UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

TRABAJO DE FIN DE GRADO

TÍTULO DEL TFG: Automatización de las interacciones con el entorno de guiado de X-Plane y estimación de los coeficientes de resistencia aerodinámica dentro del simulador

TITULACIÓN: Grado en Ingenieria en Sistemas Aeroespaciales y Sistemas de Telecomunicaciones

AUTOR: Alejandro Riera Quesada

DIRECTORES: Xavier Prats Menéndez
Enric Pastor Llorens

FECHA: 15 de junio de 2021

Título: Automatización de las interacciones con el entorno de guiado de X-Plane y estimación de los coeficientes de resistencia aerodinámica dentro del simulador
Autor: Alejandro Riera Quesada

Directores: Xavier Prats Menéndez
Enric Pastor Llorens

Fecha: 15 de junio de 2021

Resumen

Los simuladores de vuelo representan una herramienta muy útil en la investigación y diseño de aeronaves. Ayudando a predecir el comportamiento de éstas en ciertas situaciones. En este proyecto se analizará el simulador de vuelo X-Plane 11 y el avión Boeing 738.
La primera parte es un desarrollo de los modos guiados de descenso y de como aplicarlos en el avión.
La segunda parte explica como establecer una comunicación con el simulador tanto vía Plugin como vía programa externo. Esto facilita en gran medida la extracción y uso de datos, debido a que uno de los objetivos es automatizar trayectorias. En este caso se ha optado por desarrollar la comunicación mediante programa externo, en concreto vía UDP. Este método es el más indicado para cumplir con el objetivo de este proyecto.
La última parte está dedicada a aproximar los parámetros desconocidos dentro del simulador, mediante la simulación de múltiples trayectorias, usando las herramientas desarrolladas en las partes anteriores. Estos parámetros son el factor de energía compartida (ESF), el coeficiente de resistencia aerodinámica parásita y el coeficiente de resistencia aerodinámica inducido por la sustentación. Para sacar estos valores primero se han diseñado las rutas del avión para poder sacar toda la información posible, después se han extraído los datos de estas rutas y por último, se ha aplicado el método de los mínimos cuadrados. Esto ha resultado en un modelo que se aproxima bastante al observado dentro del simulador.

Title : Automated interaction with X-plane aircraft guidance environment and estimation of drag coefficients within X-plane models

Author: Alejandro Riera Quesada

Advisors: Xavier Prats Menéndez
Enric Pastor Llorens

Date: June 15, 2021

Overview

Flight simulators represent a very useful tool in aircraft research and design, helping to predict the behavior of airplanes in certain situations. In this project, the X-Plane 11 flight simulator and the Boeing 738 aircraft will be analyzed.
The first part is an analysis of the descent modes and a description of how to apply them on the plane.
The second part explains how to establish communication with the simulator either via Plugin or via an external program. This greatly facilitates the extraction and use of data. In this case, it has been decided to develop communication through an external program, specifically via UDP. This method is the most indicated to meet the goal of this project.
The last part is dedicated to approximate the unknown parameters within the simulator, by simulating multiple trajectories, using the tools developed in the previous parts. These parameters are the energy share factor (ESF), the drag coefficient in a zero lift situation, and the lift-induced drag coefficient. To obtain these values, the airplane routes have first been designed to be able to obtain all the possible information, then the data from these routes have been extracted and finally, the least squares method has been applied. This has resulted in a model that is quite close to the one observed within the simulator.

Para los que siempre me acompañan

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

GLOSARIO

L $\rightarrow$ Fuerza de sustentación.

D $\rightarrow$ Fuerza de resistencia aerodinámica.

W $\rightarrow$ Peso de la aeronave.

THR $\rightarrow$ Fuerza de empuje.

ODE $\rightarrow$ Ecuación diferencial ordinaria.

DOF $\rightarrow$ Grados de libertad.

CoG $\rightarrow$ Centro de gravedad.

FPA $\rightarrow$ Ángulo de la trayectoria de vuelo.

aoa $\rightarrow$ Ángulo de ataque de la aeronave.

IAS $\rightarrow$ Velocidad indicada.

CAS $\rightarrow$ Velocidad calibrada.

TAS $\rightarrow$ Velocidad verdadera.

GS $\rightarrow$ Velocidad relativa a la superficie terrestre.

M $\rightarrow$ Mach, velocidad relativa a la velocidad del sonido.

ATM $\rightarrow$ Gestión del tráfico aéreo.

BADA $\rightarrow$ Base de datos de aviones.

APM $\rightarrow$ Modelo de rendimiento de la aeronave.

ASCII $\rightarrow$ Código Estándar Americano para Intercambio de Información.

ISA $\rightarrow$ Atmósfera estándar internacional.

MSL $\rightarrow$ Valor medio a nivel del mar.

k $\rightarrow$ Índice adiabático del aire.

R $\rightarrow$ Constante de gases ideales.

$p_0 \rightarrow$ Presión atmosférica estándar MSL.

$\rho_0 \rightarrow$ Densidad atmosférica estándar MSL.

$g_0 \rightarrow$ Constante de aceleración gravitacional.

a $\rightarrow$ Velocidad del sonido.

m $\rightarrow$ Masa del avión

ROCD $\rightarrow$ Ratio de ascenso/descenso.

ESF $\rightarrow$ Factor de energía compartida.

$C_L \rightarrow$ Coeficiente de sustentación.

S $\rightarrow$ Superficie alar de referencia.

$C_D \rightarrow$ Coeficiente de resistencia aerodinámica.

$C_{D0} \rightarrow$ Coeficiente de resistencia parásita.

$C_{D2} \rightarrow$ Coeficiente de resistencia aerodinámica inducido por la sustentación.

$\gamma \rightarrow$ Ángulo de la trayectoria de vuelo.

FF $\rightarrow$ Flujo de combustible.

VS $\rightarrow$ Velocidad vertical.

CMD $\rightarrow$ Modo comando del piloto automático.

CWS $\rightarrow$ Control Wheel Steering.

LNAV $\rightarrow$ Modo de navegación lateral.

VNAV $\rightarrow$ Modo de navegación vertical.

ADC $\rightarrow$ Ordenador de datos del aire.

ILS $\rightarrow$ Sistema de aterrizaje instrumental.

VOR $\rightarrow$ VHF de rango omnidireccional.

LOC $\rightarrow$ Localizador.

SDK $\rightarrow$ Kit de desarrollo del sistema.

DLL $\rightarrow$ Libreria de link dinámico.

XPLM $\rightarrow$ Gestor de Plugin de X-Plane.

TCP $\rightarrow$ Protocolo de control de transmisión.

UDP $\rightarrow$ Protocolo de Datagrama de Usuario.

ACK $\rightarrow$ Reconocimiento.

IEEE $\rightarrow$ Instituto de ingenieros eléctricos y electrónicos.

VFR $\rightarrow$ Reglas de vuelo en visual.

IFR $\rightarrow$ Reglas de vuelo en instrumental.

MMO $\rightarrow$ Número de Mach máximo operativo.

VMO $\rightarrow$ Velocidad máxima operativa.

INTRODUCCIÓN

Actualmente, hay muchos tipos de simuladores de vuelo comerciales, que representan una herramienta muy útil para ingenieros e investigadores. éstos se usan para predecir el comportamiento y la manejabilidad de una aeronave en una situación concreta.

X-Plane es un simulador de vuelo muy completo, desarrollado por Laminar Research [1], con un modelo de vuelo propio para aproximarse al desempeño de un avión en el mundo real. Además X-Plane ofrece dos funciones más aparte del simulador en si, airfoil maker y plane maker, si se desea trabajar con un diseño propio.

El modelo de vuelo propio de X-Plane presenta dificultades a la hora de trabajar con este simulador, pues no se conoce el valor de algunas constantes como los coeficientes de resistencia parásita y el factor de coeficiente de resistencia aerodinámica inducido por la sustentación, los que son básicos para determinar las prestaciones de la aeronave. Además, en el aspecto operacional falta el valor del factor de transformación de energía, esta constante es necesaria para uno de los modos de descenso controlado que se detallará más adelante.

Este proyecto tiene como objetivo realizar una investigación del simulador de vuelo X-Plane versión 11 y crear fundamentos para automatizar trayectorias de vuelo. Para conseguirlo, se plantean los siguientes objetivos:

- Caracterizar los modos de guiado en el avión B737.
- Elaborar una forma de comunicación para modificar la trayectoria desde un programa exterior.
- Identificación del modelo de actuación en vuelo del X-Plane según el modelo BADA3,

La base de datos de aeronaves (BADA), ofrece modelos de actuación de aeronaves. BADA se basa en un enfoque cinético para el modelado de las prestaciones de la aeronave. El uso de BADA se centra en simulación y predicción de trayectorias en la investigación y el desarrollo de la gestión del tráfico aéreo y la planificación estratégica en las operaciones ATM terrestres [2].

La estructura de este proyecto, constará de tres capítulos bien diferenciados:

- Capítulo 1: Conceptos básicos para entender la parte teórica del proyecto.
- Capítulo 2: Formas de comunicación con el simulador y como funcionan. También se explicará el modo en el que se ha realizado la comunicación en el proyecto.
- Capítulo 3: Resultados de las simulaciones y las conclusiones de éstos.

CAPÍTULO 1. BACKGROUND TEÓRICO

1.1. Conceptos básicos

1.1.1. Ángulos relacionados con el descenso

Este trabajo se centrará en el estudio de trayectorias descendentes, en la figura 1.1 se muestran los ángulos más relevantes, principalmente el ángulo de la trayectoria de vuelo (o FPA por sus siglas en inglés, ”flight path angle”), el cabeceo y el ángulo de ataque.

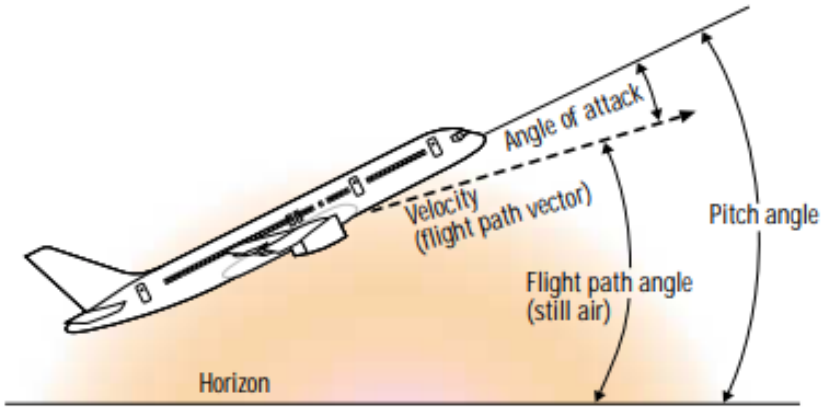


Figura 1.1: Representación del FPA, ángulo de ataque y cabeceo. [3]

Como se observa en la figura 1.1 el ángulo formado entre el suelo y la dirección en el plano vertical a la que apunta la aeronave es el cabeceo, el que esta formado entre el vector velocidad y el suelo es el FPA, y por último la diferencia entre ambos es el ángulo de ataque (aoa).

En los descensos, es especialmente importante el FPA, pues es el ángulo que determina la dirección en el plano vertical desde el punto de vista terrestre. Este ángulo queda representado de la siguiente forma:

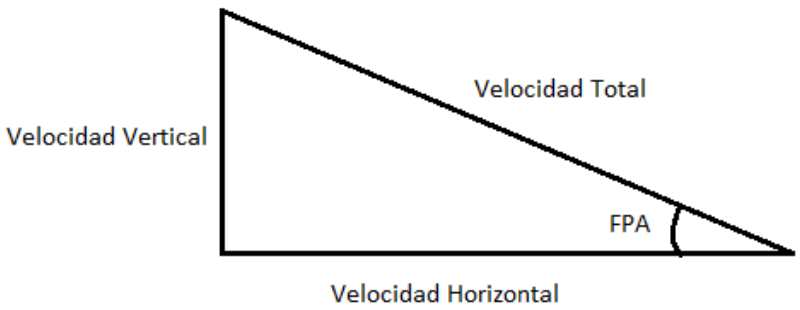


Figura 1.2: Representación del FPA

Con la representación de la figura 1.2, el cálculo del FPA es automático:

$$FPA = \arcsin\left(\frac{VelocidadVertical}{VelocidadTotal}\right) \tag{1.1}$$

Este ángulo de descenso tiene dos posibilidades, que sea el aerodinámico o el terrestre, esto depende de con que velocidades se calcule. Si se usa el vector velocidad respecto al aire se obtiene el FPA aerodinámico, si se usa el vector velocidad respecto a tierra, se obtiene el FPA terrestre.

1.1.2. Velocidades

En aviación, se describen diferentes velocidades en un avión:

- Velocidad indicada(IAS): La velocidad indicada en el panel instrumental, es la que se obtiene directamente de los objetos de medida y sensores. En esta velocidad se asume el modelo de Atmósfera Estándar Internacional (ISA) y condiciones de flujo no compresible a nivel del mar.
- Velocidad calibrada(CAS): Es la velocidad indicada corregida para los errores instrumentales. La diferencia entre la IAS y la CAS viene dada por el fabricante.
- Velocidad verdadera(TAS): Es el módulo del vector velocidad con respecto a la masa de aire que lo envuelve. Es la velocidad calibrada corregida por altitud, presión y temperatura.
- Velocidad terrestre (GS): Es la componente horizontal de la velocidad del avión relativa a la superficie terrestre. Es la TAS sumándole la velocidad del viento.
- Número de Mach (M): Es el ratio entre la TAS y la velocidad del sonido.

1.1.3. Factor de carga

Por último, el concepto de factor de carga, es la relación entre la sustentación y el peso. Este factor marca la tensión estructural que sufre la aeronave durante el vuelo.

1.2. BADA 3

Las actividades de investigación y desarrollo ATM (gestión del tráfico aéreo) en entornos de simulación, requieren información sobre el performance de los aviones, a menudo sustituyendo el avión real por un modelo aproximado. Este rol lo tiene el APM (Modelo de rendimiento de aeronaves), cuyos objetivos principales son proveer de un modelo de rendimiento del avión realista, preciso y completo [4]:

- Capaz de calcular de forma precisa los aspectos cinéticos y cinemáticos del comportamiento de la aeronave.

- Aplicable para un conjunto amplio de tipos de aeronave, para todas las fases del vuelo.
- Tener requisitos de computación y una complejidad razonable.

De esta idea nacen los modelos BADA, que son capaces de predecir el comportamiento de los aviones con bastante precisión.

El modelo BADA 3 (Base de datos de aviones) es una colección de archivos ASCII que especifican parámetros de rendimiento de operación, parámetros de procedimientos de aerolínea y tablas para 318 tipos de aviones. Esta información está diseñada para usarse en algoritmos de predicción y simulación de trayectorias dentro del dominio de gestión del tráfico aéreo. Todos estos archivos son mantenidos y configurados por EUROCONTROL[5].

BADA 3 establece las siguientes constantes [6] basándose en el modelo de atmósfera estándar a nivel del mar(MSL):

- Temperatura atmosférica estándar en MSL:

$$T_0 = 288{,}15[K]$$

- Presión atmosférica estándar en MSL:

$$p_0 = 101325[Pa]$$

- Densidad atmosférica estándar en MSL:

$$\rho_0 = 1{,}225[\frac{kg}{m^3}]$$

- Índice adiabático del aire:

$$k = 1{,}4$$

- Constante de gases ideales para el aire:

$$R = 287{,}05287[\frac{m^2}{Ks^2}]$$

También establece la aceleración gravitacional como una constante, sin tener en cuenta sus cambios por la diferencia de altura:

$$g_0 = 9{,}80665[\frac{m}{s^2}]$$

Dentro del manual de usuario de BADA 3, las fórmulas físicas más importantes para este proyecto son las siguientes:

- Cálculo de la velocidad del sonido:

$$a = \sqrt{k \cdot R \cdot T} \tag{1.2}$$

- Conversión Mach/TAS

$$V_{TAS} = M \cdot \sqrt{k \cdot R \cdot T} \tag{1.3}$$

Además de estas fórmulas físicas, BADA 3 introduce los siguientes conceptos con sus respectivas fórmulas, para realizar su modelo:

- Modelo de energía total, iguala el ratio de fuerzas ejercidas sobre la aeronave con el ratio de incremento de energía potencial:

$$(THR - D)V_{TAS} = mg_0 \frac{dh}{dt} + mV_{TAS} \frac{dV_{TAS}}{dt} \tag{1.4}$$

- Cálculo del ratio de descenso

$$ROCD = \frac{dh}{dt} = \frac{(T_{HR} - D)V_{TAS}}{mg_0}[1 + (\frac{V_{TAS}}{g_0})(\frac{dV_{TAS}}{dh})]^{-1} \tag{1.5}$$

- Factor de energía compartida (ESF):

$$ESF = (1 + \frac{v}{g}\frac{dv}{dh})^{-1} \tag{1.6}$$

- Coeficiente de lift (C_L):

$$C_L = \frac{2 \cdot m \cdot g_0}{\rho \cdot V_{TAS}^2 \cdot S \cdot \cos\phi} \tag{1.7}$$

- Coeficiente de resistencia aerodinámica (C_D):

$$C_D = C_{D0} + C_{D2} \cdot C_L^2 \tag{1.8}$$

- Resistencia aerodinámica (D):

$$D = \frac{C_D \cdot \rho \cdot V_{TAS}^2 \cdot S}{2} \tag{1.9}$$

1.2.1. Grados de libertad

Para establecer el número de grados de libertad (DOF) de una aeronave, se considera el avión como un cuerpo rígido. Se observa que son necesarias seis variables para ubicar la aeronave en el espacio. Por lo tanto, el movimiento de un avión se puede describir mediante un modelo de 6 grados de libertad, el cual consiste en un sistema de ecuaciones diferenciales ordinarias (ODE)[7] que describen las tres translaciones, arriba/abajo, izquierda/derecha y delante/atrás, descritas por cada uno de los tres ejes X, Y y Z y las tres rotaciones de los aviones, la rotación propia de cada uno de los ejes alabeo, cabeceo y guiñada. Estos grados de libertad, se pueden ver representados en la figura siguiente:

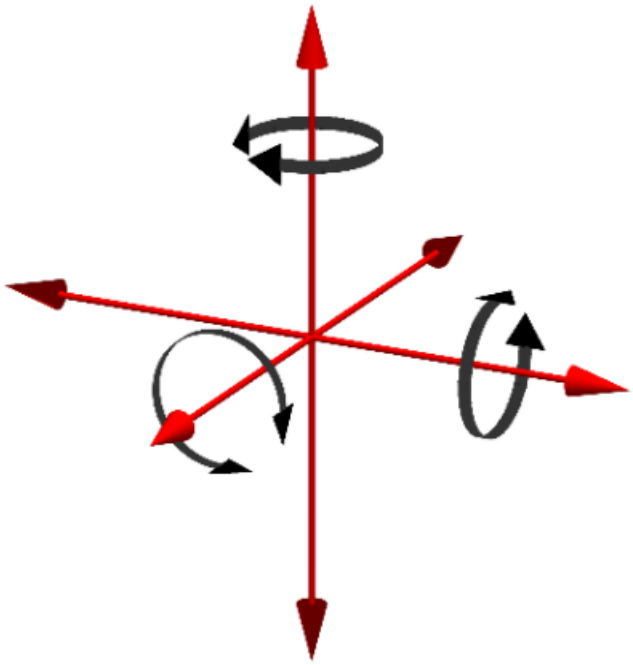


Figura 1.3: Grados de libertad en un cuerpo rígido

El alabeo es la rotación alrededor de una línea recta que va desde la parte delantera del avión hacia la trasera pasando por el centro de gravedad (CoG) del avión, eje X o longitudinal. El ángulo formado entre el eje Y y el plano horizontal se le denomina ángulo de alabeo. El movimiento alrededor del eje Z o vertical del avión corresponde en un cambio en la dirección hacia donde apunta la aeronave en el plano horizontal. Mientras que el movimiento de la aeronave alrededor del eje Y o lateral, el cual va de la parte izquierda a la parte derecha del avión pasando por el centro de gravedad se le denomina cabeceo. El ángulo que forman el eje X con el plano horizontal se le denomina, ángulo de asiento.

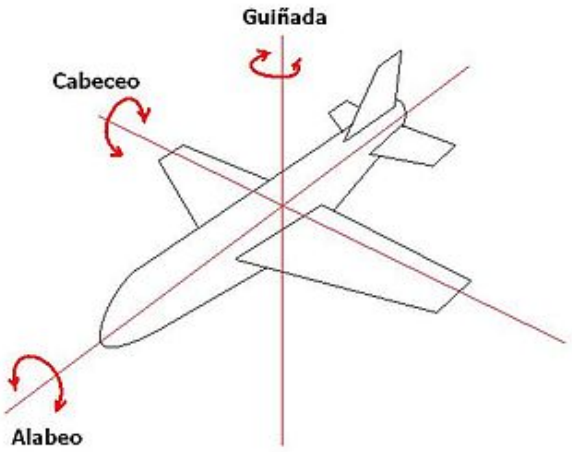


Figura 1.4: Grados de libertad en un avión. [8]

1.3. Modelo de la dinámica de aeronaves

Aunque el modelo de seis grados de libertad resulte ser el mas adecuado para la representación de la aerodinámica del avión, requiere del conocimiento del tensor de inercia de la aeronave y de un modelo aerodinámico complejo.

Las herramientas para la predicción de trayectorias básicas, usan modelos macroscópicos donde solo se consideran las ecuaciones cinemáticas. Un punto medio entre estas dos aproximaciones, se encuentran el llamado ”point-mass model”, el cual es considerado suficientemente preciso para ser usado en sistemas de predicción de trayectorias y para la mayoría de aplicaciones ATM [9]. En este modelo el sistema de movimiento de la aeronave, es reducido a tres grados de libertad, asumiendo que todas las fuerzas se aplican en el centro de gravedad de la aeronave y que el avión se mueve únicamente en el plano vertical (2D)[10].

Este proyecto se centrará en trayectorias de descenso, por lo que las simulaciones se realizaran fijando un heading concreto y se centraran en el perfil vertical de la trayectoria. *Para establecer los modos de guiado, se usará el llamado ”gamma-command model”. Este modelo sale de la simplificación del ”point-mass model”. Asumiendo un equilibrio vertical continuo[11].*

Esta hipótesis implica omitir la dinámica del ángulo de la trayectoria de vuelo (FPA), ya que éste pasa a ser considerado una variable de control.

Aunque se hayan realizado tantas simplificaciones, este modelo es lo suficientemente preciso para representar el funcionamiento de las aeronaves bajo los modos de guiado más típicos.

El movimiento de la aeronave en el plano vertical utilizando el ”gamma-command model”, puede ser descrito por este sistema de ODEs:

$$\frac{dv}{dt} = \dot{v} = \frac{THR - D}{m} - g \cdot \sin\gamma \tag{1.10}$$

$$\frac{dh}{dt} = \dot{h} = v \cdot \sin\gamma \tag{1.11}$$

$$\frac{dm}{dt} = \dot{m} = -FF \tag{1.12}$$

En este sistema de ODEs se establecen tres variables de estado: ”v” hace referencia a la TAS (true airspeed) del avión, ”h” a la altitud geométrica y ”m” a la masa del avión. También se establecen las dos variables de control: ”THR”que se refiere a la fuerza de empuje del motor (thrust) y ”γ”, que se refiere al FPA aerodinámico. Las otras variables son, ”D” el drag aerodinámico, ”g” la aceleración gravitatoria y por último ”FF” que es el flujo de combustible.

Para establecer los modos guiados de descenso y así cerrar el sistema, se necesitará cerrar los dos grados de libertad del sistema. Esto se realizará mediante las variables de control, que podemos ubicar en la palanca de gases (throttle) y el timón de profundidad. Como consecuencia de variar estos elementos, variaran las variables de control [12].

En la práctica, en maniobras de descenso, el empuje y el FPA son consecuencia de especificar el throttle y el timón de profundidad. Para saber el ángulo en el que hay que

posicionar el elevador se deberían cerrar ciertos valores para que el FPA deseado coincida con el FPA realizado. De esta idea, nacen los modos guiados de descenso:

1. Fijar cierto throttle y un FPA terrestre: En este caso se asume que el FPA aerodinámico que es una variable de control, es idéntico al al FPA terrestre que usamos como parámetro de entrada, debido a que se neglige el viento. Aquí la condición para que se cumpla este modo es la siguiente:

$$\gamma = \gamma_g \tag{1.13}$$

2. Fijar cierto throttle y una velocidad vertical (VS): En este caso se asume que la VS se obtiene a partir de la variación de la altura barométrica. Por lo que la VS y la derivada de la altura en función del tiempo se relacionan mediante la temperatura del aire y su desviación con respecto la temperatura establecida por el modelo de atmósfera estándar (ISA), de la siguiente forma:

$$VS = \frac{T - \Delta T}{T} \frac{dh}{dt} \tag{1.14}$$

   Uniendo las ecuaciones 1.14 y 1.11 se obtiene:

$$\gamma = \arcsin\left(\frac{VS}{v} \frac{T}{T - \Delta T}\right) \tag{1.15}$$

3. Fijar un throttle concreto y un ESF: En este caso se ejecuta una maniobra de descenso a un throttle concreto, fijando el valor del factor de energía compartida (ESF). El ESF, hace referencia al ratio de aumento y decremento entre energía potencial y energía cinética. Se define con la expresión 1.6.

   La restricción, en este caso, viene de la unión de las ecuaciones 1.10,1.11 y 1.6 obteniendo:

$$\gamma = \arcsin\left(\left(\frac{T_{HR} - D}{m \cdot g}\right) ESF\right) \tag{1.16}$$

1.4. Caracterización de los modos guiados de descenso en el avión B737-800

En esta sección se mapearan los modos guiados de descenso establecidos en la sección anterior dentro del piloto automático del avión B737 versión 800.

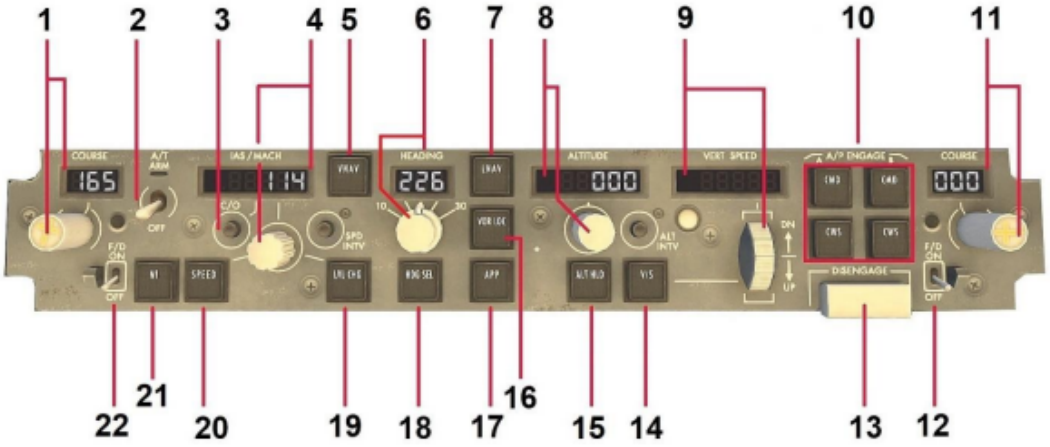


Figura 1.5: Panel del piloto automático

En esta figura se observa el panel del piloto automático del B737-800 en X-Plane. El uso de cada uno de los elementos que constituyen el panel es el mostrado en la tabla 1.1.

Cuadro 1.1: Panel del piloto automático

Número	Función
1	Monitor izquierdo del curso y selector
2	Interruptor del Autothrottle
3	Interruptor para cambiar IAS/MACH
4	Monitor IAS/MACH y selector
5	Botón VNAV(navegación vertical)
6	Monitor del heading y selector
7	Boton LNAV(navegación lateral)
8	Monitor de altitud y selector
9	Monitor de velocidad vertical y selector
10	Botones para encender el piloto automático y seleccionar sus modos de operación
11	Monitor derecho del curso y selector
12	Interruptor para el Flight director del copiloto
13	Interruptor para desconectar el piloto automático
14	Botón para activar el modo VS
15	Botón para activar el modo de mantener la altitud
16	Botón para activar el modo VOR/LOC
17	Botón para entrar en modo Approach
18	Botón para entrar en el modo heading
19	Botón para entrar en el modo level change
20	Botón para activar la velocidad deseada
21	Botón N1 indica al autothrottle avanzar a un límite predefinido de N1
22	Interruptor del piloto para el Flight director

Tras definir las funciones de cada botón, es necesario diferenciar entre 3 términos, CMD, CWS y flight director. CMD enciende el piloto automático, hay dos para añadir redundancia, después de activar esta opción se debería activar los modos deseados. CWS activan el modo ”Control Wheel Steering”, que permite al piloto insertar inputs utilizando la rueda de control, después mantendrá la configuración resultante.

Para utilizar la función de Flight Director, se necesitará insertar un plan de vuelo. Con esta opción activada se mostrará en pantalla el cabeceo y ángulo de alabeo correctos para seguir el plan de vuelo.

LNAV y VNAV hacen que el avión vuele las componentes laterales y verticales del plan de vuelo introducido.

El autothrottle es el modo en el que la palanca de throttle es controlada por el piloto automático para mantenerse en uno de los modos. Por ejemplo si se pretende hacer el vuelo en modo crucero, mantener altitud y velocidad, debe activarse este interruptor, si es un Boeing, la palanca se moverá para ajustarse el thottle que proporciona el thrust necesario; y si es un Airbus esta configuración varía, pues se debería situar la palanca en modo crucero y no se vería movimiento, pues el ADC (ordenador de datos del aire) se encargaría de calcular el thrust necesario y de aplicarlo.

Al seleccionar qué botón corresponde a cada modo de la sección 1.3., al tratarse de

trayectorias descendentes sin movimiento lateral, se asume la función de rumbo siempre activada, excepto en el aterrizaje que se hará mediante el ILS, la función approach donde el avión seguirá un curso. En caso de que haya viento, el rumbo y la derrota no siempre coinciden, pues el rumbo es la dirección a la que mira la aeronave y el curso es la dirección en la que realmente se mueve la aeronave.

Siguiendo los modos establecidos en la sección 1.3.:

1. El primer modo descrito, fijar cierta posición de la palanca de gases y un FPA, es precisamente la senda final de descenso, con el ILS (típicamente -3º). Es decir el botón de approach para interceptar la senda de descenso y el autothrottle puesto, al ser un descenso estará en modo idle ya que se mantendrá en el mínimo. En un Airbus, a diferencia del Boeing es posible establecer en el panel un FPA aerodinámico.

2. El segundo modo descrito, fijar cierta posición de la palanca de gases y una velocidad vertical, se realizará ajustando la velocidad vertical deseada, poner la palanca de throttle en una posición concreta y activar el botón de VS.

3. Por último, fijar un throttle concreto y un ESF. A efectos prácticos no se vuela conociendo el ESF, este modo será objeto de estudio para más adelante, pero para mantener la velocidad constante activaremos el modo de level change, pues es el que permite hacer descensos manteniendo una velocidad concreta.

CAPÍTULO 2. COMUNICACIÓN CON X-PLANE

X-Plane es una excelente herramienta que no solo nos permite simular trayectorias, sinó también personalizar el simulador mediante plugins o acceder a sus valores mediante plugins o desde un programa externo.

2.1. Plugins

Un plugin es un código ejecutable, que corre dentro de X-Plane. Estos ejecutables permiten ampliar las capacidades de este simulador de vuelo o tener acceso a los datos que usa X-Plane para simular el vuelo. X-Plane cuenta con su propio SDK (Kit de desarrollo de software), que admite tanto C como C++.

Un plugin en esencia, es un programa incompleto por si mismo, que se añade a X-Plane. Como se ejecuta dentro de este simulador, puede conseguir cosas que con un programa independiente resultarían imposibles de lograr. Pero, por la misma razón de que son programas incompletos, también tienen limitaciones. A continuación se especifica qué puede hacer un plugin:

- Puede ejecutar código dentro del simulador a una frecuencia concreta de ciclos de vuelo o como respuesta a un evento concreto. Por ejemplo, puede ejecutarse cada ciclo de vuelo un plugin que vuelque datos en un archivo.
- Puede leer datos del simulador. Por ejemplo, un plugin puede leer constantemente datos del velocímetro y enviarlos a un monitor externo.
- Puede modificar ciertos datos dentro del simulador, cambiando sus acciones o su comportamiento. Por ejemplo, se puede variar el ratio en el que el avión gasta combustible o cambiar la posición del propio avión.
- Puede crear interfaces de usuario dentro del propio simulador. Por ejemplo, una interfaz en la que se pueda establecer la altura de la aeronave.
- Puede controlar otros subsistemas del simulador. Por ejemplo, añadir nubes a cierta altura.

Básicamente un Plugin es un DLL (dynamically linked library), un set de funciones compiladas con sus variables asociadas. Esto resulta a la vez una ventaja y un inconveniente, ya que cada vez que se cambie el código se necesitará reiniciar el programa para enlazar la función que efectuaba el DLL en cuestión. Como ventaja, el programa únicamente ejecutara el DLL cuando se requiera esa función.

El sistema de plugins de X-Plane está basado en DLLs. Teniendo como nodo central X-Plane Plugin Manager(XPLM), que es una libreria que gestiona todos los plugins.

XPLM contiene las funciones que el plugin necesita para poder interaccionar con el simulador, por ejemplo, para leer datos o crear interfaces. Esto lo que provoca es que realmente el plugin nunca se comunicará directamente con X-Plane. Ésta comunicación siempre pasará por el XPLM. A efectos prácticos, para el plugin XPLM será el simulador[13].

2.1.1. Cómo crear un Plugin

Un Plugin es un DLL con una serie de funciones, estas funciones se dividen en:

- Funciones necesarias: son las funciones que el XPLM necesita para que el plugin se cargue correctamente. Pueden estar vacías pero tienen que aparecer.
- Funciones registradas: son las funciones que se usan para acceder a las funcionalidades adicionales, por ejemplo al crear una ventana en el simulador, se necesitará una función a la que se pueda llamar cuando el usuario haga clic en la ventana.

Las cinco funciones necesarias son[14]:

- XPluginStart(): Esta función es llamada cuando el plugin se carga por primera vez (al arrancar la simulación), esta función debe dar el nombre del plugin, la firma y una breve descripción para que aparezca en XPLM
- XPluginEnable(): Esta función es llamada cuando el plugin esta activado. Puede estar vacía.
- XPluginDisable(): Esta función es llamada al desactivarse el plugin, generalmente se usa para cerrar conexiones.
- XPluginStop(): Esta función es llamada justo antes de que se pare el plugin, aquí se debería, generalmente limpiar los recursos utilizados.
- XPluginReceiveMessage(): esta función es llamada cuando otro plugin o el propio X-Plane mandan un mensaje a este plugin.

Una cronología típica en el simulador, para un plugin que crea un menú en la pantalla, seria la siguiente:

1. El usuario arranca el simulador.
2. El simulador, via XPLM carga y llama la función *XPluginStart()*.
3. El plugin crea el menú y registra sus funciones.
4. El simulador llama *XPluginEnable()* para notificar que el plugin está en funcionamiento.
5. El usuario clica en el menú así que el simulador mediante el XPLM llama a la función del menú.
6. La función hace algo, por ejemplo, escribe un archivo con algún dato del menú.
7. El usuario sale del simulador.
8. El simulador llama a *XPluginDisable()* y a *XPluginStop()*.
9. El simulador se apaga.

Las APIs disponibles para crear plugins se muestran en el cuadro 2.1.

Cuadro 2.1: APIs disponibles

Nombre de la API	**Función**
XPLMCamera	Permite controlar el ángulo de la cámara de X-Plane
XPLMDataAccess	Permite acceder a los datos dentro del simulador y modificarlos
XPLMDefs	Contiene las definiciones básicas del SDK de X-Plane
XPLMDisplay	Permite dibujar y crear interfaces dentro de X-Plane
XPLMGraphics	Permite cambiar elementos gráficos como las texturas
XPLMInstance	Permite dibujar objetos X-Plane (archivos .obj)
XPLMMap	Permite crear nuevas capas dentro de los mapas de X-Plane
XPLMMenus	Permite crear menús en la barra de menús de X-Plane
XPLMNavigation	Da acceso a la base de datos de navegación dentro de X-Plane
XPLMPlanes	Permite controlar aviones y sus parámetros
XPLMPlugin	Permite gestionar otros plugins
XPLMProcessing	Permite crear funciones dentro del bucle de vuelo.
XPLMScenery	Permite interaccionar con el sistema de escenarios de X-Plane
XPLMUtilities	Proporciona rutas y extensiones de archivos para usar con X-Plane
XPStandardWidgets	Permite crear widgets

Un ejemplo de un plugin sencillo escrito en C++ seria el de la figura 2.1. En este plugin, cada vez que se activa, pone el heading del piloto automático a 0 grados. En el archivo de configuración del plugin se deberían establecer los siguientes puntos como poner la configuración del proyecto en librerías dinámicas, cambiar la extensión objetivo a .xpl y por último en las definiciones hay que poner tanto la versión actual de XPLM como las anteriores. Por lo tanto deberían constar las siguientes líneas:

- `<ConfigurationType>DynamicLibrary</ConfigurationType>`

  Para indicar que efectivamente es un DLL.

- `<TargetExt>.xpl</TargetExt>`

  Para indicar la extensión del Release, X-Plane lee plugins en esta extensión.

- ```
  <AdditionalIncludeDirectories>SDK\CHeaders\XPLM;
  SDK\CHeaders\Widgets;%(AdditionalIncludeDirectories)
  </AdditionalIncludeDirectories>
  ```

  Indicando la localización del SDK.

- ```
  <PreprocessorDefinitions>WINVER=0x0601;_WIN32_WINNT=0x0601;
  _WIN32_WINDOWS=0x0601;WIN32;_DEBUG;_WINDOWS;_USRDLL;
  SIMDATA_EXPORTS;IBM=1;XPLM200=1;XPLM210=1;XPLM300=1;
  XPLM301=1;XPLM302=1;XPLM303=1;
  %(PreprocessorDefinitions)</PreprocessorDefinitions>
  ```

  Indicando todas las versiones del XPLM.

```
#include "XPLMDataAccess.h"

PLUGIN_API int XPluginStart(
    char * outName,
    char * outSig,
    char * outDesc)
{
    strcpy(outName, "HeadingAutopilot");
    strcpy(outSig, "Alejandro Riera");
    strcpy(outDesc, "Changes autopilot heading");

    return 1;
}

PLUGIN_API void XPluginStop(void)
{
}

PLUGIN_API void XPluginDisable(void)
{
}

PLUGIN_API int XPluginEnable(void)
{
    XPLMSetDataf(XPLMFindDataRef("sim/cockpit/autopilot/heading"), 0);
}


PLUGIN_API void XPluginReceiveMessage(
    XPLMPluginID inFromWho,
    int inMessage,
    void* inParam)
{
}
```

Figura 2.1: Ejemplo de plugin

2.2. Programa externo

X-Plane aparte de tener la capacidad de poder establecer una comunicación mediante un plugin, tiene otra opción a través de un programa externo. Esta opción se puede realizar mediante los puertos preasignados de X-Plane, además ofrece una opción para sacar ciertos datos directamente desde el menú de configuración [15], esta opción ha sido descartada pues ofrece unas posibilidades muy limitadas, cuyas únicas salidas están divididas en grupos [16].

2.2.1. Protocolos de transporte

Para este caso, tenemos dos opciones de protocolo de transporte: el Protocolo de control de transmisión (TCP) y el Protocolo de datagramas de usuario (UDP).

TCP es un servicio completo y fiable que permite a varias aplicaciones comunicarse entre sí sin estar conectadas físicamente.

Este protocolo, requiere iniciar una conexión mediante una petición del emisor y una respuesta positiva del receptor (ACK). Tras este paso, el emisor envía la información en orden de bytes, y al acabar cierra la conexión. Cada vez que uno de los paquetes mandados por

el emisor, el receptor manda un ACK conforme ha recibido el paquete.

El protocolo UDP, proporciona un servicio de entrega de datagramas, sin verificar si la conexión es posible entre emisor y receptor. UDP simplemente manda el datagrama donde se le indica sin ninguna noción de si éste llegará o no. Este hecho hace que UDP no sea demasiado fiable, pero a la vez lo hace ideal si se requiere mandar pequeñas cantidades de datos. Pues se ahorra el tiempo de establecer y cerrar la conexión además de la espera de la confirmación de recibo.

A efectos prácticos del X-Plane se establecen las características en la tabla 2.2

Cuadro 2.2: TCP vs UDP

TCP	**UDP**
Protocolo fiable	Es considerado poco fiable
Protocolo pesado	Protocolo ligero
Con control de congestión	Sin control de congestión
Responde recibido cuando recibe	No se sabe si recibe
Requiere de plugin	Es nativo de X-Plane
No requiere ninguna conversión	Requiere conversión a datagrama

Como se entiende en la tabla anterior, aparte de las diferencias ya comentadas para comunicarse con simulador mediante TCP se requiere de un plugin como ExtPlane[17]. El uso de un plugin tiene la ventaja de que facilita la comunicación, pues no se ha de formatear el mensaje a los estándares de la IEEE (instituto de ingenieros eléctricos y electrónicos), además de que la comunicación se realiza mediante las funciones del plugin.

Por otro lado tenemos UDP, el protocolo nativo de X-Plane, con el que si que hay que tener en cuenta un formato específico que se explicará más adelante.

Al comparar ventajas e inconvenientes, se establece que para este proyecto, simulación en tiempo real, se utilizará el protocolo nativo de X-Plane, UDP.

Usando la misma lógica, también se descarta la opción del plugin por la poca capacidad de maniobra que éste ofrece, pues cada cambio de trayectoria, por ejemplo, requeriría repetir el Release y reiniciar el simulador. Lo que mediante programa externo sería inmediato, pues solo se ejecuta otro código, este proceso mediante plugin puede llevar de 3 a 10 minutos dependiendo de la capacidad del ordenador. Aunque para otros aspectos si que se destacaría el plugin frente al programa externo, por ejemplo para ejecutar el simulador con un modelo de vuelo propio, pues al correr dentro de X-Plane se ejecutaría cada vez que fuese necesario.

2.2.2. UDP en X-Plane

La información explicada en este apartado, está extraída del manual proporcionado al hacerse con una copia de X-Plane 11 y en el propio foro de X-Plane[18]. La ruta para acceder a este manual es la siguiente:

```
X-Plane 11\Instructions\X-Plane SPECS from Austin
\Exchanging Data with X-Plane.rtfd\TXT.rtf
```

Este documento es una guía de como trabajar mediante este protocolo con X-Plane.

Para empezar, añadir que con este protocolo es posible conectar el simulador a un ordenador no conectado físicamente, pero para que funcione es necesario dar permiso de acceso creando una norma en el firewall.

El primer punto que hay que tener claro, es que X-Plane tiene una serie de puertos asignados los cuales se pueden ver en el menú de network dentro de la configuración de X-Plane. Generalmente X-Plane recibe por el puerto 49000 y envía por el puerto 49001.

Los datos que usa X-Plane para sus simulaciones se denominan datarefs y están mapeadas variables de todo tipo a las que podemos acceder y, en algunos casos, editar.

Una vez conocemos este dato, ya se conoce qué puertos hay que escuchar y a qué puerto se ha de enviar la información. Falta conocer que tipo de variables están definidas dentro del simulador:

- XCHR esta variable está definida para las variables tipo carácter.
- XINT esta variable está definida para las variables tipo entero, las cuales estarán compuestas por 4 bytes.
- XFLT esta variable está definida para las variables tipo float, las cuales estarán compuestas por 4 bytes también.
- XDOB esta variable está definida por si se necesita doblar la precisión de un float.

Todas estas variables deberán enviarse en hexadecimal, pues presenta ventajas frente al sistema decimal:

- Funciona en base 16 mientras que sistema decimal es en base 10.
- Es el complemento natural a los datos en binario.
- Es un sistema que facilita el proceso de compresión.

Tras definir las variables, se requiere conocer el formato de entrada a X-Plane, este formato es el siguiente:

- Un mensaje de prólogo formado por 5 caracteres, dónde 4 indicaran el tipo de mensaje seguidos de un ’0’.
- La estructura del mensaje que se quiera mandar dependerá del tipo de mensaje.

Los tipos de mensaje que se pueden mandar dentro de esta estructura de datos son:

- ACFN sirve para cargar un avión, su estructura es la siguiente:

```
const xint net_strDIM=150; // tiene que ser suficientemente corto
para ser enviado por la red y lo suficientemente largo
para que quepa el path al .acf
struct acfn_struct
{
```

```
 xint m_acfn_p;
 xchr m_acfn_path_rel[net_strDIM];
 xchr pad[2];
 xint m_acfn_live_ind;
};
```

De esta forma se carga un avión, especificando qué avión cargar como 'p' y el camino hacia el avión como path e indicando en el último parámetro un número entre 1-19 para cargar otros aviones.

- PREL para inicializar el avión en una localización concreta, la estructura en este caso es la siguiente:

```
    const xint idDIM=8; // usado para aeropuertos y radioayudas
struct PREL_struct
{
 init_flt_enum type_start;
 xint p_idx;
 xchr apt_id[idDIM];
 xint apt_rwy_idx;
 xint apt_rwy_dir;
 xdob dob_lat_deg;
 xdob dob_lon_deg;
 xdob dob_ele_mtr;
 xdob dob_psi_tru;
 xdob dob_spd_msc;
};
```

Dependiendo del valor de Type start se inicializará el avión en un cierto modo, en este caso los más interesantes son:

  - 8 para cargar un avión más sin cambiar demasiado la localización, pero empezando en una situación apropiada para la nueva aeronave
  - 10 para empezar en rampa
  - 11 despegue en la pista
  - 12 VFR approach a pista
  - 13 IFR approach a pista

El resto indican la posición del avión si en el aeropuerto, se necesitará saber identificador, pista y dirección o en el aire, estableciendo latitud, longitud, elevación, psi y velocidad.

- CMND en este caso se utiliza para cambiar datos binarios, como por ejemplo tren de aterrizaje arriba o abajo. La lista con todos los comandos esta ubicado en la siguiente dirección:

```
X-Plane 11\Resources\plugins\Commands.txt
```

Aunque es recomendable probarlos antes de poner en código, pues no todos funcionan.

Un ejemplo para bajar el tren de aterrizaje seria:

```
CMND0+sim/flight_controls/landing_gear_down
```

- RREF este tipo es de los más útiles para este proyecto, pues permite recibir de forma periódica los dataref solicitados. El mensaje enviado al puerto 49000 ha de tener la siguiente estructura:

```
    struct dref_struct_in
{
 xint dref_freq;
 xint dref_sender_index;
 xchr dref_string[400];
};
```

En este caso, la frecuencia, hace referencia al número de veces por segundo en el que se quiere recibir los datos, la tasa de refresco. El índice es el número con el que se devolverá el dato, y string es el dataref que se quiere recibir. Alguno de los valores como el thrust está dividido por motor, por lo que se debería especificar, si se quiere motor número 1 se enviará:

```
sim/flightmodel/engine/POINT_thrust[1]
```

Se podrían acumular la petición de hasta 140 datarefs, aunque sería mejor agruparlos en ráfagas de 100.

El mensaje de vuelta es el siguiente:

```
    struct dref_struct_out
{
 xint dref_sender_index ;
 xflt dref_flt_value ;
};
```

Donde el índice es el valor con el que se solicitó el dataref y value, es el valor que corresponde a ese dataref. El mensaje generado por X-Plane sería el siguiente:

```
RREF+0+(dref_struct_out)+(dref_struct_out)+(dref_struct_out)
```

Así que el mensaje, podría ocupar como máximo 1500 bytes. Si se supera esta cantidad X-Plane generará otro paquete con el encabezado RREF. Si se desea dejar de recibir cierto dataref, se debería cambiar la frecuencia a 0.

- DREF para establecer un valor en cierto dataref, hay una lista con todos los datarefs[19]:

```
X-Plane 11\Resources\plugins\DataRefs.txt
```

En esta lista se encuentran todos los datarefs disponibles, especificando si se puede cambiar el valor o no, aunque se recomienda probar por separado, pues no todos los que indica se pueden cambiar.

La estructura de este mensaje es la siguiente:

```
const xint strDIM=500; //suficientemente largo para
que soporte los datarefs
struct dref_struct
{
 xflt var;
 xchr dref_path[strDIM];
};
```

En esta estructura var es el valor que se quiere establecer y dref hace referencia al dataref. El mensaje completo sería el siguiente:

```
DREF0+(4byte value)+dref_path+0
+espacios para completar los 509 bytes del mensaje
```

En este caso, es importante tener en cuenta la longitud total, pues sino este tipo de mensaje no hará nada.

- DATA es otro de los tipos de mensaje, este es algo especial, pues se pueden seleccionar los campos que se quiere recibir en el menú de output de configuración. Los datos recibidos por este tipo de mensajes tienen la siguiente estructura:

```
const xint VALUES_PER_DATA_ITEM=8;

struct data_struct
{
 dout_line_index_t index;
 xflt data[VALUES_PER_DATA_ITEM] ;
};
```

Este mensaje devuelve un vector de valores correspondientes al campo seleccionado. Por ejemplo, speeds, nos devolverá un vector con las velocidades del avión en ese momento. Para ver qué variables hay en cada campo, hay que acceder a la siguiente dirección:

```
www.x-plane.com/kb/data-set-output-table/
```

El índice corresponde al valor asignado por X-Plane a ese campo, éste no se puede cambiar. Este tipo de mensajes son tanto de entrada como de salida. Si se requiere cambiar algún valor en el índice se pondrá el índice correspondiente y se llenará el vector con los valores deseados, si no se requiere cambiar alguno, basta con rellenar el campo con -999.

Estos son los tipos de mensajes más interesantes. Con éstos, ya se puede desarrollar un código para mandar y recibir diferentes datos de distintas formas.

En este caso, se desarrolló un código para recibir y mandar el tipo DATA, pero debido a la falta de flexibilidad y en algunos casos de información que transmite, se decidió que era poco eficiente. Por lo que se procedió a desarrollar dos clases básicas en Python para gestionar la comunicación una para recibir datos, usando los campos RREF y otra, para enviar datos, usando los campos DREF y CMND.

2.2.2.1. Cambiar el valor de un dataref

En los primeros pasos, se recomienda hacer uso del programa Wireshark, para comprobar si realmente se está mandando el mensaje deseado. Para el ejemplo, se pausará la simulación mediante un mensaje CMND y mediante un mensaje DREF.

Una vez se entienda la estructura para el mensaje DREF, la del mensaje CMND es mas simple.

Para pausar la simulación se necesita poner el dataref que multiplica la velocidad de simulación a 0, primero se necesita encontrar el dataref:

```
"sim/time/sim_speed"
```

Para encapsular este mensaje dentro del datagrama, es necesario formatearlo, lo primero es transformar ese 0 en una variable que entienda X-Plane. Dado que es una variable de tipo entero se transformará en un XINT bajo ciertas reglas. Es una variable de cuatro bytes y se tiene que usar el método Little-Endian en la representación hexadecimal del número. En este caso es sencillo porque es un 0:

```
\x00\x00\x00\x00
```

Ahora lo mismo con el campo de dataref, recordar que este mensaje ha de tener un largo de 509 bytes totales, así que se deberá hacer pading, esto se debe realizar en este campo, pues la estructura de este mensaje era

$$DREF0 + (4bytevalue) + dref = 509Bytes$$

Así que se deberá rellenar el campo dref con ceros hasta completar el mensaje. En Python hay una librería muy útil para este caso, que nos permite codificar de forma rápida con la longitud adecuada: struct. Así que las lineas de código para comprimir el mensaje serían las siguientes:

```
value = 0
```

```
cmd = b"DREF\x00"
dref = "sim/time/sim_speed"
message = struct.pack("<5sf500s",cmd,value,dref.encode())
```

En valor asignamos el valor que se requiera, cmd es el tipo de mensaje (recordar que ha de acabar en 0), dref es el campo que se quiere cambiar. La función struct.pack, requiere que se le establezca el formato: ” < ” para Little-endian, 5s por los cinco caracteres(5 bytes), f porque siempre mandaremos floats desde esta configuración (4bytes) y por último los 500 bytes restantes de caracteres.

El paquete que se vería pasar en Wireshark contendría lo siguiente(sin el pading):

```
DREF\x00\x00\x00\x00\x00sim/time/sim_speed\x00\x00\x00
```

Una vez aclarado el funcionamiento del DREF, se analiza el del CMND, que básicamente es lo mismo pero sin longitud establecida, por lo que se pone una, suficientemente larga, para que quepan todos los comandos y así poder generalizar la función. Las líneas de código en este caso son:

```
cmd = b"CMND\x00"
cmnd = b"sim/operation/pause_toggle"
message = struct.pack("<5s50s",cmd,cmnd)
```

En este caso observamos que es bastante más sencillo, se ha establecido una longitud de 50 al campo comando debido a que es suficientemente largo como para dejar siempre un nulo al final.

En wireshark se observaria lo siguiente (sin el pading):

```
CMND\x00sim/operation/pause_toggle\x00
```

Siendo éste el mensaje de longitud mínima requerido para este caso.

2.2.2.2. XPlaneReceiver

Esta clase ha sido creada para gestionar la recepción de datos que se pretende recibir. El algoritmo que funciona dentro de esta clase es relativamente simple y funciona con solamente dos funciones:

- AddDataRef: Esta función tiene dos parámetros de entrada el dataref que se quiere añadir o quitar y la frecuencia. La frecuencia si no se establece, por defecto se pondrá a 1. El funcionamiento de esta función es el siguiente, se guarda una lista de índices en uso con su respectivo dataref, y pide automáticamente con los índices de su lista el dataref que se requiere con la frecuencia que debe (si se pone 0 se eliminará el dataref de la lista).
- GetValues: Esta función, mediante la lista de dataref y sus respectivos índices creará un diccionario con los valores devueltos por X-Plane donde se asignará el nombre del dataref con su correspondiente valor.

Puntualizar que las conexiones se establecen al inicializar la clase, donde se mandan datos desde el puerto por el que se desea que X-Plane los devuelva.

2.2.2.3. XPlaneSender

Esta clase ha sido creada para cambiar valores dentro del simulador de dos formas, o usando funciones predefinidas para los casos más usados, por ejemplo para cambiar la altitud (SetAltitude(altitud)), estando fácilmente accesibles las funcionalidades del piloto automático según las instrucciones de X-Plane [20], o usando las dos funciones más generales:

- SendDREF: funciona especificándole el dataref que se quiere cambiar y el valor que se desea establecer
- SendCMND: enviará el comando que le introduzcas como input.

2.2.2.4. WriteValues

Esta clase ha sido creada para estandarizar los datarefs a guardar en un archivo, y así facilitar el proceso de guardado. Algunos de estos campos no tienen efecto sobre este proyecto, o por ejemplo el viento será marcado a 0 para intentar sacar muestras lo más limpias posibles, por lo que estos valores no serán mostrados en gráficas. Por defecto, en esta clase se guardaran, en el orden en el que aparecen y por columnas, los siguientes datos:

- Fase del vuelo (se puede dejar a 0)
- El día del vuelo
- El tiempo (HH:MM:SS)
- Tiempo de ejecución de la simulación
- Distancia recorrida
- Altura de la aeronave
- V_{TAS}
- Masa de la aeronave
- Heading
- Curso (Track)
- V_{CAS}
- Altitud barométrica
- Mach
- Temperatura
- Presión
- Thrust

- Drag
- C_D
- C_L
- Latitud
- Longitud
- Componente del viento en dirección norte
- Componente del viento en dirección este
- Viento de cola
- Viento lateral
- Ratio del Throttle (entre 0 y 1)
- FPA
- Ratio de Speedbrakes desplegados(entre 0 y 1)
- Ratio del tren de aterrizaje (0 plegado y 1 completamente desplegado)
- Factor de carga
- Configuración de los flaps
- Aceleración($\dot{v}$)
- GS
- FF
- Velocidad vertical

Un ejemplo de las maniobras que se pueden realizar combinando estas dos funciones y extrayendo los datos utilizando esta última función para mostrarlos en la figura 2.2, es el siguiente (el código utilizado se encuentra en el apéndice A), para interceptar el ILS se establece la frecuencia de la pista en cuestión [21].

Fase final de descenso interceptando ILS:

1. Empezar maniobra final de descenso, flaps en una configuración de un grado y desaceleración a 190 nudos.
2. Se intercepta ILS, desaceleración a 170 nudos y flaps a 10º.
3. Cuando la altitud es menor a 1000 pies flaps en configuración de 25º y desaceleración a 150 nudos.
4. Cuando baja de 700 pies, se extienden completamente los flaps (40º).
5. Cuando la altitud es menor a 200 pies, desaceleración a 140 nudos.

6. Cuando se esta a menos de 15 pies, los speedbrakes se despliegan, y se empieza a frenar.

7. Cuando se para el avión, se para la simulación.

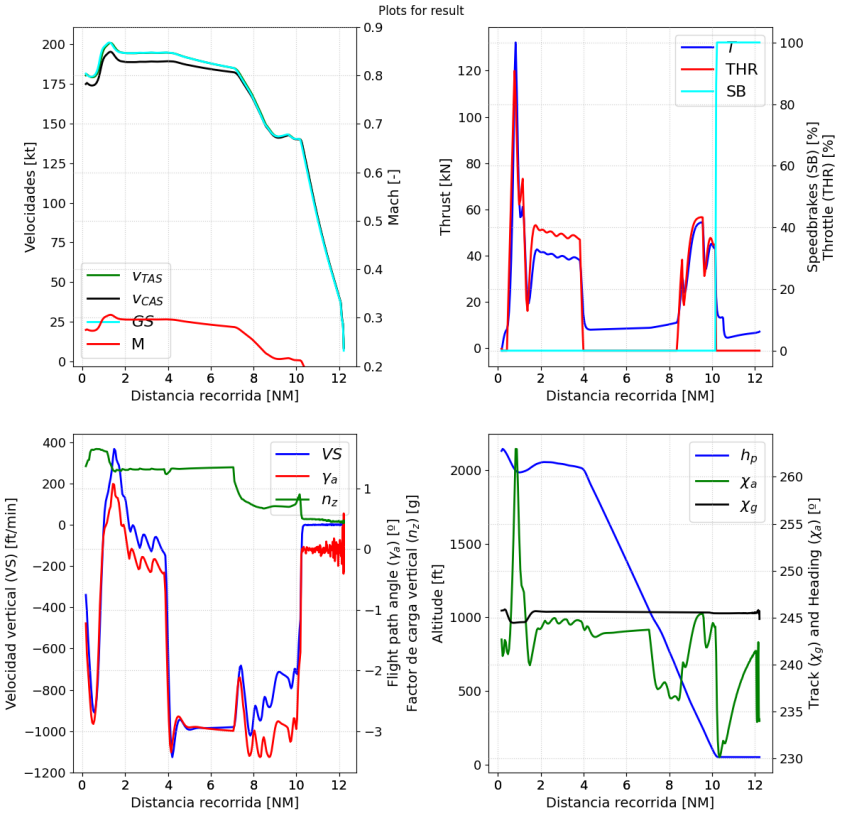


Figura 2.2: Ejemplo de trayectoria de aterrizaje

CAPÍTULO 3. IDENTIFICACIÓN DE UN MODELO BADA 3

Este simulador se diferencia de los otros implementando una variación de la BET (Blade Element Theory). Generalmente los simuladores de vuelo, simulan el comportamiento de la aeronave mediante tablas de búsqueda que contienen la información de las fuerzas aerodinámicas. Esto resulta una limitación en trabajos de diseño de aeronaves o cuando la trayectoria a simular no esta contenida dentro de las tablas.

La idea general del método BET es dividir la longitud de la hélice en un número independiente de secciones. A cada sección se le realiza un balance de fuerzas 2D, sacando el lift y el drag producidos por el empuje y el esfuerzo de torsión. Al mismo tiempo, se aplica un equilibrio de momento axial y angular, el resultado es un conjunto de ecuaciones no lineales que se resuelven de forma iterativa por cada sección. La suma de todos estos valores es el valor que se utiliza para predecir la actuación de las hélices[22].

Este método mejora este tipo de simulaciones modelando las fuerzas y momentos que actúan en el avión, evaluando individualmente las partes que lo constituyen. Este tipo de métodos se usan para calcular las fuerzas aerodinámicas en tiempo real o para crear las tablas de búsqueda con los datos de un nuevo diseño. X-Plane 11, parte de este método para modelar la aerodinámica del avión, aplicándolo sobre la superficie de este.

En este capítulo se hará una identificación del modelo BADA 3 de todos esos datos que no pueden ser extraídos del simulador. Básicamente ESF en el caso del tercer modo establecido en la sección 1.3. y los coeficientes de resistencia aerodinámica.

Para realizar esta identificación del modelo se establece el “Quad-M principle”, el cual se resume en maniobra, medida, modelo y método. Cada uno de estos términos hace referencia a uno de los siguientes pasos [23]:

1. Diseño del experimento.
2. Medición de los datos del experimento.
3. Selección del modelo matemático.
4. Aplicación del método seleccionado.

3.1. ESF en el modo de guiado 3

Para este parámetro se empezará simulando trayectorias de descenso donde se pondrá el throttle en posición de idle y se realizará una desaceleración y un descenso, usando la opción de ”level change”del piloto automático y se calculara el ESF en la trayectoria con la fórmula 1.6.

Hay que tener en cuenta que las derivadas se realizaran restando el valor anterior al valor actual, por lo que para evitar divisiones por 0, si los diferenciales son igual a 0 se cogerá el valor de ESF obtenido en el caso anterior.

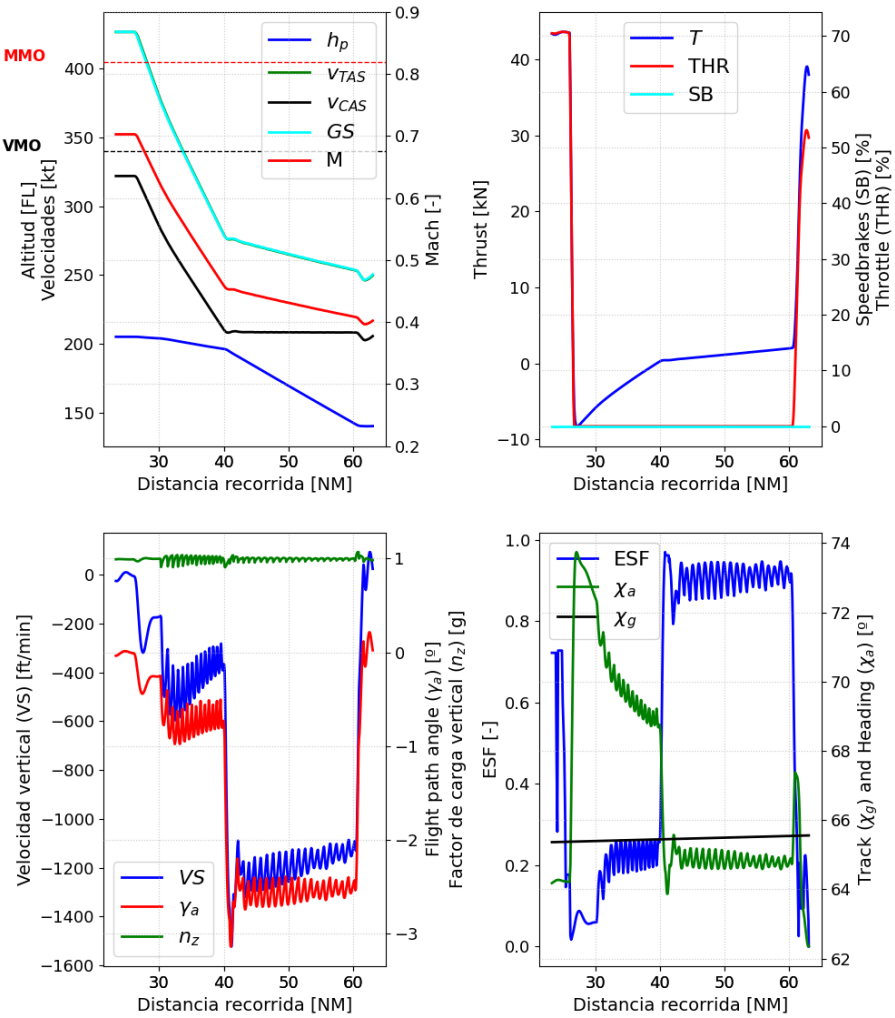


Figura 3.1: Trayectoria de desaceleración y descenso

Dado que el ESF únicamente tiene sentido en los casos en los que se intercambia energía cinética por potencial o a la inversa. En la figura 3.1 se representa los valores obtenidos de la simulación de desaceleración y descenso con la palanca de gases al mínimo. En la misma figura se observa que realmente se cierra el grado de libertad, marcando una cierta ESF:

- En el caso de desaceleración manteniendo la altitud se establece un ESF que tiende a 0.2.
- En el caso de descenso a velocidad constante, con la palanca de gases al mínimo el ESF tiende 0.95.

Este dato permite establecer un valor para cerrar el tercer modo de guiado. Debido a que sino, quedaría sin determinar el comportamiento de la aeronave con un grado de libertad.

3.2. Coeficientes de resistencia aerodinámica

El ala puede ser vista como un convertidor de resistencia aerodinámica a sustentación. Las fuerzas de arrastre y sustentación dependen del área del ala (S), la velocidad y la densidad del aire, el resto de efectos del se describen mediante coeficientes en ambas fuerzas:

$$L = \frac{1}{2}\rho S C_L v^2 \tag{3.1}$$

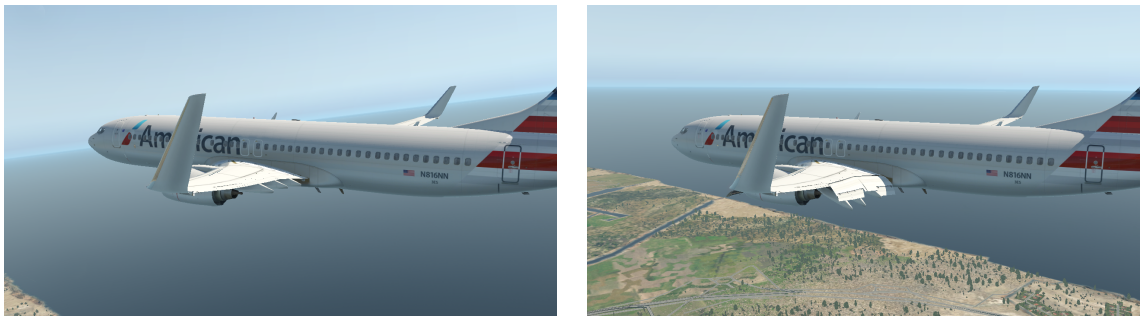
$$D = \frac{1}{2}\rho S C_D v^2 \tag{3.2}$$

Generalmente, el cálculo de estos coeficientes se realiza a partir de la polar de resistencia aerodinámica. Esta relación entre el coeficiente de resistencia aerodinámica y el coeficiente de sustentación es el factor principal en la determinación de las prestaciones de la aeronave[24].

La polar de resistencia aerodinámica se calcula con la ecuación 1.8. Los dos parámetros C_{D0} y C_{D2} son, respectivamente, el coeficiente de resistencia parásita (resistencia aerodinámica en condiciones de sustentación igual a cero) y el factor de coeficiente de resistencia aerodinámica inducido por la sustentación. Estos valores son considerados como constantes para cada configuración aerodinámica de la aeronave.

Cada configuración aerodinámica tiene valores propios de coeficientes de resistencia aerodinámica y cada avión tiene sus propias configuraciones de flaps.

Los flaps y slats son básicamente, dispositivos en forma de panel con bisagras montados en la parte de atrás y de delante del ala respectivamente, que se pueden extender para cambiar la geometría del ala. Al extender estos paneles, se aumenta la superficie efectiva del ala lo que permite mantener una misma sustentación a velocidades mas bajas o también retrasar la velocidad de pérdida.



(a) Ala del avión B737 con flaps contraídos(clean configuration)

(b) Ala del avión B737 con flaps completamente extendidos (40º)

Figura 3.2: Ejemplo de configuraciones de los flaps

Un incremento en el ángulo de los flaps causa un incremento en el coeficiente de sustentación, para un mismo ángulo de ataque, a expensas de obtener más resistencia aerodinámica.

Otro elemento que añade resistencia aerodinámica y cambia la configuración aerodinámica de la aeronave es el tren de aterrizaje, aunque como este elemento no proporciona sustentación, debería únicamente sumar al coeficiente de resistencia aerodinámica parásita.

Para sacar esta polar se seguirá la siguiente metodología:

1. Se simularán dos trayectorias para cada configuración en crucero, una con el peso máximo y la otra con el peso mínimo desde la velocidad máxima hasta la mínima que permite el piloto automático o la perdida.
2. Como resultado de estas simulaciones se extraerán el Cl y el Cd que usa x-Plane en cada punto de la trayectoria. Obteniendo una nube de puntos para cada configuración en una gráfica Cl vs Cd.
3. Esta nube de puntos recordará la forma de una parábola, que se parametrizará usando el método de los mínimos cuadrados.

Todas las configuraciones aerodinámicas posibles en términos de slats y flaps del avión son las representadas en la figura 3.3.

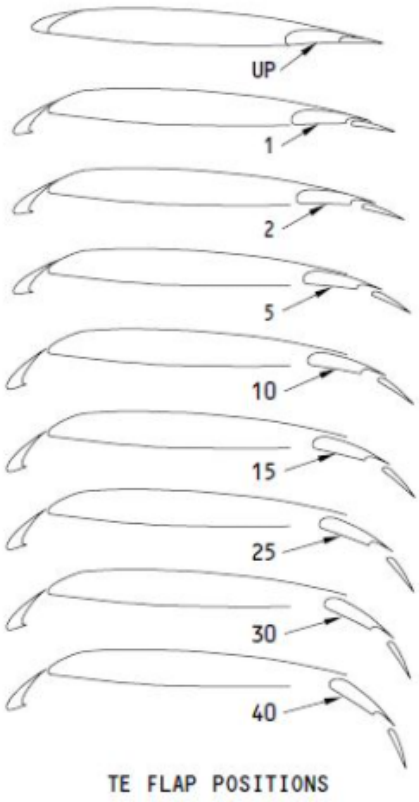


Figura 3.3: Configuraciones posibles de slats y flaps en el avión B737 [25]

3.2.1. Mínimos cuadrados

Para aproximar la dependencia entre el Cl y el Cd, se usará este método. La esencia de este método es que se intenta determinar el polinomio que minimiza la distancia global de todos los puntos a ese polinomio. Este ajuste se realiza haciendo que sea mínima la suma de los cuadrados de las distancias de cada dato Cl, Cd a el valor que tiene el polinomio en ese punto del eje x, esta distancia se le denomina error. Creando una función de coste que será la media de los errores elevados al cuadrado(error cuadrático medio). La

consecuencia de elevar al cuadrado es la de penalizar en mayor medida aquellos puntos más alejados.

Para sacar los valores de los parámetros se deriva la función error al cuadrado y se iguala a cero.

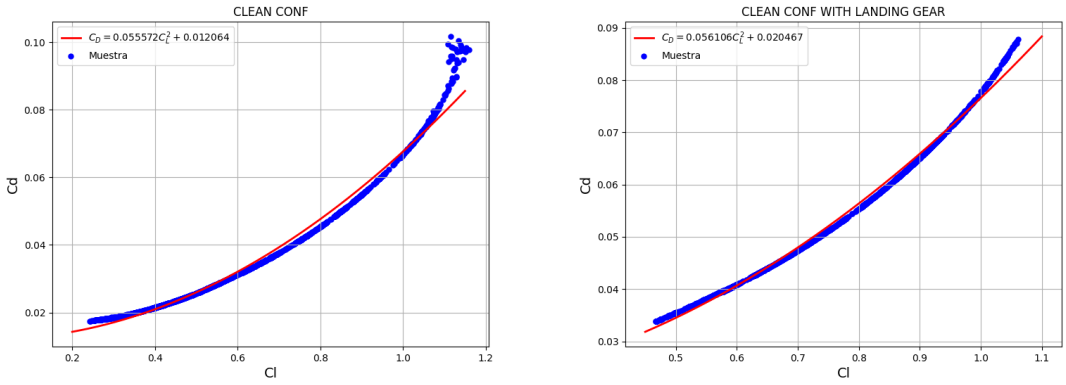
$$ECM = \frac{1}{n}\sum_{i=1}^{n}(datos_{observados(i)} - datos_{predichos(i)})^2 \tag{3.3}$$

En la ecuación 3.3 ECM hace referencia al error cuadrático medio y n al numero de muestras. Los datos observados y predichos son los valores de C_{D0} y C_{D2} en cada punto.

CAPÍTULO 4. RESULTADOS

En este apartado se exponen los resultados y una breve explicación de estos. Todas las trayectorias, han sido simuladas a 3000 pies de altura, empezando por la velocidad máxima, marcada por la limitación estructural de los elementos desplegados (flaps o tren de aterrizaje). En el caso de la configuración clean (flaps a 0º) y sin tren de aterrizaje, al no haber limitación por ninguno de estos dos elementos, se realizará la trayectoria en el punto de crossover altitude. Este punto es la altitud a la que podemos recorrer mas rango de velocidades, marcado por la intersección entre la velocidad máxima por límite aerodinámico por cada altura y la velocidad máxima por la limitación por estructura para cada altura.

Configuración de flaps a 0º (clean):



(a) Cl vs Cd: flaps en 0º

(b) Cl vs Cd: flaps en 0º con tren de aterrizaje

Figura 4.1: Cl vs Cd: flaps en 0º con y sin tren de aterrizaje

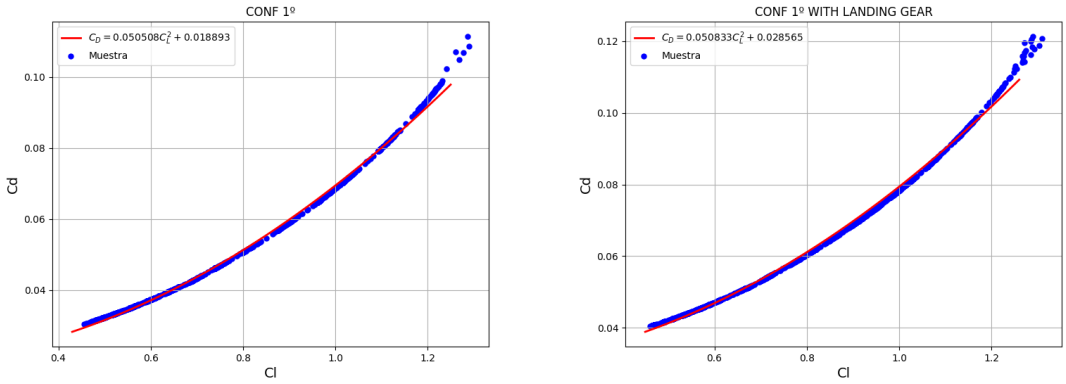
En las gráficas de la figura 4.1, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 0º. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}055572C_L^2 + 0{,}012064 \tag{4.1}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}056106C_L^2 + 0{,}020467 \tag{4.2}$$

Configuración de flaps a 1º:



(a) Cl vs Cd: flaps en 1º (b) Cl vs Cd: flaps en 1º con tren de aterrizaje

Figura 4.2: Cl vs Cd: flaps en 1º con y sin tren de aterrizaje

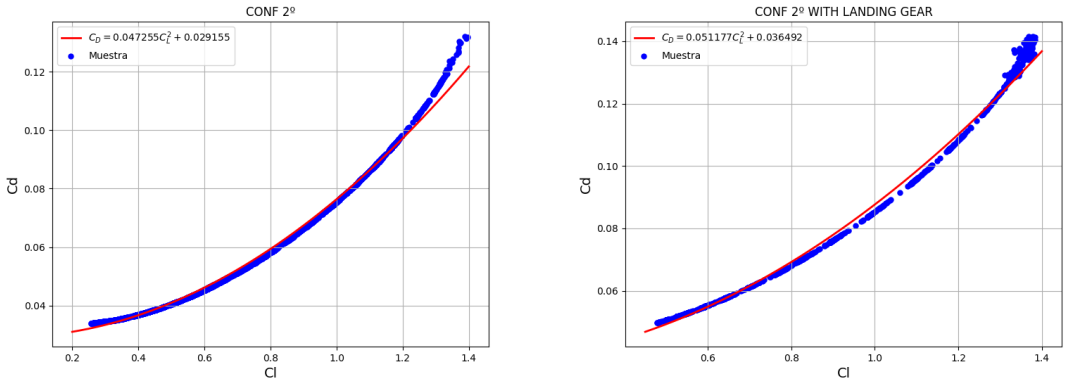
En las gráficas de la figura 4.2, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 1º. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}050508C_L^2 + 0{,}018893 \tag{4.3}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}050833C_L^2 + 0{,}028565 \tag{4.4}$$

Configuración de flaps a 2º:



(a) Cl vs Cd: flaps en 2º (b) Cl vs Cd: flaps en 2º con tren de aterrizaje

Figura 4.3: Cl vs Cd: flaps en 2º con y sin tren de aterrizaje

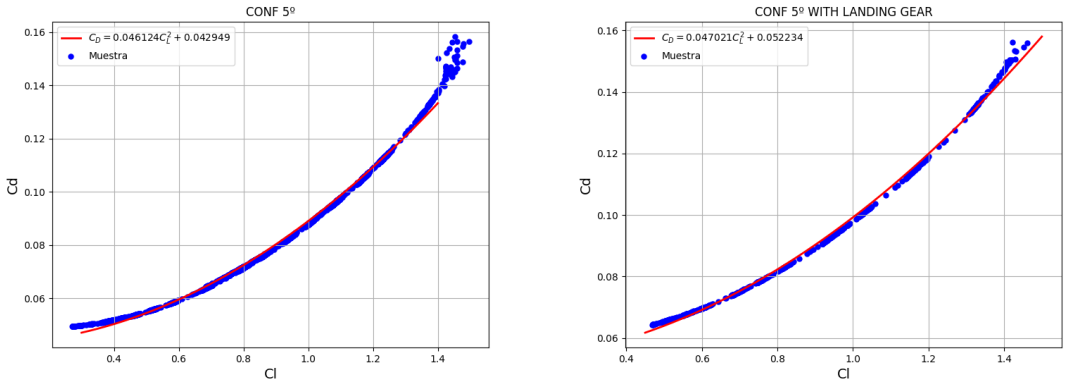
En las gráficas de la figura 4.3, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 2º. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}047255C_L^2 + 0{,}029155 \tag{4.5}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}051177C_L^2 + 0{,}036492 \tag{4.6}$$

Configuración de flaps a 5º:



(a) Cl vs Cd: flaps en 5º (b) Cl vs Cd: flaps en 5º con tren de aterrizaje

Figura 4.4: Cl vs Cd: flaps en 5º con y sin tren de aterrizaje

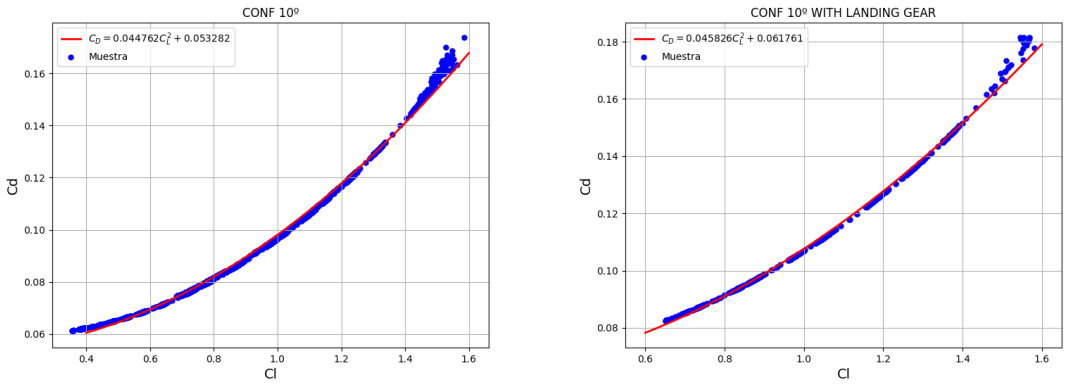
En las gráficas de la figura 4.4, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 5º. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}046124C_L^2 + 0{,}042949 \tag{4.7}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}047021C_L^2 + 0{,}052234 \tag{4.8}$$

Configuración de flaps a 10º:



(a) Cl vs Cd: flaps en 10º (b) Cl vs Cd: flaps en 10º con tren de aterrizaje

Figura 4.5: Cl vs Cd: flaps en 10º con y sin tren de aterrizaje

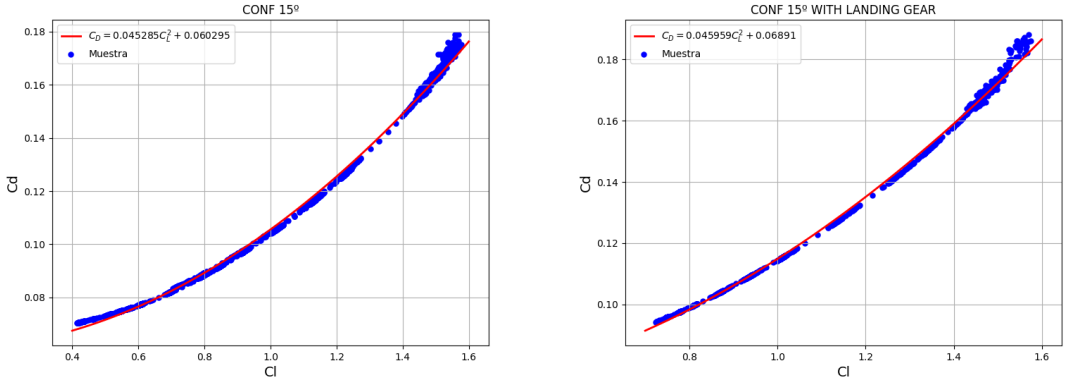
En las gráficas de la figura 4.5, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 10º. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}044762C_L^2 + 0{,}053282 \tag{4.9}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}045826C_L^2 + 0{,}061761 \tag{4.10}$$

Configuración de flaps a 15º:



(a) Cl vs Cd: flaps en 15º (b) Cl vs Cd: flaps en 15º con tren de aterrizaje

Figura 4.6: Cl vs Cd: flaps en 15º con y sin tren de aterrizaje

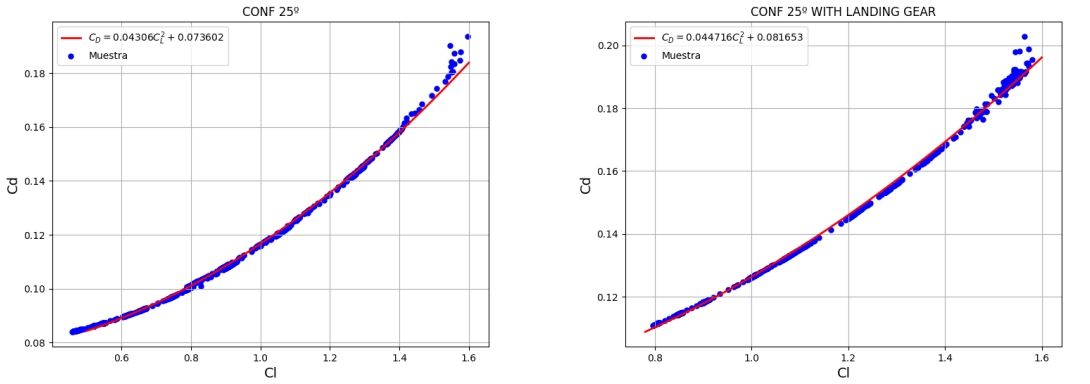
En las gráficas de la figura 4.6, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 15º. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}045285C_L^2 + 0{,}060295 \tag{4.11}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}045959C_L^2 + 0{,}06891 \tag{4.12}$$

Configuración de flaps a 25º:



(a) Cl vs Cd: flaps en 25º (b) Cl vs Cd: flaps en 25º con tren de aterrizaje

Figura 4.7: Cl vs Cd: flaps en 25º con y sin tren de aterrizaje

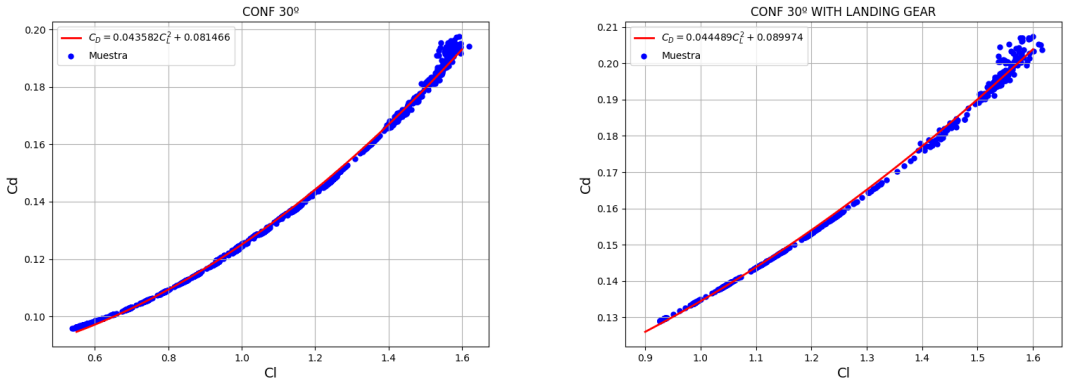
En las gráficas de la figura 4.7, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 25º. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}04306C_L^2 + 0{,}073602 \tag{4.13}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}044716C_L^2 + 0{,}081653 \tag{4.14}$$

Configuración de flaps a 30º:



(a) Cl vs Cd: flaps en 30º (b) Cl vs Cd: flaps en 30º con tren de aterrizaje

Figura 4.8: Cl vs Cd: flaps en 30º con y sin tren de aterrizaje

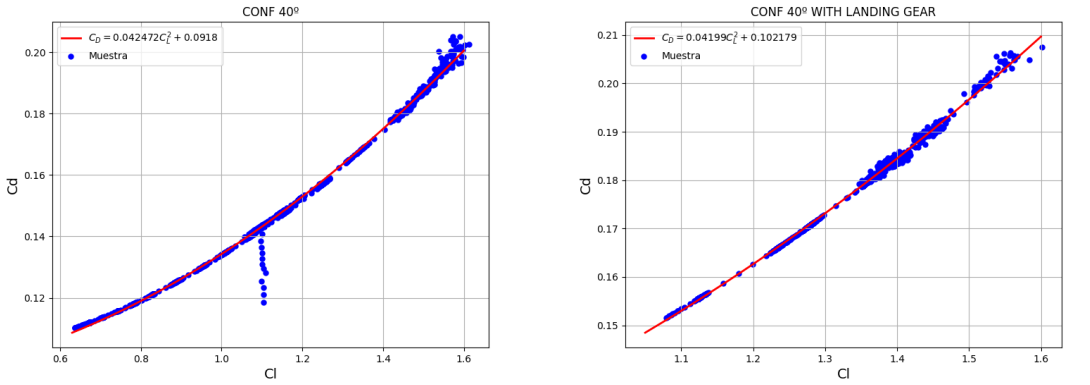
En las gráficas de la figura 4.8, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 30. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}043582C_L^2 + 0{,}081466 \tag{4.15}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}044489C_L^2 + 0{,}089974 \tag{4.16}$$

Configuración de flaps a 40º:



(a) Cl vs Cd: flaps en 40º

(b) Cl vs Cd: flaps en 40º con tren de aterrizaje

Figura 4.9: Cl vs Cd: flaps en 40º con y sin tren de aterrizaje

En las gráficas de la figura 4.9, se observa el resultado obtenido al parametrizar la muestra extraída del simulador para el caso con configuración 40º. La línea roja representa la ecuación polinómica resultante, en estos casos: Para la configuración sin tren de aterrizaje:

$$C_D = 0{,}042472C_L^2 + 0{,}0918 \tag{4.17}$$

Para la configuración con tren de aterrizaje:

$$C_D = 0{,}04199C_L^2 + 0{,}102179 \tag{4.18}$$

Tabla resumen de los valores obtenidos:

En la tabla 4.1 se introduce una columna nueva llamada R^2. Este valor hace referencia a la similitud de los resultados obtenidos con respecto al valor que se obtendría con el modelo extraído. Cuanto más cercano a 1 sea este valor, más precisa será la aproximación.

Como se puede observar, se ha caracterizado la polar de resistencia aerodinámica para todas las configuraciones. Los resultados obtenidos siguen la lógica. Donde el coeficiente de resistencia parásita (se tiene en cuenta la geometría) es el único parámetro que cambia de forma considerable cuando se despliega el tren de aterrizaje. Mientras que el coeficiente de resistencia aerodinámica inducida por el lift, permanece mas o menos constante.

En el caso de cambiar la configuración de flaps, lógicamente la curva cambia para cada caso, a medida que aumenta el ángulo de los flaps, el coeficiente debido a la geometría aumenta mientras que el otro disminuye. Lo que disminuye la curvatura, pero levanta la curva, retrasando así la entrada en pérdida del avión.

Cuadro 4.1: Tabla resumen coeficientes de drag

Flaps	**Tren aterrizaje**	C_{D2}	C_{D0}	R^2
0º	No	0.055572	0.012064	0.985733
0º	Si	0.056106	0.020467	0.992518
1º	No	0.050508	0.018893	0.998378
1º	Si	0.050833	0.028565	0.998078
2º	No	0.047255	0.029155	0.996533
2º	Si	0.051177	0.036492	0.995204
5º	No	0.046124	0.042949	0.993927
5º	Si	0.047021	0.052234	0.997894
10º	No	0.044762	0.053282	0.996605
10º	Si	0.045826	0.061761	0.999519
15º	No	0.045285	0.060295	0.997527
15º	Si	0.045959	0.06891	0.998497
25º	No	0.04306	0.073602	0.997794
25º	Si	0.044716	0.081653	0.998714
30º	No	0.043582	0.081466	0.998254
30º	Si	0.044489	0.089974	0.99744
40º	No	0.042472	0.0918	0.994927
40º	Si	0.04199	0.102179	0.997416

CONCLUSIONES

En este proyecto han habido dos objetivos claros que tienen que ser diferenciados:

El primero ha sido la creación de una forma de comunicación con el simulador, mediante la cual poder recoger datos o incluso automatizar trayectorias. Durante el desarrollo del código necesario para la comunicación, se observa que tanto el método a partir de plugin, como el método a partir de programa externo, tienen sus ventajas e inconvenientes. El plugin resulta más eficiente para programas que afecten al entorno de simulación, mientras que el programa externo permite mayor flexibilidad a la hora de ejecutar el programa y mayor margen de maniobra cada vez que sea necesario cambiar el código.

Después de la realización del proyecto se concluye que X-Plane es una herramienta de simulación muy completa que no solo permite realizar simulaciones, además se puede personalizar e incluso ampliar según se necesite.

Como segundo objetivo importante, el de adaptar los parámetros de los que no tenemos información a un modelo BADA 3. Se observa que el modelo de vuelo usado por X-Plane 11 en el Boeing 737 versión 800 se puede adaptar al modelo en cuestión de forma bastante precisa. Además, en el modo de descenso en el que no se mostraba toda la información para que fuese un descenso controlado, se observa que el valor del ESF es bastante constante.

Observando los resultados, se puede declarar la utilidad de este proyecto de cara a la investigación del cálculo de trayectorias.

Como trabajo futuro, si bien es cierto que las clases realizadas durante este trabajo servirían para cualquier trayectoria, creando el código para ésta, faltaría un análisis de aquellos tipos de trayectorias controladas que no fuesen descensos. También, en caso de requerirse más precisión, se podrían aplicar modelos de machine learning a los datos extraídos.

BIBLIOGRAFÍA

[1] XPlane 11 Flight Simulator https://www.x-plane.com/

[2] Angela Nuic, Damir Poles, Vincent Mouillet. BADA: An advanced aircraft performance model for present and future ATM systems. Publicado en Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/acs.1176

[3] Boeing: https://www.boeing.com/

[4] Sophie Gillet, Angela Nuic, Vincent Mouillet. Enhancement in realism of ATC simulations by improving aircraft behaviour models

[5] Hoekstra, Jacco. Aircraft performance models for ATM Research BADA Overview.

[6] Eurocontrol Experimental Centre.User manual for the base of aircraft data (BADA) REVISION 3.8 EEC Technical/Scientific Report No. 2010-003.

[7] Dalmau Codina, R. Optimal trajectory management for aircraft descent operations subject to time constraints. Tesis doctoral, UPC, 2019. Disponible en: ¡http://hdl.handle.net/2117/166150¿

[8] Imagen de los grados de libertad en el avión: https://es.wikipedia.org/wiki/Ejes_del_avión CC BY-SA 3.0 (Accedido el 03/01/2021)

[9] Vilardaga García-Cascón, S. An integrated framework for trajectory optimisation, prediction and parameter estimation for advanced aircraft separation concepts. Tesis doctoral, UPC, 2019. Disponible en: http://hdl.handle.net/2117/173620

[10] J.L. De Prins, K.F.M. Schippers, M. Mulder, M.M. van Paassen, J.P. Clark. Enhanced Self-Spacing Algorithm for Three-Degree Decelerating Approaches

[11] Jesper Bronsvoort, MSc. Contributions to trajectory prediction theory and its application to arrival management for air traffic control. Tesis doctoral, UPM,2014. Disponible en: http://oa.upm.es/32198/

[12] Ramon Dalmau, Marc Pérez-Batlle y Xavier Prats. Real-time Identification of Guidance Modes in Aircraft Descents Using Surveillace Data An Interacting Multiple-model (IMM) filter application.

[13] Plugin compatibility guide for X-Plane 11.50 X-Plane Developer: https://developer.x-plane.com/article/plugin-compatibility-guide-for-x-plane-11-50/

[14] Developing Plugins X-Plane Developer: https://developer.x-plane.com/article/developing-plugins/

[15] X-Plane 11 Desktop Manual X-Plane Developer: https://www.x-plane.com/manuals/desktop/

[16] Data Set Output Table X-Plane: https://www.x-plane.com/kb/data-set-output-table/

[17] ExtPlane: https://github.com/vranki/ExtPlane

[18] Foro de X-Plane: https://forums.x-plane.org/

[19] X-Plane datarefs: https://www.siminnovations.com/xplane/dataref/

[20] Accessing the X-Plane autopilot from datarefs: https://developer.x-plane.com/article/accessing-the-x-plane-autopilot-from-datarefs/

[21] Airport information X-Plane: https://apxp.uber.space/

[22] Gur, Ohad, Rosen, A.. (2008). The Aeronautical Journal. Comparison between Blade-Element models of propellers. 112. 689-704. 10.1017/S0001924000002669.

[23] N. Fezans, C. Deiler, M. S. Roeser. GENERATION OF A DATASET FOR SYSTEM IDENTIFICATIONWITH VIRTTAC-CASTOR

[24] Junzi Sun, Jacco M. Hoekstra, Joost Ellerbroek. Aircraft Drag Polar Estimation Based on a Stochastic Hierarchical Model

[25] Flight Controls http://www.b737.org.uk/flightcontrols.htm

APÉNDICES

APÉNDICE A. EJEMPLO DE CÓDIGO

```
import XPlaneReceiver as receiver
import XPlaneSender as sender
import WriteValues
'''
b738:
landing gear limit (ias) in X-Plane ias-cas 1:1
270K

flaps limit(ias)
1-250k
2-250k
5-250k
10-210k
15-200k
25-190
30-175
40-162k
230k alt flap extend

for flaps position in B738:
    1->1/8
    2->2/8
    5->3/8
    10->4/8
    15->5/8
    20->6/8
    30->7/8
    40->8/8
to change in the following dref:
sim/flightmodel/controls/flaprqst
or use the predefined function in XPlaneSender: SetFlapsPosition() and the ratio.
'''

xprec = receiver.XPlaneReceiver()
xpsen = sender.XPlaneSender()
write = WriteValues
save = write.WriteValues("Ejemploaterrizaje.dat")
#https://apxp.uber.space/airport/LEBL/
#Aquí se encuentran las frecuencias de los aeropuertos.
ils_freq = 10950
course = 245


#datarefs to receive
```

```
datarefs_needed = save.datarefs_to_add()
for data in datarefs_needed:
    xprec.AddDataRef(data)


xpsen.SendDREF("sim/cockpit2/radios/actuators/nav1_frequency_hz",ils_freq)
xpsen.SendDREF("sim/cockpit2/radios/actuators/nav1_obs_deg_mag_pilot",course)
xpsen.SetAutopilotOn()
xpsen.SetAutothrottleOn()
xpsen.SetAutopilotSpeed(201)
xpsen.ToggleSpeedHoldButton()
xpsen.SendCMND("sim/autopilot/approach")

#vpath: "sim/flightmodel2/position/vpath" vertical angle

xpsen.SetFlapsPosition(1/8)
phase = 0
while True:
    values = xprec.GetValues()
    save.SaveValues(phase,values)
    speed = values.get("sim/cockpit2/gauges/indicators/true_airspeed_kts_pilot")
    altitude = values.get("sim/cockpit2/gauges/indicators/radio_altimeter_height_ft_

    if speed >=199 and phase == 0:
        xpsen.SetFlapsPosition(2/8)
        xpsen.SetAutopilotSpeed(189)
        phase = 1

    if values.get("sim/cockpit2/autopilot/glideslope_armed") == 0:
        phase = 2
        xpsen.SetAutopilotSpeed(170)
    if speed < 184 and phase == 2:
        xpsen.SetFlapsPosition(4/8)
        xpsen.SetAutopilotSpeed(170)
        phase = 3
    if altitude < 1000:
        phase = 4
        xpsen.SetAutopilotSpeed(150)
        xpsen.SetFlapsPosition(6/8)
        xpsen.LandingGearDown()
    if altitude <700:
        phase = 5
        xpsen.SetFlapsPosition(8/8)
    if altitude <500:
        phase = 6
        xpsen.SetAutopilotSpeed(145)
    if altitude < 200:
        phase = 7
```

```
        xpsen.SetAutopilotSpeed(140)
    if altitude <15:
        phase = 8
        xpsen.SetAutothrottleOff()
        xpsen.SetSpeedBrakes()
        xpsen.SendDREF("sim/cockpit2/engine/actuators/throttle_ratio_all",0)
    if speed <= 50:
        xpsen.PauseSimulation()
        break
```