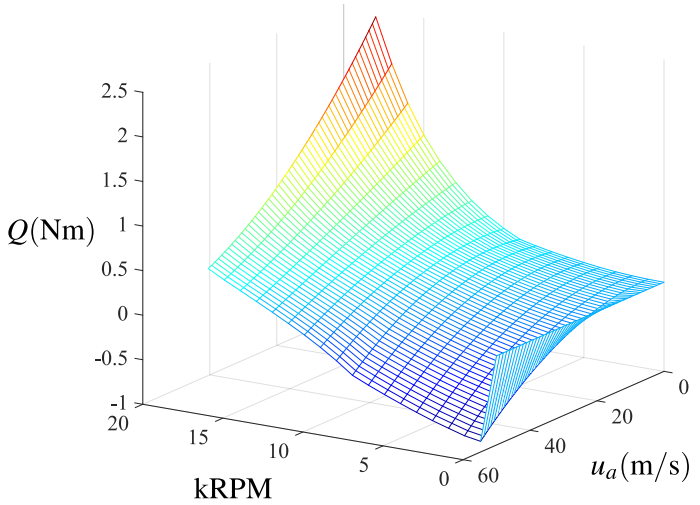


Figura 2.7 Torque requerido por la hélice, Q, en función de las RPM del motor y la velocidad incidente del viento u_a.

Modelado del motor eléctrico BLDC

Los motores más comunmente usados en UAVs pequeños son los BLDC (del inglés *Brush-Less Direct Current* o *de corriente directa sin escobillas*). Este tipo de motor se modela con la siguiente ecuación:

$$Q_{\text{motor}} = Q_{\text{prop}} = K_Q \cdot I_{\text{motor}} = K_Q \left[\frac{V_{\text{in}} - K_V \Omega}{R} - i_0 \right] \tag{2.49}$$

donde R es la resistencia del bobinado del motor, K_Q es la constante de par del motor (que relaciona linealmente el par motor con la corriente que circula por el mismo), K_V es la constante de fuerza electromotriz inversa (back-EMF) del motor e i_0 es la corriente de vacío del motor (usualmente despreciable).

El modelo asume que el torque del motor es proporcional a la corriente de entrada del mismo, I_{motor}. Esta corriente es la suma de la corriente que atraviesa el bobinado del motor y por ende genera un movimiento de rotación, y de otra corriente de vacío o pérdidas que suele ser despreciable. Finalmente la corriente que atraviesa las bobinas del motor se corresponde con la diferencia entre la tensión de entrada del motor y la fuerza electromotriz inducida por el campo magnético de los imanes permanentes del mismo, dividido por la resistencia de las bobinas.

El valor de esta fuerza electromotriz se asume que es proporcional a la velocidad de rotación del motor Ω, donde la constante de proporcionalidad es K_V. El valor de esta constate suele venir dada por los fabricantes de motores de la siguiente manera: $[1/K_V]$ =RPM/V. Esta fuerza electromotriz surge debido a la variación del campo magnético generado por los imanes permanentes del motor (y por ende, la variación del flujo magnético ϕ_B) en el interior de las bobinas del motor, lo cual genera un efecto inductivo, que se opone al cambio del flujo magnético. Por esta razón, al aumentar la velocidad de giro del motor, aumenta la velocidad de variación del flujo, es decir aumenta el valor de su derivada $d\phi_B/dt$, lo que genera una tensión más alta.

Cabe destacar que se ha asumido que el voltaje en bornas entregado por la ESC (*Electronic Speed Controller* o variador) cumple

$$V_{\text{in}} = V_{\text{max}}\ \delta_t \tag{2.50}$$

donde V_{max} es la tensión máxima de a batería. Hay que tomar en cuenta que por la descarga de la batería, esta tensión máxima va disminuyendo con el pasar del tiempo, es decir: $V_{\text{max}} = V_{\text{max}}(t)$, $\dot{V}_{\text{max}} < 0$. Este efecto también se desprecia en el presente estudio.

Por último, se asume que no existe deformación en el rotor, de forma que el torque generado por el motor es directamente el transmitido a la hélice o que $Q_{\text{motor}} = Q_{\text{prop}}$.

2.3 Modelo ambiental

Los principales parámetros ambientales a determinar a lo largo de la simulación son la densidad del aire y la aceleración gravitatoria. Los modelo que predicen sus valores se detallan a continuación.

2.3.1 Modelo gravitatorio

El modelo gravitatorio se basa en la distribución uniforme de la masa (equipotencial) del modelo elipsoidal de la tierra WGS84 (World Geodetic System 1984) utilizado para modelar la forma de la Tierra y definir coordenadas de ubicaciones geográficas en su superficie (desarrollado y mantenido por el Departamento de Defensa de los Estados Unidos de América y la Agencia Nacional de Inteligencia Geoespacial). Se realiza una aproximación mediante Series de Taylor ya que no se necesita exagerada precisión en el modelo para trabajar con drones de corto alcance y baja altitud, ganando ligereza en la simulación.

Posteriormente, se detalla a implantación de este modelo en `Simulink`.

2.3.2 Modelo atmosférico

La velocidad de la aeronave vista desde tierra $\mathbf{V}$ (ground velocity) es la suma de la velocidad del avión respecto a la masa de aire circundante V_a, también llamada velocidad aerodinámica (en inglés, *air velocity*) más la velocidad de dicha masa de aire respecto a tierra $\mathbf{V}_w$, es decir, la velocidad del viento (en inglés, *wind velocity*).

$$\mathbf{V} = \mathbf{V}_a + \mathbf{V}_w \tag{2.51}$$

Cabe destacar que la "*airspeed*" el módulo o magnitud de la velocidad aerodinámica: $|\mathbf{V}_a| = V_a = \sqrt{u_a^2 + v_a^2 + w_a^2} \equiv$ airspeed.

La velocidad del viento puede descomponerse en una componente determinista y mensurable $\mathbf{V}_{w_s}$ (steady wind $\partial/\partial t = 0$) y una aleatoria $\mathbf{V}_{w_g}$ (también llamada *random gusts*, o turbulencias):

$$\mathbf{V} = \mathbf{V}_a + \mathbf{V}_w = \mathbf{V}_a + \mathbf{V}_{w_s} + \underbrace{\mathbf{V}_{w_g}}_{\text{Random}} = \begin{bmatrix} u_a \\ v_a \\ w_a \end{bmatrix}_b + \begin{bmatrix} u_w \\ v_w \\ w_w \end{bmatrix}_b = \begin{bmatrix} u_a \\ v_a \\ w_a \end{bmatrix}_b + \underbrace{\begin{bmatrix} w_{n_s} \\ w_{e_s} \\ w_{d_s} \end{bmatrix}_e + \begin{bmatrix} u_{w_g} \\ v_{w_g} \\ w_{w_g} \end{bmatrix}_b}_{\mathbf{V}_w} \tag{2.52}$$

El steady wind $\mathbf{V}_{w_s}$ puede dividirse en dos componentes: una componente responsable de los vientos cortantes o de cizalladura $\mathbf{V}_{w_{\text{shear}}}$(también llamada *shear wind*) y una componente de viento uniforme en todo el espacio $\mathbf{V}_{w_{\text{unif}}}$ ($\nabla(\cdot) = [0,0,0]^T$):

$$\mathbf{V} = \mathbf{V}_a + \mathbf{V}_{w_s} + \mathbf{V}_{w_g} = \mathbf{V}_a + \mathbf{V}_{w_{\text{shear}}} + V_{w_{\text{unif}}} + \mathbf{V}_{w_g} = \begin{bmatrix} u_a \\ v_a \\ w_a \end{bmatrix}_b + \begin{bmatrix} w_n^{\text{shear}} \\ w_e^{\text{shear}} \\ w_d^{\text{shear}} \end{bmatrix}_e + \begin{bmatrix} w_n^{\text{unif}} \\ w_e^{\text{unif}} \\ w_d^{\text{unif}} \end{bmatrix}_e + \begin{bmatrix} u_{w_g} \\ v_{w_g} \\ w_{w_g} \end{bmatrix}_b \tag{2.53}$$

3 Implementación de modelos en Simulink

El modelado en el entorno Simulink se apoya en las arquitecturas de bloques descritas por Mendoza y Fernández [5], y por MathWorks [6], realizando las modificaciones pertinentes para ajustarse a los objetivos del presente trabajo.

3.1 Generalidades

Los modelos de físicos anteriormente mencionados, se implementan finalmente mediante bloques de Simulink, los cuales en función de como se agrupen, dan lugar a los FWS V1, V2 o V3. Es entonces la herramienta Simulink, la que se encarga de resolver un *sistema de ecuaciones algebraica-diferenciales (DAEs, Differential Algebraic Equations) no lineales*, planteado en forma de Diagrama de Bloques.

Posteriormente en el presente capítulo, se detalla como se realiza esta implementación, pero antes, se pretenden explicar los aspectos operativos (o de uso) de los diferentes simuladores desarrollados.

3.1.1 Realización de una simulación simple

Para simular un vuelo del UAV, es necesario abrir el modelo simulador2017b_v*.slx, donde el símbolo (*) ha de sustituirse por 1, 2 o 3, en función de la versión del simulador con la que se desee trabajar. En estos modelos, es donde se resuelve el sistema *DAE* no lineal antes mencionado, para determinar las variables de estado $\mathbf{x}$ y las variables algebraicas $\mathbf{x}_a$ del vuelo. Es decir:

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{x}_a,t) \\ 0 = g(\mathbf{x},\mathbf{x}_a,t) \end{cases} \tag{3.1}$$

Un ejemplo de una variable algebraica puede ser la velocidad aerodinámica $\mathbf{V}_a$ o el ángulo de ataque α. Además, cabe destacar que el vector de estados del avión $\mathbf{x}$, se representa en el simulador con la señal X VEC. Ver Figura3.1.

Los valores iniciales de las variables de estado, son por defecto los correspondientes al trimado de un vuelo en crucero a altitud constante $h = 100\,\text{m}$. Los valores iniciales de los estados se establecen dentro de la *Mask* del bloque UAV. Ver Figura 3.2.

Las entradas del sistemas se encuentran a la izquierda del bloque UAV, donde se especifica el nombre de cada entrada y el rango de valores que admite. Las salidas se encuentran a la derecha. Existen 4 salidas en total.

- AC BUS. Bus que maneja todos los datos del avión, incluyendo variables de estado $\mathbf{x}$ y variables algebraicas $\mathbf{x}_a$.
- X LAT BUS. Variables de estado de la dinámica lateral del avión $\mathbf{x}_{\text{lat}}$.
- X LONG BUS. Variables de estado de la dinámica longitudinal del avión $\mathbf{x}_{\text{long}}$.
- X BUS. Conduce a las variables de estado del UAV, $\mathbf{x}$.

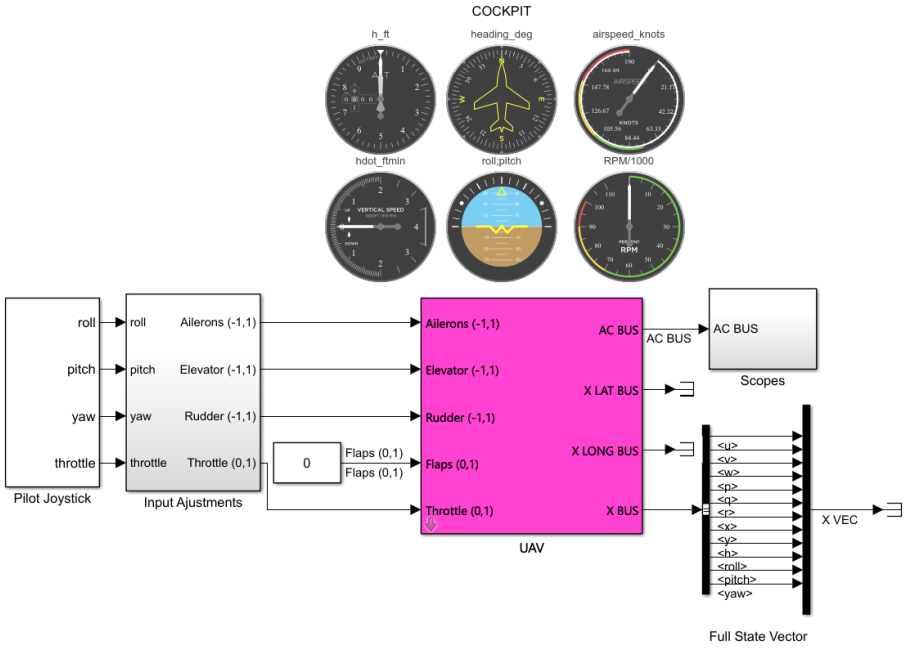


Figura 3.1 Modelo de UAV de propulsión por hélice con sus entradas y salidas. El modelo presenta una serie de instrumentos en la parte superior (`COCKPIT`) para comprobar el estado del vuelo mientras transcurre la simulación.

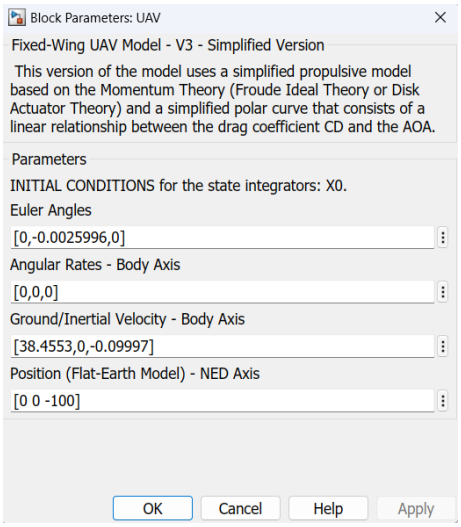


Figura 3.2 *Mask* del bloque `UAV`.

El bloque `Scopes` aloja una serie de monitores (o en inglés `scope`) para visualizar los datos del vuelo en función del tiempo.

3.1.2 Carga y configuración de un UAV específico para simulación

Todos los parámetros de la aeronave, se modifican desde el archivo `config_model.m`. Los únicos parámetros que se modifican directamente desde los menús contextuales (*mask*) de los bloques, son los del modelo de vientos. Cabe destacar que los parámetros almacenados en este archivo de configuración son globales para los 3 modelos de simulador. Además existen 2 archivos de configuración, exactamente idénticos; uno sirve a los simuladores de la carpeta `fixed_wing_uav`, y el otro sirve a los modelos para identificación de la carpeta `fixed_wing_system_identification`.

Los modelos almacenados en estas carpetas, son exactamente iguales. Lo que varía en cada una de las

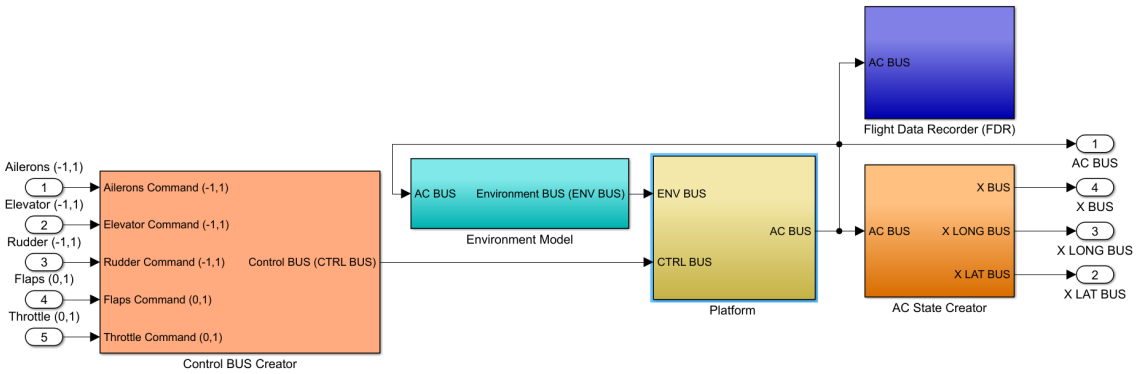


Figura 3.3 Diagrama de bloques dentro del bloque `UAV`. Se tienen 5 bloques principales: `Control BUS Creator`, `AC State Creator`, `Environment Model`, `Flight Data Recorder` y `Platform`. El bloque `Platform` integra toda la dinámica del UAV asi como también modela el sistema de datos de aire (bloque `Platform/Air Data System`) para el cálculo del AOA, el *sideslip angle* y el módulo de la velocidad del avión respecto al aire (en inglés *airspeed*). El bloque `AC State Creator` es un bloque auxiliar que crear los buses de estado para un fácil manejo de estos datos.

carpeta es la presencia o no de ciertos scripts necesarios para la identificación de parámetros a partir de datos de vuelo.

Por defecto se tiene configurado como UAV por defecto el Aerosonde UAV. Los parámetros de este avión se encuentran disponibles en el libro *Small Unmanned Aircraft: Theory and Practice* de Randy Beard y Tim McLain [2]. A esta plataforma se le han realizado ciertas modificaciones que permiten realizar vuelos con los 3 modelos de simulador disponibles.

- **Modelo aerodinámico**. La aerodinámica longitudinal de la aeronave, es decir, los coeficientes de sustentación C_L y de resistencia aerodinámica C_D, se corresponden con los de la aeronave Jaleo-I, del equipo VANTUS AeroDesign Team de la Escuela Técnica Superior de Ingeniería. Los detalles del cálculo de estos coeficientes se encuentra en el Informe Técnico del Jaleo-I [3] realizado para la competencia inter-universitaria XtraChallenge 2023.
- **Modelo propulsivo**. Para los modelos de la propulsión por hélice se los modelos `V2` Y `V1`, se ha utilizado el modelo ajustado con datos en túnel de viento proporcionados por la empresa APC para la hélice APC 10×6E, la cual se utilizó también en el Jaleo-I de VANTUS, y cuyos datos ya habían sido procesados por mi persona para realizar el análisis de la propulsión y actuaciones de la aeronave mencionada anteriormente. Para el caso del `V3` se a ha utilizado un modelo sencillo de empuje proporcionado en el libro de referencia *Small Unmanned Aircraft: Theory and Practice*, correspondiente al Aerosonde UAV.

Como es evidente, las 3 versiones del simulador presentan aeronaves diferentes, ya que *en este estudio en particular no es de interés hacer una comparativa entre modelos* de aeronaves, sino que se busca evaluar la capacidad de identificación de las aeronaves para distintos grados de complejidad en el modelo. Para más detalles sobre los parámetros del Aerosonde UAV, ver la Tabla 3.1, mientras que para los parámetros de interés del Jaleo-I ver la Tabla 3.2.

El script de configuración se divide en diferentes secciones:

1. `Load Propulsion Data`. Se cargan las *Look-Up tables* del empuje y torque de la hélice del vehículo (`data_PER310x6E_arrays.mat`) mediante el script `LOOKUP_TABLES.m`. Cada hélice del fabricante tiene asociada un archivo `.csv` con datos provenientes de ensayos experimentales. Estos datos deben ser procesados antes de poder ser utilizados (este proceso es el que genera el archivo `.mat` mencionado anteriormente). Este procesado no se detalla en el presente trabajo, sino que se ha importado el archivo `.mat` directamente del proyecto Jaleo-I de VANTUS.
2. `Geometry and mass properties`. Propiedades geométrica y másicas del UAV, como pueden ser: MTOW (`mass`), la superficie de referencia del modelo (`S`), la envergadura de las alas (`b`), la cuerda media del ala (`c_bar`) o el tensor de inercia (`J`).

Tabla 3.1 Parámetros del AerosondeUAV.

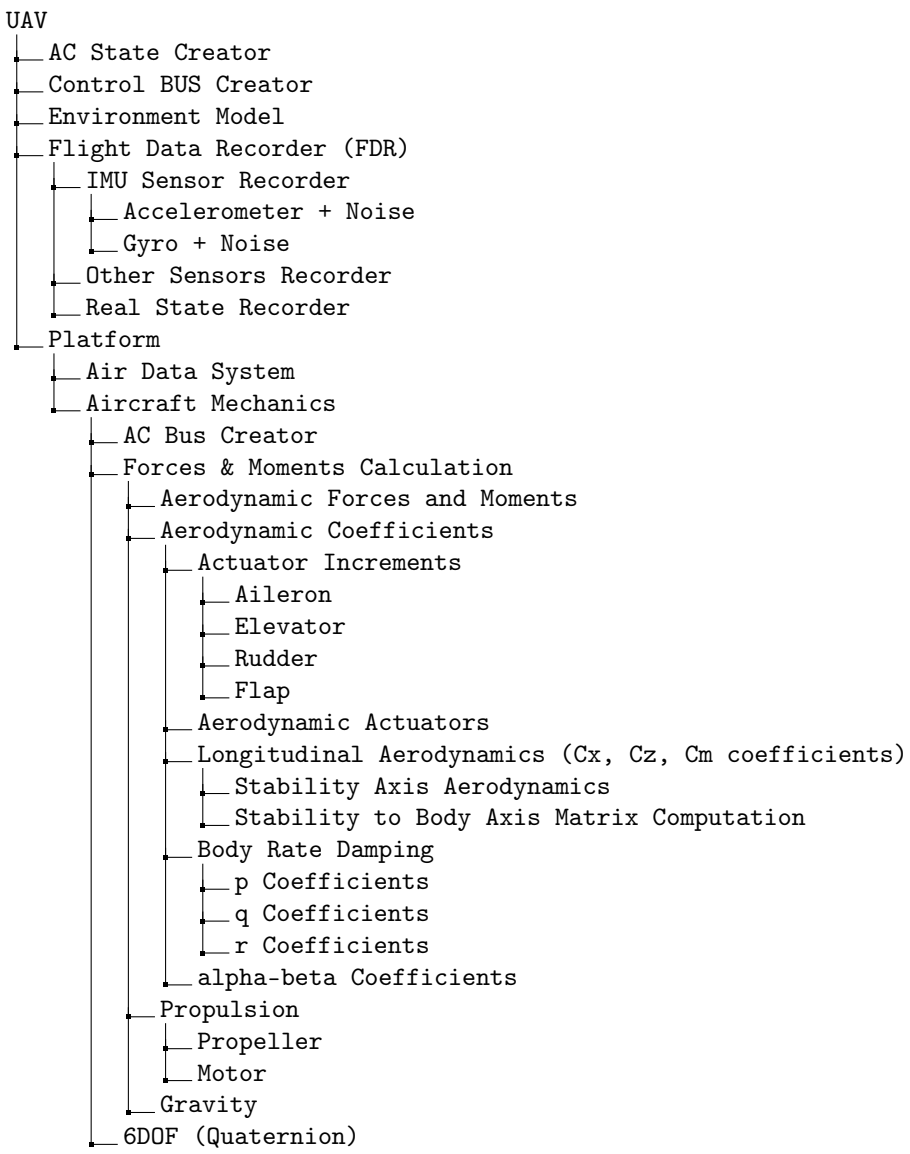
Parámetros	Valores	Parámetros	Valores
m	11 kg	C_{Y_0}	0
J_x	0.824 kg m^2	C_{l_0}	0
J_y	-1.135 kg m^2	C_{n_0}	0
J_z	-1.759 kg m^2	$C_{Y_{\delta_a}}$	0.075
J_{xz}	-0.120 kg m^2	$C_{l_{\delta_a}}$	0.17
S	0.55 m^2	$C_{n_{\delta_a}}$	-0.011
b	2.9 m	$C_{m_{\delta_e}}$	-0.99
$\bar{c}$	0.19 m	$C_{X_{\delta_r}}$	0
$\delta_{\max}$	0.35 rad	$C_{Y_{\delta_r}}$	0.19
$\text{RPM}_{\max}$	11,000	$C_{l_{\delta_r}}$	0.0024
C_{prop}	1	$C_{n_{\delta_r}}$	-0.069
k_{motor}	80	$C_{X_{\delta_f}}$	0
S_{prop}	0.196 m^2	$C_{Z_{\delta_f}}$	0
i_0	1.5 A	$C_{m_{\delta_f}}$	0
$V_{\max}$	44.4 V	$C_{Y_{\delta_p}}$	0
R	0.042 Ω	$C_{l_{\delta_p}}$	-0.51
K_V	0.0659 V-s/rad	$C_{n_{\delta_p}}$	-0.069
K_Q	0.0659 N-m	C_{Y_r}	0
C_{D_q}	0	C_{l_r}	0.25
$C_{D_{\delta_e}}$	0.0135	C_{n_r}	-0.095
C_{L_q}	7.95	$C_{Y_{\delta_p}}$	0
$C_{L_{\delta_e}}$	0.13	$C_{l_{\delta_p}}$	-0.51
C_{m_0}	0.0135	C_{m_q}	-38.21
C_{m_α}	-2.74	C_{n_β}	0.073
C_{Y_β}	-0.83	C_{l_β}	-0.13

Tabla 3.2 Parámetros del Jaleo-I.

Parámetros	Valores
$C_{L_{\text{cruise}}}$	0.38797
α_{cruise}	0.0282
k_1	0.0537
k_2	-0.0241
C_{D_p}	0.0234
C_{D_0}	0.0193
C_{D_α}	0.0987
C_{L_0}	0.23
C_{L_α}	5.61

3. `Simulation initial conditions and atmospheric conditions`. Latitud y longitud iniciales del modelo para la determinación de la aceleración gravitatoria en dicho lugar, así como los parámetros que definen las ráfagas de viento que se deseen simular durante el vuelo. El modelo de vientos y atmosférico se detalla más adelante.
4. `Propulsion`. Parámetros propulsivos como la superficie frontal de la hélice (`Sprop`) o la constante propulsiva del modelo (`C_prop`).
5. `Aerodynamics`. Coeficientes adimensionales (derivadas de estabilidad) que definen todo el comportamiento aerodinámico del UAV.

3.1.3 Diagrama de árbol del modelo de Simulink



3.2 Buses de datos

A continuación se presenta una descripción breve de las diferentes señales que viajan a través de los buses de datos más importantes del simulador, señalando sus unidades físicas como su símbolo matemático y nombre de la señal en Simulink.

3.2.1 Control BUS (CTRL BUS)

Maneja las señales de control del avión en unidades del sistema internacional y con saturación al ser creadas, de forma que se encuentran acotadas en sus respectivos valores de operación (de los que se detalla en el capítulo siguiente).

3.2.2 Air Data BUS (AD BUS)

Maneja las señales asociadas a el flujo del aire incidente, utilizadas para el cálculo de las fuerzas aerodinámicas.

Tabla 3.3 Señales de control - `Control BUS (CTRL BUS)`.

Nombre	Símbolo	Unidades	Descripción
`deltaa`	δ_a	rad	Deflexión de los alerones
`deltae`	δ_e	rad	Deflexión del elevador
`deltar`	δ_r	rad	Deflexión del rudder
`deltaf`	δ_f	rad	Deflexión de los flaps
- `deltat`	δ_t	-	Palanca de gases (de 0 a 1)

Tabla 3.4 Datos de aire - `Air Data BUS (AD BUS)`.

Nombre	Símbolo	Unidades	Descripción
`alpha`	α	rad	Ángulo de Ataque (AOA)
`beta`	β	rad	Sideslip Angle
`Va`	$\|\mathbf{V}_a\| = V_a$	m/s	Módulo de la Velocidad Aerodinámica
`pqr_tot`	$\boldsymbol{\omega}^b_{b/a} = [p,q,r]^T_b$	rad/s	Velocidad Angular Total
`q_bar`	$\bar{q} = \frac{1}{2}\rho V^2$	Pa	Presión Dinámica
`Va - Aerodynamic Velocity`	$\mathbf{V}^b_a = [u_a,v_a,w_a]^T_b$	m/s	Velocidad Aerodinámica

3.2.3 `Propulsion Data BUS (PD BUS)`

Maneja las señales asociadas al sistema propulsivo del avión. En la versión más sencilla del simulador (`V3`) la única señal no nula es la del empuje, debido a que no se posee un modelo para las RPM del motor del UAV y se ha supuesto un torque nulo alrededor del eje x_b.

Tabla 3.5 Señales de control - `Propulsion Data BUS (PD BUS)`.

Nombre	Símbolo	Unidades	Descripción
`RPM`	RPM	-	Revoluciones por minuto del motor
`Thrust`	T	N	Empuje
`Torque`	Q	Nm	Torque del motor

3.2.4 `Wind BUS (WIND BUS)`

Maneja las señales relativas al viento generadas en el bloque `Wind Models`, del que se hablará más adelante. Estas señales son utilizadas para calcular las fuerzas aerodinámicas o la velocidad respecto a tierra.

Tabla 3.6 Señales del modelo de vientos - `Wind BUS (WIND BUS)`.

Nombre	Símbolo	Unidades	Descripción
`uvw_wind`	$\mathbf{V}^b_w = [u_w, v_w, w_w]^T_b$	m/s	Velocidad del viento respecto a tierra
`pqr_wind`	$\boldsymbol{\omega}^b_{w/n} = [p_w, q_w, r_w]^T_b$	rad/s	Velocidad Angular del viento respecto a tierra

3.2.5 `Environment BUS (ENV BUS)`

Maneja las señales asociadas al modelo ambiental sobre el que se desenvuelve el UAV (es decir, información relativa al modelo gravitatorio y al modelo atmosférico).

3.2.6 `Aerodynamic BUS (AERO BUS)`

Maneja todos las señales correspondientes a los coeficientes aerodinámicos de fuerzas y momentos en ejes cuerpo. Es utilizado para la generación de datos sintéticos de vuelo. Además, este bus es creado en el bloque

Tabla 3.7 Señales de datos ambientales - Environment BUS (ENV BUS).

Nombre	Unidades	Descripción
g	m/s^2	Aceleración gravitatoria
rho	kg/m^3	Densidad del aire
Wind BUS	-	-

AC Bus Creator. Ver Tabla 3.8.

Tabla 3.8 Señales de los coeficientes aerodinámicos Aerodynamic BUS (AERO BUS).

Nombre	Símbolo	Unidades	Descripción
CX	C_X	-	Coeficiente de fuerza en x_b
CY	C_Y	-	Coeficiente de fuerza en y_b
CZ	C_Z	-	Coeficiente de fuerza en z_b
Cl	C_l	-	Coeficiente de momento en x_b
Cm	C_m	-	Coeficiente de momento en y_b
Cn	C_n	-	Coeficiente de momento en z_b

3.2.7 Aircraft BUS (AC BUS)

Maneja todos los datos del simulador. Integra a todos los demás buses con los datos provenientes del bloque 6DOF (Quaternion) relativos a la dinámica de la aeronave.

Tabla 3.9 Señales del estado y demás buses - Aircraft BUS (AC BUS).

Nombre	Símbolo	Unidades	Descripción
xyh	$[x,y,h] = [x,y,-z]$	m	Posición en el modelo de tierra plana
xdot ydot zdot	$\mathbf{V}_g^n = [\dot{x},\dot{y},\dot{z}]_n^T$	m/s	Velocidad respecto a tierra (Ejes N)
uvw	$\mathbf{V}^b = [u,v,w]_b^T$	m/s	Velocidad respecto a tierra (Ejes B)
hdot	$\dot{h}$	m/s	Climb Rate
phi theta psi	$[\phi,\theta,\psi]^T$	rad	Ángulos de Euler
DCMbe	$\mathbf{DCM}_a^b$	-	Matriz de Coseno Directores ($A \rightarrow B$)
pqr	$\boldsymbol{\omega}_{b/a}^b = [p,q,r]_b^T$	rad/s	Velocidad Angular Total
Control BUS	-	-	-
Air Data BUS	-	-	-
Propulsion Data BUS	-	-	-
Environment BUS	-	-	-
Aerodynamic BUS	-	-	-

3.3 Bloques de Simulink más importantes

3.3.1 Condicionado de las señales de control - Control BUS Creator

Este bloque se encarga de crear el bus de datos CTRL BUS , preparando las señales de entrada (PWM en rangos entre 0 y 1, y entre -1 y 1) para ser implementadas en la ecuaciones dinámicas. Ver Figura 3.4.

1. Las señales PWM de los alerones, elevador y rudder (Ailerons Command (-1,1), Elevator Command (-1,1), Rudder Command (-1,1)) se multiplican por $\delta_{\max} > 0, [\delta_{\max}] \equiv$ rad, de forma que se gene-

ran las señales $\delta_a, \delta_e, \delta_r \in (-\delta_{\max}, \delta_{\max})$. Este parámetro determina la deflexión máxima de las superficies de control en radianes. Se especifica en el archivo de configuración del modelo `config_model.m`.

2. La señal PWM de los flaps puede ir de 0 a 1. Los flaps tienen un bloque de saturación y un *rate limiter* para emular la variación progresiva del ángulo de los flaps δ_f. Además las posiciones de los flaps están limitadas a 0 %, 20 %, 60 % o 100 % de la deflexión máxima posible, es decir $\delta_f \in [0,\ 0.2 \times \delta_{\max},\ 0.6 \times \delta_{\max},\ \delta_{\max}]$.

3. La señal de la palanca de potencia del motor (`throttle`) solo es pasada por un saturador para prevenir que esta escape del rango permitido, $\delta_t \in (0,1)$.

4. Finalmente se crea el bus de control `CTRL BUS`.

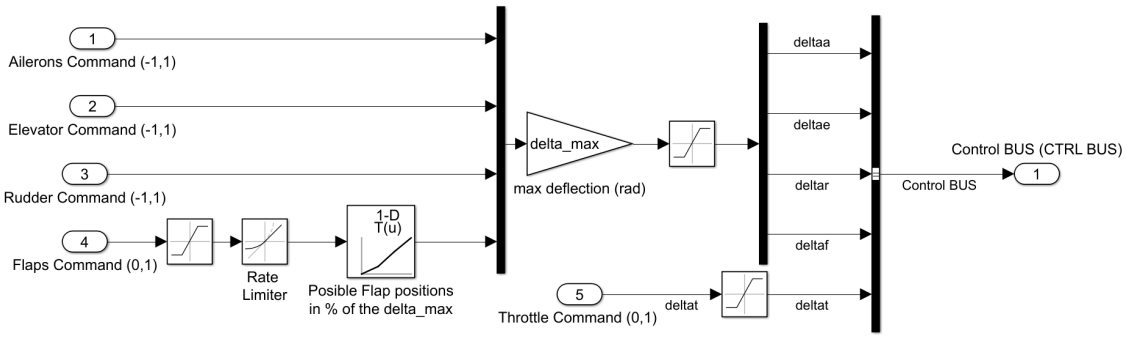


Figura 3.4 Diagrama de bloques del bloque `Control BUS Creator`.

3.3.2 Modelos de fuerzas y momentos - `Forces & Moments Calculation`, `6DOF (Quaternion)` y `AC BUS Creator`

El bloque `6DOF (Quaternion)` del `Aerospace Blockset` de `Simulink`, recibe como entradas los vectores de fuerzas `Fxyz (N)` y momentos `Mxyz (N-m)` totales sobre el UAV en ejes *B*, y realiza la integración de las ecuaciones (2.8), (2.11), (2.13) y (2.15). A la salida de este bloque se encuentran las variables de estado que conforman el vector de estado **x**. Ver Figura 3.5.

El bloque `Forces & Moments Calculation` calcula los vectores de fuerza y momentos comentados anteriormente. En sus distintos bloques internos se realizan los cálculos de las fuerzas y momentos aerodinámicos, propulsivos y la fuerza gravitatoria, para las 3 componentes de los ejes *B*. Además cabe destacar que por la diferencia que existe en el modelo propulsivo de la versión simplificada `V3`, las entradas del bloque encargado de la propulsion del UAV son diferentes. Comparar las Figuras 3.6 y 3.7.

Los vectores `Fxyz (N)` y `Mxyz (N-m)` se definen como:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = mg \begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} + \frac{1}{2}\rho V_a^2 S \begin{bmatrix} C_X(\alpha) + C_{X_q}(\alpha)(c/2V_a)\, q + C_{X_{\delta_e}}(\alpha)\delta_e \\ C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}(b/2V_a)\, p + C_{Y_r}(b/2V_a)\, r + C_{Y_{\delta_r}}\delta_r + C_{Y_{\delta_a}}\delta_a \\ C_Z(\alpha) + C_{Z_q}(\alpha)(c/2V_a)\, q + C_{Z_{\delta_e}}(\alpha)\delta_e \end{bmatrix} + \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} l \\ m \\ n \end{bmatrix} = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} b\left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}(b/2V_a)\, p + C_{l_r}(b/2V_a)\, r + C_{l_{\delta_r}}\delta_r + C_{l_{\delta_a}}\delta_a\right] \\ c\left[C_m(\alpha) + C_{m_q}(c/2V_a)\, q + C_{m_{\delta_e}}\delta_e\right] \\ b\left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}(b/2V_a)\, p + C_{n_r}(b/2V_a)\, r + C_{n_{\delta_r}}\delta_r + C_{n_{\delta_a}}\delta_a\right] \end{bmatrix} + \begin{bmatrix} Q \\ 0 \\ 0 \end{bmatrix}$$

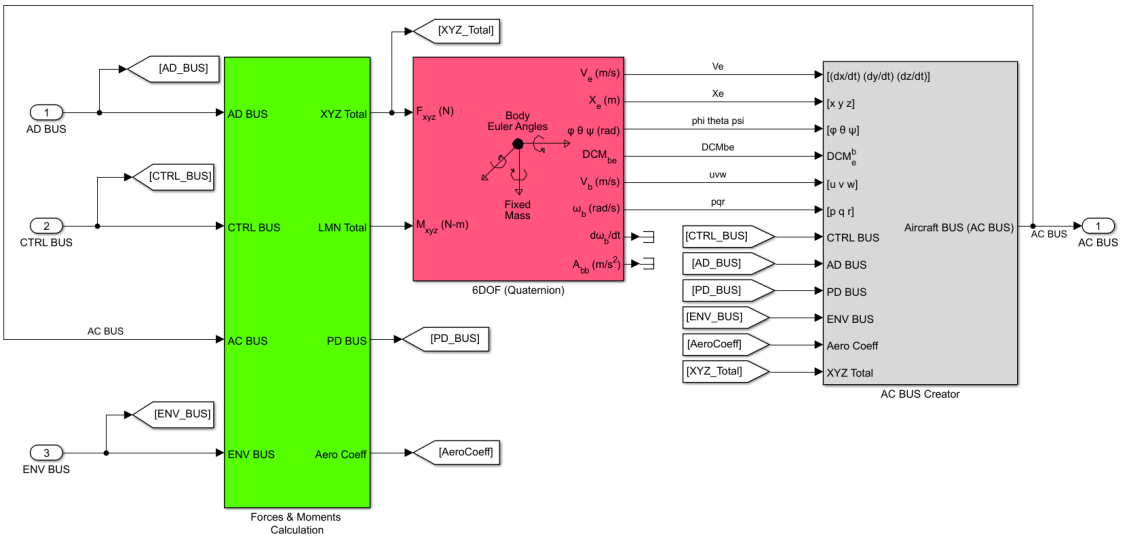


Figura 3.5 Diagrama de bloques del bloque `Aircraft Mechanics`, el cual contiene en su interior los bloques `Forces & Moments Calculation`, `6DOF (Quaternion)` y `AC BUS Creator`.

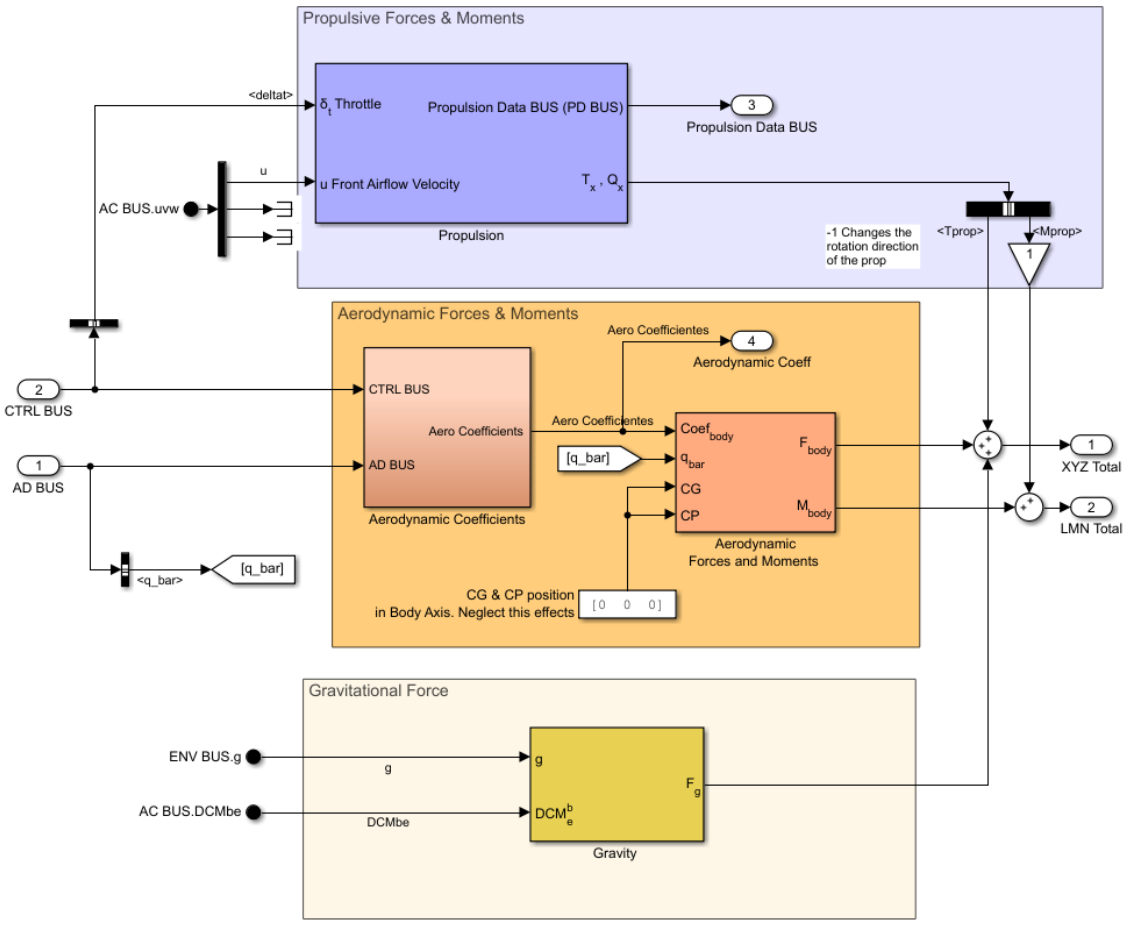


Figura 3.6 Diagrama de bloques del bloque `Forces & Moments Calculation` para las versiones `V1` y `V2` del simulador.

El bloque `AC BUS Creator` se encarga de agrupar todas las señales del estado del avión junto con todos los buses del avión como se muestra en la Figura 3.8. En este bloque también es creado el bus `Aerodynamic BUS - AERO BUS`.

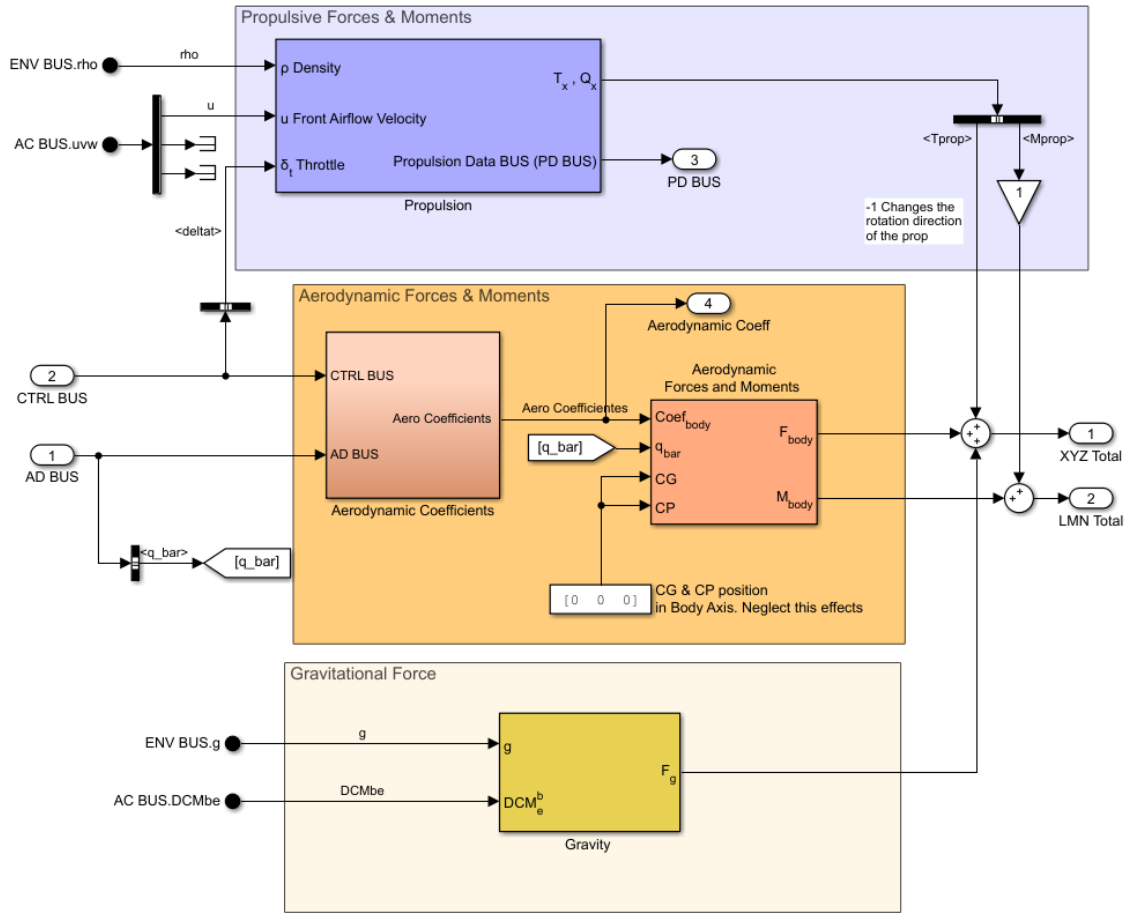


Figura 3.7 Diagrama de bloques del bloque `Forces & Moments Calculation` para la version `V3`.

3.3.3 Modelo Aerodinámico - `Aerodynamic Coefficients` y `Aerodynamic Forces and Moments`

El bloque `Aerodynamic Forces and Moments` del `Aerospace Blockset` se encarga de multiplicar la presión dinámica $\bar{q}$ por los distintos coeficientes aerodinámicos proporcionados, para calcular las fuerzas y momentos aerodinámicos. Además permite establecer la posición del centro de presiones (`CP`) y el centro de gravedad de la aeronave (`CG`) para tomar en cuenta los momentos generados en caso de no estar en la misma posición. Para simplificar el modelo, se asumen que ambos se ubican en las coordenadas $[0,0,0]_b^T$. Además se utiliza tanto para las entradas como para las salidas los ejes B (o `Body Axes`).

Por otra parte, el bloque `Aerodynamic Coefficients` se utiliza para realizar el cálculo de los coeficientes aerodinámicos del UAV, y presenta la estructura mostrada en la Figura 3.9.

Calculo de los coeficientes de la dinámica longitudinal - `Longitudinal Aerodynamics (Cx, Cz, Cm coefficients)`

El bloque `Longitudinal Aerodynamics (Cx, Cz, Cm coefficients)` se encarga de realizar el cálculo de los coeficientes aerodinámicos de la dinámica longitudinal en ejes B. Su diagrama de bloques interno se observa en la Figura 3.10.

El contenido del bloque `Stability Axis Aerodynamics` permite calcular los coeficientes aerodinámicos de la dinámica longitudinal del UAV en ejes estabilidad S. Este posee 2 versiones, la versión utilizada en los modelos `V1` y `V2`, y otra versión para el modelo `V3`. Estas versiones se diferencian por la polar C_D que implementan. En las Figuras 3.11 y 3.12 se muestran sus diferencias.

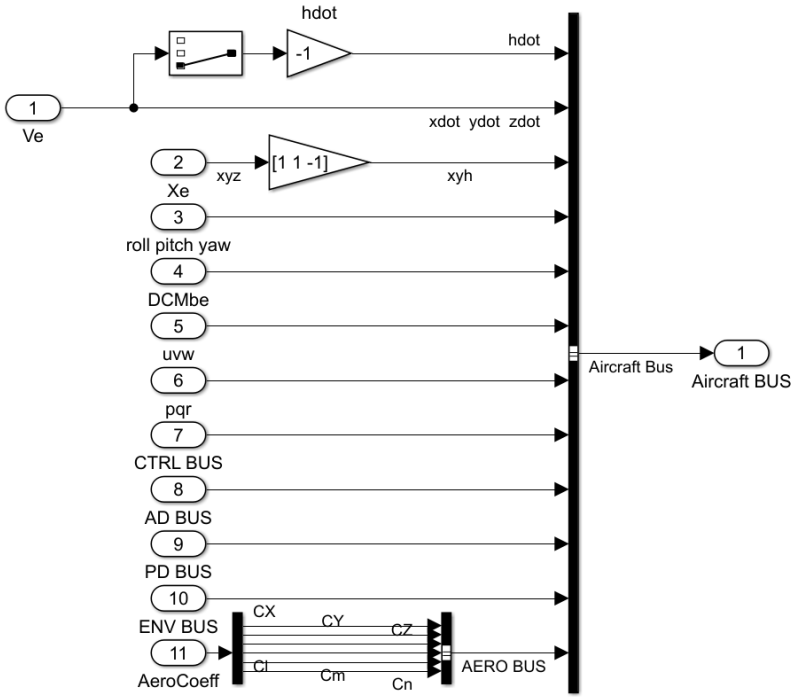


Figura 3.8 Diagrama de bloques del bloque `AC BUS Creator`.

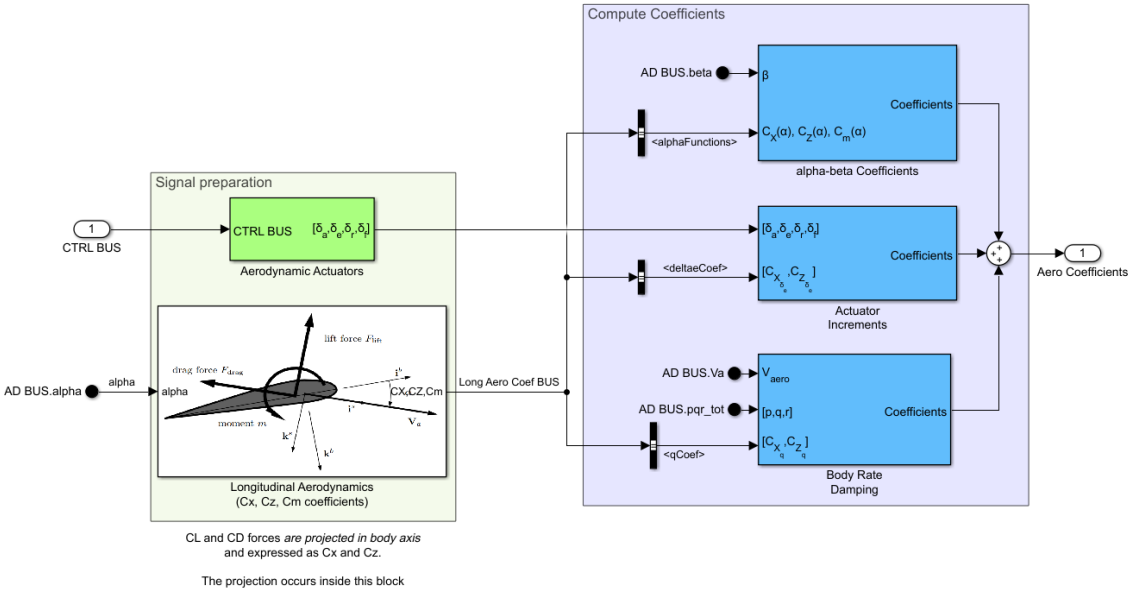


Figura 3.9 Diagrama de bloques del bloque `Aerodynamic Coefficients`.

Por último, cabe detallar el bloque `Stability to Body Axis Matrix Computation`. En este bloque se calcula la matriz de rotación en función del ángulo de ataque α que permite realizar la rotación de ejes S a ejes B. Ver Figura 3.13. Esta matriz tiene la expresión mostrada en la ecuación (3.2).

$$\mathbf{R}_s^b(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.2}$$

Cálculo de los términos aerodinámicos

Se utilizan 3 bloques para calcular de forma ordenada los diferentes términos que contribuyen a las fuerzas y momentos aerodinámicos que actúan sobre el UAV. Cabe destacar que para el cálculo de estos coeficientes

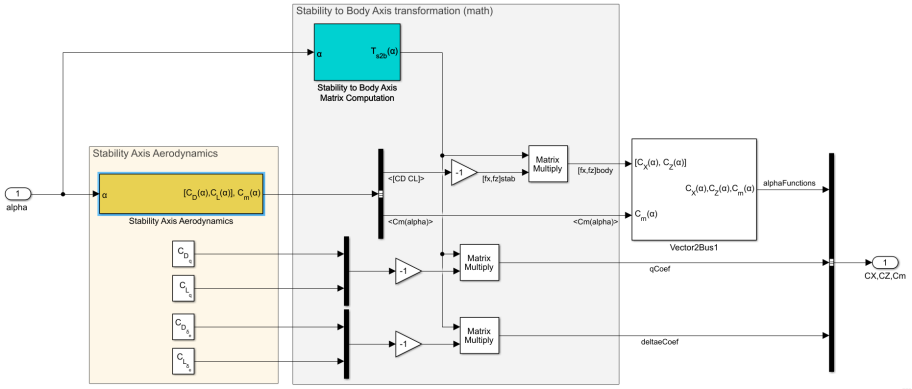


Figura 3.10 Diagrama de bloques del bloque `Longitudinal Aerodynamics (Cx, Cz, Cm coefficients)`.

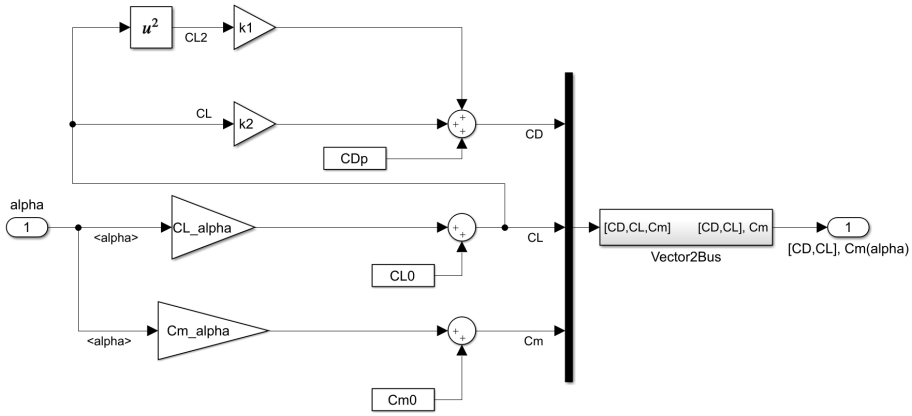


Figura 3.11 Diagrama de bloques del bloque `Stability Axis Aerodynamics` para las versiones del simulador `V1` y `V2`.

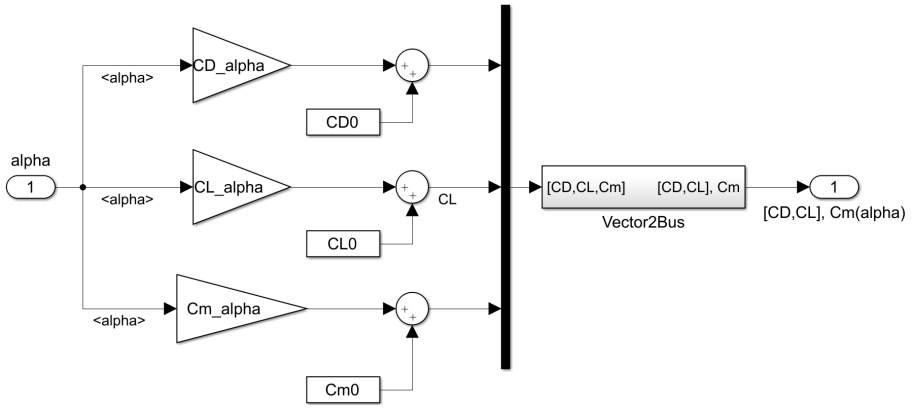


Figura 3.12 Diagrama de bloques del bloque `Stability Axis Aerodynamics` para la versión `V3` del simulador.

se realiza la suma de términos sobre un vector de la forma $[C_X, C_Y, C_Z, C_l, C_m, C_n]_b^T$. Esto es así ya que la entrada de coeficientes aerodinámicas del bloque `Aerodynamic Forces and Moments` es un vector de seis componentes de esta forma.

- `alpha-beta Coefficients`. Términos asociados a los coeficientes de las derivadas aerodinámicas respecto a α y β. Ver Figura .

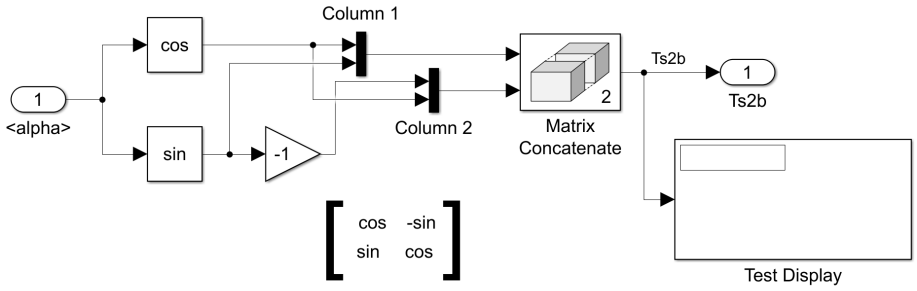


Figura 3.13 Cálculo de la matriz de rotación de S a B.

- `Actuator Increments`. Términos asociados a las derivadas aerodinámicas respecto a las diferentes señales de control δ_e, δ_e, δ_r, δ_f y δ_t. Ver Figura .

- `Body Rate Damping`. Términos asociados a los *Coeficientes de Amortiguamiento*. Estos coeficientes son las derivadas aerodinámicas respecto a las velocidades angulares de rotación p, q y r. Ver Figura .

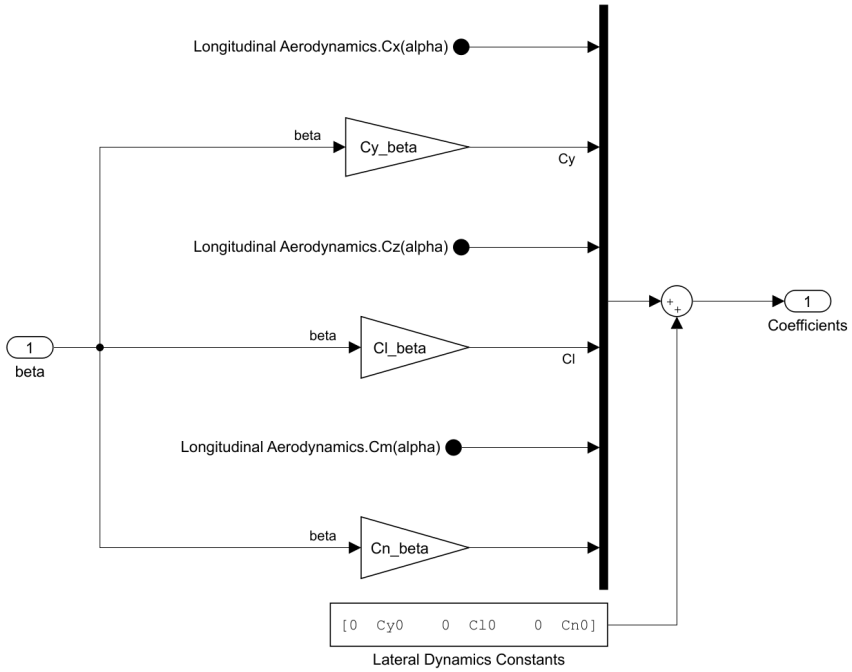


Figura 3.14 Diagrama de bloques del bloque `alpha-beta Coefficients`.

3.3.4 Modelos propulsivos - `Propulsion`

A continuación se presentan los diferentes implementaciones de modelos propulsivos realizas en `Simulink` para las 3 versiones del simulador.

Modelo propulsivo muy simplificado - Simulador versión `V3`

Como se vio en la sección 2.2.1, el modelo más sencillo de propulsión consiste en las siguientes ecuaciones:

$$T = \frac{1}{2}\rho S_{\text{prop}} C_{\text{prop}} \left[(k_{\text{motor}}\delta_t)^2 - V_a^2 \right]$$
$$Q = 0$$

Esta ecuaciones son implementadas como se muestra en la Figura 3.17.

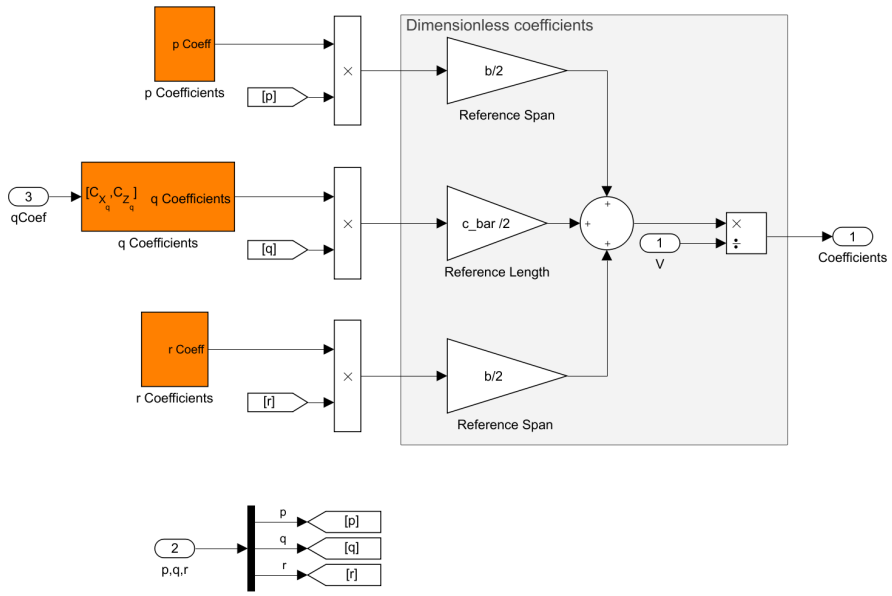


Figura 3.15 Diagrama de bloques del bloque `Body Rate Damping`.

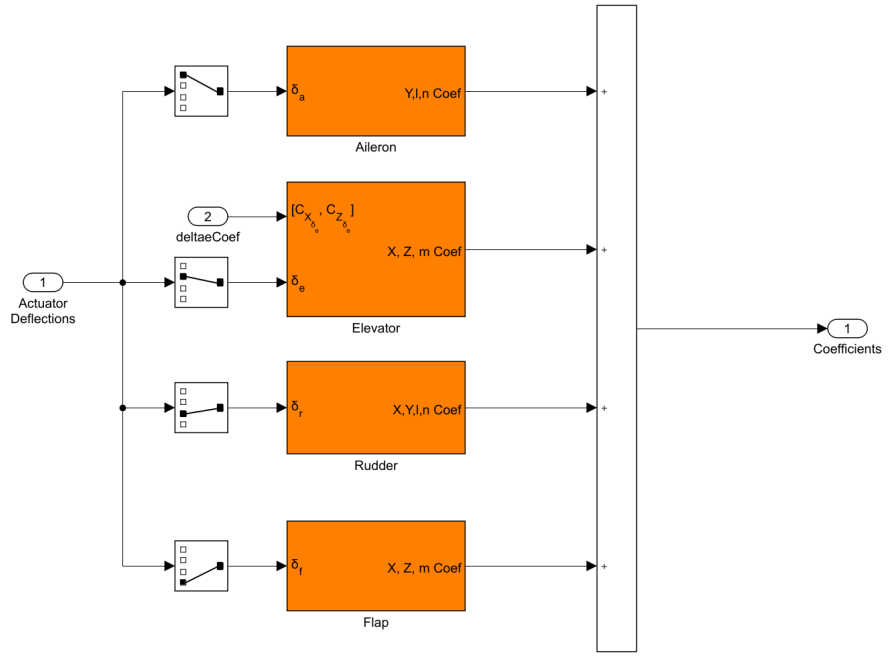


Figura 3.16 Diagrama de bloques del bloque `Actuator Increments`.

Modelo simple de motor y modelo semi-analítico de la hélice - Simulador versión `V2`

El diagrama de bloques utilizado se presenta en la Figura 3.18. En esta versión se implementa un modelo de motor que posee como entrada el valor de la palanca de potencia δ_t y como salida las RPM estimadas del motor.

$$\text{RPM} = 9000\tanh\left(\frac{9}{2}\delta_t - 2\right) + 9000 \tag{3.3}$$

Esta formula se implementa mediante un bloque `Lookup Table`, el cual posee una gráfica como la mostrada en la Figura 3.19.

Para modelar la hélice, simplemente se utilizan 2 bloques `Lookup Table`, pero a diferencia del modelo

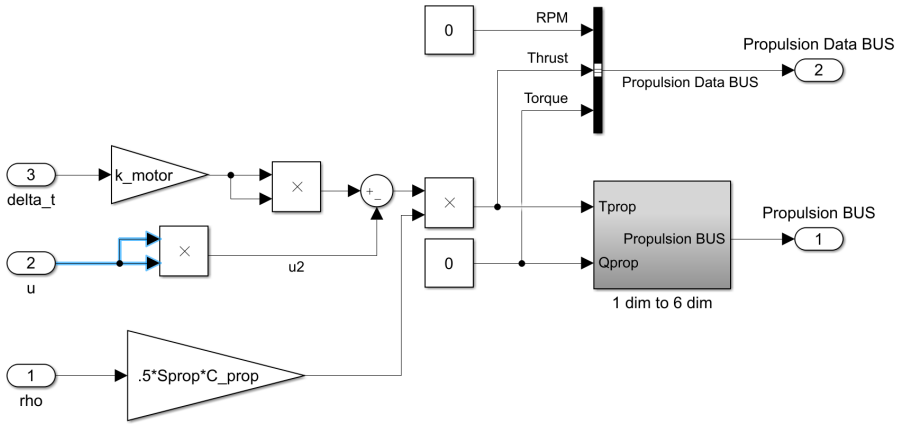


Figura 3.17 Diagrama de bloques del bloque `Propulsion` para la versión `V3` del simulador..

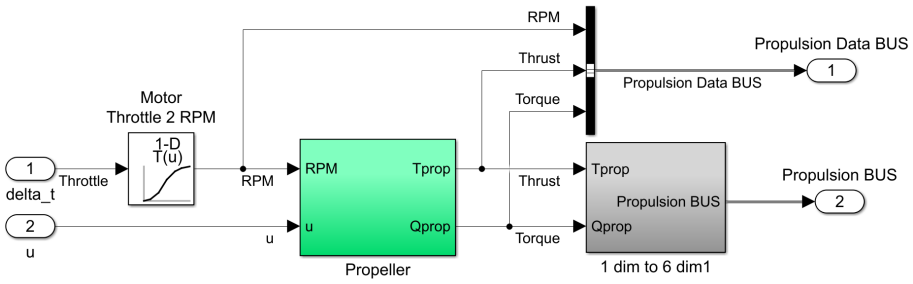


Figura 3.18 Diagrama de bloques del bloque `Propulsion` para la versión `V2` del simulador..

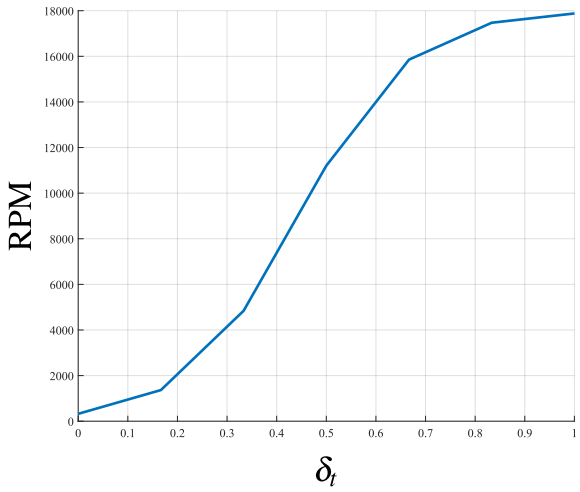


Figura 3.19 Modelo del motor en la versión `V2` del simulador.

anterior, en este caso se implementas *Lookup Tables* bidimensionales, una para el empuje, y otra para el torque del motor, generando sendas superficies como las mostradas en las Figuras 2.6 y 2.7 del capítulo anterior. Ver Figura 3.20. Las matrices de datos utilizadas para generar las *Lookup Tables* (matriz de torque `M`, matriz de empuje `T`, matriz de velocidad `V` y matriz de RPM/1000 `N`), son generadas a partir de los datos del fabricante de la hélice, y se obtienen mediante los scripts A.4.1, A.4.2 y A.4.3 adjuntos como anexos al presente trabajo.

Modelo eléctrico del motor y modelo semi-analítico de la hélice. - Simulador versión `V1`

El diagrama de bloques del bloque de propulsión para esta versión se muestra en la Figura 3.21. La única diferencia con el modelo anterior es que se implementa un modelo de motor eléctrico BLDC, descrito por la siguiente ecuación.

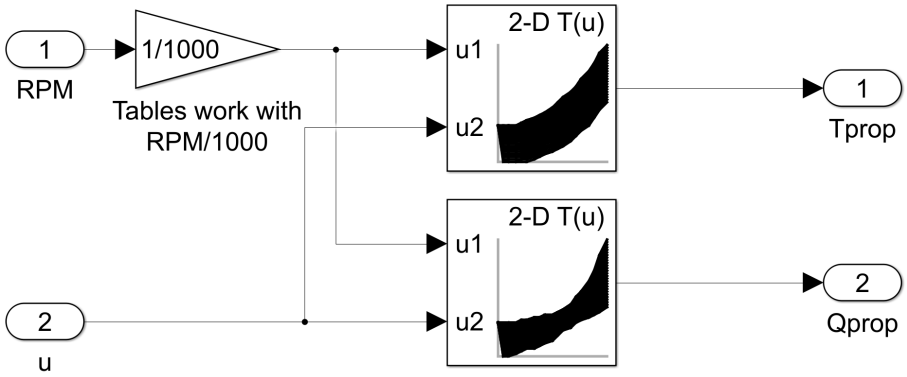


Figura 3.20 Modelo de la hélice implementado en el bloque `Propeller`.

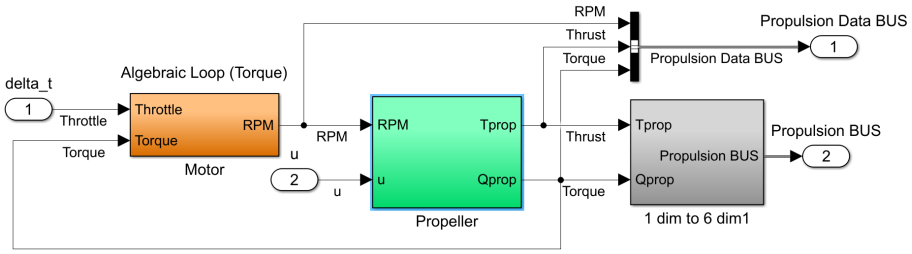


Figura 3.21 Diagrama de bloques del bloque `Propulsion` para la versión `V1` del simulador..

$$Q_{\text{motor}} = Q_{\text{prop}} = K_Q \cdot I_{\text{motor}} = K_Q \left[\frac{V_{\text{in}} - K_V \Omega}{R} - i_0 \right]$$

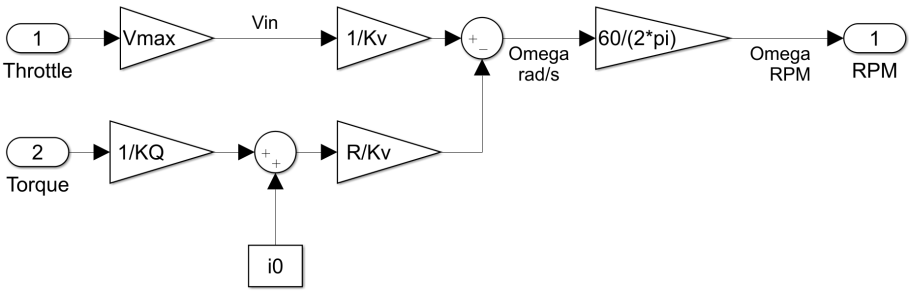


Figura 3.22 Diagrama de bloques del bloque `Motor`.

3.3.5 Modelo de gravedad - `Gravity`

Dado el vector de aceleración gravitatoria generado en el bloque `Environment Model` se, procede a calcular el peso que actúa sobre la aeronave. Este puede escribirse como

$$\mathbf{g} \equiv \mathbf{g}^a = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}_a$$

Para pasar este vector a ejes cuerpo B (ejes en los cuales están mecanizadas las ecuaciones), se multiplica el vector mostrado anteriormente por la *matriz de cosenos directores*, generada en el bloque `6DOF (Quaternion)`, es decir:

$$\mathbf{DCM}_a^b = \underbrace{\begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Roll - } \phi} \underbrace{\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}}_{\text{Pitch- } \theta} \underbrace{\begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Yaw - } \psi}$$

Finalmente, se multiplica $\mathbf{g}^b$ por la masa del avión m y por la matriz $\mathbf{DCM}_a^b$

$$\mathbf{F}_g = \mathrm{m}\mathbf{g} \equiv \mathbf{F}_g^b = \mathrm{m}\mathbf{g}^b = \mathrm{m} \cdot \mathbf{DCM}_a^b \, \mathbf{g}^a = \mathrm{m}g \begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix}$$

Se implementa en Simulink como se muestra en la Figura 3.23.

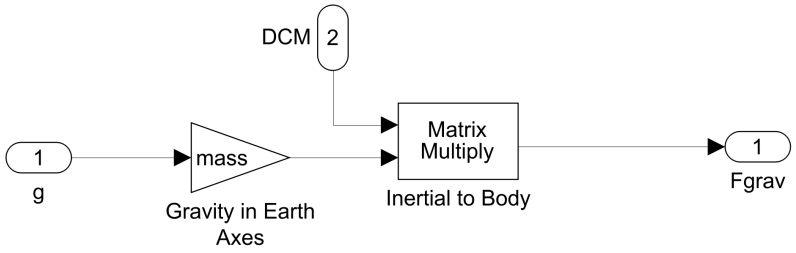


Figura 3.23 Diagrama de bloques del bloque Gravity.

3.4 Modelo linealizado de la plataforma

3.4.1 Trimado del UAV

El trimado del UAV consiste en hallar los estados de equilibrio $\mathbf{x} = \mathbf{x}_{\text{eq}}$ y variables algebraicas de equilibrio $\mathbf{x}_a = \mathbf{x}_{a,\text{eq}}$ que cumplan:

$$\begin{cases} \dot{\mathbf{x}}_{\text{eq}} = f(\mathbf{x}_{\text{eq}}, \mathbf{x}_{a,\text{eq}}, t) = \mathbf{C} \\ g(\mathbf{x}_{\text{eq}}, \mathbf{x}_{a,\text{eq}}, t) = 0 \end{cases} \tag{3.4}$$

Es decir, es necesario resolver un sistema de ecuaciones algebraicas no lineales definido por el vector $\mathbf{C}$ constante y por los valores que se definan como conocidos dentro de $\mathbf{x}_{\text{eq}}$ y $\mathbf{x}_{a,\text{eq}}$ (siempre y cuando den pie a un sistema compatible y con solución).

Cabe destacar que el vector $\dot{\mathbf{x}}_{\text{eq}} = \mathbf{C}$ no tiene que tener todas sus componentes nulas, es decir, existen maniobras o puntos de operación tales que $\dot{\mathbf{x}}_{\text{eq}} = \mathbf{C} \neq 0$. Por ejemplo, en un ascenso con *climb-rate* constante se tiene que $\dot{h} = \text{const} \neq 0$ ft/min, siendo este un punto de operación (o equilibrio) perfectamente válido. Lo mismo puede suceder en un viraje uniforme a altura constante donde se cumple que $\dot{\psi} = \text{const} \neq 0$ rad/s, $R(t) = \sqrt{x^2(t) + y^2(t)} = \text{const}$, $h = \text{const}$ (donde x, y, h son componentes del estado de equilibrio $\mathbf{x}_{\text{eq}}$).

Los pasos recomendados para realizar el trimado del modelo son los siguientes:

1. Se recomienda realizar una copia del modelo, en la cual se sustituyan las entradas y salidas predeterminadas con Inports y Outports. Se suministra el modelo de trimado simulador2017bTRIM.slx para ahorrarnos este paso. Observe que se ha establecido un empuje del 50 % y flaps a 0° para el presente ejemplo, mientras que se han establecido como entradas variables los ángulos de alerones, elevador y rudder, de allí los bloques Inport (estos ángulos serán calculados por el programa de trimado). Ver Figura 3.24.

2. Posteriormente, se abre la herramienta Linear Analysis Tool. Una forma de abrirla rápidamente es haciendo click derecho sobre el bloque UAV y luego, dentro del menú contextual seleccionando Linear

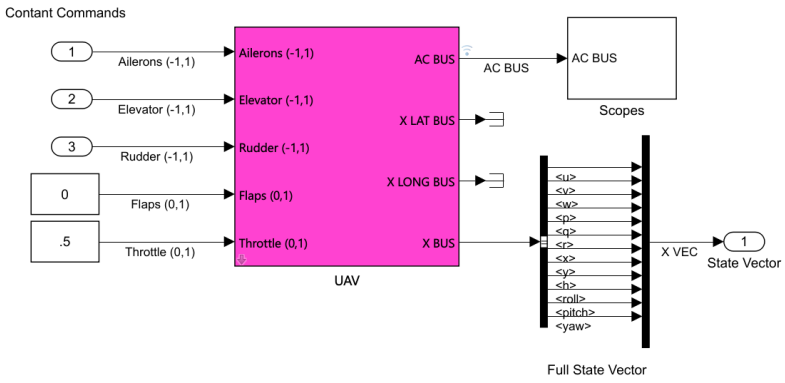


Figura 3.24 Modelo `simulador2017bTRIM.slx`.

`Analysis > Linearize Block`. Ver Figura 3.25.

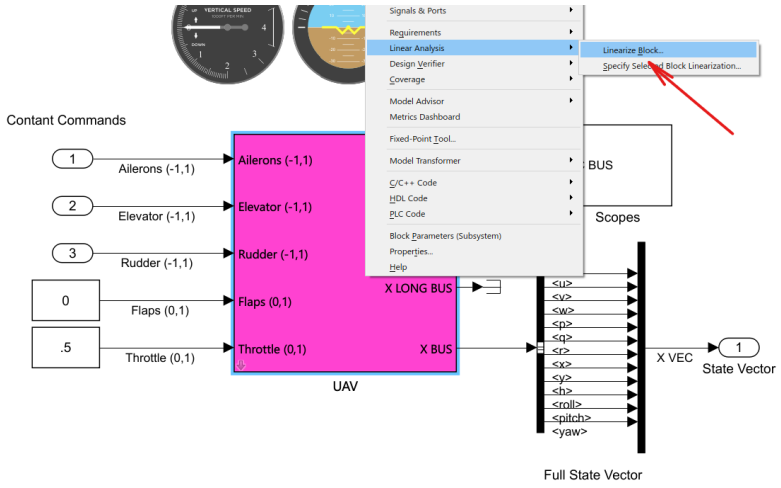


Figura 3.25 Menu contextual del bloque `UAV`.

3. Se selecciona la opción `Trim Model`. Ver Figura 3.26
4. Finalmente, se configuran las variables que se desean en *steady-state* ($\dot{x} = 0 \Rightarrow x = \text{const}$), se establecen los valores iniciales para la búsqueda del punto de equilibrio (comúnmente llamados *seeds*. En nuestro caso se establecen en la columna `Value` y se imponen los valores conocidos del punto de equilibrio (columna `Know`). Además puede configurarse el rango de búsqueda de las soluciones mediante las columnas `Minumum` y `Maximum`. Para empezar el trimado se hace click en el botón `Start trimming`. Ver Figura 3.27.

3.4.2 Análisis de los modos de vuelo del UAV

Esta sección se fundamenta en los apuntes de la asignatura *Sistema de Control y Guiado* impartida en el Grado de Ingeniería Aeroespacial de la Universidad de Sevilla. A continuación se busca describir el proceso de análisis de los modos de vuelo de los UAVs que se pretenden modelar.

Una vez realizada la linealización del modelo en un vuelo de crucero (recto y nivelado), se obtienen las matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ y $\mathbf{D}$, que cumplen

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3.5}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{3.6}$$

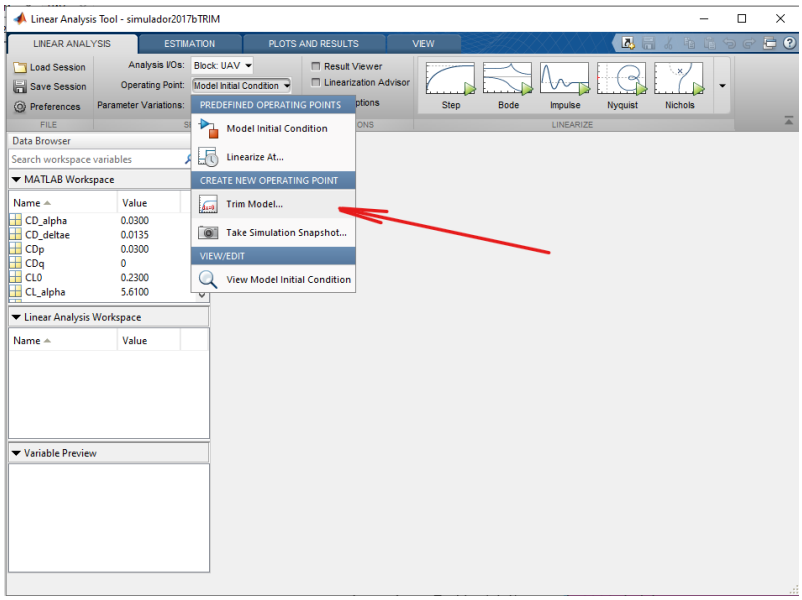


Figura 3.26 Herramienta `Linear Analysis Tool`.

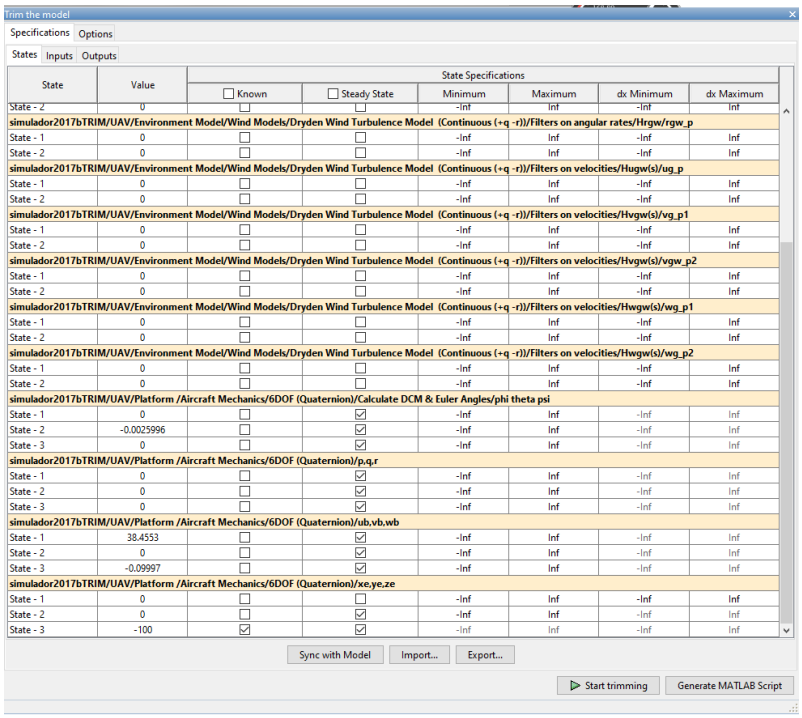


Figura 3.27 Menú contextual `Trim the model`.

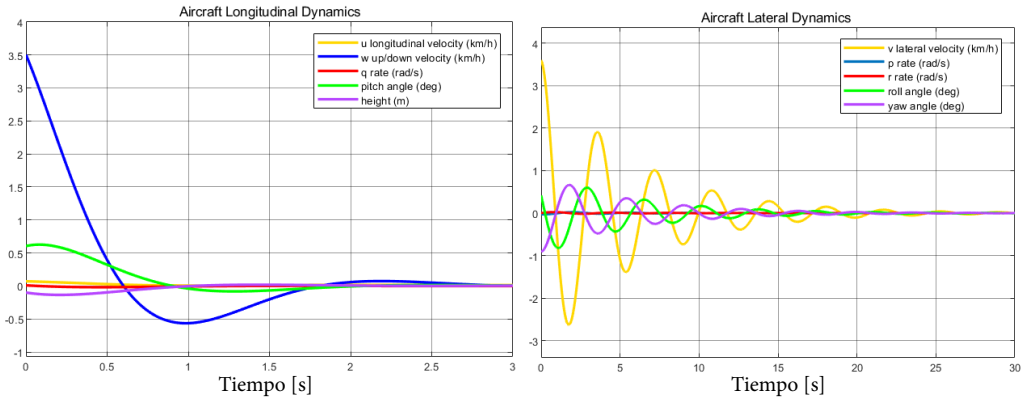
El análisis de la matriz de estado $\mathbf{A}$, permite conocer los modos de vuelo de la aeronave, los cuales son de gran importancia al momento de desarrollar sistemas de aumento de estabilidad o SAS (*Stability Augmentation System*).

En primer lugar se separan las dinámicas lateral y longitudinal del avión, teniendo 2 vectores de estados diferentes, uno para cada tipo de dinámica. Definiendo los siguientes vectores de estado, $\mathbf{x}_{\text{long}} = [u,w,q,h,\theta]^T$, $\mathbf{x}_{\text{lat}} = [v,p,r,\phi]$, y ejecutando la función `longAndLatMatrices()`, se crean las matrices de estado $\mathbf{A}_{\text{long}}$ y $\mathbf{A}_{\text{lat}}$, correspondientes.

Luego, estas se introducen como entrada en la función `eigDynamics()`, la cual calcula los autovalores de

las matrices de estado para posteriormente devolver los modos de vuelo de la dinámicas lateral y longitudinal, así como las variaciones en las condiciones iniciales que excitan estos modos en particular. Estas condiciones iniciales, vienen dadas por la parte real de los autovectores de las matrices de estado.

En la Figura 3.28 se ejemplifica como al cambiar las condiciones iniciales del avión y dejando intactas las entradas de trimado, se obtiene una respuesta característica de alguno de los modos de vuelo.



a) Short Period Mode **b)** Dutch Roll Mode

Figura 3.28 Modos de vuelo para el simulador `V3`.

4 Interfaz usuario-simulador

Todas las versiones de los simuladores, independientemente de si se encuentran o no en las carpetas de identificación, incorporan una serie de bloques especiales que permiten la recepción de entradas externas a Simulink y la salida de datos para visualización, de forma que se pueda ejecutar el pilotaje del UAV de forma online, es decir, en directo mientras corre la simulación.

El objetivo de tener entradas en directo y un medio de visualización, es poder recrear un vuelo con maniobras reales, lo más parecidas a las realizables con una aeronave RC, de forma que los datos recolectados y utilizados posteriormente para la identificación del sistema, sean de la mayor calidad posible. Además, puede utilizarse como medio de entrenamiento para pilotos, pruebas de nuevas plataformas, entre otras aplicaciones.

A continuación se detalla el proceso de conexión de las entradas y salidas del modelo, utilizando siempre bloques de la librería `Aerospace Blockset` de `Simulink`.

Cabe destacar que las simulaciones deben simularse con un *time step* fijo, establecido con la variable `ST`. Este intervalo de tiempo debe ser exactamente igual para todos los bloques del sistema, tanto para los bloques de entrada como para los bloques de salida, de forma que el sistema este completamente sincronizado.

4.1 Entradas al modelo

Las entradas del sistema posee dos segmentos: el segmento hardware y el segmento software. De los cuales, se detallará a continuación.

4.1.1 Segmento hardware

Se hizo uso del *Gamepad Controller Logitech Dual Action G-UF13A* como dispositivo hardware para realizar el control.

El mismo posee 2 joysticks, 8 botones simples y 4 botones en forma de cruz. Ver Figura. Esto implica que el controlador (o mando) posee un total de 14 canales, mediante los cuales se transmiten 14 señales independientes desde el controlador hasta el ordenador donde se pretende correr la simulación, todo mediante un cable USB.

Cabe destacar que cada joystick posee 2 canales independientes, uno correspondiente a los movimientos verticales y otro correspondiente a los movimientos horizontales. De forma que se utilizan 4 canales del mando para realizar el control de la aeronave se corresponden con las 4 salidas del bloque `Pilot Joystick` del `Aerospace Blockset` de Simulink, del cual se detallará más adelante.

- **Canal 1** (`roll` en el bloque de Pilot Joystick): se asigna a la señal $\delta_r \in (-1,1)$ del timón de cola (en `Simulink`, señal `Rudder (-1,1)`).
- **Canal 2** (`pitch` en el bloque de Pilot Joystick): se asigna a la señal $\delta_t \in (0,1)$ de la palanca de gases (en `Simulink`, señal `Throttle (0,1)`).

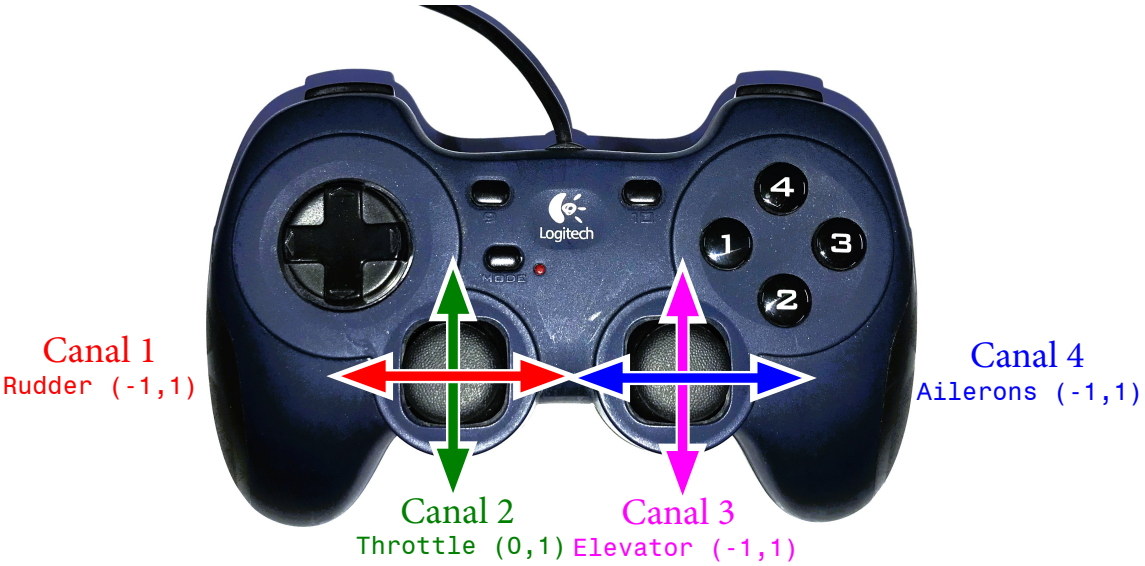


Figura 4.1 *Gamepad Controller Logitech Dual Action G-UF13A*, el nombre de los canales correspondientes a cada señal de los joysticks y los nombres de las correspondientes señales de entrada al bloque `UAV`.

- **Canal 3** (`yaw` en el bloque de Pilot Joystick): se asigna a la señal $\delta_e \in (-1,1)$ del timón de profundidad (en `Simulink`, señal `Elevator (-1,1)`).
- **Canal 4** (`throttle` en el bloque de Pilot Joystick): se asigna a la señal $\delta_a \in (-1,1)$ de los alerones (en `Simulink`, señal `Ailerons (-1,1)`).

Las señales generadas por el mando suelen ser muy ruidosas y llenas de sesgo, Ver Figura **??**. Además, los rangos de salida de los canales no corresponde con los del modelo del UAV. Este inconveniente es corregido posteriormente mediante software.

4.1.2 Segmento software

Mediante el bloque `Pilot Joystick` del `Aerospace Blockset`, se detecta y configura automáticamente el controlador hardware, de forma que las señales son recibidas por medio del cable USB, se reciben directamente en los puertos de salida de el bloque antes mencionado. La configuración del bloque se muestra en la Figura.

La señales pertenecientes a las superficies de control deben moverse en el rango $(-1,1)$ mientras que la señal correspondiente a la palanca de gases, debe moverse en el rango $(0,1)$. El bloque `Pilot Joystick` arroja los siguientes rangos:

- **Canal 1** (`roll`): $(-1,1)$
- **Canal 2** (`pitch`): $(-1,1)$
- **Canal 3** (`yaw`): $(-1,1)$
- **Canal 4** (`throttle`): $(0,1)$

Por ende, es necesario colocar bloques de saturación (algo redundantes) precediendo a las entradas del modelo de Simulink, además de cambiar la señales de orden como se muestra en la Figura 4.4.

El sesgo en las señales es el mayor inconveniente al momento de pilotar el UAV, de forma que para corregirlo, se procede añadiendo una constante de mismo valor y signo opuesto al sesgo detectado cuando el mando se encuentra libre (es decir, sin desplazar ninguno de los joysticks). Es necesario que las señales obtenidas sean nulas tras realizar este proceso.

Una vez realizado el paso anterior, es claro que el UAV no tendrá un crucero recto y nivelado teniendo las superficies de control sin deflectar y la palanca de gases con un valor nulo. Por tanto, es necesario sumar

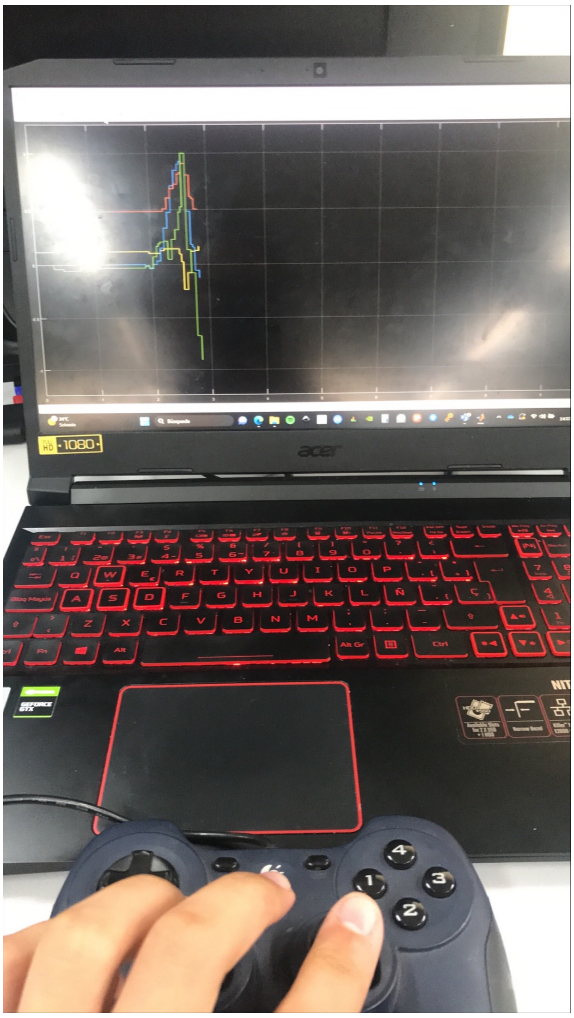


Figura 4.2 Señales de entrada al modelo de `Simulink`, sin procesamiento alguno.

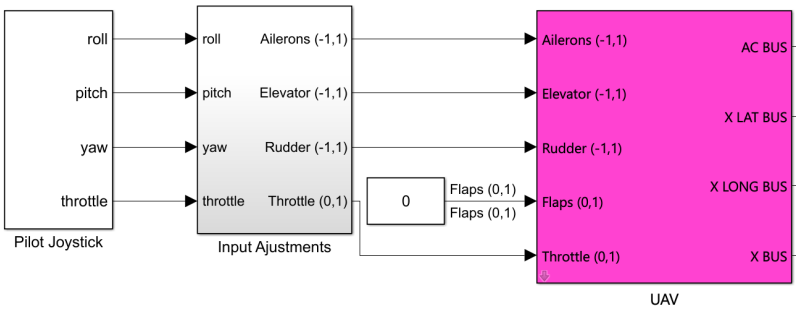


Figura 4.3 Bloques de entrada al modelo.

los valores de trimado de la superficies de control y de la palanca de gases, a los comando suministrados mediante el mando. Estos valores se obtienen como se ha mostrado en el capítulo anterior.

Por último, es interesante detallar la creación de las variables de validación `ailerons_val`, `elevator_val`, `rudder_val` y `throuttle_val`. Estas son utilizadas posteriormente como las entradas en el modelo de validación tras la identificación del sistema, de forma que se pueda realizar una comparación entre el modelo de generación de datos y el modelo identificado al estimularse por entradas idénticas.

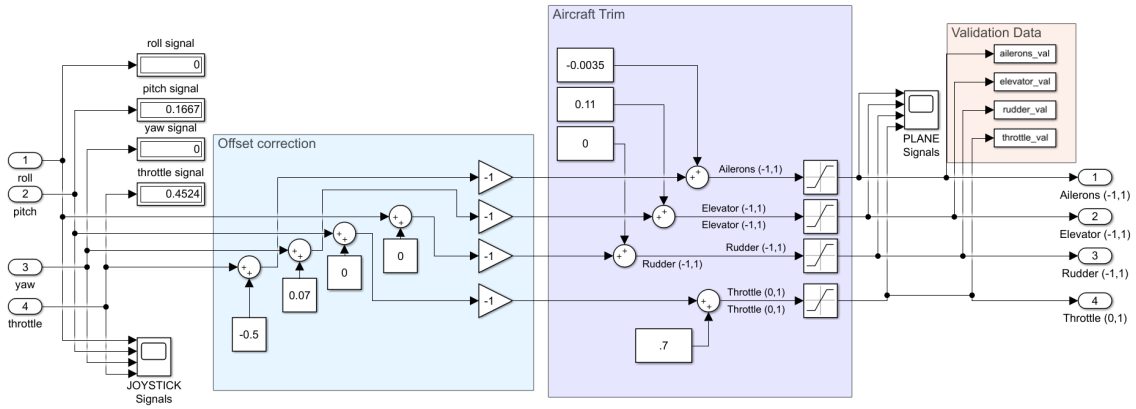


Figura 4.4 Bloques de adaptación de las señales de entrada al modelo. Se observa como los valores de las constantes para la corrección del offset son iguales y opuestos a los valores reflejados en los displays en la parte superior izquierda del diagrama. Cabe destacar que en esta imagen los joysticks se encuentran libres, sin deflexión alguna.

4.2 Salida del modelo y visualización

Para visualizar el estado de la aeronave, y de esta forma ser capaz de pilotarla, se utiliza el software de simulación `Flight Gear`. Como comentan en su página oficial, `FlightGear` es un simulador de vuelo de código abierto desarrollado por una amplia comunidad de voluntarios como alternativa a los simuladores comerciales. Si es cierto que el mismo integrar un modelo físico para emular el vuelo real de diferentes aeronaves, en nuestro caso se pretende utilizar únicamente como un medio para representar el estado de la aeronave, de forma que el modelo físico utilizado es el desarrollado mediante `Simulink`.

Para pilotar un vuelo, es interesante visualizar dos importantes aspectos del estado de la aeronave: su posición y su actitud. La posición del UAV puede definirse por una terna de números en un sistema de referencia de preferencia, por ejemplo, si se utilizan *coordenadas geodésicas*: longitud, latitud y altitud o (λ, φ, h). De igual forma, la actitud puede representarse por otra terna de números dada por los *Ángulos de Euler*: ángulo de alabeo, ángulo de cabeceo y ángulo de guiñada, o (ϕ, θ, ψ). De forma que es necesario proporcionar estas 6 variables al software de visualización para representar al UAV en vuelo.

Para cumplir este objetivo, se utiliza el protocolo UDP (que significa protocolo de datagramas de usuario, del inglés *User Datagram Protocol*). Este es un protocolo de comunicaciones en redes de ordenadores que se utiliza para proporcionar un medio de transporte para datos provenientes de diferentes fuentes así como el control del flujo a través de redes de ordenadores. Los paquetes de información que se transmiten mediante UDP se denominan *datagramas* o *paquetes*. Estos, al tener cabeceras muy simples (contando únicamente con la dirección o puerto de destino y la longitud del mensaje), son bastante ligeros y permiten la retransmisión de datos para visualización en tiempo real con un retardo imperceptible. Para establecer la comunicación entre `Flight Gear` y `Simulink` mediante este protocolo, ambas aplicaciones deben configurarse apropiadamente, como se enseña a continuación.

4.2.1 Configuración en `Simulink`

El `Aerospace Blockset` posee un bloque especial para la comunicación con la aplicación de `Flight Gear` mediante UDP, llamado `FlightGear Preconfigured 6DoF Animation`. Ver Figura 4.5.

Este bloque se configura de la siguiente forma: como dirección IP de destino de establece la IP del ordenador donde va a iniciarse `Flight Gear`, en este caso simplemente se podría colocar `localhost`; se asigna como puerto de destino el 5502 (el cual es el puerto predeterminado que utiliza la aplicación de `Flight Gear` para escuchar y recibir paquetes externos); se establece como tiempo de muestreo el de la simulación `ST` y por último, se selecciona la versión de Flight Gear con la que se pretenda trabajar. Si se posee una versión superior a las v2017.1, basta con seleccionar esta última en el bloque de configuración, quedando todo finalmente configurado como se muestra en la Figura 4.6.

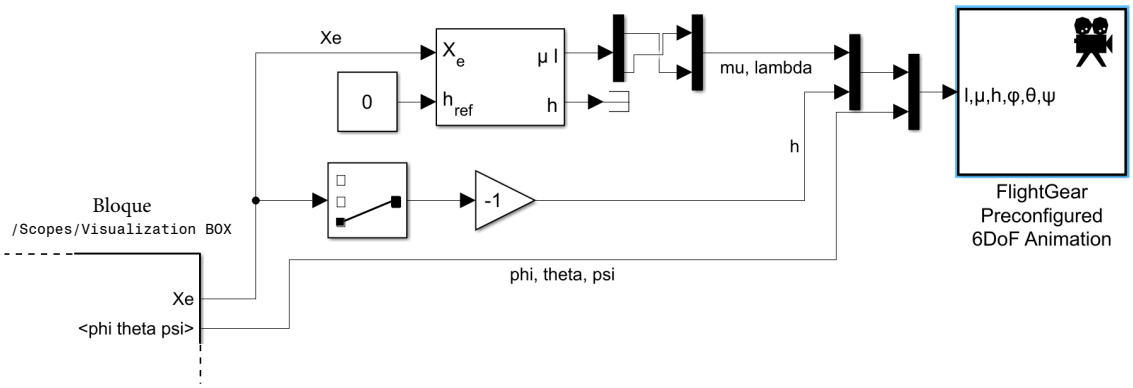


Figura 4.5 Transformación de las señales de posición en ejes N a coordenadas geodésicas y conexión al bloque de comunicación con `Flight Gear`.

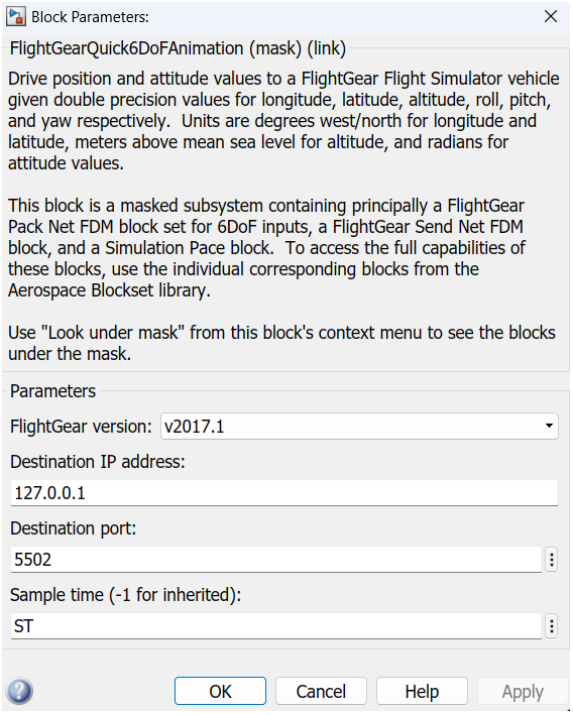


Figura 4.6 Configuración del bloque `FlightGear Preconfigured 6DoF Animation`.

4.2.2 Configuración en `Flight Gear`

En este caso, se ha utilizado la última versión disponible del software para la fecha de redacción del presente trabajo. Además se ha consultado el manual de uso del herramienta para realizar la correcta configuración del entorno [7]

En la ventana de *Ajustes* en la sección de *Ajustes Adicionales*, es necesario añadir los siguientes comandos:

```
SET FG_ROOT=C:\Program Files\FlightGear 2020.3\data
SET FG_SCENERY=C:\Program Files\FlightGear 2020.3\data\Scenery;
.\\bin\win64\fgfs --fdm=null --enable-auto-coordination
 --native-fdm=socket,in,30,localhost,5502,udp --fog-disable --enable-clouds3d
 --start-date-lat=2004:06:01:09:00:00 --enable-sound --visibility=15000 --in-air
 --prop:/engines/engine0/running=true --disable-freeze --offset-distance=0
 --offset-azimuth=0 --enable-rembrandt
```

Código 4.1 Script de ajuste de `Flight Gear` mediante comandos.

Los comandos más importantes son:

- `-fdm=null`: desactiva las dinámicas de vuelo modeladas por el propio `Flight Gear`.
- `-native-fdm=socket,in,30,localhost,5502,udp`: configura la dirección `localhost:5502` como la puerta por medio de la cual recibirá paquetes UDP con los valores de $\lambda, \varphi, h, \phi, \theta$ y ψ. De esta forma toda la dinámica de la aeronave que se esta visualizando correspondera con la del modelo de `Simulink`.

La configuración queda finalmente como se muestra en la Figura 4.7. Los comandos establecidos despues de la configuración UDP, son todos opcionales y corresponden con parámetros propios de la visualización como la presencia de niebla, la forma de las nubes en el entorno, entre otros.

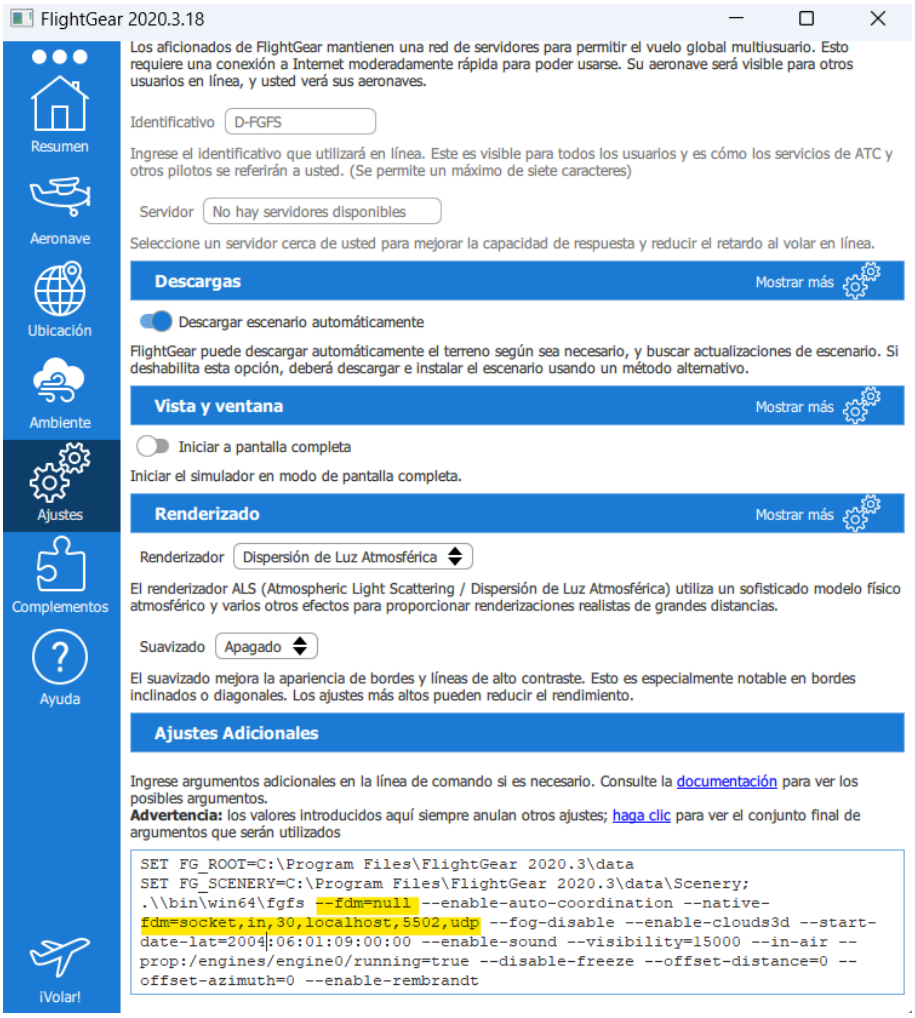


Figura 4.7 Configuración de `Flight Gear`.

Para realizar un vuelo, simplemente hay que elegir un avión y un aeropuerto determinado, todo esto tras la configuración mencionada anteriormente. Además cabe destacar que las coordenadas geodésicas de partida establecidas en `Simulink` corresponden con las coordenadas del centro FADA-CATEC de Sevilla, La Rinconada, de forma que independientemente del aeropuerto elegido en `Flight Gear`, una vez iniciado el vuelo, el avión se transportará inmediatamente a la locación antes mencionada antes de empezar a desplazarse.

5 Identificación de parámetros del modelo a partir de datos de vuelo sintéticos

El presente estudio, además de enfocarse en el proceso de desarrollo de varios simuladores de vuelo, también tiene como objetivo implementar un conjunto de algoritmos para determinar los coeficientes aerodinámicos de UAVs utilizando únicamente una gran cantidad de datos recolectados de vuelos reales mediante los sensores que integra la aeronave. En la Figura 5.1, se enseña un mapa o diagrama de flujo, que explica cómo es la interacción entre los distintos modelos de `Simulink` y scripts de `MATLAB`, que permiten la identificación de un UAV y su posterior validación.

El proceso seguido es el siguiente:

1. Se realiza una simulación de vuelo pilotado, donde se generan datos de vuelo sintéticos y donde además, se guardan las entradas del sistema, para introducirlas en el modelo de validación. Esto es así ya que de esta manera se puede verificar si antes los mismos estímulos de entrada, el UAV de identificado reacciona igual que el UAV que generó los datos.
2. Luego, se toman estos datos de vuelo y se introducen en el algoritmo de identificación gestionado por el script `NONLINEAR_IDENTIFICATION.m`. Este calcula los coeficientes aerodinámicos estimados, y los almacena en un archivo `fittedCoefficients.mat`.
3. Posteriormente, dentro del archivo de configuración del modelo de validación, se asignan los valores almacenados en `fittedCoefficients.mat` a cada uno de los coeficientes aerodinámicos del modelo.
4. El modelo de validación en `Simulink`, lee el archivo de configuración y realiza la simulación del UAV identificado. Las entradas de este modelo son idénticas a las utilizadas en el vuelo de generación de datos sintéticos.
5. Por último, se pretende comparar los datos del estado durante el vuelo de generación de datos, con los datos del estado durante el vuelo de validación.

En este capítulo se detalla el proceso seguido para cumplir este objetivo, además de describir el conjunto de herramientas utilizadas para lograrlo. Cabe destacar que solo se pretende trabajar con los modelos `V2` y `V3` por motivos de tiempo, ya que se ha asumido que identificar un modelo más complejo como el `V1`, podría requerir mayor inversión de tiempo en el caso de que la identificación no fuese satisfactoria.

5.1 Modelos usados para la identificación

Como se comentó anteriormente, es necesario generar un modelo de simulación que permita recrear no solo la dinámica del UAV, sino también las mediciones realizadas por sus sensores, para poder generar datos sintéticos que permitan emular un vuelo real con adquisición de datos.

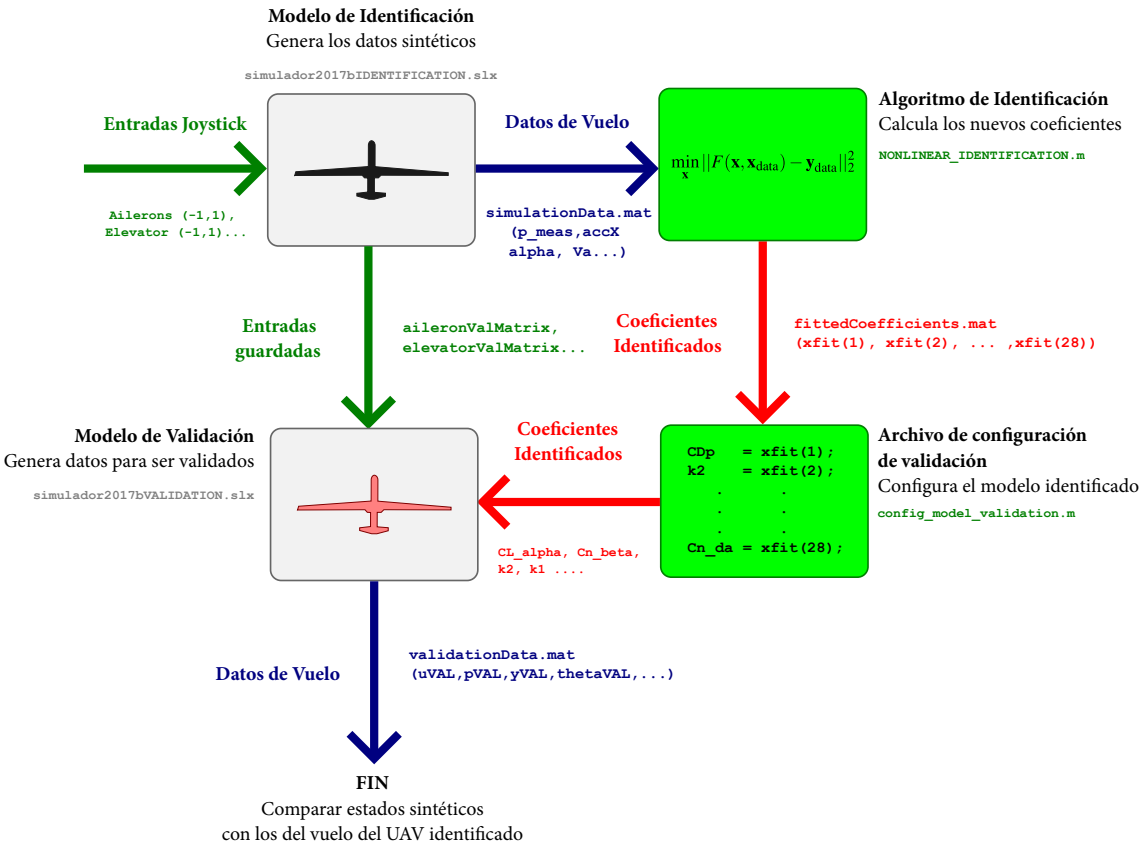


Figura 5.1 Procedimiento de generación de datos sintéticos, identificación de UAVs y validación de resultados mediante modelos de `Simulink` y scripts de `MATLAB`. Las flechas representan el flujo de datos, que pueden ser cargas de archivos `.mat` o lecturas de archivos de configuración `.m`. Conectan los distintos tipos de archivos con los que se trabaja (estos pueden ser modelos de `Simulink`, `.slx` o scripts de `MATLAB`, `.m`).

El objetivo de recrear estas señales reales, es poder poner a prueba de forma sintética y controlada, el algoritmo de identificación a implementar con datos de vuelo reales y verificar su correcto funcionamiento. En este caso, se procede a modelar de forma simple, un acelerómetro, un giroscopio (ambos en el bloque `IMU Sensor Recorder`) y otro conjunto de sensores (en el bloque `Other Sensors Recorder`) de los que se detalla en lo que sigue. Además, se pretende detallar el contenido del bloque `Flight Data Recorder (FDR)` de los modelos de `Simulink`, donde se tratan de modelar todos los sistemas mencionados anteriormente.

5.1.1 Modelo de generación de datos sintéticos

En la Figura 5.2 se muestra la estructura interna del bloque `Flight Data Recorder (FDR)`. Los 3 bloques que lo conforman tiene funciones de grabación de datos al *Workspace* de `MATLAB`. Estos se diferencian fundamentalmente en el tipo de datos que graban cada uno de ellos.

En la Figura 5.3 se muestra el contenido del bloque `IMU Sensor Recorder` el cual engloba las funciones de generación de datos sintéticos del acelerómetro (`Accelerometer + Noise`) y el giroscopio (`Gyro + Noise`).

Por último, las desviaciones típicas de las medidas de los sensores artificiales se han tomando del trabajo *Aerodynamic Parameter Identification and Uncertainty Quantification for Small Unmanned Aircraft* [8], y se muestran en la Tabla 5.1.

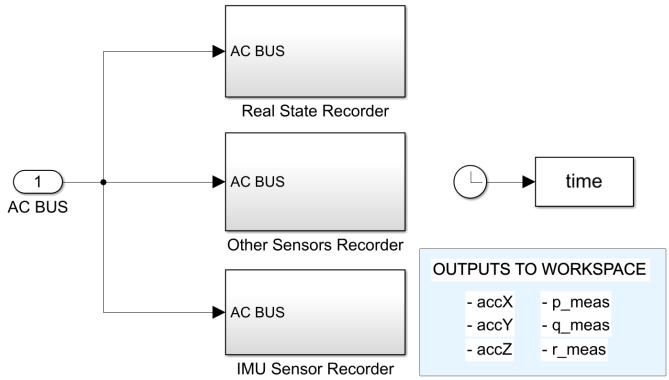


Figura 5.2 Contenido del bloque `Flight Data Recorder (FDR)`.

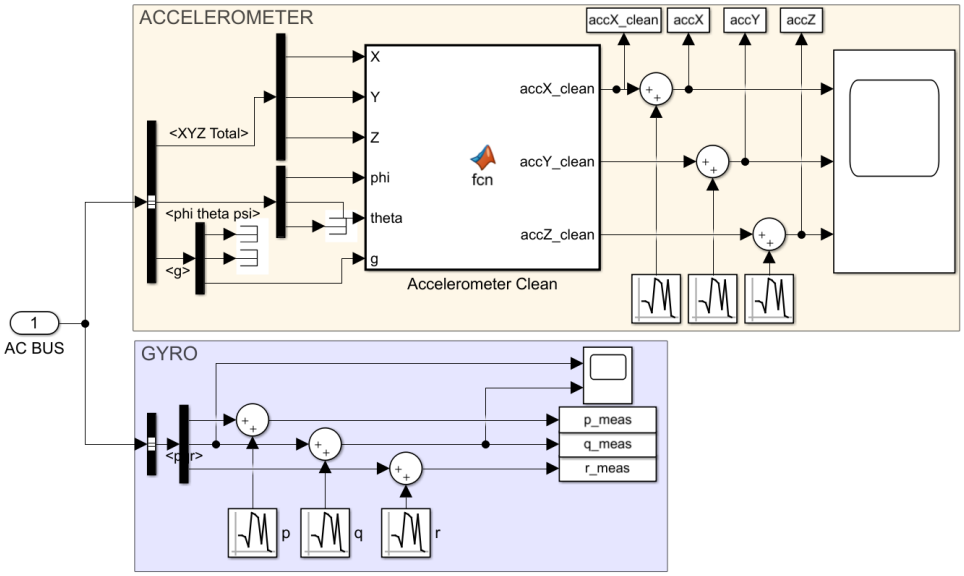


Figura 5.3 Contenido del bloque `IMU Sensor Recorder`.

Tabla 5.1 Desviaciones típicas en las medidas de los sensores modelados.

Variable	Desviación Típica (σ_v)
$y_{\text{accel},x}, y_{\text{accel},y}, y_{\text{accel},z}$	$0.16\,\text{m}\,\text{s}^{-2}$
$\alpha, \beta, \delta_a, \delta_e, \delta_r$	0.1°
p, q, r	$0.2\,^\circ/\text{s}$
ρ	$0.001\,\text{kg}/\text{m}^3$
V	$0.16\,\text{m}\,\text{s}^{-1}$

Modelado del acelerómetro del UAV

Para modelar la IMU (o *Inertial Measurement Unit*) de la aeronave, se ha utilizado el modelo propuesto por Beard y McLain [2]. En este caso, se asume que esta solo se compone de un acelerómetro y un giroscopio que miden aceleraciones y velocidades angulares alrededor de 3 ejes cartesianos.

Se asume que las señales obtenidas de estos dispositivos pueden presentar un sesgo dependiente de la temperatura, así como un ruido blanco gaussiano de media cero, es decir $v_{\text{accel},i} \sim N(0, \sigma_{v_{\text{accel}}})$ con $i = x, y, z$, y $\sigma_{v_{\text{accel}}} = 0.16\,\text{m}/\text{s}^2$. Para simplificar el modelo, se considera el caso en el cual se calibra el instrumento previo al vuelo, de forma que el sesgo del mismo puede despreciarse. De igual forma, se asume que este dispositivo se encuentra ubicado muy cerca del centro de gravedad del UAV y que sus ejes coinciden con los

ejes del sistema B. De forma que finalmente, las mediciones del acelerómetro quedan como:

$$y_{\text{accel},\,x} = \frac{X}{\text{m}} + g\sin\theta + \nu_{\text{accel},\,x} \tag{5.1}$$

$$y_{\text{accel},\,y} = \frac{Y}{\text{m}} - g\cos\theta\sin\phi + \nu_{\text{accel},\,y} \tag{5.2}$$

$$y_{\text{accel},\,z} = \frac{Z}{\text{m}} - g\cos\theta\cos\phi + \nu_{\text{accel},\,z} \tag{5.3}$$

Cabe destacar que se sustraen las aceleraciones gravitatorias en cada uno de los ejes debido a que estos dispositivos, por su forma de funcionamiento, miden la aceleración gravitacional de la Tierra incluso cuando el avión no se encuentra acelerado Este modelo se implementa mediante el siguiente bloque `MATLAB Function`:

```
function [accX_clean,accY_clean,accZ_clean] = fcn(X,Y,Z,mass, phi, theta, g)
%% X measurement
accX_clean = (X./mass) + g.*sin(theta);

%% Y measurement
accY_clean = (Y./mass)-g.*cos(theta).*sin(phi);

%% Z measurement
accZ_clean = (Z./mass)-g.*cos(theta).*cos(phi);
```

Código 5.1 Modelo de acelerómetro de 3 ejes sin los efectos del ruido ν_{accel}.

En esta función, la masa del UAV es pasada como un parámetro `mass`, el cual se extrae del `Workspace` de `MATLAB`. Además se tienen que:

$$\begin{aligned}
\texttt{accX} &\equiv y_{\text{accel},\,x} \\
\texttt{accY} &\equiv y_{\text{accel},\,y} \\
\texttt{accZ} &\equiv y_{\text{accel},\,z} \\
\texttt{noiseX} &\equiv \nu_{\text{accel},\,x} \\
\texttt{noiseY} &\equiv \nu_{\text{accel},\,y} \\
\texttt{noiseZ} &\equiv \nu_{\text{accel},\,z}
\end{aligned}$$

Los resultados obtenidos se muestran en la Figura 5.4.

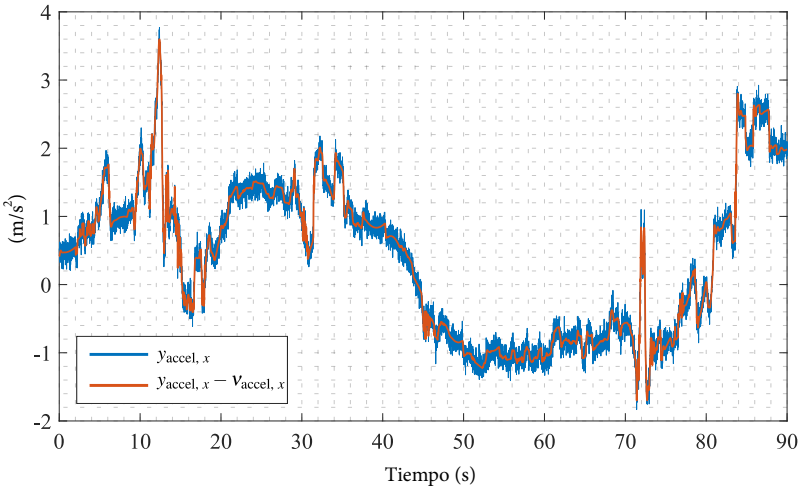


Figura 5.4 Señal real de la aceleración lineal $y_{\text{accel},\,x} - \nu_{\text{accel},\,x}$ y su equivalente medido por el acelerómetro artificial $y_{\text{accel},\,x}$.

Modelado del giroscopio del UAV

En el caso del giroscopio se procede de forma similar. Se tiene que las medidas del mismo cumplen:

$$y_{\text{gyro},\,x} = p + \nu_{\text{gyro},\,x} \tag{5.4}$$
$$y_{\text{gyro},\,y} = q + \nu_{\text{gyro},\,y} \tag{5.5}$$
$$y_{\text{gyro},\,z} = r + \nu_{\text{gyro},\,z} \tag{5.6}$$

Al igual que en el caso anterior, se asume que los ejes del instrumento se encuentran alineados con los ejes cuerpo B, que los sesgos en las señales se pueden mitigar mediante calibración y que el ruido arrojado por el sensor en sus 3 ejes es ruido blanco gaussiano tal que cumple que $\nu_{\text{gyro},\,i} \sim N(0, \sigma_{\nu_{\text{gyro}}})$ con $i = x, y, z$, y $\sigma_{\nu_{\text{gyro}}} = 0.2^\circ/\text{s}$. Además, la nomenclatura utilizada en el simulador es

$$\texttt{p\_meas} \equiv y_{\text{gyro},\,x}$$
$$\texttt{q\_meas} \equiv y_{\text{gyro},\,y}$$
$$\texttt{r\_meas} \equiv y_{\text{gyro},\,z}$$

Los resultados obtenidos se muestran en la Figura 5.5.

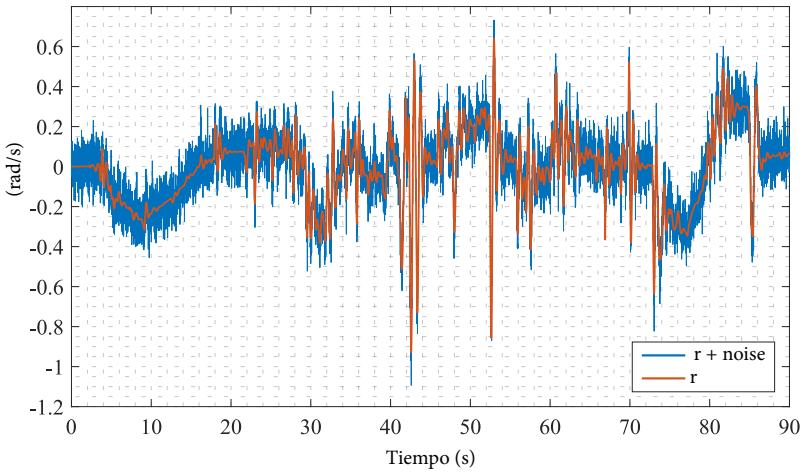


Figura 5.5 Señal real de la velocidad angular $r \equiv$ r y su equivalente medido por el giroscopio artificial $y_{\text{gyro},z} \equiv$ r + noise.

Modelado de otros sensores del UAV para su uso en la identificación del sistema

Además de la IMU, es necesario modelar otro conjunto de sensores necesarios para la identificación efectiva del sistema. Cada uno de ellos se encarga de medir una variable del estado $\mathbf{x}$ o una variable algebraica $\mathbf{x}_a$, dependiendo del caso. A continuación se presentan cuales son estas variables y los nombres con los que son enviados posteriormente al `Workspace` de `MATLAB` para su posterior análisis.

- **Velocidad aerodinámica** V_a. La variable almacenada en el `Workspace` por este sensor tiene por nombre `Va`.
- **Ángulo de ataque** α. La variable almacenada en el `Workspace` tiene por nombre `alpha`.
- **Ángulo de resbalamiento** β. La variable almacenada en el `Workspace` tiene por nombre `beta`.
- **Ángulo de deflexión de los alerones** δ_a. La variable almacenada en el `Workspace` tiene por nombre `deltaa`.
- **Ángulo de deflexión del elevador** δ_e. La variable almacenada en el `Workspace` tiene por nombre `deltae`.
- **Ángulo de deflexión del timón de cola** δ_r.La variable almacenada en el `Workspace` tiene por nombre `deltar`.

- **Densidad del aire** ρ.La variable almacenada en el `Workspace` tiene por nombre `rho`.

En la Figura 5.6 se muestra como se han implementado estas medidas artificiales en el modelo de `Simulink`. Además, en la Figura 5.7 se muestra el resultado de agregar el ruido de los sensores a las señales limpias.

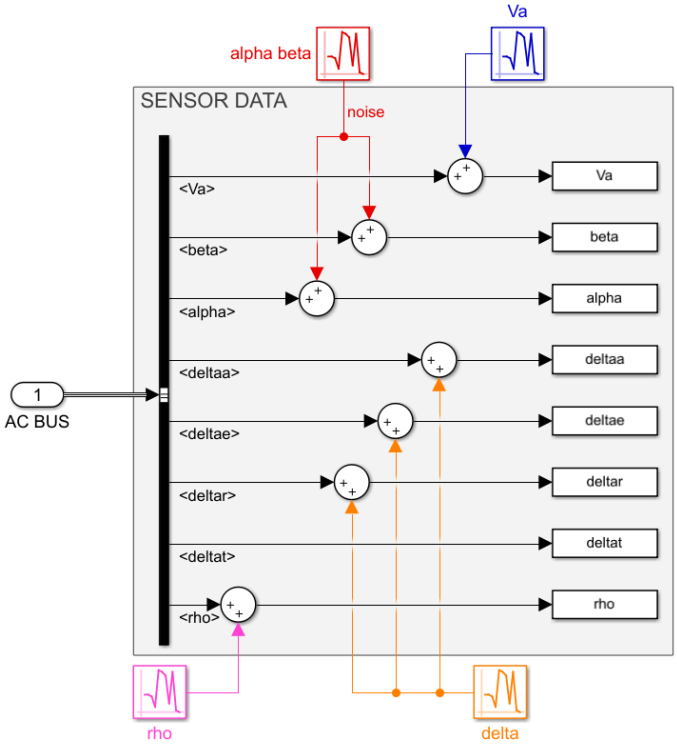


Figura 5.6 Estructura interna del bloque `Other Sensors Recorder`.

Otros datos almacenados tras la simulación

Mediante el `Callback` de parada del modelo, `StopFcn`, que se ejecuta al finalizar una simulación, se realiza el guardado de las variables que se encuentran en el Workspace. Estas se conforman por las variables son las variables del estado **x**, las cuales se almacenan en el archivo `fullStateData.mat`, así como por las variables medidas por los sensores en la sección anterior, las cuales se almacenan en el archivo `simulationData.mat`.

Por último, se crean las variables `aileronValMatrix`, `elevatorValMatrix`, `rudderValMatrix` y `throttleValMatrix`, las cuales son utilizadas como entradas para el modelo de validación. Estas son matrices cuya primera columna es el vector de tiempo de la simulación `time`, y cuya segunda columna corresponde con los vectores `ailerons_val`, `elevator_val`, `rudder_val` o `throuttle_val`, creados en el bloque Input Ajustments. Esto es así debido a que el bloque de `Simulink`, `From Workspace`, utilizado para importar variables desde el `Workspace` de `MATLAB` como señales nuevas, requiere que las variables importadas tengan esta estructura de matriz en particular.

En la Figura 5.8 se observa la configuración utilizada en el bloque `UAV`.

5.1.2 Modelo de validación

Los modelos de `Simulink` para la validación, son versiones del simulador que implementan los coeficientes aerodinámicos identificados por el algoritmo de identificación. De forma que representan al UAV finalmente identificado tras utilizar datos de vuelo reales, y sobre los cuales se pretenden realizar diferentes estudios de la dinámica del avión, así como el diseño de autopilotos para el mismo.

Estos modelos tienen como principales caracteristicas:

- Entradas del tipo `From Workspace` que toman los vectores de los inputs medidos por el modelo de identificación, para recrear los movimientos del piloto en vuelo.

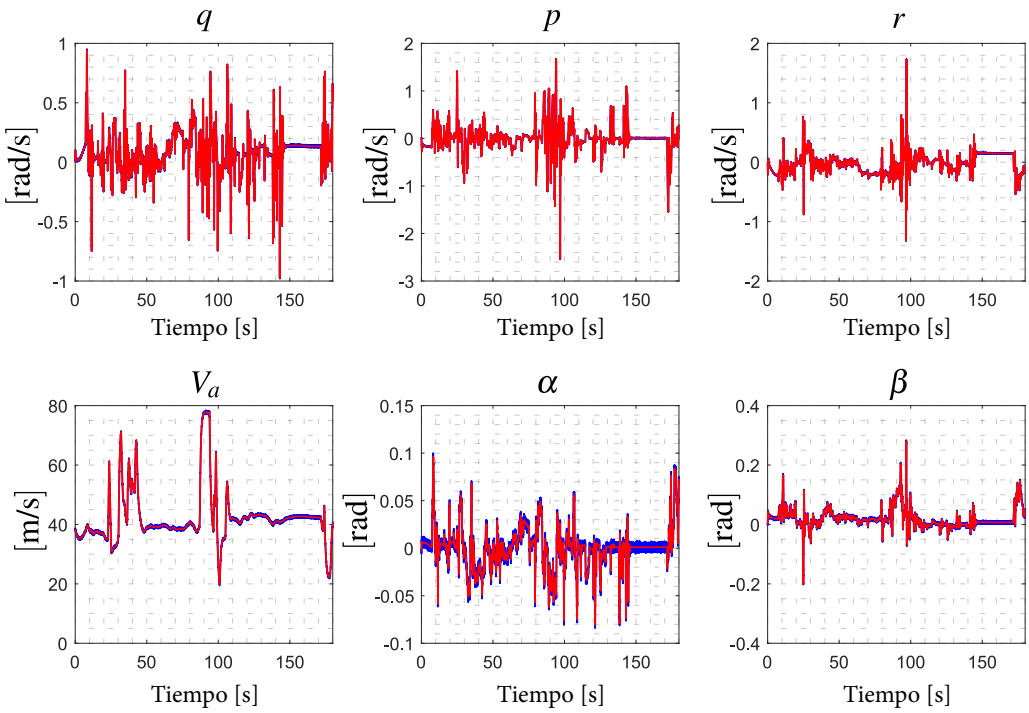


Figura 5.7 Resultado de añadir ruido a las señales medidas por los demás sensores (vuelo de modelo `V3` de 180 s de duración). En rojo se muestra la señal original y en azul la señal con ruido. Como puede apreciarse, el ruido añadido a las señales, en principio, no es muy significativo en el caso de p, q y r, mientras que se aprecia mejor en las medidas de α, β y V_a.

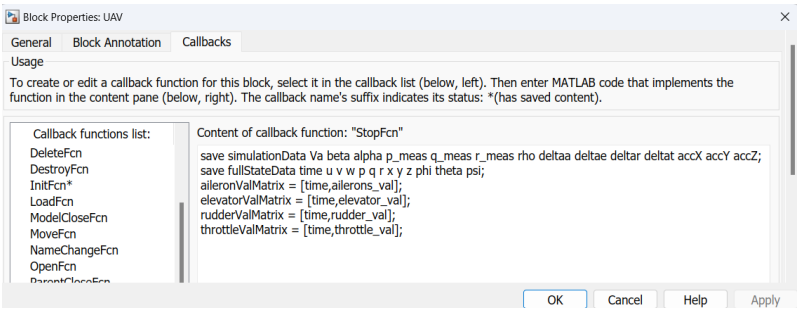


Figura 5.8 Configuración del `Callback/StopFcn` del bloque UAV. Es igual para todas las versiones del simulador.

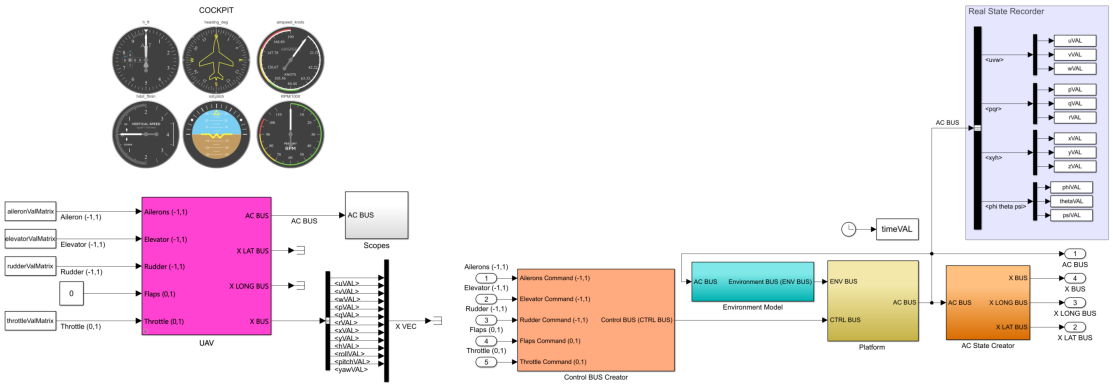
- Modificación del bloque `Flight Data Recorder (FDR)` de forma que únicamente registre las variables reales del estado para realizar la validación, contrastando el valor de estas variables con los estados reales registrados durante el vuelo de identificación.

- Modificación en los `callbacks` del bloque `UAV` para:

  – Configurar los coeficientes recientemente identificados como los coeficientes a utilizar en el modelo (todo esto mediante el archivo de configuración `config_model_validation.m`).

  – Crear una rutina de parada que permita almacenar los estados del modelo de validación en un archivo llamado `validationStateData.mat`.

En la Figura 5.9 se observan las modificaciones indicadas anteriormente. Cabe destacar que la única diferencia en los modelos de validación de las versiones `V3` y `V2` es el bloque `UAV/platform`.



a) Modificación de entradas b) FDR modificada

Figura 5.9 Modelo de validación.

5.2 Identificación de los parámetros del UAV

A continuación se detalla el algoritmo de identificación implementado en el archivo `NONLINEAR_IDENTIFICATION.m`, detallado y comentado en el Apéndice A.6.5.

5.2.1 Planteamiento del problema

Como se muestra en el trabajo *Aerodynamic Parameter Identification and Uncertainty Quantification for Small Unmanned Aircraft* [8], para la identificación de parámetros en aeronaves de ala fija, es común utilizar métodos como el de *mínimos cuadrados*. Estos proporcionan valores estimados de los parámetros que se aproximan a la mejor estimación (o valor real del parámetro si el modelo es apropiado) a medida que se aumenta la cantidad de datos proporcionados al algoritmo. Para realizar este proceso, es necesario plantear en primer lugar las ecuaciones que relacionan los parámetros de interés con las señales que podemos medir, es decir, las ecuaciones dinámicas del UAV.

Al desarrollar las ecuaciones (2.15) de la dinámica rotacional del avión, haciendo referencia al trabajo de Hale, Patil y Roy [9], y sustituyendo en estas las definiciones del C_l, C_m y C_n, se obtienen las expresiones mostradas a continuación:

$$\frac{J_x}{\frac{1}{2}\rho {V_a}^2 Sb}\left[\dot{p}+\frac{J_z-J_y}{J_x}qr-\frac{J_{xz}}{J_x}(pq+\dot{r})\right]=C_{l_0}+C_{l_\beta}\beta+C_{l_p}(b/2V_a)\,p+C_{l_r}(b/2V_a)\,r+C_{l_{\delta_r}}\delta_r+C_{l_{\delta_a}}\delta_a \tag{5.7}$$

$$\frac{J_y}{\frac{1}{2}\rho {V_a}^2 Sc}\left[\dot{q}+\frac{J_x-J_z}{J_y}pr+\frac{J_{xz}}{J_y}(p^2-r^2)\right]=C_m(\alpha)+C_{m_q}\left(\frac{c}{2V_a}q\right)+C_{m_{\delta_e}}\delta_e \tag{5.8}$$

$$\frac{J_z}{\frac{1}{2}\rho {V_a}^2 Sb}\left[\dot{r}+\frac{J_y-J_x}{J_z}pq+\frac{J_{xy}}{J_z}(qr-\dot{p})\right]=C_{n_0}+C_{n_\beta}\beta+C_{n_p}(b/2V_a)\,p+C_{n_r}(b/2V_a)\,r+C_{n_{\delta_r}}\delta_r+C_{n_{\delta_a}}\delta_a \tag{5.9}$$

donde en color rojo, se muestran las señales medidas en la FDR, mientras que en color azul se destacan los parámetros a determinar. Hay tener en cuenta que las velocidades angulares p, q y r han de ser sustituidas por sus equivalentes medidos $y_{\text{gyro},x}, y_{\text{gyro},y}$ y $y_{\text{gyro},z}$, respectivamete. Además se considera que p, q y r son señales ruidosas (o *raw signals*, señales sin ningún procesamiento). Posteriormente se detalla el cálculo y procesado de sus derivadas.

Realizando el mismo procedimiento con el modelo del acelerómetro mostrado en la sección (5.1.1), se llega a las siguientes expresiones.

$$-\frac{T}{\frac{1}{2}\rho {V_a}^2 S}+\frac{\mathrm{m}}{\frac{1}{2}\rho {V_a}^2 S} y_{\text{accel, } x} = C_X(\alpha)+C_{X_q}(\alpha)(c/2V_a)\ q+C_{X_{\delta_e}}(\alpha)\delta_e \tag{5.10}$$

$$\frac{\mathrm{m}}{\frac{1}{2}\rho {V_a}^2 S} y_{\text{accel, } y} = C_{Y_0}+C_{Y_\beta}\beta+C_{Y_p}(b/2V_a)\ p+C_{Y_r}(b/2V_a)\ r+C_{Y_{\delta_r}}\delta_r+C_{Y_{\delta_a}}\delta_a \tag{5.11}$$

$$\frac{\mathrm{m}}{\frac{1}{2}\rho {V_a}^2 S} y_{\text{accel, } z} = C_Z(\alpha)+C_{Z_q}(\alpha)(c/2V_a)\ q+C_{Z_{\delta_e}}(\alpha)\delta_e \tag{5.12}$$

donde $T = T(\rho, V_a, \delta_t)$. Estas ecuaciones relacionan los coeficientes aerodinámicos (parámetros a identificar) con las variables medidas durante el vuelo simulado. Son estas las ecuaciones que se pretenden ajustar utilizando la función `lsqcurvefit()`.

5.2.2 Función `lqscurvefit()`

Para realizar la identificación de los coeficientes aerodinámicos de interés, se hace uso de la función `lsqcurvefit()` del `Optimization Toolbox` de `MATLAB`. Tal y como se comenta en la documentación de dicha función, esta proporciona una interfaz práctica para resolver problemas de ajustes de datos como el planteado en la sección anterior, utilizando el mismo algoritmo de la función `lsqnonlin()`, la cual tiene como objetivo resolver problemas de mínimos cuadrados no lineales.

Utilizando la nomenclatura proporcionada por `Mathworks` en su documentación [9], el problema se plantea de la siguiente manera:

$$\min_{\mathbf{x}} \|\mathbf{F}(\mathbf{x},\mathbf{X}_{\text{data}}) - \mathbf{Y}_{\text{data}}\|_2^2 \tag{5.13}$$

donde el subíndice 2, hace referencia a la normal euclídea, definida como la raíz cuadrada de la suma de los cuadrados de las componentes del vector [10]. Además se tiene que

- $\mathbf{Y} = \mathbf{F}(\mathbf{x},\mathbf{X})$: es una función matricial equivalente al modelo de la aeronave, que implementa las ecuaciones de la sección (5.2.1).
- $\mathbf{x}$: es un vector que representa a los coeficientes aerodinámicos a identificar.
- $\mathbf{X}$data: es una matriz con las señales de entrada del modelo de UAV.
- $\mathbf{Y}$data: es una matriz con las señales de salida del modelo de UAV.

Claramente, lo que a continuación se plantea es un problema de optimización no lineal. Para resolverlo, la función `lsqnonlin()` hace uso de los métodos *Levenberg-Marquardt* y *trust-region-reflective*, cuyos detalles se escapan del alcance de este trabajo.

Es importante destacar que además de las entradas del modelo $\mathbf{X}$, la función $\mathbf{F}$ recibe como entradas los parámetros a identificar $\mathbf{x}$ (de forma que puedan ser ajustados durante el problema de minimización a resolver). En el caso de la función `lqscurvefit()`, también recibe un valor inicial o valor semilla para la búsqueda de soluciones al problema de optimización, $\mathbf{x}_0$, así como un objeto `optimoptions()` para configurar las opciones del problema de optimización, en nuestro caso, para definir las tolerancias en los cálculos. La función $\mathbf{F}$ se implementa mediante el fichero `FUN_v2.m` o `FUN_v3.m` en función de la versión del simulador que se este usando (detallado y comentado en el Apéndice A.6.3), el cual es utilizado por la función `lsqcurvefit()` dentro del script de identificación `NONLINEAR_IDENTIFICATION.m`. Si se utiliza el código de colores de la sección anterior, se puede ver claramente que las ecuaciones de dicha sección puede representarse de la siguiente forma

$$\mathbf{Y}_{\text{data}} = \mathbf{F}(\mathbf{x},\mathbf{X}_{\text{data}})$$

Como un último detalle, hay que considerar que estos métodos no toman en cuenta perturbaciones como podrían ser el viento (uniforme o turbulento). Es por esto que en el trabajo desarrollado a lo largo de este capítulo, no se utiliza el bloque de modelado de vientos del simulador, y se asume que los vuelos son

realizados en días con viento en calma y poca turbulencia.

5.2.3 Cálculo de las derivadas $\dot{p}, \dot{q}, \dot{r}$

Claramente, existe la necesidad de calcular las derivadas $\dot{p}, \dot{q}$ y $\dot{r}$, es decir, las derivadas de las señales medidas por los 3 ejes del giroscopio. Este es un problema de cálculo numérico mal condicionado que aumenta considerablemente los errores de calculo respecto a la integración numérica típica.

En este trabajo se procede a realizar el cálculo aproximado de estas derivadas mediante el script `ratesDerivatives.m`, detallado y comentado en el Apéndice A.6.1. En el mismo, se sigue el siguiente procedimiento:

1. Mediante la función `filtfilt()` del `Signal Processing Toolbox`, se le aplica un filtro de fase nula a las señales del giroscopio, ajustando los parámetros del filtro mediante la función propia `optimalFiltering()`, la cual selecciona el `windowSize` óptimo para minimizar el error relativo entre la señal filtrada y la señal original.
2. Se procede a calcular el vector de diferencias finitas de cada una de las señales, tal que $\dot{y} \approx \Delta y / \mathtt{ST}$, $y = p, q, r$, donde `ST` es el *sample time* utilizado en la simulación.
3. Por último, se vuelven a filtrar las señales recién calculadas por diferencias finitas.

En la Figura 5.10 se muestra un resumen de las señales obtenidas a lo largo de este proceso. Sin pérdida de generalidad, se estimuló al sistema con una entrada cosenoidal en el elevador, manteniendo el resto de entradas en la posición de trimado, de forma que este resultado sea extrapolable a cualquier otra señal de entrada (que pueda expresarse como una suma infinita de senos y cosenos).

Como se puede observar, la señal azul con ruido es la medición realizada en por el giroscopio en el eje y_b. La señal roja, es el resultado de pasar la señal medida por el LPF. Luego, la señal rosa o magenta, representa las diferencias finitas calculadas sobre la señal roja y por último, se tiene a el senoide azul, que representa el resultado de filtrar la señal rosa para quitarle sus componentes de alta frecuencia.

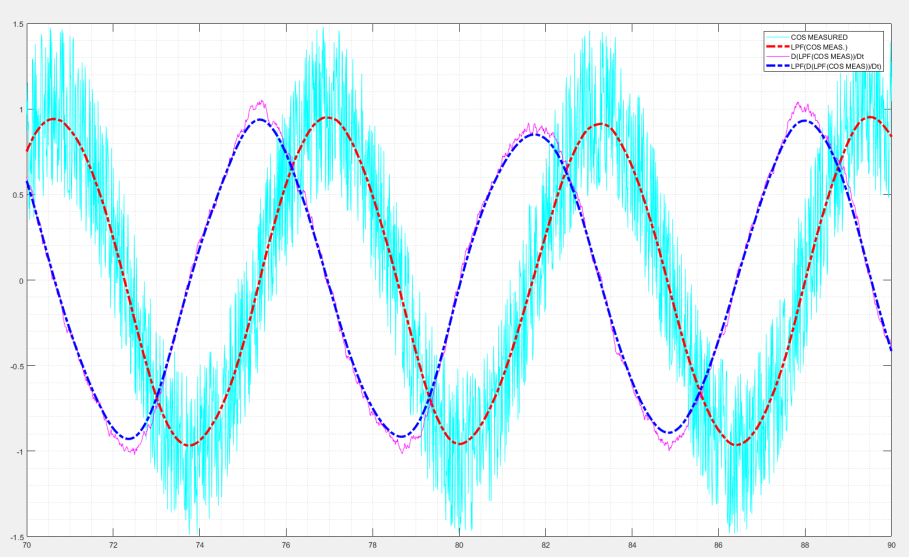


Figura 5.10 Señales del giroscopio a lo largo del proceso de filtrado y derivación..

5.2.4 Script de identificacion final

A continuación, se explica brevemente las diferentes secciones del código de identificación. El mismo se puede encontrar detallado y comentado en el Apéndice A.6.5. Cabe destacar que existen 2 códigos de identificación diferentes debido a las diferencias en los sistemas propulsivos y modelos aerodinámicos de los simuladores

V3 y V2 con los que se trabajan en este capítulo. Estas diferencias se detallan más adelante.

Sección `Ajustment of vector sizes`

En esta sección se realiza el procesado y calculo de las derivadas de las señales del giroscopio, mediante el llamado al script `ratesDerivatives.m`. Esto se realiza inmediatamente después de realizar una simulación, ya que es necesario que los vectores que almacenan los valores de las velocidades angulares p, q y r, se encuentren en el Workspace.

Debido a que se realiza el cálculo de las derivadas de las velocidades angulares por diferencias finitas, estas poseen una longitud de $N-1$, donde N es el número de pasos de integración tal que el tiempo total de la simulación es $T = N \cdot \mathtt{ST}$. Es por ello que es necesario eliminar 2 componentes del resto de vectores para tener congruencia dimensional. Esto se hace mediante la función propia `myResize()`.

Sección `Build the DATA MATRICES`

En este bloque, se crean todas las variables de entrada necesarias para el correcto funcionamiento de la función `lsqcurvefit()`.

En primer lugar se calculan las estimaciones de los valores de C_l, C_m y C_n, las cuales reciben los nombres `Cl_meas`, `Cm_meas` y `Cn_meas` respectivamente. Estas se corresponden con los primeros miembros de las ecuaciones (5.7), (5.8) y (5.9).

Inmediatamente después se establecen los extremos inferiores y superiores de los intervalos de búsqueda para cada uno de los coeficientes, de forma que se calculen más rápidamente. En este caso, se establece el signo de algunos coeficientes. Estos se conocen ya que la estabilidad de la aeronave dependerá de los mismos, por tanto, para que la aeronave sea estable, es necesario que estos tengan unos signos particulares.

Por último, se crean, el vector de coeficientes semilla $\mathbf{x}$, y las matrices con datos de entrada y salida del modelo, es decir $\mathbf{X}_{\text{data}}$ y $\mathbf{Y}_{\text{data}}$. Estos reciben los nombres `x0`, `XDATA` y `YDATA`. Usando la sintaxis de `MATLAB`, se tiene que

$$\mathtt{XDATA} = \mathtt{[Va,beta,alpha,p,q,r,rho,deltaa,deltae,deltar,deltat]}$$
$$\mathtt{YDATA} = \mathtt{[accX,accY,accZ,Cl\_meas,Cm\_meas,Cn\_meas]}$$

Sección `nonlinear regression function`

Incluye únicamente a la función `lsqcurvefit()` para ser ejecutada individualmente. La salida de esta función, es decir, el vector de coeficientes identificado, se almacena en la variable `xfit`, de la siguiente manera.

```
xfit = lsqcurvefit(FUN,x0,XDATA,YDATA,lb,ub,options)
```

Sección `Results`

En esta sección se presentan y se analizan los resultados obtenidos. El análisis cuantitativo consiste esencialmente en comparar los coeficientes obtenidos con los coeficientes reales utilizados en el modelo que generó los datos para la identificación.

6 Análisis de los resultados

A continuación, se presentan los resultados obtenidos de la simulación e identificación de los sistemas descritos en los capítulos anteriores. Además de los resultados finales para la identificación de los modelos `V2` y `V3`, se presentan resultados intermedios obtenidos a lo largo del proceso de desarrollo, más puntualmente, la identificación de los parámetros responsables únicamente de las fuerzas sobre la aeronave.

A continuación, se considera como un experimento de recolección de datos básicos, un vuelo pilotado mediante transmisora RC convencional, por vuelo completamente visual y de unos 25 min. Asumiendo un escenario hipotético en el cual se realizan 2 vuelos diarios durante una campaña de recolección de datos de un único día, se tendrían en total $T_{\text{flight}} = 50\,\text{min}$ de vuelo registradas.

Tomando además una frecuencia de muestreo relativamente baja para los instrumentos de medida, $f_s = 60\,\text{Hz}$, se tendría que cada variable medida estaría conformada por un vector de 180 000 componentes, es decir, $N = 180\,000$ puntos de medida. Se ha optado por una frecuencia de muestreo baja de forma que se pueda tomar en cuenta mediciones con sensores de baja gama, entendiendo que evidentemente, sistemas con mayores frecuencias de muestreo recolectan mayor cantidad de datos en el mismo intervalo de tiempo, y por ende, se puede mejorar el resultado final obtenido para el mismo tiempo de vuelo (dadas las características de los algoritmos utilizados, ver sección 5.2.1).

Como se puede notar, se han supuesto escenarios pesimistas que minimizan el número de datos recolectados, de forma que se realiza la identificación en el peor caso posible en términos de cantidad de datos adquiridos.

Para evitar realizar vuelos tan largos en la generación sintética de datos, se disminuye el *time step* o tiempo de muestreo de la simulación a $\Delta t^* = 0.001\,\text{s}$ lo cual implica que, para la misma cantidad de puntos de datos N, se tiene un tiempo de vuelo o tiempo de simulación de $T^* = N \cdot \Delta t^* = 3\,\text{min} = 180\,\text{s}$.

Se procederá a comparar los resultados de identificar el UAV con ambos simuladores, las mejoras existentes al aumentar el número de datos recolectados y las diferencias por realizar la identificación únicamente de los coeficientes responsables de las fuerzas aerodinámicas.

6.1 Simulador `V3`. Modelo Simplificado.

En el caso del simulador `V3`, se ha procedido a realizar simulaciones con y sin ruido en los sensores. Además la identificación se ha aplicado de dos maneras diferentes: identificación de los coeficientes aerodinámicos correspondientes a las fuerzas únicamente, y la identificación de todos los coeficientes aerodinámicos.

6.1.1 Identificación con sensores ideales

Tras realizarse una simulación de 3 min de duración sobre el simulador `V3`, y ejecutar el script de identificación, se obtienen los resultados mostrados en la Tabla 6.1. Estos se puede analizar de una forma más cualitativa en

la Figura 6.1

Tabla 6.1 Coeficientes identificados en el modelo `V3`. Sensores ideales, 3 min de simulación.

ID	Nombres de los coeficientes	Valor Identificado	Valor Real	Error Relativo (%)
1	`CD0`	0.019301	0.0193	0
2	`CD_alpha`	0.098156	0.0987	1
3	`CDq`	-0.0027029	0	100
4	`CD_deltae`	0.013316	0.0135	1
5	`CL0`	0.23003	0.23	0
6	`CL_alpha`	5.6317	5.61	0
7	`CLq`	7.7079	7.65	1
8	`CL_deltae`	0.13569	0.13	4
9	`Cy_beta`	-0.83	-0.83	0
10	`Cy_p`	0	0	0
11	`Cy_r`	0	0	0
12	`Cy_da`	0.075	0.075	0
13	`Cy_dr`	0.19	0.19	0
14	`Cl_beta`	-0.11091	-0.13	15
15	`Cl_p`	-0.45763	-0.51	10
16	`Cl_r`	0.21778	0.25	13
17	`Cl_dr`	0	0.0024	100
18	`Cl_da`	0.15047	0.17	11
19	`Cm0`	0.016197	0.0135	20
20	`Cm_alpha`	-3.2848	-2.74	20
21	`Cm_q`	-45.843	-38.21	20
22	`Cm_de`	-1.1871	-0.99	20
23	`Cn_beta`	-0.27103	0.073	271
24	`Cn_p`	-1.1149	-0.069	1516
25	`Cn_r`	0.52144	-0.095	449
26	`Cn_dr`	-0.0050145	-0.069	93
27	`Cn_da`	0.36687	-0.011	3235

Como puede observarse, la identificación del sistema se realiza casi a la perfección para los coeficientes de fuerzas (desde el 1 hasta el 13). Se tiene que en la gran mayoría de estos, el error relativo es inferior al 1 %.

Para el resto de coeficientes correspondientes a los momentos sobre la aeronave, se tiene un ajuste relativamente decente en casos como el del parámetro C_{l_p} (donde el error es de un 10 %), como también se observan desviaciones muy importantes como en el caso del $C_{n_{\delta_a}}$ de hasta un 3235 %. Existe la posibilidad de que se esté identificando al ruido, lo cual afecta negativamente a los resultados.

Para comprobar la utilidad de estos resultados, se procede a realizar la validación del modelo pero únicamente con los coeficientes de fuerzas identificados, es decir, los parámetros desde el 1 hasta el 13, corriendo la simulación de `simulador2017bVALIDATION_v3.slx`, y posteriormente, ejecutando el script de graficación `state_compare_plot.m`, que nos permite comparar las variables de estado x, y, z, ϕ, θ y ψ. El resultado se muestra en la Figura 6.2. No se realiza esta comparación con un modelo de validación que incorpore todos los coeficientes identificados ya que por los valores obtenidos, el avión se vuelve completamente inestable y realiza maniobras sin ninguna clase de sentido físico o interés.

Como se puede observar, las variables de estado son prácticamente idénticas durante todo el vuelo, salvo leves desviaciones en la variable x durante los últimos minutos del vuelo, las cuales son casi imperceptibles.

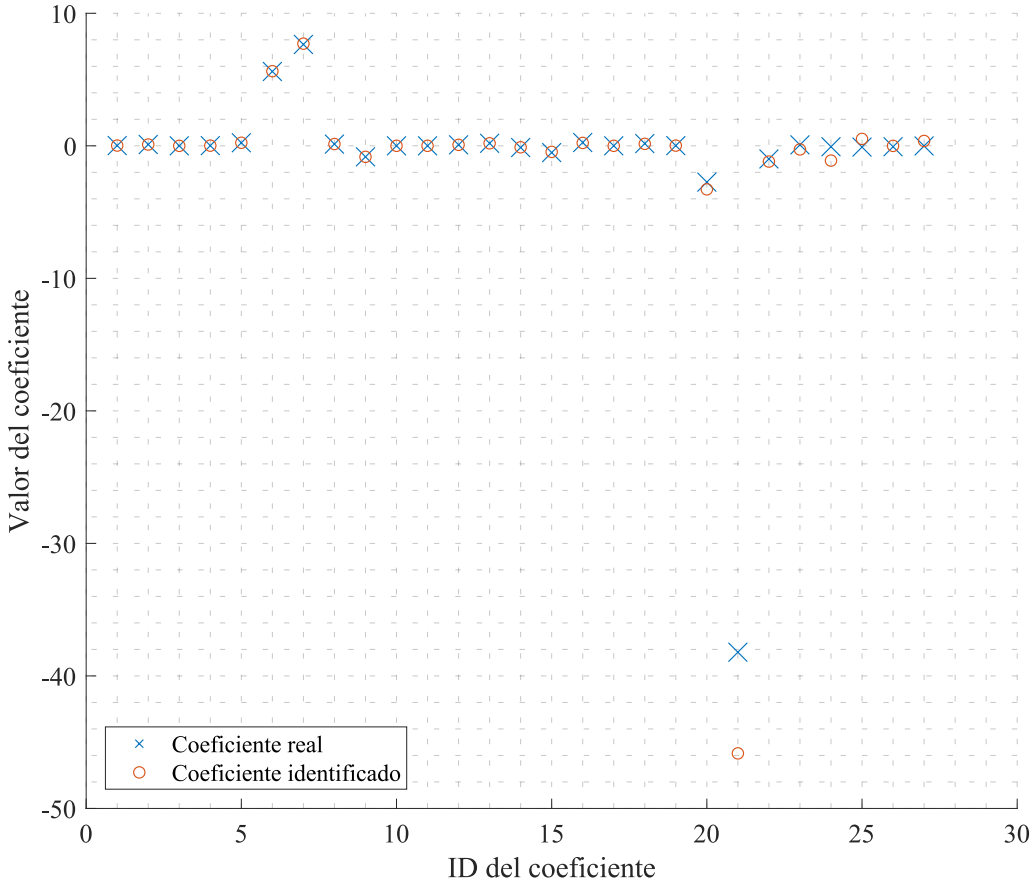


Figura 6.1 Resultados de la identificación del sistema V3 con sensores ideales y una simulación de 3 min.

6.1.2 Identificación con sensores reales

Utilizando sensores reales, es decir, añadiendo ruido blanco gaussiano con las varianzas típicas de los sensores utilizados, e identificando el UAV con datos de vuelo de una simulación de 3 min de duración, se obtiene la Tabla 6.2. En la Figura 6.3 se obtiene una visión más amplia de los resultados.

6.2 Simulador V2. Modelo intermedio.

Para el modelo V2 se sigue el mismo procedimiento con la diferencia de que se utiliza un script de identificación adaptado para ajustarse al modelo propulsivo y aerodinámico de este simulador en particular. En principio, al hacer más complejos los modelos se espera que la identificación sea menos satisfactoria, debido a que son requeridos más cálculos, lo cual propicia al propagación y acumulación de errores numéricos.

6.2.1 Identificación de todos los coeficientes con 3 min de vuelo

A igual que en la sección anterior, en esta sección se emulan sensores reales para medir las variables de interés para la identificación. En este caso se pretende realizar un simulación de 3 minutos de duración que nos asegura 180000 puntos. Tras realizar el vuelo y ejecutar el respectivo script de identificación se obtienen los resultados mostrados en la Tabla 6.3 o en la Figura 6.4.

En este caso, se logran identificar satisfactoriamente algunos coeficientes como el C_{Y_β} o el $C_{Y_{\delta_r}}$ pero en general, se obtienen peores resultados que al identificar un vuelo con el modelo V3. De hecho, podríamos considerar valor promedio de errores relativos. En este caso, tiene un valor de 231 %. Se procede a continuación a realizar una simulación del doble de duración, aspirando a mejorar este último resultado.

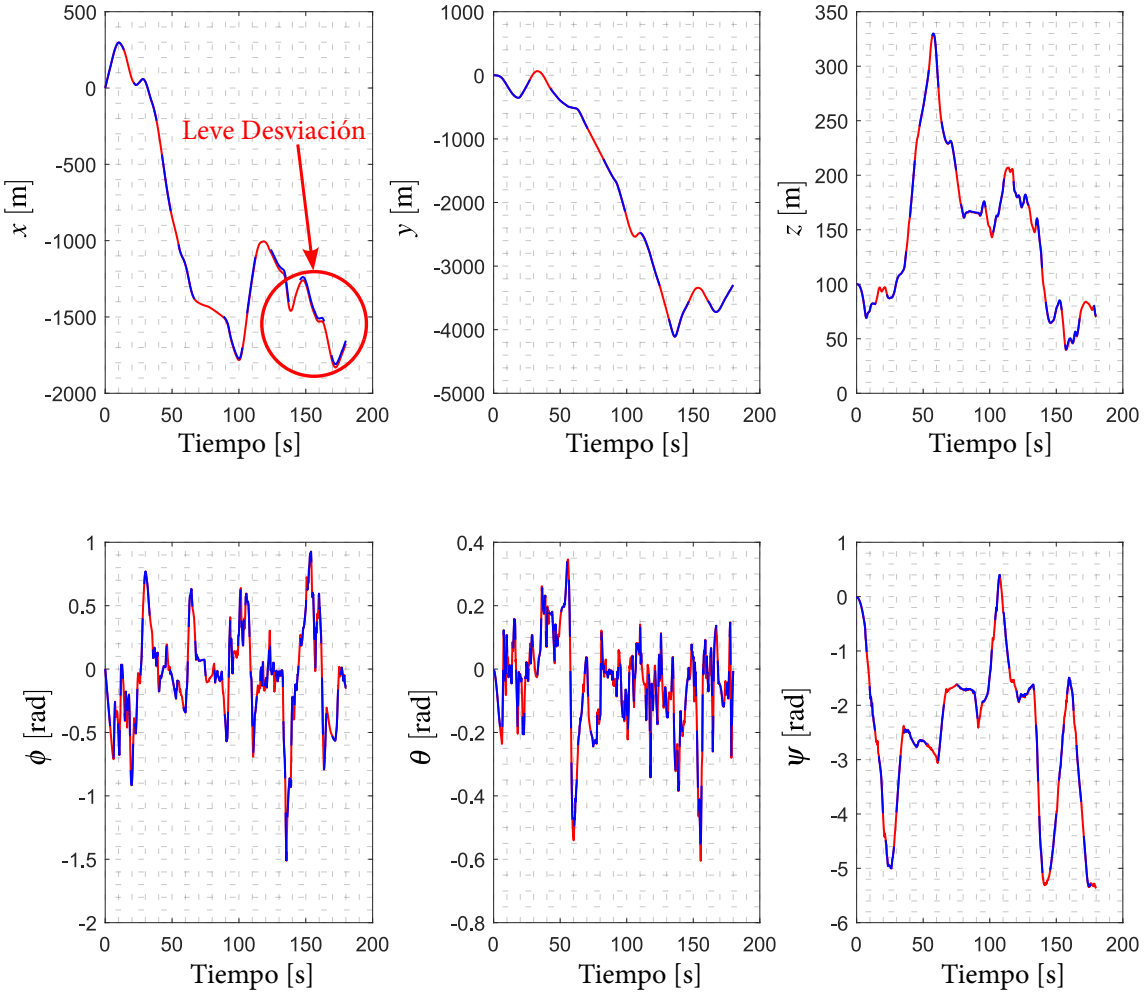


Figura 6.2 Validación de las variables de estado para la identificación utilizando una simulación de 3 min en el modelo `V3` e incorporando al modelo de validación únicamente los coeficientes de fuerzas. En línea roja continua se muestra el estado del modelo de generación de datos, mientras que la línea azul discontinua, representa el estado del modelo de validación.

6.2.2 Identificación de todos los coeficientes con 6 min de vuelo

Por último, se procede a realizar una simulación de vuelo de 6 min de duración, y nuevamente, se utilizan los modelos de los sensores reales para la recolección de datos. El duplicar el tiempo de simulación nos asegura un total de 360 000 puntos, el doble que en la simulación anterior. Tras realizar el vuelo y ejecutar el respectivo script de identificación se obtienen los resultados mostrados en la Tabla 6.4 o en la Figura 6.5.

Es evidente la mejoría en coeficientes como el C_{L_α}, el cual reduce su error en un 4 % (7$^{\text{mo}}$ punto en la Figura 6.5, de izquierda a derecha), o por ejemplo el C_{m_q}, que disminuye su error de un 13 % a un 2 %. Además, se tiene en promedio un error relativo de 229 %, lo cual se traduce en que, a pesar de duplicar el número de puntos medidos, el error relativo promedio solo disminuye en un 2 %. Una disminución tan poco significativa sugiere la existencia de un error sistemático en el procedimiento de identificación, lo cual podría justificar la invarianza del estadístico calculado anteriormente.

Tabla 6.2 Coeficientes identificados en el modelo V3. Sensores reales, 3 min de simulación.

ID	Nombres de los coeficientes	Valor Identificado	Valor Real	Error Relativo (%)
1	CD0	0.021955	0.0193	14
2	CD_alpha	-0.36324	0.0987	268
3	CDq	-7.6938	0	669
4	CD_deltae	-0.15931	0.0135	1080
5	CL0	0.2379	0.23	3
6	CL_alpha	4.2526	5.61	24
7	CLq	-15.037	7.65	97
8	CL_deltae	-0.37781	0.13	191
9	Cy_beta	-0.8706	-0.83	5
10	Cy_p	-0.12498	0	88
11	Cy_r	0.086302	0	91
12	Cy_da	0.12085	0.075	61
13	Cy_dr	0.19845	0.19	4
14	Cl_beta	-0.10668	-0.13	18
15	Cl_p	-0.44026	-0.51	14
16	Cl_r	0.21014	0.25	16
17	Cl_dr	0	0.0024	100
18	Cl_da	0.14474	0.17	15
19	Cm0	0.012607	0.0135	7
20	Cm_alpha	-2.6471	-2.74	3
21	Cm_q	-34.891	-38.21	9
22	Cm_de	-0.94423	-0.99	5
23	Cn_beta	-0.25443	0.073	249
24	Cn_p	-1.0525	-0.069	1425
25	Cn_r	0.50071	-0.095	427
26	Cn_dr	-0.0053403	-0.069	92
27	Cn_da	0.3458	-0.011	3044

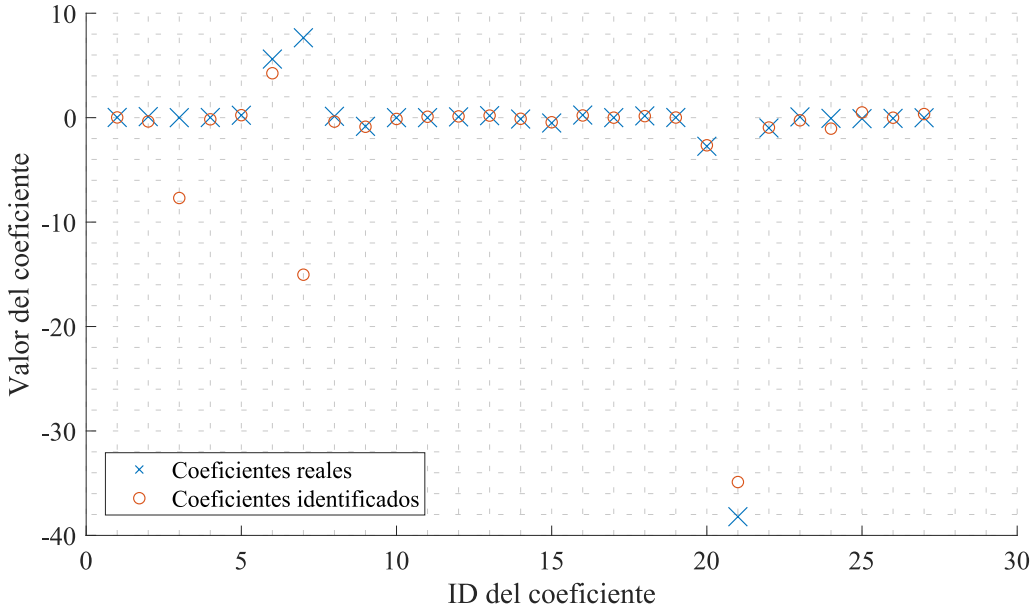


Figura 6.3 Resultados de la identificación del sistema V3 con sensores reales y una simulación de 3 min.

Tabla 6.3 Coeficientes identificados en el modelo V2. Sensores reales, 3 min de simulación.

ID	Nombres de los coeficientes	Valor Identificado	Valor Real	Error Relativo (%)
1	CDp	0.026795	0.0234	15
2	k2	-0.043709	-0.0241	81
3	k1	0.070444	0.0537	31
4	CDq	0.25624	0	74
5	CD_deltae	0	0.0135	100
6	CL0	0.23282	0.23	1
7	CL_alpha	4.9188	5.61	12
8	CLq	0	7.65	100
9	CL_deltae	-0.11485	0.13	188
10	Cy_beta	-0.83844	-0.83	1
11	Cy_p	-0.0040247	0	100
12	Cy_r	0.043303	0	96
13	Cy_da	0.08194	0.075	9
14	Cy_dr	0.19695	0.19	4
15	Cl_beta	-0.10793	-0.13	17
16	Cl_p	-0.4459	-0.51	13
17	Cl_r	0.21535	0.25	14
18	Cl_dr	0	0.0024	100
19	Cl_da	0.14677	0.17	14
20	Cm0	0.012461	0.0135	8
21	Cm_alpha	-2.7457	-2.74	0
22	Cm_q	-33.248	-38.21	13
23	Cm_de	-0.96263	-0.99	3
24	Cn_beta	-0.2381	0.073	426
25	Cn_p	-0.98271	-0.069	1324
26	Cn_r	0.47647	-0.095	602
27	Cn_dr	-0.0035086	-0.069	95
28	Cn_da	0.32186	-0.011	3026

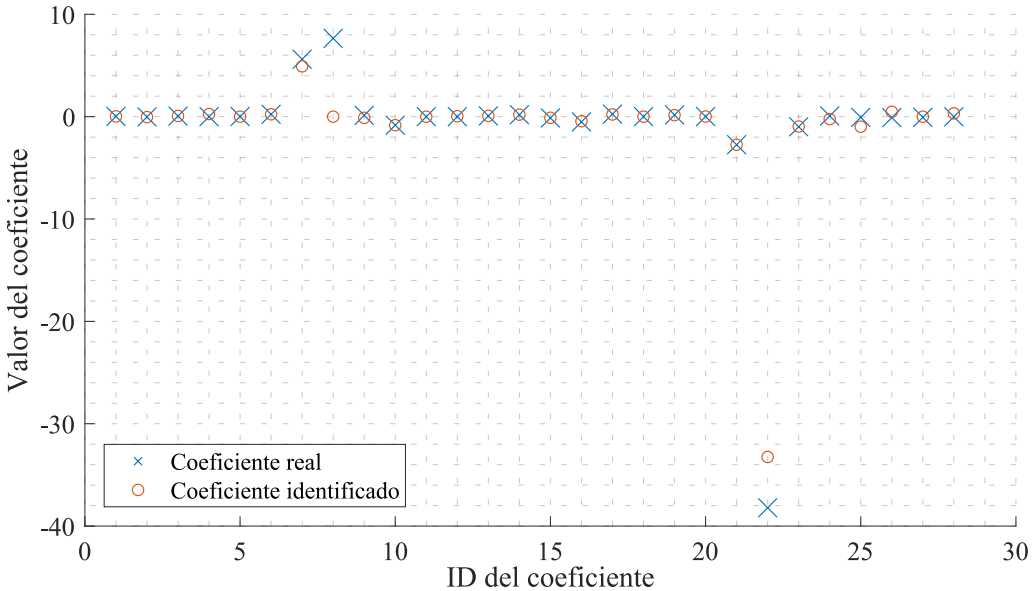


Figura 6.4 Resultados de la identificación del sistema V2 con sensores reales y una simulación de 3 min.

Tabla 6.4 Coeficientes identificados en el modelo V2. Sensores reales, 6 min de simulación.

ID	Nombres de los coeficientes	Valor Identificado	Valor Real	Error Relativo (%)
1	CDp	0.025325	0.0234	8
2	k2	-0.038359	-0.0241	59
3	k1	0.064824	0.0537	21
4	CDq	-0.29842	0	130
5	CD_deltae	0	0.0135	100
6	CL0	0.23231	0.23	1
7	CL_alpha	5.1778	5.61	8
8	CLq	0	7.65	100
9	CL_deltae	-0.037756	0.13	129
10	Cy_beta	-0.85931	-0.83	4
11	Cy_p	-0.083628	0	108
12	Cy_r	0.06506	0	93
13	Cy_da	0.10617	0.075	42
14	Cy_dr	0.1975	0.19	4
15	Cl_beta	-0.10658	-0.13	18
16	Cl_p	-0.44111	-0.51	14
17	Cl_r	0.21406	0.25	14
18	Cl_dr	0	0.0024	100
19	Cl_da	0.14476	0.17	15
20	Cm0	0.013823	0.0135	2
21	Cm_alpha	-2.9476	-2.74	8
22	Cm_q	-37.515	-38.21	2
23	Cm_de	-1.045	-0.99	6
24	Cn_beta	-0.23654	0.073	424
25	Cn_p	-0.97944	-0.069	1319
26	Cn_r	0.43741	-0.095	560
27	Cn_dr	-0.0082433	-0.069	88
28	Cn_da	0.32526	-0.011	3057

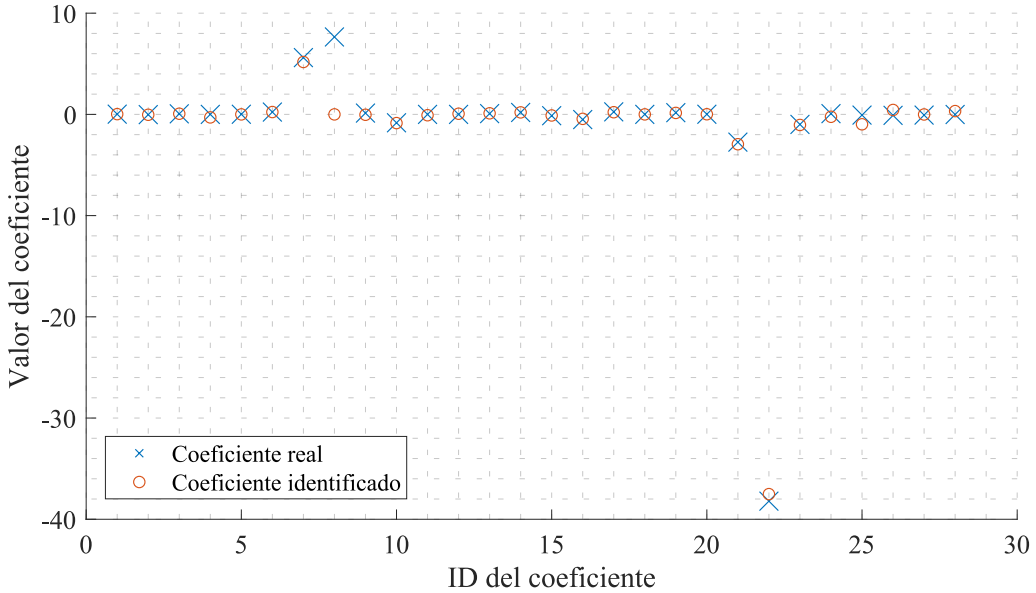


Figura 6.5 Resultados de la identificación del sistema V2 con sensores reales y una simulación de 6 min.

7 Conclusiones y perspectiva futura

En este trabajo, se han desarrollado con éxito múltiples simuladores de vuelo para UAVs, así como un método de identificación de parámetros aerodinámicos a través del método de mínimos cuadrados.

En primer lugar, se han creado tres simuladores de vuelo que involucran dos modelos de resistencia aerodinámica distintos: la polar parabólica extendida y el modelo de resistencia lineal con el AOA (ángulo de ataque), junto con tres modelos propulsivos. El primero de ellos se basó en la Teoría Ideal de Froude para hélices, mientras que los otros dos utilizan datos experimentales en túneles de viento para modelar el empuje y el par proporcionados por la hélice. El modelo propulsivo más complejo integra un modelo de motor BLDC con parámetros que pueden medirse perfectamente mediante instrumentos de laboratorio sencillos, como un multímetro. Además, en todos los modelos de `Simulink` se incorporó una interfaz usuario-máquina que permitía pilotar el UAV en tiempo real mientras se visualizaba el vuelo a través del software `Flight Gear`. Se desarrolló un método para determinar los modelos linealizados de los UAVs a simular, así como un algoritmo en `MATLAB` para estudiar los modos de vuelo de dichos UAVs. Se logró pilotar exitosamente todos los simuladores, proporcionando al piloto una experiencia realista comparable a la de simuladores de RC comerciales como `Real Flight`.

Posteriormente, se diseñó un algoritmo de identificación de parámetros aerodinámicos basado en el método de mínimos cuadrados. En este, se buscó minimizar una función objetivo compuesta por la norma de segundo orden del vector de error que resultaba de la diferencia entre los datos medidos en los vuelos sintéticos y los datos predichos por el modelo de identificación. Estos datos se midieron mediante bloques que emulaban la captación de datos por sensores reales, incorporando ruido blanco gaussiano con media cero y las varianzas típicas presentes en este tipo de sensores. Se logró identificar exitosamente un UAV modelado con la versión `V3` del simulador (la más simplificada), obteniendo resultados satisfactorios para los coeficientes aerodinámicos relativos a la fuerza bajo la condición de poseer sensores ideales (sin ruido). Por otra parte, no se pudo realizar una correcta identificación de parámetros aerodinámicos relacionados con los momentos que actúan sobre la aeronave, ni una identificación adecuada en general de los parámetros cuando los modelos de simulación registraban ruido en sus sensores.

Como principales implicaciones, se destaca que la implementación de un modelo de empuje y par basado en datos de túneles de viento permite contar con un modelo más preciso al desarrollar sistemas de control en velocidad o altitud, como los presentes en el Autopiloto CATEC. Esto se logra al usar datos proporcionados por el fabricante de la hélice, así como mediciones propias si están disponibles. La integración con el modelo de motor BLDC presentado por Beard y McLain en su libro [2] genera un modelo lo suficientemente preciso para la mayoría de aplicaciones prácticas, incluido el análisis de las actuaciones de los UAVs diseñados por el equipo VANTUS AeroTeam de la Escuela Técnica Superior de Ingeniería. Esto permite conocer con mayor precisión valores característicos del rendimiento de una aeronave, como su autonomía, alcance, velocidad máxima y de crucero, tiempo de despegue, entre otros. Además, si se tienen sensores de alta calidad y alta tolerancia al ruido, es posible considerar el uso de los algoritmos de identificación aquí presentados para calcular los parámetros aerodinámicos de fuerzas, y así analizar las actuaciones de un UAV real en estudio (es decir, estudiar el movimiento de la aeronave con tan solo 3 grados de libertad).

Un aspecto a mejorar en este trabajo es abordar el cálculo de las derivadas $\dot{p}, \dot{q}$ y $\dot{r}$ a partir de las señales

ruidosas del giroscopio. Dado que la derivación numérica enfrenta problemas de mal condicionamiento, y considerando el ruido presente en las señales a derivar, resolver este problema sin incurrir en grandes errores numéricos resulta complicado. Por lo tanto, los coeficientes de momentos, que dependen principalmente de estas señales derivadas, presentan los peores resultados. Además, la posible existencia de errores sistemáticos en el modelo de identificación implementado en las funciones `FUN_v2` y `FUN_v3` podría explicar la escasa mejora en el ajuste de datos al duplicar el número de puntos medidos durante la simulación. Se podría usar el modelo de simulador presentado para generar datos y luego identificar y modelar el avión mediante métodos como las redes neuronales, que podrían mejorar la identificación y abordar las no-linealidades de la dinámica del UAV, como los mostrados en [10]. Otra área de mejora sería el modelado de un magnetómetro para la aeronave, que mejoraría la estimación del ángulo ψ. Además, se podría calcular la actitud de la aeronave utilizando los datos de los acelerómetros, como se muestra en [2].

Este proyecto puede dar lugar a futuros trabajos que amplíen las capacidades del simulador presentado, incluida la integración con autopilotos (como el que se pretende desarrollar en VANTUS), la navegación mediante waypoints e incluso el estudio del rendimiento y las actuaciones de la aeronave (autonomía, alcance y otros datos relevantes). Además, sería interesante modelar la interacción del tren de aterrizaje de la aeronave con el terreno para prever la distancia de despegue o las cargas de impacto en el aterrizaje. Todas estas variables son de gran importancia en competencias de aeromodelismo universitario como SAE AeroDesign o XtraChallenge2023, en las que el equipo VANTUS participa, así como para la planificación de misiones con UAVs comerciales disponibles en FADA-CATEC.

Apéndice A
Códigos de MATLAB

Los códigos desarrollados para la configuración del simulador, el estudio de los modelos linealizados y la obtención de los parámetros de los mismos, se incluyen en el presente apéndice.

Se listan de más bajo nivel a más alto nivel de integración, de forma que al final de cada sección se muestra un script dependiente de todos los anteriores (el de más alto nivel), para generar la salida deseada.

Como se ha comentado en el capítulo 5, debido a la existencia de 2 simuladores con modelos aerodinámicos diferentes y por tanto, con parámetros a identificar distintos, se tienen 2 versiones de los scripts de identificación, de forma que se pueda realizar la identificación del sistema correctamente en ambos casos.

A.1 Script de configuración de una aeronave para el Simulador y los modelos de identificación - config_model.m

```
%% CONFIGURATION SCRIPT
%
% This script sets the default the parameters of the Simulink Simulators.
% These parameters should be ajusted in order to simulate the desired UAV.
%
% All variables most be expressed in SI UNITS.
%
% The default aircraft has the AerosondeUAV airframe parameters but with
% the Polar Curves and Lift Curve of the JALEO-1, designed by VANTUS
% AeroDesign Team from the University of Seville.
%
% The Aerosonde UAV parameters are reported in the book: "Small Unmanned
% Aircraft - Beard&McLain" while the JALEO-1 technical report can be
% requested to: blancovillafane@gmail.com
%
%% Load Propulsion Data
run('LOOKUP_TABLES.m');
ST = 0.01;           % SAMPLE TIME
%% Geometry and mass properties

% ============================== %

b          = 2.9;     % wing span
S          = 0.55;    % wing surface
c_bar      = 0.19;    % mean chord
delta_max  = 0.35;    % max deflection of the control surfaces

% ============================== %
```

```
29
30 mass    = 11;          % Mass of the AC
31 Jx      = 0.8244;
32 Jy      = 1.135;
33 Jz      = 1.759;
34 Jxz     = 0.120;
35 J       = [Jx,   0,      -Jxz;...
36            0,    Jy,       0;...
37           -Jxz,  0,       Jz];    % Inertia Tensor
38 Jxy     = 0;
39 % ================================ %
40
41 % Computation of gamma inertial variables
42 gamma  = Jx*Jz-Jxz^2;
43 gamma1 = Jxz*(Jx - Jy + Jz)/(gamma);
44 gamma2 = (Jz*(Jz - Jy)+Jxz^2)/gamma;
45 gamma3 = Jz/gamma;
46 gamma4 = Jxz/gamma;
47 gamma5 = (Jz-Jx)/Jy;
48 gamma6 = Jxz/Jy;
49 gamma7 = (Jx*(Jx - Jy) + Jxz^2)/gamma;
50 gamma8 = Jx/gamma;
51
52 % ================================ %
53 %% Simulation initial conditions and atmospheric conditions
54
55 latitude0  = 37.431430;
56 longitude0 = -5.859596;
57 LatLong0   = [latitude0,longitude0];
58
59 % ================================ %
60
61 % Controlled Wind Gust
62 gust_start_time = 20;        % (s)
63 gust_length    = [120 120 80]; % (m)
64 gust_amplitude = [3.5 3.5 3.0];% (m/s) Body Axis
65
66 %  Click the check-boxes in the Simulink Block
67 %  "UAV/Enviroment Model/Wind Models/Discrete Wind Gust Model"
68 %  to activated the gust.
69
70 % ================================ %
71
72 % Controlled Uniform Wind around space
73 Vw_unif        = [0 0 0]';    % (m/s) NED Axis
74
75 % ================================ %
76 %% Propulsion
77 maxRPM = 18e3;     % For visualization
78
79 % V3 PROPULSION
80 C_prop = 1;
81 k_motor = 80;
82
83 % V1 PROPULSION
84 Sprop  = 0.196349; % (m2)
85 i0     = 1.5;      % (A)
86 Vmax   = 44.4;     % (V)
87 R      = 0.042;    % (ohm)
88 Kv     = 0.0659;   % (V*s/rad)
89 KQ     = 0.0659;   % (N*m)
90
```

```
%% Aerodynamics
CL_cruise      = 0.38797;
alpha_cruise   = 0.0282;

% Stability Axis Aerodynamics (JALEO-1)
k1          = 0.0537;    % d(CD)/d(CL2)
k2          = -0.0241;   % d(CD)/d(CL)
CDp         = 0.0234;    % parasite drag coef.

CD0         = 0.0193;    % linear parasite drag coef.
CD_alpha    = 0.0987;    % d(CD)/d(alpha)

% The polar formula used: CD = CDp + k1*CL2 + k2*CL
% These values of correspond to the JALEO 1 aerodynamics.

CDq         = 0;       % damping drag due to q rate
CD_deltae   = 0.0135;  % elevator drag derivative

% ================================== %

CL0         = 0.23;    % zero-AOA lift coef.
CL_alpha    = 5.61;    % lift AOA derivative
CLq         = 7.95;    % lift due to q rate

CL_deltae   = 0.13;    % elevator lift derivative

Cm0         = 0.0135;  % zero-AOA moment m coef.
Cm_alpha    = -2.74;   % moment m-AOA derivative

% ========================================================================= %
% Airflow (alpha-beta) Dependent Derivatives
Cy_beta     = -0.83;
Cl_beta     = -0.13;
Cn_beta     = 0.073;

Cy0         = 0;    % zero for symmetric AC
Cl0         = 0;    % zero for symmetric AC
Cn0         = 0;    % zero for symmetric AC

% ========================================================================= %
% Actuator Increments Derivatives
%  da = aleron incremental deflection
    Cy_da       = 0.075;
    Cl_da       = 0.17;
    Cn_da       = -0.011;

%  de = elevator incremental deflection
    % Cx_deltae = f(CL_deltae,CD_deltae,alpha)
    % Cz_deltae = f(CL_deltae,CD_deltae,alpha)
    Cm_de       = -0.99;

%  dr = rudder incremental deflection
    Cx_dr   = 0;
    Cy_dr   = 0.19;
    Cl_dr   = 0.0024;
    Cn_dr   = -0.069;

%  df = flaps incremental deflection
    Cx_df   = 0;
    Cz_df   = 0;
    Cm_df   = 0;

```

```
153 % ==================================================================== %
154 % Damping Derivatives
155 Cy_p       = 0;
156 Cl_p       = -0.51;
157 Cn_p       = -0.069;
158
159 % Cx_q = f(CL_deltae,CD_deltae,alpha)
160 % Cz_q = f(CL_deltae,CD_deltae,alpha)
161 Cm_q    = -38.21;
162
163 Cy_r    = 0;
164 Cl_r    = 0.25;
165 Cn_r    = -0.095;
```

A.2 Script de configuración del modelo de validación V2 - config_model_validation.m

```
1  %% CONFIGURATION SCRIPT - VALIDATION V2
2  %
3  %  This script sets the default the parameters of the Simulink Simulators.
4  %  These parameters should be ajusted in order to simulate the
5  %  identified UAV.
6  %
7  %  All variables most be expressed in SI UNITS.
8  %
9  %% Load Propulsion Data
10 run('LOOKUP_TABLES.m');
11 load ../fittedCoefficients.mat;
12 ST = 0.01;            % SAMPLE TIME
13 %% Geometry and mass properties
14
15 % ================================ %
16
17 b          = 2.9;     % wing span
18 S          = 0.55;    % wing surface
19 c_bar      = 0.19;    % mean chord
20 delta_max  = 0.35;    % max deflection of the control surfaces
21
22 % ================================ %
23
24 mass    = 11;          % Mass of the AC
25 Jx      = 0.8244;
26 Jy      = 1.135;
27 Jz      = 1.759;
28 Jxz     = 0.120;
29 J       = [Jx,   0,     -Jxz;...
30            0,    Jy,       0;...
31            -Jxz,  0,     Jz];    % Inertia Tensor
32 Jxy     = 0;
33 % ================================ %
34
35 % Computation of gamma inertial variables
36 gamma   = Jx*Jz-Jxz^2;
37 gamma1  = Jxz*(Jx - Jy + Jz)/(gamma);
38 gamma2  = (Jz*(Jz - Jy)+Jxz^2)/gamma;
39 gamma3  = Jz/gamma;
40 gamma4  = Jxz/gamma;
41 gamma5  = (Jz-Jx)/Jy;
42 gamma6  = Jxz/Jy;
43 gamma7  = (Jx*(Jx - Jy) + Jxz^2)/gamma;
```

```
gamma8 = Jx/gamma;

% ================================ %
%% Simulation initial conditions and atmospheric conditions

latitude0  = 37.431430;
longitude0 = -5.859596;
LatLong0   = [latitude0,longitude0];

% ================================ %

% Controlled Wind Gust
gust_start_time = 20;        % (s)
gust_length    = [120 120 80]; % (m)
gust_amplitude = [3.5 3.5 3.0];% (m/s) Body Axis

%  Click the check-boxes in the Simulink Block
%  "UAV/Enviroment Model/Wind Models/Discrete Wind Gust Model"
%  to activated the gust.

% ================================ %

% Controlled Uniform Wind around space
Vw_unif        = [0 0 0]';    % (m/s) NED Axis

% ================================ %
%% Propulsion
maxRPM = 11e3;

% V3 PROPULSION
C_prop = 1;
k_motor = 80;

% V1 PROPULSION
Sprop  = 0.2027;
i0     = 1.5;     % (A)
Vmax   = 44.4;    % (V)
R      = 0.042;   % (ohm)
Kv     = 0.0659;  % (V*s/rad)
KQ     = 0.0659;  % (N*m)
%% Aerodynamics

% forces
CDp        = xfit(1);
k2         = xfit(2);
k1         = xfit(3);
CDq        = xfit(4);
CD_deltae  = xfit(5);

CL0        = xfit(6);
CL_alpha   = xfit(7);
CLq        = xfit(8);
CL_deltae  = xfit(9);

Cy_beta = xfit(10);
Cy_p    = xfit(11);
Cy_r    = xfit(12);
Cy_da   = xfit(13);
Cy_dr   = xfit(14);

% moments
Cl_beta = xfit(15);
```

```
Cl_p    = xfit(16);
Cl_r    = xfit(17);
Cl_dr   = xfit(18);
Cl_da   = xfit(19);

Cm0        = xfit(20);
Cm_alpha   = xfit(21);
Cm_q       = xfit(22);
Cm_de      = xfit(23);

Cn_beta = xfit(24);
Cn_p    = xfit(25);
Cn_r    = xfit(26);
Cn_dr   = xfit(27);
Cn_da   = xfit(28);
```

A.3 Script de configuración del modelo de validación V3 - config_model_validation.m

```
%% CONFIGURATION SCRIPT - VALIDATION V3
%
%  This script sets the default the parameters of the Simulink Simulators.
%  These parameters should be ajusted in order to simulate the
%  identified UAV.
%
%  All variables most be expressed in SI UNITS.
%
%% Load Propulsion Data
run('LOOKUP_TABLES.m');
load ../fittedCoefficients.mat;
ST = 0.001;            % SAMPLE TIME
%% Geometry and mass properties

% ================================ %

b          = 2.9;     % wing span
S          = 0.55;    % wing surface
c_bar      = 0.19;    % mean chord
delta_max  = 0.35;    % max deflection of the control surfaces

% ================================ %

mass    = 11;          % Mass of the AC
Jx      = 0.8244;
Jy      = 1.135;
Jz      = 1.759;
Jxz     = 0.120;
J       = [Jx,   0,     -Jxz;...
           0,    Jy,      0;...
           -Jxz,  0,     Jz];    % Inertia Tensor
Jxy     = 0;
% ================================ %

% Computation of gamma inertial variables
gamma  = Jx*Jz-Jxz^2;
gamma1 = Jxz*(Jx - Jy + Jz)/(gamma);
gamma2 = (Jz*(Jz - Jy)+Jxz^2)/gamma;
gamma3 = Jz/gamma;
gamma4 = Jxz/gamma;
gamma5 = (Jz-Jx)/Jy;
```

```
gamma6 = Jxz/Jy;
gamma7 = (Jx*(Jx - Jy) + Jxz^2)/gamma;
gamma8 = Jx/gamma;

% ================================ %
%% Simulation initial conditions and atmospheric conditions

latitude0  = 37.431430;
longitude0 = -5.859596;
LatLong0   = [latitude0,longitude0];

% ================================ %

% Controlled Wind Gust
gust_start_time = 20;        % (s)
gust_length    = [120 120 80]; % (m)
gust_amplitude = [3.5 3.5 3.0];% (m/s) Body Axis

%  Click the check-boxes in the Simulink Block
%  "UAV/Enviroment Model/Wind Models/Discrete Wind Gust Model"
%  to activated the gust.

% ================================ %

% Controlled Uniform Wind around space
Vw_unif        = [0 0 0]';    % (m/s) NED Axis

% ================================ %
%% Propulsion
maxRPM = 11e3;

% V3 PROPULSION
C_prop = 1;
k_motor = 80;

% V1 PROPULSION
Sprop  = 0.2027;
i0     = 1.5;     % (A)
Vmax   = 44.4;    % (V)
R      = 0.042;   % (ohm)
Kv     = 0.0659;  % (V*s/rad)
KQ     = 0.0659;  % (N*m)
%% Aerodynamics

% forces
CDp        = xfit(1);
CD_alpha   = xfit(2);
CDq        = xfit(3);
CD_deltae  = xfit(4);

CL0        = xfit(5);
CL_alpha   = xfit(6);
CLq        = xfit(7);
CL_deltae  = xfit(8);

Cy_beta = xfit(9);
Cy_p    = xfit(10);
Cy_r    = xfit(11);
Cy_da   = xfit(12);
Cy_dr   = xfit(13);

% moments
```

```
Cl_beta = xfit(14);
Cl_p    = xfit(15);
Cl_r    = xfit(16);
Cl_dr   = xfit(17);
Cl_da   = xfit(18);

Cm0        = xfit(19);
Cm_alpha   = xfit(20);
Cm_q       = xfit(21);
Cm_de      = xfit(22);

Cn_beta = xfit(23);
Cn_p    = xfit(24);
Cn_r    = xfit(25);
Cn_dr   = xfit(26);
Cn_da   = xfit(27);
```

Cabe destacar que los últimos 2 scripts, a pesar de poseer el mismo y cumplir la misma función, no son confundidos debido a que se encuentran en proyectos diferentes.

A.4 Análisis de datos de APC para el modelado de la hélice del UAV

A.4.1 Conversión de unidades imperiales a sistema internacional de la matriz de datos de la hélice APC - siUnits.m

```
function matrizSI = siUnits(matriz)
%% matrizSI: TRANSFORMA UNA MATRIZ DE DATOS EN SISTEMA IMPERIAL A SISTEMA INTERNACIONAL
    matrizSI      =   matriz;
    matrizSI(:,1) =   matriz(:,1).*0.44704; % Velocidad V
    matrizSI(:,6) =   matriz(:,6).*745.7;   % Potencia propulsiva P
    matrizSI(:,7) =   matriz(:,7).*0.11;    % Torque
    matrizSI(:,8) =   matriz(:,8).*4.44822; % Empuje
end
```

A.4.2 Filtrado de datos de la hélice en función de las RPM de operación - extractRPM.m

```
function data = extractRPM(helice,krpm)
%% extractRPM: extrae los datos de una matriz "helice" para un cierto régimen de RPM.
%
% helice: matriz de los datos de la hélice de 8 columnas
% krpm: rpm de la hélice en miles
    data = helice((30*krpm-29):30*(krpm),:);
end
```

A.4.3 Creación de las matrices de velocidad, RPM/1000, empuje y par para el modelado de la hélice mediante Look-up Tables - LOOKUP_TABLES.m

```
%clear, close all
format short
load("data_PER310x6E_arrays.mat") % Se carga un ejemplo de array tras la importación

PER310x6E=siUnits(PER310x6E);

N = 1:18;  % 1e3 RPM vector
V = 0:1:50; % airspeed vector (m/s)

```

```
T = zeros(length(N)+1,length(V)); % Thrust lookup matrix
                                  % +1 row for N=0
M = T;                            % Moment lookup matrix

for i=1:length(N) % kRPM
    array = extractRPM(PER310x6E,i);
    V_pts = array(:,1);
    M_pts = array(:,7);
    T_pts = array(:,8);

    pt    = polyfit(V_pts,T_pts,2);
    T_fun = @(v) pt(1).*v.^2+pt(2).*v+pt(3);

    pm    = polyfit(V_pts,M_pts,2);
    M_fun = @(v) pm(1).*v.^2+pm(2).*v+pm(3);

    T(i+1,:) = T_fun(V); % +1 row bc i=1->T=0
    M(i+1,:) = M_fun(V); % +1 row bc i=1->M=0
end

N = [0,N]; % adds 0 RPM
```

A.5 Análisis de modelos linealizados - LINEAR_ANALYSIS.m

```
%% LINEAR ANALYSIS OF THE UAV
% The objective of this script is to analyze the flight modes of a fixed
% wing UAV in a specific operating point given by its linealized model
% matrices A, B, C and D.
%
% CHANGE THE NAME OF THE STATE SPACE OBJECT (ss) IF NEEDED. By default the
% ss object has the name "linsys1".
%
% This analysis its necessary in order to develop a Stability Augmentation
% System (SAS) for the Autopilot.
%
%% Reorganize the state vector and matrices in the correct order
Atemp     = linsys1.A;
A         = zeros(12,12);
Btemp     = linsys1.B;
B         = zeros(12,5);
tempState = [10 11 12 1 2 3 6 7 8 4 5 6];
% state   = [1 2 3 4 5 6 7 8 9 10 11 12];

for i = 1:12
    for j = 1:12
        A(tempState(i),tempState(j))= Atemp(i,j);
        B(tempState(i),:)           = Btemp(i,:);
    end
end

%% Extract the longitudinal and lateral dynamic matrices
x_long = [1,3,5,11];
x_lat  = [2,4,6,10];
u_lat  = [1,3];
u_long = [2,4,5];
[A_long,B_long,C_long,D_long,A_lat,B_lat,C_lat,D_lat] = longAndLatMatrices(A,B,x_long,
    u_long,x_lat,u_lat);

%% Analyze the different flight modes
```

```
longDynamicSys = ss(A_long,B_long,C_long,D_long);
figure, pzplot(longDynamicSys), grid on, title('Longitudinal Dynamics Modes');

latDynamicSys = ss(A_lat,B_lat,C_lat,D_lat);
figure, pzplot(latDynamicSys), grid on, title('Lateral Dynamics Modes');

[Inc_X0_long_modes,E_long,Inc_X0_lat_modes,E_lat] = eigDynamics(A_long,A_lat)
figure, plot(real(E_lat),imag(E_lat),'rx','MarkerSize',1,'LineWidth',20)
hold on, plot(real(E_long),imag(E_long),'bx','MarkerSize',1,'LineWidth',20)

% myzeros = zeros(length(E_lat),1);
% quiver(myzeros,myzeros,real(E_lat),imag(E_lat),'r')
%
% myzeros = zeros(length(E_long),1);
% quiver(myzeros,myzeros,real(E_long),imag(E_long),'b')

%% AUXILIARY FUNCTIONS USED
function [A_long,B_long,C_long,D_long,A_lat,B_lat,C_lat,D_lat] = longAndLatMatrices(A,B,
    x_long,u_long,x_lat,u_lat)
% Separates the lateral and longitudinal state matrices, generating 2
% systems of equations with specific A, B, C and D matrices each.

% ---------------------- LATERAL DYNAMICS ----------------------------- %
% x_lat  = [2,4,6,10,12];
num_lat_x = length(x_lat); % number of lateral state var
% u_lat  = [2,3];
num_lat_u = length(u_lat); % number of lateral inputs

A_lat  = zeros(num_lat_x,num_lat_x);
for i=1:num_lat_x
    for j=1:num_lat_x
        A_lat(i,j)=A(x_lat(i),x_lat(j));
    end
end

B_lat=zeros(num_lat_x,num_lat_u);
for i=1:num_lat_x
    for j=1:num_lat_u
        B_lat(i,j)=B(x_lat(i),u_lat(j));
    end
end

C_lat = eye(num_lat_x,num_lat_x);
D_lat = zeros(num_lat_x,num_lat_u);

% ---------------------- LONGITUDINAL DYNAMICS -------------------------- %
% x_long = [1,3,5,9,11];
num_long_x = length(x_long); % number of lateral state var
% u_long = [1,4];
num_long_u = length(u_long); % number of lateral inputs

A_long=zeros(num_long_x,num_long_x);
for i=1:num_long_x
    for j=1:num_long_x
        A_long(i,j)=A(x_long(i),x_long(j));
    end
end

B_long=zeros(num_long_x,num_long_u);
for i=1:num_long_x
    for j=1:num_long_u
        B_long(i,j)=B(x_long(i),u_long(j));
```

```
96      end
97  end
98
99  C_long = eye(num_long_x,num_long_x);
100 D_long = zeros(num_long_x,num_long_u);
101 end
102 function [Inc_X0_long_modes,E_long,Inc_X0_lat_modes,E_lat] = eigDynamics(A_long,A_lat)
103 % Computes de eigenvalues and eigenvectors of the state transition
104 % matrices A_lat and A_long to determine the lateral and longitudinal
105 % flight modes and the initial conditions to excite those specific modes.
106
107 % --------------------- LONGITUDINAL DYNAMICS ------------------------- %
108 [V_long,D_long] = eig(A_long);     % Tranform. and diagonal matrices
109
110 n_long = length(D_long(:,1));      % number of eigenvalues
111 E_long = zeros(n_long,1);
112 for k = 1:n_long
113     E_long(k) = D_long(k,k);       % eigenvalues vector
114 end
115
116 long_oscillatory_modes_positions = oscillatoryModesPositions(E_long);
117 unique_long_modes_positions      = modesPositions(E_long,
        long_oscillatory_modes_positions);
118 E_long=E_long(unique_long_modes_positions);
119
120 % Computes the initial conditions to estimulate unique modes
121 long_variables = [1 3 5 11]; % longitudinal dynamics variables
122 Inc_X0_long_modes = real(V_long(:,unique_long_modes_positions));
123
124 % ------------------------ LATERAL DYNAMICS ---------------------------- %
125 [V_lat,D_lat] = eig(A_lat);
126
127 n_lat  = length(D_lat(:,1));
128 E_lat  = zeros(n_lat,1);
129
130 for k = 1:n_lat
131     E_lat(k) = D_lat(k,k);
132 end
133
134 lat_oscillatory_modes_positions  = oscillatoryModesPositions(E_lat);
135 unique_lat_modes_positions       = modesPositions(E_lat,lat_oscillatory_modes_positions)
        ;
136 E_lat                            = E_lat(unique_lat_modes_positions);
137
138 % Computes the initial conditions to estimulate unique modes
139 lat_variables = [2 4 6 10]; % lateral dynamics variables
140 Inc_X0_lat_modes = real(V_lat(:,unique_lat_modes_positions));
141 end
142
143 function oscillatory_modes_positions = oscillatoryModesPositions(E)
144 % DEVUELVE LA POSICION DE LOS MODOS OSCILATORIOS COMPLEJOS UNICOS
145 % DENTRO DEL VECTOR DE AUTOVALORES. Solo se devuelve la posicion
146 % del primer número con parte imaginaria positiva.
147
148 i      = 1;       % actual mode position number
149 j      = 1;       % oscillatory_modes_position counter
150 oscillatory_modes_positions = zeros(length(E),1);
151 while j<=length(E) && i<=length(E)
152     if ~(imag(E(i)) == 0)
153         oscillatory_modes_positions(j) = i;
154         i = i + 2;                 % +2 to jump over the conjugate pair
155     else
```

```
        i = i + 1;
    end
    j=j+1;
end
% delete zeros in the vector
oscillatory_modes_positions(oscillatory_modes_positions==0) = [];
end

function modes_positions = modesPositions(E,oscillatory_modes_positions)
% DEVUELVE LA POSICION DE LOS MODOS UNICOS DENTRO DEL VECTOR DE
% AUTOVALORES. De los pares complejos conjugados solo se devuelve
% la posicion del primer número con parte imaginaria positiva. Los
% autovalores reales se devuelven sus posiones sin modificacion alguna.

i       = 1;        % actual mode position number
j       = 1;        % [modes_position] counter
modes_positions = zeros(length(E),1);
while j<=length(E) && i<=length(E)
    if (sum(i==oscillatory_modes_positions))
        modes_positions(j) = i;
        i = i + 2;                      % +2 to jump over the conjugate pair
    else
        modes_positions(j) = i;
        i = i + 1;
    end
    j=j+1;
end
modes_positions(modes_positions==0) = []; % clear the vector of zeros
end
```

A.6 Identificación del modelo

A.6.1 Cálculo de las derivadas de las señales medidas por el giroscopio $\dot{\hat{p}}, \dot{\hat{q}}, \dot{\hat{r}}$ - ratesDerivatives.m

```
%% This scripts compute the derivatives (vectors) of the signals p,q and r
% The process consists of the following steps:
%   1. clean_signal       = LPF(signal)
%   2. noisy_derivative   = d(clean_signal)/dt
%   3. signal_derivative  = LPF(noisy_derivative)
% Where "signal" is any vector in the set {p,q,r} and LPF() is a Low Pass
% Filter Function with zero-phase to prevent delays.
%
%
%
% ------------------------------------------------------------------------ %
% --------------------------- q RATE  ---------------------------- %
% ------------------------------------------------------------------------ %
load simulationData
% filter setup
optimal_windowSize_q = optimalFiltering(q_meas,q,tout);
windowSize = optimal_windowSize_q;
bfilt = (1/windowSize)*ones(1,windowSize);
afilt = 1;
q_filt = filtfilt(bfilt,afilt,q_meas);
q = q_filt;

% q_meas VS q = LPF(q_meas) plots
    % figure
    % plot(tout,q_meas,'c','LineWidth',.01)
```

```
    % hold on
    % plot(tout,q,'-.b','LineWidth',3)
    % hold on
    % plot(tout,q_filt,'-.r','LineWidth',3)
    % grid minor
    % title('q rate signal filtering')
    % legend('Measured q','Real q','Filtered q')

% "finite difference" derivative.
% Note it is one element shorter than y and x
qdot = diff(q_filt)./diff(tout);
% this is to assign yd an abscissa midway between two subsequent t
tdot = (tout(2:end)+tout(1:(end-1)))/2;

    % this should be a rough plot of your derivative
    % plot(tdot,qdot,'m','LineWidth',0.01)

windowSize = windowSize + 2;
bfilt = (1/windowSize)*ones(1,windowSize).*1.1;
afilt = 1;
qdot = filtfilt(bfilt,afilt,qdot);
    % hold on
    % plot(tdot,qdot,'-.b','LineWidth',3)
    % ylim([-2 2])
    % xlim([13 19])

qdot = normalizeLengths(qdot);

% ------------------------------------------------------------------------ %
% --------------------------- p RATE  ---------------------------- %
% ------------------------------------------------------------------------ %
%% filter analysis -> FIND THE BEST WINDOW SIZE FOR THE FILTER
optimal_windowSize_p = optimalFiltering(p_meas,p,tout);
% filter setup
windowSize = optimal_windowSize_p;
bfilt = (1/windowSize)*ones(1,windowSize);
afilt = 1;
p_filt = filtfilt(bfilt,afilt,p_meas);
p = p_filt;

% r_meas VS r = LPF(r_meas) plots
    % close all
    % figure
    % plot(tout,p_meas,'c','LineWidth',.01)
    % hold on
    % plot(tout,p,'-.b','LineWidth',3)
    % hold on
    % plot(tout,p_filt,'-.r','LineWidth',3)
    % grid minor
    % title('r rate signal filtering')
    % legend('Measured p','Real p','Filtered p')

% "finite difference" derivative.
% Note it is one element shorter than y and x
pdot = diff(p_filt)./diff(tout);

    % this is to assign yd an abscissa midway between two subsequent t
    % tdot = (tout(2:end)+tout(1:(end-1)))/2;
    % this should be a rough plot of your derivative
    % plot(tdot,pdot,'m','LineWidth',0.01)

windowSize = windowSize +4;
```

```
bfilt = (1/windowSize)*ones(1,windowSize).*1.025;
afilt = 1;
pdot = filtfilt(bfilt,afilt,pdot);
    % hold on
    % plot(tdot,pdot,'-.b','LineWidth',3)
    % ylim([-7 7])
    % xlim([0 10])

pdot = normalizeLengths(pdot);

% ------------------------------------------------------------------------ %
% --------------------------- r RATE  ---------------------------- %
% ------------------------------------------------------------------------ %
%% filter analysis -> FIND THE BEST WINDOW SIZE FOR THE FILTER
optimal_windowSize_r = optimalFiltering(r_meas,r,tout);
windowSize = optimal_windowSize_r;
bfilt = (1/windowSize)*ones(1,windowSize);
afilt = 1;
r_filt = filtfilt(bfilt,afilt,r_meas);
r = r_filt;

% r_meas VS r = LPF(r_meas) plots
    % close all
    % figure
    % plot(tout,r_meas,'c','LineWidth',.01)
    % hold on
    % plot(tout,r,'-.b','LineWidth',3)
    % hold on
    % plot(tout,r_filt,'-.r','LineWidth',3)
    % grid minor
    % title('r rate signal filtering')
    % legend('Measured r','Real r','Filtered r')

% "finite difference" derivative.
% Note it is one element shorter than y and x
rdot = diff(p_filt)./diff(tout);

    % this is to assign yd an abscissa midway between two subsequent t
    % tdot = (tout(2:end)+tout(1:(end-1)))/2;
    % this should be a rough plot of your derivative
    % plot(tdot,rdot,'m','LineWidth',0.01)

windowSize = windowSize +4;
bfilt = (1/windowSize)*ones(1,windowSize).*1.01;
afilt = 1;
rdot = filtfilt(bfilt,afilt,rdot);

    % hold on
    % plot(tdot,rdot,'-.b','LineWidth',3)
    % ylim([-7 7])
    % xlim([0 10])
    % length(rdot)

rdot = normalizeLengths(rdot);

    % length(rdot)
    % hold on
    % tnorm = tout;
    % tnorm(end) = [];
    % tnorm(1)  = [];
    % length(tnorm)
    % plot(tnorm,rdot,'-.g','LineWidth',2.5)
```

```
150
151
152 %% AUXILIARY FUNCTIONS
153 function optimal_windowSize = optimalFiltering(signal_meas,signal,tout)
154 % Filter Analysis -> FINDS THE BEST WINDOW SIZE FOR THE FILTER
155 % Computes the error between the real signal and its noisy version after
156 % filtering, and returns "optimal_windowSize" for the filter such that
157 % the error is minimum.
158
159 means_vector = zeros(10,1);
160 k=1;
161 for i = 1:2:20
162     % filter setup
163     windowSize = i;
164     b = (1/windowSize)*ones(1,windowSize);
165     a = 1;
166     signal_filt = filtfilt(b,a,signal_meas);
167     % error plot
168     error_signal = 100*abs(signal-signal_filt)/max(signal);
169     means_vector(k) = mean(error_signal);
170     k=k+1;
171 end
172
173 % ERROR VS WINDOW SIZE PLOT
174 windowSizes = 1:2:20;
175 signalName = inputname(2);
176
177     % figure, plot(windowSizes,means_vector);
178     %
179     % xlabel('window size')
180     % ylabel("average error between "+signalName+" and filtered "+signalName)
181
182
183 optimal_windowSize_index = find(means_vector==min(means_vector));
184 optimal_windowSize = windowSizes(optimal_windowSize_index);
185 minimum_errorMean = means_vector(optimal_windowSize_index);
186
187 % ERROR(t) and mean(ERROR)(t) PLOT
188 % optimal filter setup
189 windowSize = optimal_windowSize;
190 b = (1/windowSize)*ones(1,windowSize);
191 a = 1;
192 signal_filt = filtfilt(b,a,signal_meas);          % filtering
193 error_signal = 100*abs(signal-signal_filt)/max(signal); % error
194
195     % figure, plot(tout,error_signal);                  % plot error
196     %
197     % mean_error = mean(error_signal)*ones(length(error_signal),1);
198     % mytitle = "error in the estimated "+ signalName +...
199     %          "  - windowSize = "+num2str(optimal_windowSize);
200     % title(mytitle)
201     % hold on, plot(tout,mean_error)                    % plot mean error
202     % ylim([0,100])
203
204 end
205
206 function normalSignal = normalizeLengths(signalDerivative)
207 % Normalize the pdot, qdot and rdot sizes to N-2. N = length(p,q or r)
208 Nminus1 = length(signalDerivative);
209 j=1;
210 normalSignal = zeros(Nminus1-1,1);
211 for l = 1:(Nminus1-1)
```

```
    linspace_vector = linspace( signalDerivative(j+1), signalDerivative(j), 3);
    normalSignal(l) = linspace_vector(2);
    j=j+1;
end

end
```

A.6.2 Función que integra el modelo propulsivo del simulador V2 usado en identificación - propulsiveModel.m

```
function [Thrust,Torque] = propulsiveModel(Va,deltat,parameters)
%% COMPLETE PROPULSIVE MODEL FOR THE V2 SIMULATOR. Computes the thrust and
%  torque of the propulsive system of the UAV based on the aerodynamic
%  velocity "Va" and the throttle "deltat".
%
%  The model parameters are required as inputs. These have to be passed to
%  the function as a structure object with the attributes:
%      - deltat_pts:  interpolation points vector of the throttle signal
%      - RPM_pts:     interpolation points vector of the RPMs
%      - Nmatrix:     interpolation points matrix of the RPM/1000
%      - Vmatrix:     interpolation points matrix of the Va
%      - M:           interpolation points matrix of the torque M
%      - T:           interpolation points matrix of the thrust T
%
%%  MOTOR MODEL
%  RPM_function   = @(deltat)9000.*tanh(0.5.*(9.*deltat-4))+9000;
    deltat_pts     = parameters.deltat_pts;
    RPM_pts        = parameters.RPM_pts;
    RPM = interp1(deltat_pts,RPM_pts,deltat); % RPM motor LOOKUP TABLE

%  plot(deltat_pts,RPM_pts), hold on, fplot(RPM_function), xlim([0 1]),ylim([0 18000])

%%  PROPELLER MODEL
%  load propulsiveParameters.mat propulsiveParameters;
    Nmatrix = parameters.Nmatrix;
    Vmatrix = parameters.Vmatrix;
    M       = parameters.M;
    T       = parameters.T;

%%  OUTPUTS
    Thrust = interp2(Nmatrix,Vmatrix,T,RPM/1000,Va);
    Torque = interp2(Nmatrix,Vmatrix,M,RPM/1000,Va);
end
```

A.6.3 Función que integra el modelo V2 al que se le desea realizar el ajuste - FUN_v2.m

```
function [YDATA] = FUN(x,XDATA)
% FUN: implements de nonlinear function that models
% the aerodynamic forces and moments of the aircraft, as a vectorial
% function of the vector of parameters, X (coeff. vector) and the state
% matrix of the A/C, XDATA (state measurements).

%% parameters of the A/C
c_bar = 0.19;
b = 2.9;
S = 0.55;
mass = 11;
load propulsiveParameters.mat propulsiveParameters

%% x vector translation
```

```
% forces
CDp        = x(1);
k2         = x(2);
k1         = x(3);
CDq        = x(4);
CD_deltae  = x(5);

CL0        = x(6);
CL_alpha   = x(7);
CLq        = x(8);
CL_deltae  = x(9);

Cy_beta = x(10);
Cy_p    = x(11);
Cy_r    = x(12);
Cy_da   = x(13);
Cy_dr   = x(14);

% moments
Cl_beta = x(15);
Cl_p    = x(16);
Cl_r    = x(17);
Cl_dr   = x(18);
Cl_da   = x(19);

Cm0        = x(20);
Cm_alpha   = x(21);
Cm_q       = x(22);
Cm_de      = x(23);

Cn_beta = x(24);
Cn_p    = x(25);
Cn_r    = x(26);
Cn_dr   = x(27);
Cn_da   = x(28);

%% XDATA matrix translation
Va      = XDATA(:,1);
beta    = XDATA(:,2);
alpha   = XDATA(:,3);

p       = XDATA(:,4);
q       = XDATA(:,5);
r       = XDATA(:,6);

rho  = XDATA(:,7);

deltaa = XDATA(:,8);
deltae = XDATA(:,9);
deltar = XDATA(:,10);
deltat = XDATA(:,11);

%% Formulas
% Aerodynamic coeff.
CL = CL0 + CL_alpha.*alpha;
CD = CDp + k2.*CL + k1.*CL.*CL;

% trigonometric functions of AOA
ca = cos(alpha);
sa = sin(alpha);

% Force Coeff.
```

```
Cx = (-CD.*ca+CL.*sa) + (-CDq.*ca+CLq.*sa).*(c_bar.*q.*0.5./Va) + ...
     (-CD_deltae.*ca+CL_deltae.*sa).*deltae;
Cy = Cy_beta.*beta + Cy_p.*p.*0.5.*b./Va + Cy_r.*r.*0.5.*b./Va + ...
     Cy_da.*deltaa + Cy_dr.*deltar;
Cz = (-sa.*CD-ca.*CL) + (-sa.*CDq-ca.*CLq).*(c_bar.*q.*0.5./Va) + ...
     (-sa.*CD_deltae-ca.*CL_deltae).*deltae;

% Auxiliary variables
a0         = rho.*Va.*Va.*S./(2.*mass);
[Thrust,~] = propulsiveModel(Va,deltat,propulsiveParameters);

% Acceletometer model
accX  = a0.*Cx + Thrust/mass;
accY  = a0.*Cy;
accZ  = a0.*Cz;

% Moment coeff.
Cl = beta.*Cl_beta + Cl_p.*p.*b./(2.*Va) + Cl_r.*r.*b./(2.*Va) +...
     Cl_dr.*deltar + Cl_da.*deltaa;
Cm = Cm0 + Cm_alpha.*alpha + Cm_q.*c_bar.*q./(2.*Va) + Cm_de.*deltae;
Cn = beta.*Cn_beta + Cn_p.*p.*b./(2.*Va) + Cn_r.*r.*b./(2.*Va) +...
     Cn_dr.*deltar + Cn_da.*deltaa;

% WE ARE ASSUMING THAT THE A/C IS SYMMETRIC: Cl0 = Cn0 = Cy0 = 0

%% FUN Output
YDATA = [accX,accY,accZ,Cl,Cm,Cn];
end
```

A.6.4 Función que integra el modelo V3 al que se le desea realizar el ajuste - FUN_v3.m

```
function [YDATA] = FUN(x,XDATA)
% FUN: implements de nonlinear function that models
% the aerodynamic forces and moments of the aircraft, as a vectorial
% function of the vector of parameters, X (coeff. vector) and the state
% matrix of the A/C, XDATA (state measurements).

%% parameters of the A/C
c_bar = 0.19;
b = 2.9;
C_prop = 1;
k_motor = 80;
Sprop = 0.2027;
S = 0.55;
mass = 11;

%% x vector translation
% forces
CDp        = x(1);
CD_alpha   = x(2);
CDq        = x(3);
CD_deltae  = x(4);

CL0        = x(5);
CL_alpha   = x(6);
CLq        = x(7);
CL_deltae  = x(8);

Cy_beta = x(9);
Cy_p    = x(10);
```

```
Cy_r    = x(11);
Cy_da   = x(12);
Cy_dr   = x(13);

% moments
Cl_beta = x(14);
Cl_p    = x(15);
Cl_r    = x(16);
Cl_dr   = x(17);
Cl_da   = x(18);

Cm0        = x(19);
Cm_alpha   = x(20);
Cm_q       = x(21);
Cm_de      = x(22);

Cn_beta = x(23);
Cn_p    = x(24);
Cn_r    = x(25);
Cn_dr   = x(26);
Cn_da   = x(27);

%% XDATA matrix translation
Va      = XDATA(:,1);
beta    = XDATA(:,2);
alpha   = XDATA(:,3);

p       = XDATA(:,4);
q       = XDATA(:,5);
r       = XDATA(:,6);

rho  = XDATA(:,7);

deltaa = XDATA(:,8);
deltae = XDATA(:,9);
deltar = XDATA(:,10);
deltat = XDATA(:,11);

%% Formulas
% Aerodynamic coeff.
CL = CL0 + CL_alpha.*alpha;
CD = CDp + CD_alpha.*alpha;

% trigonometric functions of AOA
ca = cos(alpha);
sa = sin(alpha);

% Force Coeff.
Cx = (-CD.*ca+CL.*sa) + (-CDq.*ca+CLq.*sa).*(c_bar.*q.*0.5./Va) + ...
     (-CD_deltae.*ca+CL_deltae.*sa).*deltae;
Cy = Cy_beta.*beta + Cy_p.*p.*0.5.*b./Va + Cy_r.*r.*0.5.*b./Va + ...
     Cy_da.*deltaa + Cy_dr.*deltar;
Cz = (-sa.*CD-ca.*CL) + (-sa.*CDq-ca.*CLq).*(c_bar.*q.*0.5./Va) + ...
     (-sa.*CD_deltae-ca.*CL_deltae).*deltae;

% Auxiliary variables
a0          = rho.*Va.*Va.*S./(2.*mass);
% In the case of the V2 version, uncomment this line
% [Thrust,~] = propulsiveModel(Va,deltat,propulsiveParameters);
% For the V3 SIMPLIFIED VERSION, uncomment this line
Thrust = 0.5.*rho.*Sprop.*C_prop.*((k_motor.*deltat).^2-Va.^2);

```

```
% Acceletometer model
accX  = a0.*Cx + Thrust/mass;
accY  = a0.*Cy;
accZ  = a0.*Cz;

% Moment coeff.
Cl = beta.*Cl_beta + Cl_p.*p.*b./(2.*Va) + Cl_r.*r.*b./(2.*Va) +...
     Cl_dr.*deltar + Cl_da.*deltaa;
Cm = Cm0 + Cm_alpha.*alpha + Cm_q.*c_bar.*q./(2.*Va) + Cm_de.*deltae;
Cn = beta.*Cn_beta + Cn_p.*p.*b./(2.*Va) + Cn_r.*r.*b./(2.*Va) +...
     Cn_dr.*deltar + Cn_da.*deltaa;

% WE ARE ASSUMING THAT THE A/C IS SYMMETRIC: Cl0 = Cn0 = Cy0 = 0

%% FUN Output
YDATA = [accX,accY,accZ,Cl,Cm,Cn];
end
```

A.6.5 Identificación del modelo no lineal V2 (calculo de coeficientes) mediante mínimos cuadrados - NONLINEAR_IDENTIFICATION_v2.m

```
%% Aerodynamic Coeffiecients Identification using lsqcurvefit()
% This script calculates the aerodynamic coefficients (vector "x" 28x1)
% of an airplane model (vectorial function "FUN" 6xN) based on the measured
% state data (matrix "XDATA" MxN) and the accelerometer ouput data
% (matrix "YDATA" 6xN), by using Least Squares Fitting, lsqcurvefit().
%
% ---------------------------------------------------------------------- %
%
% In our case exist 7 state variables & 4 input
% variables to be measured or estimated (XDATA):
%
%      -   Va:           Airspeed       (m/s)
%      -   beta:         Sideslip angles (rad)
%      -   alpha:        AOA            (rad)
%      -   p:            roll rate      (rad/s)
%      -   q:            Pitch rate     (rad/s)
%      -   r:            yaw rate       (rad/s)
%      -   rho:          air density    (kg/m3)
%      ........................................
%      -   deltaa:       Aileron angle  (rad)
%      -   deltae:       Elevator angle (rad)
%      -   deltar:       Rudder angle   (rad)
%      -   deltat:       Thrust command (-)
%
% ---------------------------------------------------------------------- %
% Besides the rates (p,q and r) derivatives, the accelerations measured by
% the accelerometer are also used to build YDATA matrix:
%
%      -   accX:      non-gravitational acceleration in Xb axis (m/s2)
%      -   accY:      non-gravitational acceleration in Yb axis (m/s2)
%      -   accZ:      non-gravitational acceleration in Zb axis (m/s2)
%
% ---------------------------------------------------------------------- %
%
% The FIRST PART of the coefficients vector corresponds with the FORCE
% COEFF. It has 14 components.
%      -   CDp:          Parasite drag
%      -   k2:           d(CD)/d(CL)
%      -   k1:           d(CD)/d(CL^2)
```

```
%       -   CDq:            damping drag due to q rate
%       -   CD_deltae:      d(CD)/d(deltae)
%       ..............................................
%       -   CL0:            zero-AOA lift coeff.
%       -   CL_alpha:       d(CL)/d(AOA)
%       -   CLq:            damping lift due to q rate
%       -   CL_deltae:      d(CL)/d(deltae)
%       ..............................................
%       -   Cy_beta:        d(Cy)/d(beta)
%       -   Cy_p:           d(Cy)/d(p)
%       -   Cy_r:           d(Cy)/d(p)
%       -   Cy_da:          d(Cy)/d(delta)
%       -   Cy_dr:          d(Cy)/d(deltar)
%
% The SECOND PART of the coefficients vector corresponds with the MOMENT
% COEFF. It has 14 components.
%       -   Cl_beta:        roll static stability derivative < 0
%       -   Cl_p:           roll damping derivative
%       -   Cl_r:           d(Cl)/d(r)
%       -   Cl_dr:          cross-control derivative of yaw
%       -   Cl_da:          primary roll control derivative
%       ..............................................
%       -   Cm0:            zero-AOA pitching moment coeff.
%       -   Cm_alpha:       longitudinal static stability derivative
%       -   Cm_q:           pitch damping derivative
%       -   Cm_de:          primary pitch control derivative
%       ..............................................
%       -   Cn_beta:        yaw static stability derivative > 0
%       -   Cn_p:           d(Cn)/d(p)
%       -   Cn_r:           yaw damping derivative
%       -   Cn_dr:          primary yaw control derivative
%       -   Cn_da:          cross-control derivative of yaw
%
% In total, the "x" vector has 14+14=28 coefficients.
%
% ------------------------------------------------------------------------ %

%% Ajustment of vector sizes
% WARNING:
% Do not run this section multiple times (only 1 run)

run ratesDerivatives.m % computes pdot, qdot and rdot,
                       % and filter the gyro signals p,q and r

load simulationData
Va     = myResize(Va);
beta   = myResize(beta);
alpha  = myResize(alpha);
p      = myResize(p);
q      = myResize(q);
r      = myResize(r);
rho    = myResize(rho);
deltaa = myResize(deltaa);
deltae = myResize(deltae);
deltar = myResize(deltar);
deltat = myResize(deltat);

accX   = myResize(accX);
accY   = myResize(accY);
accZ   = myResize(accZ);

%% Build the DATA MATRICES.
```

```
format short

% Measured independent-terms of the moment coeff.
qS = 0.5.*rho.*Va.*Va.*S; % Dynamic pressure * Surface

[~,Torque]=propulsiveModel(Va,deltat,propulsiveParameters);
% Torque = zeros(length(Va),1);
% for i=1:length(Va)
%     [~,B]=propulsiveModel(Va(i),deltat(i),propulsiveParameters);
%     Torque(i) = B;
%     progress=(i*100)/length(Va);
%     disp("Progress: "+num2str(progress))
% end

Cl_meas = (Jx./(qS.*b)).*( pdot +((Jz-Jy)/Jx).*q.*r - Jxz.*(p.*q+rdot)./Jx)...
           - Torque./(qS.*b);
Cm_meas = (Jy./(qS.*c_bar)).*( qdot +((Jx-Jz)/Jy).*p.*r + Jxz.*(p.^2-r.^2)./Jy);
Cn_meas = (Jz./(qS.*b)).*( rdot +((Jy-Jx)/Jz).*q.*p + Jxy.*(r.*q-pdot)./Jz);

lb      = -inf.*ones(28,1);
ub      = inf.*ones(28,1);

ub(2)   = 0; % k2        negative
ub(22)  = 0; % Cm_q      negative
lb(1)   = 0; % CDp       positive
lb(8)   = 0; % CLq       positive
lb(3)   = 0; % k1        positive
lb(5)   = 0; % CD_deltae positive


% lsqcurvefit() setup
x0      = rand(28,1).*ones(28,1);
XDATA   = [Va,beta,alpha,p,q,r,rho,deltaa,deltae,deltar,deltat];
YDATA   = [accX,accY,accZ,Cl_meas,Cm_meas,Cn_meas];

options = optimoptions('lsqcurvefit','TolFun',1e-16);

% minutes = 1000/60;
% Ts      = 0.01;
% Fs      = 1/Ts;

%% nonlinear regression function
% x = lsqcurvefit(@FUN,x0,XDATA,YDATA,lb,ub,options);
xfit = lsqcurvefit(@FUN,x0,XDATA,YDATA,lb,ub,options);
length(xfit)
%% Results
% table of results
xreal = [0.03,-0.0241,0.0537,0,0.0135,0.23,5.61,7.65,0.13,-0.83,0,0,0.075,0.19...
        -0.13, -0.51, 0.25, 0.0024, 0.17, 0.0135, -2.74, -38.21, -0.99, 0.073, -0.069,
            -0.095, -0.069, -0.011]';

% Data cleaning
minValCoef = min(abs(xreal(xreal~=0)));
xfit(abs(xfit)<minValCoef) = 0;    % Round small values to zero


coeffName = {'1. CDp';...
             '2. k2';...
             '3. k1';...
             '4. CDq';...
             '5. CD_deltae';...
             '6. CL0';...
```

```
163             '7. CL_alpha';...
164             '8. CLq';...
165             '9. CL_deltae';...
166             '10. Cy_beta';...
167             '11. Cy_p';...
168             '12. Cy_r';...
169             '13. Cy_da';...
170             '14. Cy_dr';...
171             '15. Cl_beta';...
172             '16. Cl_p';...
173             '17. Cl_r';...
174             '18. Cl_dr';...
175             '19. Cl_da';...
176             '20. Cm0';...
177             '21. Cm_alpha';...
178             '22. Cm_q';...
179             '23. Cm_de';...
180             '24. Cn_beta';...
181             '25. Cn_p';...
182             '26. Cn_r';...
183             '27. Cn_dr';...
184             '28. Cn_da'};
185         xreal_zeros = xreal;
186         xreal_zeros(xreal_zeros==0)=1;
187
188  for i=1:28
189      if (xfit(i)==0 && xreal(i)==0)
190          relErrorPorcentage(i) = 0;
191      else
192          relErrorPorcentage(i) = round(100*abs((xreal_zeros(i)-xfit(i))./xreal_zeros(i))
                 );
193      end
194  end
195
196
197 T = table(xfit,xreal,relErrorPorcentage','RowNames',coeffName);
198 disp(T)
199
200
201
202 % scatter plot
203 coeff = linspace(1,28,28);
204 markerSize = 200;
205 figure, scatter(coeff,xreal,markerSize*ones(28,1),'x'), hold on, scatter(coeff,xfit),
      grid minor
206 legend('REAL COEFF','FITTED COEFF')
207 xlabel('Nro del Coeficiente')
208 ylabel('Valor del Coeficiente')
209
210 % percentage of error plot
211 relErrorPorcentage(relErrorPorcentage>100) = 100;
212 figure, scatter(coeff,relErrorPorcentage,markerSize*ones(28,1),'x'),grid minor
213 title('PERCENTAGE OF ERROR')
214 xlabel('Nro del Coef')
215 ylim([0,100])
216 % save results
217 save fittedCoefficients xfit;
218
219
220 %% Auxiliary functions
221
222 function vecResized = myResize(vec)
```

```
    aux        = vec;
    aux(end)   = []; % delete the last component
    aux(1)     = []; % delete the firts component
    vecResized = aux;
end
```

A.6.6 Identificación del modelo no lineal V3 (calculo de coeficientes) mediante mínimos cuadrados - NONLINEAR_IDENTIFICATION_v3.m

```
%% Aerodynamic Coeffiecients Identification using lsqcurvefit()
% This script calculates the aerodynamic coefficients (vector "x" 27x1)
% of an airplane model (vectorial function "FUN" 6xN) based on the measured
% state data (matrix "XDATA" MxN) and the accelerometer ouput data
% (matrix "YDATA" 6xN), by using Least Squares Fitting, lsqcurvefit().
%
% ------------------------------------------------------------------------ %
%
% In our case exist 7 state variables & 4 input
% variables to be measured or estimated (XDATA):
%
%      -   Va:           Airspeed       (m/s)
%      -   beta:         Sideslip angles (rad)
%      -   alpha:        AOA            (rad)
%      -   p:            roll rate      (rad/s)
%      -   q:            Pitch rate     (rad/s)
%      -   r:            yaw rate       (rad/s)
%      -   rho:          air density    (kg/m3)
%      ........................................
%      -   deltaa:       Aileron angle  (rad)
%      -   deltae:       Elevator angle (rad)
%      -   deltar:       Rudder angle   (rad)
%      -   deltat:       Thrust command (-)
%
% ------------------------------------------------------------------------ %
% Besides the rates (p,q and r) derivatives, the accelerations measured by
% the accelerometer are also used to build YDATA matrix:
%
%      -   accX:      non-gravitational acceleration in Xb axis (m/s2)
%      -   accY:      non-gravitational acceleration in Yb axis (m/s2)
%      -   accZ:      non-gravitational acceleration in Zb axis (m/s2)
%
% ------------------------------------------------------------------------ %
%
% The FIRST PART of the coefficients vector corresponds with the FORCE
% COEFF. It has 13 components.
%      -   CDp:          Parasite drag
%      -   CD_alpha:     d(CD)/d(alpha)
%      -   CDq:          damping drag due to q rate
%      -   CD_deltae:    d(CD)/d(deltae)
%      ..............................................
%      -   CL0:          zero-AOA lift coeff.
%      -   CL_alpha:     d(CL)/d(AOA)
%      -   CLq:          damping lift due to q rate
%      -   CL_deltae:    d(CL)/d(deltae)
%      ..............................................
%      -   Cy_beta:      d(Cy)/d(beta)
%      -   Cy_p:         d(Cy)/d(p)
%      -   Cy_r:         d(Cy)/d(p)
%      -   Cy_da:        d(Cy)/d(delta)
%      -   Cy_dr:        d(Cy)/d(deltar)
```

```
%
% The SECOND PART of the coefficients vector corresponds with the MOMENT
% COEFF. It has 14 components.
%       -   Cl_beta:       roll static stability derivative < 0
%       -   Cl_p:          roll damping derivative
%       -   Cl_r:          d(Cl)/d(r)
%       -   Cl_dr:         cross-control derivative of yaw
%       -   Cl_da:         primary roll control derivative
%       ..............................................
%       -   Cm0:           zero-AOA pitching moment coeff.
%       -   Cm_alpha:      longitudinal static stability derivative
%       -   Cm_q:          pitch damping derivative
%       -   Cm_de:         primary pitch control derivative
%       ..............................................
%       -   Cn_beta:       yaw static stability derivative > 0
%       -   Cn_p:          d(Cn)/d(p)
%       -   Cn_r:          yaw damping derivative
%       -   Cn_dr:         primary yaw control derivative
%       -   Cn_da:         cross-control derivative of yaw
%
% In total, the "x" vector has 13+14=27 coefficients.
%
% ------------------------------------------------------------------------ %

%% Ajustment of vector sizes
% WARNING:
% Do not run this section multiple times (only 1 run)

run ratesDerivatives.m % computes pdot, qdot and rdot,
                       % and filter the gyro signals p,q and r

load simulationData
Va     = myResize(Va);
beta   = myResize(beta);
alpha  = myResize(alpha);
p      = myResize(p);
q      = myResize(q);
r      = myResize(r);
rho    = myResize(rho);
deltaa = myResize(deltaa);
deltae = myResize(deltae);
deltar = myResize(deltar);
deltat = myResize(deltat);

accX   = myResize(accX);
accY   = myResize(accY);
accZ   = myResize(accZ);

%% Build the DATA MATRICES.
format short

% Measured independent-terms of the moment coeff.
qS = 0.5.*rho.*Va.*Va.*S; % Dynamic pressure * Surface

% In the case of the V2 version, uncomment this line
% [~,Torque]=propulsiveModel(Va,deltat,propulsiveParameters);

% For the V3 SIMPLIFIED VERSION, uncomment this line
Torque = 0;

Cl_meas = (Jx./(qS.*b)).*( pdot +((Jz-Jy)/Jx).*q.*r - Jxz.*(p.*q+rdot)./Jx)...
           - Torque./(qS.*b);
```

```
Cm_meas = (Jy./(qS.*c_bar)).*( qdot +((Jx-Jz)/Jy).*p.*r + Jxz.*(p.^2-r.^2)./Jy);
Cn_meas = (Jz./(qS.*b)).*( rdot +((Jy-Jx)/Jz).*q.*p + Jxy.*(r.*q-pdot)./Jz);

lb      = -inf.*ones(27,1);
ub      = inf.*ones(27,1);

% Signs for stable aircrafts
% forces
lb(1) = 0; % CDp      positive
lb(2) = 0; % CD_alpha positive
% moments
ub(14) = 0;    % Cl_beta negative
ub(15) = 0;    % Cl_p negative
lb(16) = 0;    % Cl_r positive
lb(17) = 0;    % Cl_dr positive
lb(18) = 0;    % Cl_dr positive

ub(20) = 0;    % Cm_alpha negative
ub(21) = 0;    % Cm_q    negative
ub(22) = 0;    % Cm_de   negative

lb(23) = 0;    % Cn_beta positive
ub(24) = 0;    % Cn_p negative
ub(25) = 0;    % Cn_r negative
ub(26) = 0;    % Cn_dr negative
ub(27) = 0;    % Cn_da negative


% lsqcurvefit() setup
x0      = rand(27,1).*ones(27,1);
XDATA   = [Va,beta,alpha,p,q,r,rho,deltaa,deltae,deltar,deltat];
YDATA   = [accX,accY,accZ,Cl_meas,Cm_meas,Cn_meas];

options = optimoptions('lsqcurvefit','TolFun',1e-16);

% minutes = 1000/60;
% Ts      = 0.01;
% Fs      = 1/Ts;

%% nonlinear regression function
% x = lsqcurvefit(@FUN,x0,XDATA,YDATA,lb,ub,options);
xfit = lsqcurvefit(@FUN,x0,XDATA,YDATA,lb,ub,options);

%% Results
% table of results
xreal = [0.03,0.3,0,0.0135,0.23,5.61,7.65,0.13,-0.83,0,0,0.075,0.19...
        -0.13, -0.51, 0.25, 0.0024, 0.17, 0.0135, -2.74, -38.21, -0.99, 0.073, -0.069,
            -0.095, -0.069, -0.011]';

% Data cleaning
minValCoef = min(abs(xreal(xreal~=0)));
xfit(abs(xfit)<minValCoef) = 0;    % Round small values to zero


coeffName = {'CDp';...
           'CD_alpha';...
           'CDq';...
           'CD_deltae';...
           'CL0';...
           'CL_alpha';...
           'CLq';...
           'CL_deltae';...
```

```
175             'Cy_beta';...
176             'Cy_p';...
177             'Cy_r';...
178             'Cy_da';...
179             'Cy_dr';...
180             'Cl_beta';...
181             'Cl_p';...
182             'Cl_r';...
183             'Cl_dr';...
184             'Cl_da';...
185             'Cm0';...
186             'Cm_alpha';...
187             'Cm_q';...
188             'Cm_de';...
189             'Cn_beta';...
190             'Cn_p';...
191             'Cn_r';...
192             'Cn_dr';...
193             'Cn_da'};
194         xreal_zeros = xreal;
195         xreal_zeros(xreal_zeros==0)=1;
196
197  for i=1:27
198      if (xfit(i)==0 && xreal(i)==0)
199          relErrorPorcentage(i) = 0;
200      else
201          relErrorPorcentage(i) = round(100*abs((xreal_zeros(i)-xfit(i))./xreal_zeros(i))
                 );
202      end
203  end
204
205
206 T = table(xfit,xreal,relErrorPorcentage','RowNames',coeffName);
207 disp(T)
208
209
210
211 % scatter plot
212 coeff = linspace(1,27,27);
213 markerSize = 200;
214 figure, scatter(coeff,xreal,markerSize*ones(27,1),'x'), hold on, scatter(coeff,xfit),
      grid minor
215 legend('REAL COEFF','FITTED COEFF')
216 xlabel('Nro del Coeficiente')
217 ylabel('Valor del Coeficiente')
218
219 % percentage of error plot
220 relErrorPorcentage(relErrorPorcentage>100) = 100;
221 figure, scatter(coeff,relErrorPorcentage,markerSize*ones(27,1),'x'),grid minor
222 title('PERCENTAGE OF ERROR')
223 xlabel('Nro del Coef')
224 ylim([0,100])
225 % save results
226 save fittedCoefficients xfit;
227
228
229 %% Auxiliary functions
230
231 function vecResized = myResize(vec)
232     aux         = vec;
233     aux(end)    = []; % delete the last component
234     aux(1)      = []; % delete the firts component
```

```
235     vecResized = aux;
236 end
```

Bibliografía

[1] Damián Rivas Rivas. *Mecánica de vuelo 1*, chapter TEMA 1. Ecuaciones del movimiento. Universidad de Sevilla, Escuela Técnica Superior de Ingenieros, 2012.

[2] Randy Beard y Tim McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[3] Francisco Castro Zafra y otros. Informe técnico - jaleo 1. Technical report, VANTUS AeroDesign Team, 2022.

[4] Anderson. *Introduction to flight*. McGraw-Hill, 7th edition, 2012.

[5] Antonio Mendoza y Eduardo Fernández Camacho. Diseño e implementación de algoritmos de control y guiado en vuelo para un cessna-172. Master's thesis, Universidad de Sevilla . Escuela Técnica Superior de Ingeniería, 2018.

[6] MathWorks. Fly the de havilland beaver. https://es.mathworks.com/help/aeroblks/fly-the-dehavilland-beaver.html. Consultado: 05/2023.

[7] Michael Basler y Martin Spott y otros. *The FlightGear Manual*, 2007.

[8] Lawrence E. Hale y Mayuresh Patil y Christopher J Roy. Aerodynamic parameter identification and uncertainty quantification for small unmanned aircraft. *Aerospace Research Central*, 2016.

[9] MathWorks. lsqcurvefit. https://es.mathworks.com/help/optim/ug/lsqcurvefit.html. Consultado: 07/2023.

[10] Steven L. Brunton y J. Nathan Kutz. *Data-Driven Science and Engineering. Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2021.