UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ETSI Aeroespacial y Diseño Industrial

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño e Implementación de un sistema de auto-tuning mediante optimización numérica para autopilotos basados en el stack de control PX4.

Trabajo Fin de Grado

Grado en Ingeniería Aeroespacial

AUTOR/A: Torres Huesca, Francisco José

Tutor/a: García-Nieto Rodríguez, Sergio

CURSO ACADÉMICO: 2023/2024

Agradecimientos

“Agradecer a mi familia, novia, amigos y compañeros por todo el apoyo durante estos cuatro años”

Resumen

0.1 Resumen

Este trabajo final de grado tiene como finalidad el desarrollo de un sistema de auto-ajuste de parámetros de control mediante optimización global empleando algoritmos genéticos, utilizando un sistema de auto-piloto basado en el stack de control de PX4. El sistema se diseña con el establecimiento de un algoritmo genético en la plataforma Matlab y con una implementación del sistema de auto-piloto en Simulink con una validación mediante la técnica SITL (software-in-the-loop). El trabajo establece tanto un modelo completo de auto-tuning basado en algoritmos genéticos, como un sistema de Simulink equivalente a PX4.

0.2 Abstract

This final degree project aims to develop a auto-tuning system for control parameters using global optimization through genetic algorithms, using an autopilot system based on the PX4 control stack. The system is designed with a genetic algorithm on the Matlab platform and an implementation of the autopilot system in Simulink, validated using the SITL (software-in-the-loop) technique. The work establishes both a complete auto-tuning model based on genetic algorithms and a Simulink system equivalent to PX4.

Índice general

Resumen I

0.1 Resumen . . . III
0.2 Abstract . . . III

Índice general V

I Memoria 1

1 Introducción 3

1.1 Motivación . . . 3
1.2 Vehículo aéreo no tripulado . . . 3

2 Objetivo y Alcance del proyecto 9

3 Descripción 11

3.1 Aeronave . . . 11
3.2 Auto-piloto . . . 13
3.3 Dinámica . . . 21
3.4 Estación en tierra . . . 21
3.5 Software de implementación . . . 22
3.6 Software in the loop . . . 24

4 Condicionantes y limitaciones del proyecto 29

4.1 Software . . . 29
4.2 Dinámica . . . 30

5 Soluciones alternativas 31

5.1 Software . . . 31

5.2 Ajuste parámetros 33

6 Implementación PX4 en Simulink 37

6.1 Controladores 37
6.2 Validación Simulink 59

7 Implementación auto-ajuste 65

7.1 Algoritmo genético 65
7.2 Planteamiento del problema 67
7.3 Función objetivo 68
7.4 Ajuste dinámica longitudinal 69
7.5 Ajuste dinámica lateral 70
7.6 Ajuste global 71
7.7 Interfaz gráfica 71

8 Resultados 75

9 Conclusiones 89

9.1 Conclusiones 89

10 Anexo 91

10.1 Objetivos del desarrollo sostenible agenda 2030 91
10.2 Funciones variables de entrada 92
10.3 Funciones Control de posición 93
10.4 Funciones Control Actitud 95
10.5 Funciones auto-tuning 96

Bibliografía 101

Índice de figuras

1.1. RPA. Fuente: [12] 4
1.2. RPAS. Fuente: [12] 4
1.3. Clasificación UAV. Fuente: [8] 5
1.4. Categoría abierta. Fuente:[19] 6
1.5. Categoría específica. Fuente:[19] 6
1.6. Categoría certificada. Fuente:[19] 7

3.1. Método de los paneles. Fuente: [14] 12
3.2. Diagrama auto-piloto PX4. Fuente: [5] 14
3.3. Diagrama conceptual seguimiento waypoints. Fuente: propia 15
3.4. Radio de transición. Fuente: propia 15
3.5. Distancia de anticipación. Fuente: [22] 16
3.6. Diagrama Total Energy Control Loop Teórico. Fuente: [5] 18
3.7. Diagrama Total Energy Balance Control Loop Teórico. Fuente: [5] 18
3.8. Diagrama Attitude Control Teórico. Fuente: [5] 19
3.9. Entorno Ubuntu 20.04. Fuente: propia 20
3.10. Diagrama Dinámica. Fuente: propia 21
3.11. Entorno QGroundControl. Fuente: propia 22
3.12. Entorno Matlab. Fuente: propia 23
3.13. Entorno Simulink. Fuente: propia 24
3.14. Esquema conexión PX4. Fuente: [16] 24
3.15. Interfaz PX4 (Ubuntu 20.04). Fuente: propia 26
3.16. Esquema Conexión SITL. Fuente: [11] 26

3.17. Diagrama Pixhak SIL Connector y Dinámica. Fuente: propia 27
3.18. Diagrama Pixhak SIL Connector. Fuente: propia 28
3.19. Conexión QGroundControl. Fuente: propia 28

5.1. Diagrama conceptual implementacion PX4 en Simulink. Fuente: propia 32
5.2. Diagrama flujo algoritmo genético. Fuente: [2] 34
5.3. Machine learning. Fuente: [17] 35
5.4. Reinforcement Learning. Fuente: [21] 36

6.1. Bloques cambio a tiempo discreto y continuo. Fuente: propia 38
6.2. Bloque Path Manager. Fuente: propia 39
6.3. Bloque Waypoint Follower. Fuente: propia 40
6.4. Bloque Seguimiento Waypoints. Fuente: propia 41
6.5. Diagrama Simulink controlador L1. Fuente: propia 43
6.6. Diagrama Total Energy Control Loop. Fuente: propia 43
6.7. Diagrama variación energía específica estimada. Fuente: propia 45
6.8. Diagrama variación energía específica referencia. Fuente: propia 46
6.9. Diagrama Integrador Total Energy Control Loop. Fuente: propia 46
6.10. Diagrama Derivador Total Energy Control Loop. Fuente: propia 47
6.11. Diagrama ganancia feedforward Total Energy Control Loop. Fuente: propia . . . 47
6.12. Diagrama corrección throttle con ratio underspeed. Fuente: propia 48
6.13. Diagrama Total Energy Balance Control Loop. Fuente: propia 49
6.14. Diagrama variación balance de energía estimado. Fuente: propia 50
6.15. Diagrama variación balance de energía referencia. Fuente: propia 51
6.16. Diagrama Integrador Total Energy Balance Control Loop. Fuente: propia 51
6.17. Diagrama Derivador y ganancia feedforward Total Energy Balance Control Loop. Fuente: propia 52
6.18. Diagrama correccion incremento pitch. Fuente: propia 53
6.19. Diagrama Attitude Control. Fuente: propia 53
6.20. Diagrama Euler Rate Setpoint Pitch y Yaw. Fuente: propia 55
6.21. Diagrama Roll Body Rate Setpoint. Fuente: propia 55
6.22. Diagrama Pitch Body Rate Setpoint. Fuente: propia 56
6.23. Diagrama Yaw Body Rate Setpoint. Fuente: propia 56
6.24. Diagrama ganancia feedforward. Fuente: propia 57

6.25. Diagrama Scaler_1 ganancia feedforward. Fuente: propia 57
6.26. Diagrama Control PI. Fuente: propia 57
6.27. Diagrama Scaler_2 Control PI. Fuente: propia 58
6.28. Diagrama Mix (Control Allocation). Fuente: propia 58
6.29. Trayectoria QGroundControl. Fuente: propia 59
6.30. Validación Trayectoria x-y. Fuente: propia 60
6.31. Validación Altitud. Fuente: propia 61
6.32. Validación Velocidad. Fuente: propia 62
6.33. Validación Actitud. Fuente: propia 63
6.34. Validación Controles. Fuente: propia 64

7.1. Diagrama planteamiento algoritmo genético. Fuente: propia 67
7.2. Flujograma sistema algoritmo genético. Fuente: propia 68
7.3. Interfaz: Definición variables. Fuente: propia 72
7.4. Interfaz: Resultados ajuste longitudinal. Fuente: propia 72
7.5. Interfaz: Resultados ajuste lateral. Fuente: propia 73
7.6. Interfaz: Resultados parámetros ajuste global. Fuente: propia 73
7.7. Interfaz: Resultados gráficos parámetros ajuste global. Fuente: propia 74
7.8. Interfaz: Resultados actitud ajuste global. Fuente: propia 74

8.1. Iteraciones parámetros pitch. Fuente: propia 76
8.2. Error longitudinal. Fuente: propia 76
8.3. Comparación referencias longitudinales. Fuente: propia 77
8.4. Iteraciones parámetros roll. Fuente: propia 78
8.5. Iteraciones parámetros yaw. Fuente: propia 79
8.6. Error lateral. Fuente: propia 79
8.7. Comparación referencias laterales. Fuente: propia 80
8.8. Iteraciones parámetros roll finales. Fuente: propia 81
8.9. Iteraciones parámetros pitch finales. Fuente: propia 82
8.10. Iteraciones parámetros yaw finales. Fuente: propia 82
8.11. Error global. Fuente: propia 83
8.12. Comparación referencias en ángulos. Fuente: propia 83
8.13. Comparación referencias en velocidades. Fuente: propia 84

8.14. Trayectoria PX4 vs PX4 auto-tuning. Fuente: propia 85
8.15. Altitud PX4 vs PX4 auto-tuning. Fuente: propia 86
8.16. Velocidad PX4 vs PX4 auto-tuning. Fuente: propia 86
8.17. Trayectoria Simulink auto-tuning vs PX4 auto-tuning. Fuente: propia 87
8.18. Altitud Simulink auto-tuning vs PX4 auto-tuning. Fuente: propia 87
8.19. Velocidad Simulink auto-tuning vs PX4 auto-tuning. Fuente: propia 88

10.1. Función ste 92
10.2. Función V_rate 92
10.3. Función weight 92
10.4. Función ratio_underspeed 92
10.5. Función cour_gnds 93
10.6. Función waypoints_alt 93
10.7. Función vel 93
10.8. Función T_predicted 93
10.9. Función T_integ 94
10.10Función seb_to_pitch 94
10.11Función p_integ 94
10.12Función pitch_incre 94
10.13Función pitch_incre_lim 95
10.14Función eta 95
10.15Función euler_rate 95
10.16Función roll_crtl 95
10.17Función pitch_crtl 95
10.18Función yaw_crtl 96
10.19Auto-tuning longitudinal 97
10.20Función objetivo longitudinal 98
10.21Función objetivo lateral 99

Índice de tablas

1.1. Categorías en función de MTOW. Fuente: [19] 6

3.1. Características aeronave 12
3.2. Derivadas aerodinámicas 13

6.1. Constantes Controlador L1 42
6.2. Constantes Total Energy Control Loop 44
6.3. Constantes Total Energy Balance Control Loop 50
6.4. Valores Attitude Control 54
6.5. Waypoints Trayectoria Validación 59

7.1. Rango valores control pitch 69
7.2. Rango valores control roll y yaw 70

8.1. Valores estimados control pitch 75
8.2. Valores estimados control roll y yaw 78
8.3. Valores estimados control final 81

10.1. Caption 91

Parte I

Memoria

Capítulo 1

Introducción

1.1 Motivación

Las aeronaves no tripuladas se encuentran en gran evolución en las últimas décadas, siendo aplicadas cada vez en un mayor abanico de sectores gracias a sus numerosas ventajas. Por ese motivo un gran número de empresas del sector aeroespacial están apostando por esta nueva tecnología. Dentro de estas pequeñas aeronaves hay numerosos campos a desarrollar pero uno de los más importantes y de mayor interés de mejora reside en el sistema de control autónomo que permite un vuelo sin necesidad de intervención humana. Prosiguiendo en la investigación de esta rama de desarrollo de establece el tema de este proyecto.

1.2 Vehículo aéreo no tripulado

1.2.1 Definición

Un vehículo aéreo no tripulado (UAV) es cualquier aeronave no tripulada capaz de llevar a cabo una determinada misión. Este vuelo puede ser realizado de forma autónoma controlando la aeronave con un sistema de auto-piloto incorporado o mediante el control manual de un operador en tierra; este tipo de control se establece por las siglas RPA (Remotely Piloted Aircraft).

Como ha sido comentado, un vehículo aéreo no tripulado requiere de un conjunto de sistemas adicionales para llevar a cabo su vuelo todos estos sistemas se engloban en lo conocido como Unmanned Aircraft System (UAS) este sistema esta compuesto por el vehículo aéreo, un sistema de comunicaciones y una estación en tierra; especificando para el sistema de los UAVs con control manual, este sistema se conoce como Remotely Piloted Aircraft System (RPAS) [12]

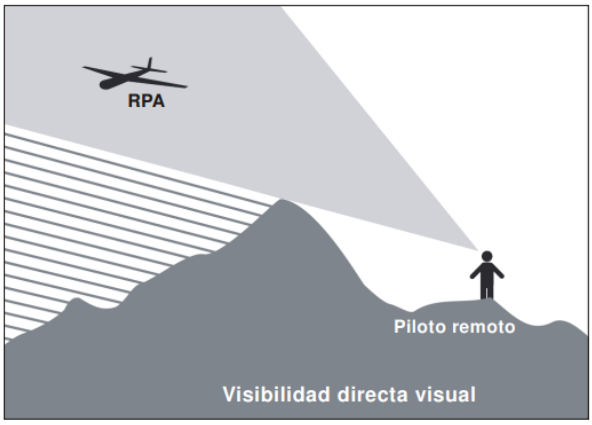

Figura 1.1: RPA. Fuente: [12]

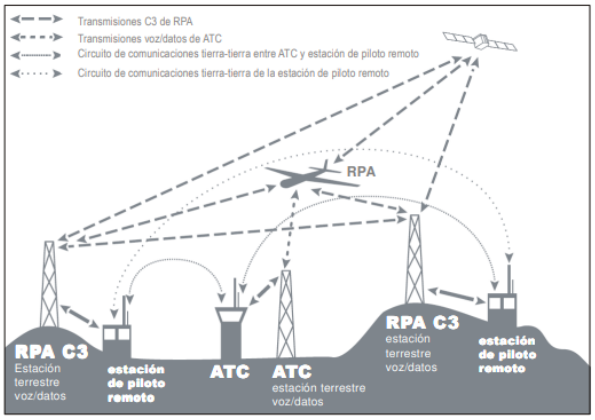

Figura 1.2: RPAS. Fuente: [12]

1.2.2 Historia

Para comprender mejor la evolución actual de este tipo de aeronaves es necesario conocer un poco de su historia. Aunque el termino UAV suene moderno, la primera aparición de una aeronave no tripulada de remonta a tiempos de Leonardo da Vinci con su invento el ornitóptero, del cual existía una pequeña maqueta la cual realizaba vuelos sin tripulación; posteriormente, otro de los momentos históricos de las primeras apariciones de este tipo de aeronaves fue en en 1849 cuando los austriacos utilizaron globos no tripulados para el bombardeo de Venecia. Posteriormente, durante la primera guerra mundial, su uso aumento, utilizando aeronaves no tripuladas para tareas de vigilancia; durante esta época surgieron grandes avances en el campo tecnológico como son la radio (permitiendo el radio control) y el giroscopio, que supusieron una evolución de estos vehículos, este tipo de tecnología fue la utilizada durante la segunda guerra mundial. Finalmente, la última gran evolución de las aeronaves no tripuladas ocurrió en los años 80 y 90 con el desarrollo de la computación y sistemas de control electrónico, lo que ha llevado a los drones a su forma actual. [4]

1.2.3 Clasificación

La clasificación de este tipo de aeronaves puede llegar a ser una ardua tarea debido a la gran cantidad de posibles combinaciones existentes.
La primera clasificación a tratar consiste en una diferenciación en función de como se genera la sustentación.El conjunto global de aeronaves se pueden dividen entre aerostatos y aerodinos los primero requieren de un gas para generar sustentación (globos aerostáticos) y lo segundos requieren de propulsión para volar (aviones); los UAVs se encuentran dentro de este último tipo. A su vez los UAVs se pueden dividir en tres grandes áreas [25]:

- UAV de ala fija: más ligeros y de mayor autonomía pero requieren de estar siempre en movimiento, se pueden clasificar en función de la posición de sus alas en: ala alta, ala media, ala baja o ala volante.
- UAV de ala rotatoria: tienen la capacidad de despegue en vertical y vuelo estático, en función de su número de motores se pueden clasificar en: aeronaves con un rotor principal y uno de cola, drones con un único rotor, drones con dos rotores en configuración coaxial, con configuración en tándem o multirrotores.
- UAV híbrido: combinan las tecnologías anteriores, tienen la capacidad de despegue vertical y de realizar vuelos a altas velocidades.

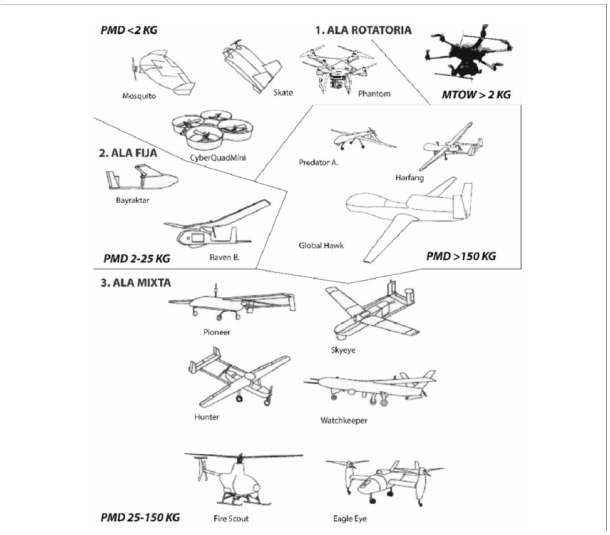

Figura 1.3: Clasificación UAV. Fuente: [8]

Centrado la atención en una clasificación más legislativa, la agencia estatal de seguridad aérea (EASA) divide este tipo de aeronaves en tres categorías [19]:

- Categoría abierta: cubre las operaciones de bajo riesgo y no se requiere de autorización. Este tipo se divide en subcategorías en función de su masa máxima al despegue (MTOW) definidas por:

  Se utiliza esta medida como referencia ya que esta masa establece la energía cinética en caso de colisión de la aeronave y de esta forma la peligrosidad de la misma.

Categoría	MTOW
C0	<250 g
C1	<900 g
C2	<4 kg
C3	<25 kg
C4	<25 kg y control manual

Tabla 1.1: Categorías en función de MTOW. Fuente: [19]

Figura 1.4: Categoría abierta. Fuente:[19]

- Categoría específica: cubre las operaciones de riesgo medio y es necesario obtener una autorización personal emitida por EASA para el vuelo.

Figura 1.5: Categoría específica. Fuente:[19]

- Categoría certificada: se establece para operaciones de alto riesgo siendo necesario la certificación del UAS por EASA, la del operador y la obtención de la licencia del piloto a distancia.

Al igual que se han desarrollado estas dos clasificaciones, existen numerosas más en función de la estructura, del tipo de propulsión, de la propiedad, entre otras. Las dos desarrolladas son las

Figura 1.6: Categoría certificada. Fuente:[19]

de mayor interés para la aeronave de estudio del proyecto que será comentada más adelante en el proyecto.

1.2.4 Aplicaciones

Como ha sido comentado anteriormente, este tipo de aeronaves poseen una gran versatilidad lo que hace que sean utilizadas en un gran abanico de sectores tanto civiles como militares. Comentado previamente las aplicaciones en el ámbito civil de estas aeronaves destacan [10]:

- Agricultura.
- Inspección de infraestructuras.
- Control de fauna.
- Control ambiental: valores atmosféricos, contaminación, erosión, control de glaciales.
- Topografía y fotogrametría
- Ocio: competiciones.
- Filmación.
- Salvamento marítimo y extinción de incendios.

Por otro lado, las aplicaciones militares son más reducidas, pero han supuesto un antes y un después en el desarrollo de las guerras en las que han sido utilizadas, permitiendo reducir riegos de las tropas en tareas de vigilancia y bombardeo de objetivos.

Capítulo 2

Objetivo y Alcance del proyecto

El objeto del proyecto es el desarrollo de un sistema de auto-ajuste de los parámetros de control del modulo de actitud de un sistema de auto-piloto basado en PX4, este auto-tuning se realiza con optimización global mediante algoritmos genéticos.
Este proyecto centra su atención en la implementación del stack de control de PX4 en un diagrama de bloques de la herramienta Simulink que simule las actuaciones del auto-piloto con conexión con el sistema de estimación de dinámica de la aeronave, estableciendo mediante SITL las característica del vuelo de la aeronave. La optimización global del sistema permite obtener los parámetro de control del modulo de actitud haciendo uso de un código conjunto de ejecución del algoritmo genético desarrollado en Matlab.Con todo esto se persigue el objetivo final del proyecto que reside en el desarrollo de un sistema de auto-tuning de un sistema de auto-piloto.

Capítulo 3

Descripción

3.1 Aeronave

Para el desarrollo del proyecto se hace uso de una aeronave perteneciente a la clasificación UAV de ala fija, desarrollando en más profundidad las características de este tipo de aeronaves: un UAV de ala fija se caracteriza por poseer una estructura similar a la de un avión, es decir la sustentación es generada por alas fijas no por alas rotatorias, la propulsión de este tipo de aeronaves puede ser muy diversa, desde hélices con motores eléctricos, motores de combustión interna o sistemas híbridos en función del tamaño y misión del vehículo.
Una vez conocido la definición se desarrollan una serie de características de este tipo de vehículos:

- Gran aerodinámica: esto es debido a su estructura simular a la de un avión permitiendo que la aeronave planee y se desplace de forma horizontal a gran velocidad, mayor que un aeronave de ala rotatoria.
- Autonomía: en general poseen un gran autonomía debido a su reducido tamaño y peso junto con una aerodinámica muy bien optimizada.
- Estabilidad: este tipo de aeronave posee una gran estabilidad de vuelo.
- Maniobrabilidad: la capacidad de maniobra de este tipo de aeronave es menor que las de ala rotatoria, esto es debido a sus limitaciones en las capacidades de giro y cambio de dirección ya que no esta permitido el giro sobre su propio eje.
- Para el despegue y aterrizaje se necesita una pista, siendo este punto una desventaja comparando con los drones de alas rotatorias que poseen la capacidad de despegue y aterrizaje en vertical.
- Posee una gran capacidad de carga de pago[6].

La aeronave utilizada para el proyecto se encuentra definida en el artículo *Design, implementation and flight verfication of a versatile and rapidly reconfigurable UAV GNC research platform*[14]. La definición del tipo de aeronave viene determinada por las características aerodinámicas y de

la planta propulsora, estas vienen determinadas por el perfil alar, la geometría de la aeronave, la dispersión de la masa y las características del motor.
Las características de la aeronave son:
Para el cálculo de los coeficientes aerodinámicos se emplea el método de los paneles para resolver

Parámetro	Valor	Unidades
Envergadura	2.795	m
Cuerda	0.35134	m
Masa	8.16466	kg
I_{xx}	4.12	(kg/m^2)
I_{yy}	9.58	(kg/m^2)
I_{zz}	9.85	(kg/m^2)

Tabla 3.1: Características aeronave

las ecuaciones de Prandtl-Glauert, el modelo utilizado es:
Con la resolución de estas ecuaciones se obtienen los coeficientes aerodinámicos adimensionales

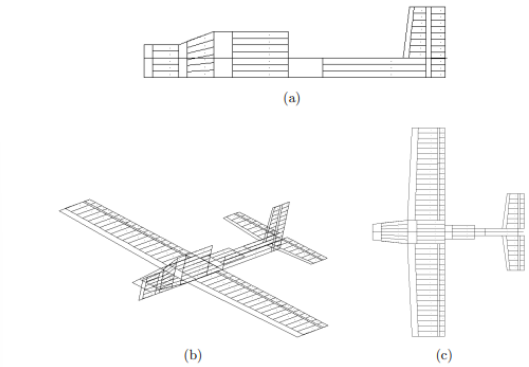

Figura 3.1: Método de los paneles. Fuente: [14]

requeridos para el modelado de la aeronave.

El esquema propulsivo de la aeronave se rige la ecuación que permite obtener el empuje [lbs] a partir de una velocidad de vuelo y una posición de la palanca de gases, la ecuación es:

$$Thrust = [T_o - 0{,}0047 * \sqrt{T_o} * V] \tag{3.1}$$

$$T_o = 0{,}0399299 * (0{,}163492 + dT)^2 \tag{3.2}$$

Esta segunda ecuación es la responsable de establecer las características del motor, máximas y mínimas revoluciones, diámetro y empujes característicos.

Parámetros de Fuerzas	Valor	Parámetros de Momentos	Valor
CL_0	0.38	Cl_b	-0.296
CL_a	6	Cl_p	-1.96
$CL_{\dot{a}}$	2.64	Cl_r	0.103
CL_q	7.4	Cl_{dA}	0.1695
CL_{dE}	0.24	Cl_{dR}	0.106
CL_{dF}	0.4		
		Cm_0	0.3
CD_0	0.022	Cm_a	-1.239
A_1	0.007	$Cm_{\dot{a}}$	-7
A_{Polar}	0.057	Cm_q	-2.4
		Cm_{dE}	-3.2
CY_b	-1.098	Cm_{dF}	-0.021
CY_{dR}	0.143		
		Cn_b	0.277
		Cn_p	-0.0889
		Cn_r	-0.19997
		Cn_{dA}	-0.023
		Cn_{dR}	-0.1997

Tabla 3.2: Derivadas aerodinámicas

3.2 Auto-piloto

Un auto-piloto es un sistema diseñado para tomar decisiones en función de unos parámetros de la aeronave predefinidos y de condiciones de contorno, para conseguir que el vehículo cumpla con unos requerimientos establecidos en la misión a desarrollar.

3.2.1 PX4

Para el caso de estudio, el software utilizado como referencia para el vuelo autónomo pertenece a PX4.
PX4 es un software de código abierto para el control de vuelo para drones y vehículos no tripulados. Un software de código abierto significa que cualquier usuario puede usarlo y modificarlo en función del proyecto a desarrollar, esto permite una constante actualización del sistema. Este software se puede encontrar en la plataforma GitHub[23] desde la cual se puede descargar o modificar.
De este software se pueden extraer las siguientes características:

- Posee una estructura modular, lo que facilita la fácil incorporación de nuevos algoritmos y funciones.
- Permite soporte multiplataforma, siendo un software de gran flexibilidad gracias a su amplia compatibilidad.

- Admite gran variedad de vehículos, tanto aéreos como terrestres en numerosas con diversas configuraciones: multicópteros, aviones de ala fija, VTOL, vehículos terrestre y submarinos.
- Permite un control de vuelo avanzado en el que se pueden encontrar diversidad de modos de vuelo, los diferentes modos proporcionan distintos niveles de automatización y auto-piloto. En los modos autónomos no se requiere de ninguna intervención externa y el vehículo de controla de forma total por un auto-piloto, mientras que en los modos de control manual, las ordenes se producen por el usuario y el auto-piloto solo asiste.
- Integración robusta y profunda.
- Planificación, navegación, telemetría y monitorización de trayectorias en tiempo real.

La base de este proyecto se centra en la parte de pilotos automáticos para ello primero es necesario definir este concepto, un piloto automático es el çerebro"de un drone, formado por una parte de software que es la que se encarga de obtener los requisitos para la estabilización y seguridad de la aeronave. Para el caso de este proyecto este auto piloto será utilizado para la simulación de las trayectorias de ajuste[11].
El auto-piloto se puede dividir en diferentes bloques siguiendo el siguiente esquema:

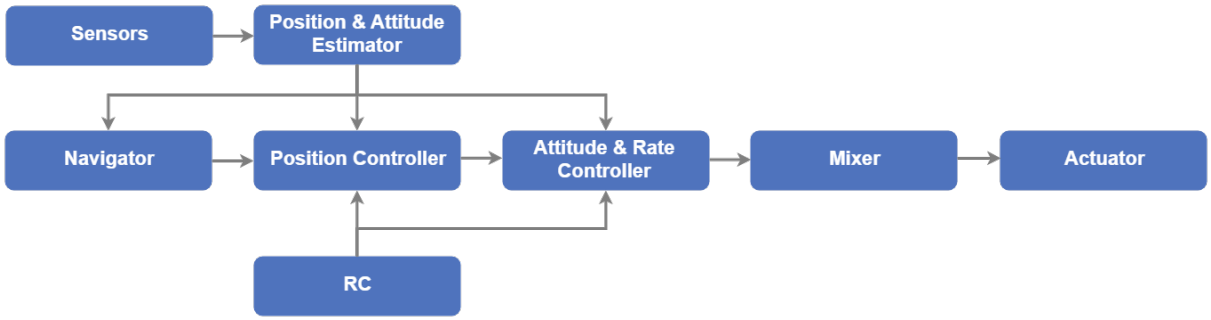

Figura 3.2: Diagrama auto-piloto PX4. Fuente: [5]

Dentro de este diagrama, los dos subsistemas de mayor importancia y los determinantes del sistema de control autónomo de vuelo son el de control de posición y el de control de actitud. Comenzando la explicación teórica por el primero mencionado, el controlador de posición se encarga de obtener unos ángulos de actitud de referencia para la aeronave y el empuje necesario a partir de unas condiciones predefinidas de la misión a seguir, estas condiciones viene establecidas por la definición de la trayectoria.
Este controlador de posición se divide en tres bloques:

- Seguimiento de Waypoints
  Este subsistema del control de posición realiza la función de navegación del sistema, para ello se establece como entrada los requerimiento de la trayectoria a seguir definiendo posición de los waypoints (latitud, longitud y altitud) y velocidades en de referencia. El esquema teórico a seguir viene determinado como:

  Este contiene un primer bloque que determina dos posiciones de waypoints para trazar la trayectoria a seguir en cada fase de vuelo, con estos datos el segundo bloque realiza el

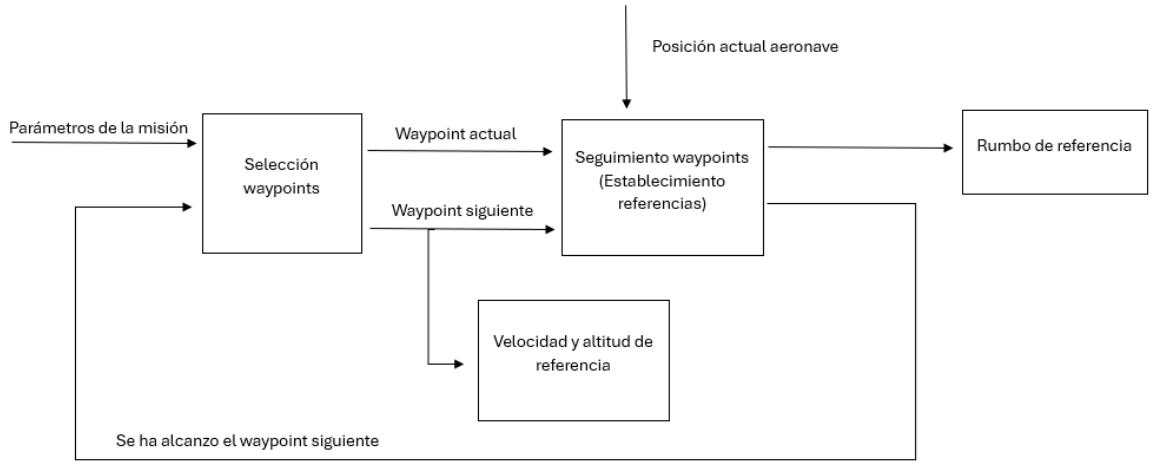

Figura 3.3: Diagrama conceptual seguimiento waypoints. Fuente: propia

seguimiento y calcula un rumbo de referencia que sera utilizado en posteriores bloques. Este segundo bloque requiere de un radio de transición para determinar cuando se ha alcanzado el waypoint final del segmento y se cambia al siguiente.

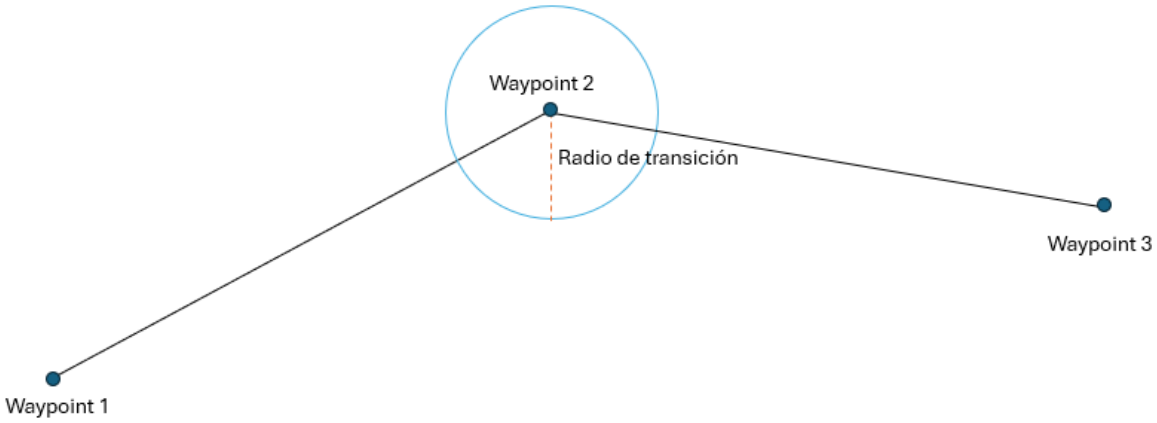

Figura 3.4: Radio de transición. Fuente: propia

Cuando la aeronave ingresa en dicho radio el sistema detectará que el waypoint ha sido alcanzado y se proseguirá con el siguiente segmento.

- Controlador L1
  Este controlador encargado de la obtención de un ángulo de alabeo a partir de una distancia de anticipación de la trayectoria (L1), las entradas de este sistema son: rumbo de referencia, un rumbo estimado y una distancia de anticipación.
  El ángulo de alabeo de referencia se establece por la aceleración lateral necesaria para

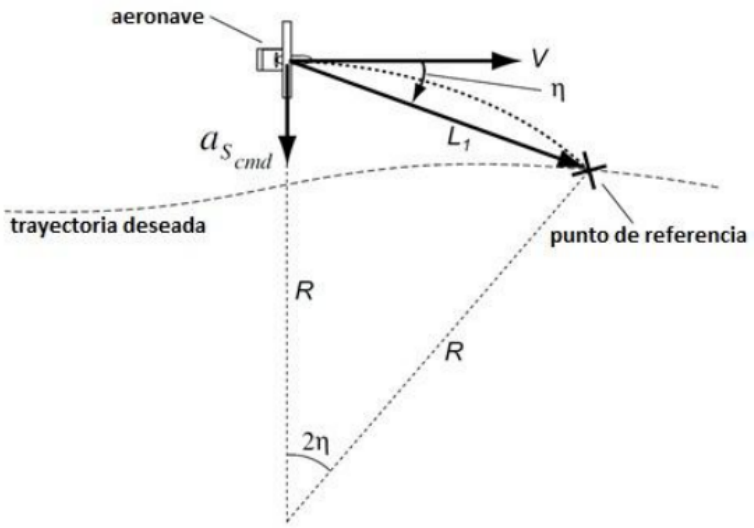

Figura 3.5: Distancia de anticipación. Fuente: [22]

llevar a cabo el giro requerido para un buen seguimiento de la trayectoria deseada. Esta aceleración lateral viene establecida por la velocidad, el seno del error en el rumbo y la distancia de anticipación.

- Total energy control system (TECS)
  Este subsistema se encarga de abordar el problema desde el punto de vista energético para obtener los valores de referencia de ángulo de cabeceo y posición de la palanca de gases del motor tomando como valores de partida la velocidad y altitud de referencia establecidas para la trayectoria a realizar.
  Para realizar un control simultáneo de la velocidad y de la altitud es necesario tener en cuenta que al aumentar el ángulo de cabeceo se provoca un aumento de la altitud pero una disminución de la velocidad, sin embargo si se produce un aumento de la palanca de gases se aumenta la velocidad, lo que provoca un aumento de la sustentación; resumiendo, se esta ante un problema en el que ambas variables de entrada afectan a ambas salidas.
  Para el desarrollo del problema, previamente es necesario conocer que la energía total es igual a la suma de la energía cinética más la energía potencial; la primera de ellas se define como la energía producida cuando un objeto está en movimiento, y la segunda esta asociada a la relación entre un cuerpo y un campo de fuerza; la ecuación resultante que queda de ambas es la siguiente:

$$E_T = \frac{1}{2} * m * V_T^2 + m * g * h \tag{3.3}$$

  Correspondiendo el primer término a la energía cinética y el segundo a la potencial.
  Este método de energía permite desacoplar el sistema estableciendo un subsistema para determinar un ángulo de cabeceo de referencia y por otro lado establecer una posición de palanca de gases.
  Observando la influencia de los controles en términos de energía se ve que la palanca de gases produce una variación en el estado energético total de una aeronave, esto se puede lograr con combinaciones arbitrarias de energía cinética y potencial (volar a gran altitud y

baja velocidad es lo mismo que volar a baja altitud y alta velocidad), estas combinaciones definen el balance de energía específico. Desarrollando esta variación de energía total en ecuaciones se obtiene:

$$\dot{E}_T = m * V_T * \dot{V}_T + m * g * \dot{h} \tag{3.4}$$

A partir de la variación de energía total se puede obtener la variación de energía especifica de la siguiente forma:

$$\dot{E} = \frac{\dot{E}_T}{m * g * V_T} = \frac{\dot{V}_T}{g} + \frac{\dot{h}}{V_T} = \frac{\dot{V}_T}{g} + sin(\gamma) \tag{3.5}$$

Si la pendiente de vuelo es pequeña se establece:

$$\dot{E} \approx \frac{\dot{V}_T}{g} + \gamma \tag{3.6}$$

La relación entre la variación de energía y el empuje se determina a través de la ecuación dinámica de sumatorio de fuerzas en dirección longitudinal.

$$T - D = m * g * (\frac{\dot{V}_T}{g} + sin(\gamma)) \approx m * g * (\frac{\dot{V}_T}{g} + \gamma) = \Delta T \tag{3.7}$$

De esta forma se obtiene una relación directa entre el empuje y la variación de energía.
Poniendo la atención ahora en el ángulo de cabeceo, este se encarga del balance de energía específico, transfiriendo energía cinética a potencial y viceversa, por lo que se encarga de determinar como se consigue el estado energético total. El balance de energía específico se expresa a partir de la siguiente ecuación:

$$\dot{B} = \gamma - \frac{\dot{V}_T}{g} \tag{3.8}$$

Como conclusión se obtiene que al utilizar este método se consigue desacoplar el sistema y establecer cada una de las variables de referencia independientemente.
Aplicando todo lo desarrollado anteriormente, los esquemas teóricos a seguir son los siguientes [5]:

El segundo de los bloques importantes determina el control de actitud de la aeronave, el control de actitud permite obtener los valores de deflexión de las superficies que controlan la aeronave a partir de unos valores de ángulos de alabeo y cabeceo de referencia establecidos con anterioridad en el control de posición [5].
Comenzando por una explicación teórica, el control de actitud funciona con un método de bucle en cascada formado por un bucle externo y un bucle interno, el esquema teórico es el siguiente:

El bucle externo es el encargado de calcular el error entre la actitud de referencia y la estimada, este error al multiplicarlo por una ganancia P se genera un punto de ajuste de velocidad que se toma como entrada del bucle interno; en este segundo se calcula el error en las velocidades de variación de actitud y se aplica un controlador PI junto con una ganancia feedforward generando la aceleración angular necesaria, las acciones de control son escaladas por la velocidad ya que las superficies de control son más efectivas a altas velocidades que a bajas. Con esta aceleración

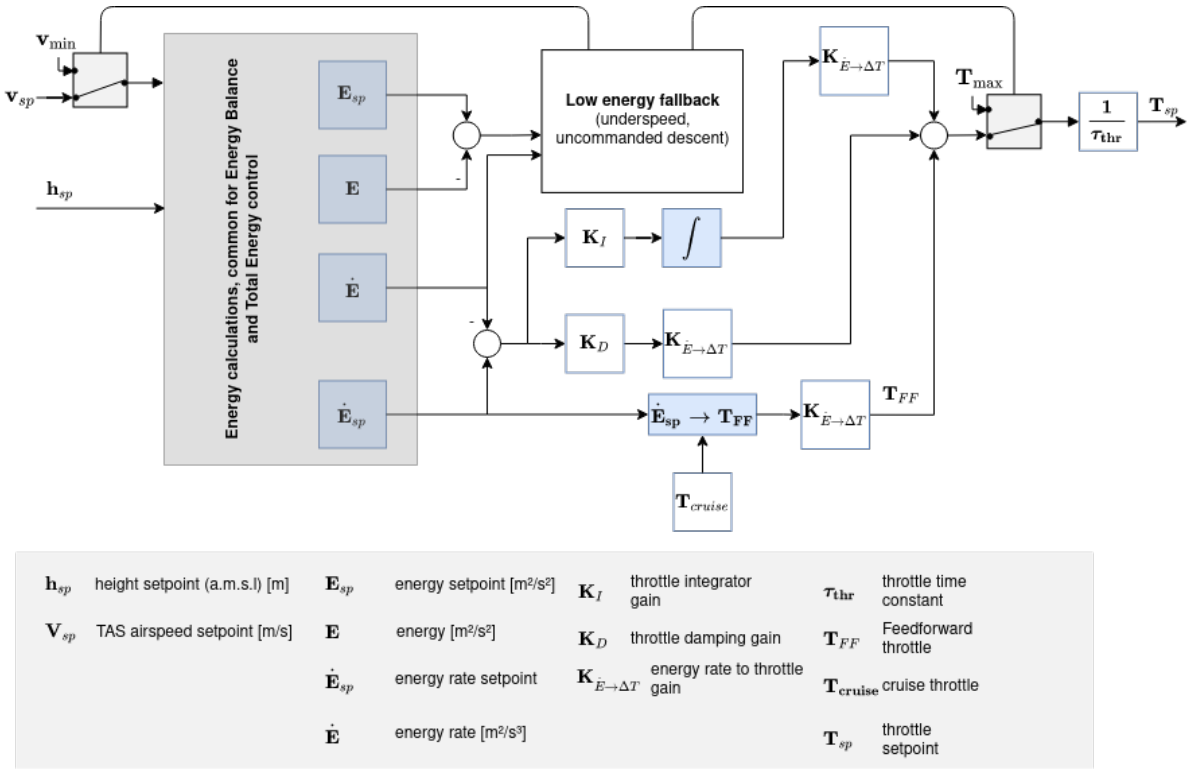

Figura 3.6: Diagrama Total Energy Control Loop Teórico. Fuente: [5]

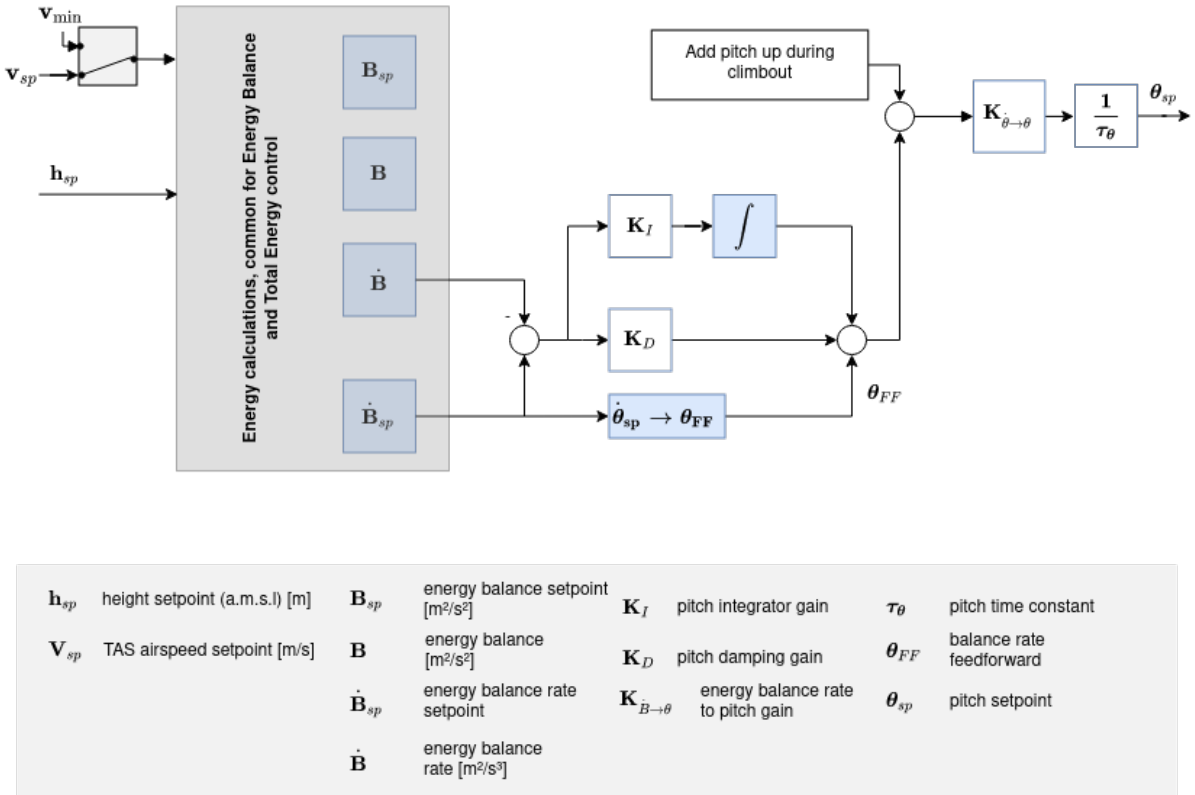

Figura 3.7: Diagrama Total Energy Balance Control Loop Teórico. Fuente: [5]

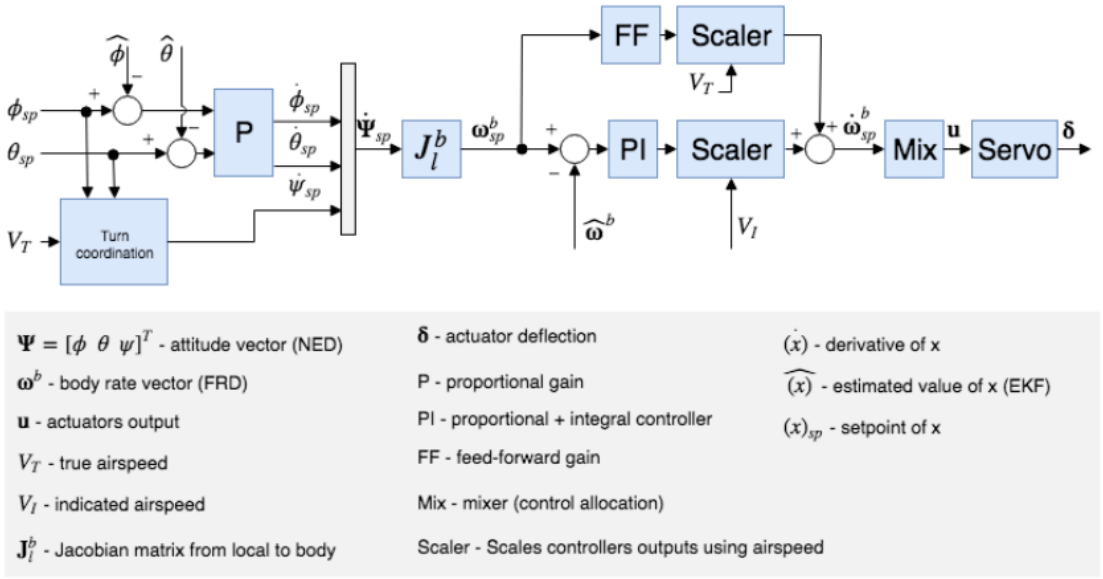

Figura 3.8: Diagrama Attitude Control Teórico. Fuente: [5]

angular es posible calcular las deflexiones de los controles mediante asignación si previamente es conocido el sistema.
Las componentes de los momentos generados en el eje del cuerpo del avión son producidos por las superficies de control y la amortiguación aerodinámica, para compensar esta última se hace uso de la ganancia feedforward que permite mantener una variación constante.
Otro bloque de especial interés es el giros coordinados, este sistema se encarga de obtener una velocidad de guiñada; esta velocidad se encuentra restringida por la realización de giros coordinados que minimizan la aceleración lateral. Al realizar el cálculo de la velocidad de guiñada de esta forma también se contrarrestan efectos adversos de guiñada y se amortigua el modo de balanceo holandés ya que se proporciona una amortiguación direccional adicional.
Este bloque de giros coordinados esta definido por la función:

$$\dot{\psi}_{sp} = \frac{g}{V_T} * tan(\phi_{sp}) * cos(\theta_{sp}) \tag{3.9}$$

Como los términos utilizados pueden no ser conocidos por todo el público lector se procede a realizar una breve definición sobre los términos efectos adversos de guiñada y modo de balanceo holandés:

- El efecto adverso de guiñada es un tendencia natural e indeseada de una aeronave de guiñar en la dirección contraria a la que alabea, esto es debido a que el ala que genera una mayor sustentación en el giro también genera una mayor resistencia inducida, lo que provoca la guiñada; en caso de aeronaves con propulsión por hélices, este efecto puede ser producido por pares inducidos o por la corriente generada.[27]
- El modo de balanceo holandés es una oscilación lateral del avión en la que se combinan movimientos de alabeo y guiñada. [9]

Este software de control requiere de un entorno de ejecución Linux, para este requerimiento se hace uso de un software conocido como Ubuntu.
Ubuntu es una distribución de código abierto basada en Debian que permite establecer un sistema operativo Linux en un entorno de trabajo, este entorno de trabajo utiliza un interfaz gráfica conocida como GNOME. El objetivo de desarrollo de este software se centra en obtener un sistema sencillo, rápido y seguro. Desde su creación se ha convertido en una de las distribuciones de Linux más populares y reconocidas .
Las características de este software son las siguientes:

- Actualizaciones regulares del software.
- Compromiso de calidad.
- Comunidad activa para prestar servicio y soporte.
- Fácil uso del software.
- Interés de internacionalización del software.

Esta distribución de Linux se centra en resolver los problemas existentes en otras distribuciones para crear una en la que adoptar Linux como sistema operativo para nuevos usuarios se realice de la forma lo más sencilla posible [26].
Para el caso del problema a desarrollar se requiere la instalación de Linux en un subsistema de Windows para Linux (WSL) seleccionando la versión Ubuntu 20.04.

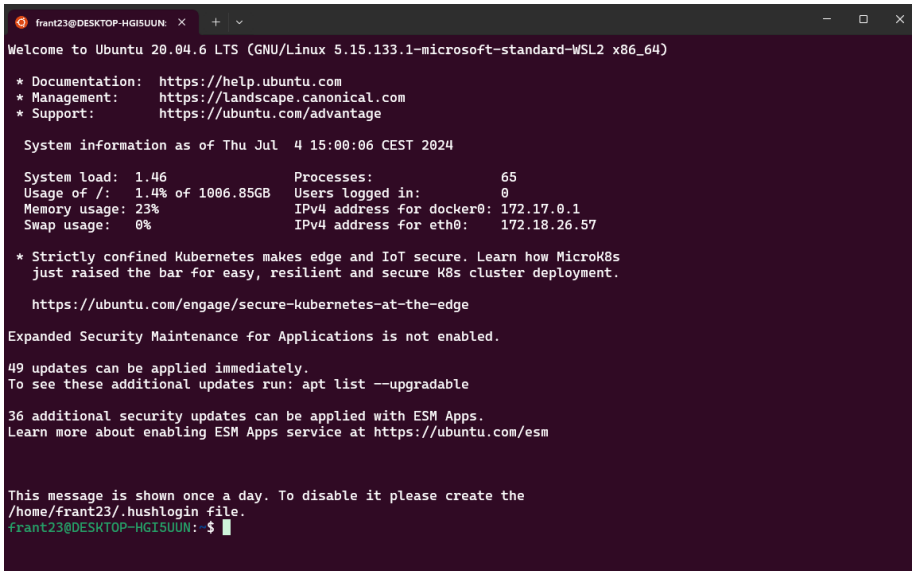

Figura 3.9: Entorno Ubuntu 20.04. Fuente: propia

3.3 Dinámica

Haciendo uso de Simulink es posible simular las actuaciones de un drone de ala fija obteniendo su comportamiento ante una señal de entrada formada por deflexiones de superficies de control. Este sistema conocido como dinámica se encuentra dividido principalmente en dos bloques uno encargado de definir los valores de fuerzas y momentos y otro, que a partir de lo anterior, calcula la actitud de la aeronave. Este segundo bloque se llama *6DOF Rigid body dynamics*, pertenece a la librería Aeroespace Blockset de Simulink, y representa la implementación a través de cuaterniones de las ecuaciones de movimiento de seis grados de libertad respecto a ejes cuerpo. [1]
El subsistema de fuerzas obtiene las fuerzas y momentos generadas en la aeronave al actuar sobre las palancas de control, para ello es necesario implementar todo un sistema de fuerzas formado por un término aerodinámico,uno producido por el motor, gravedad, y fuerza y momento de contacto (en caso de accidente).
Este bloque se encuentra desarrollado en mayor profundidad en la referencia [14] de donde se ha tomado para ser utilizado en esta implementación.
De esta forma el esquema global del sistema de dinámica queda de la siguiente forma:

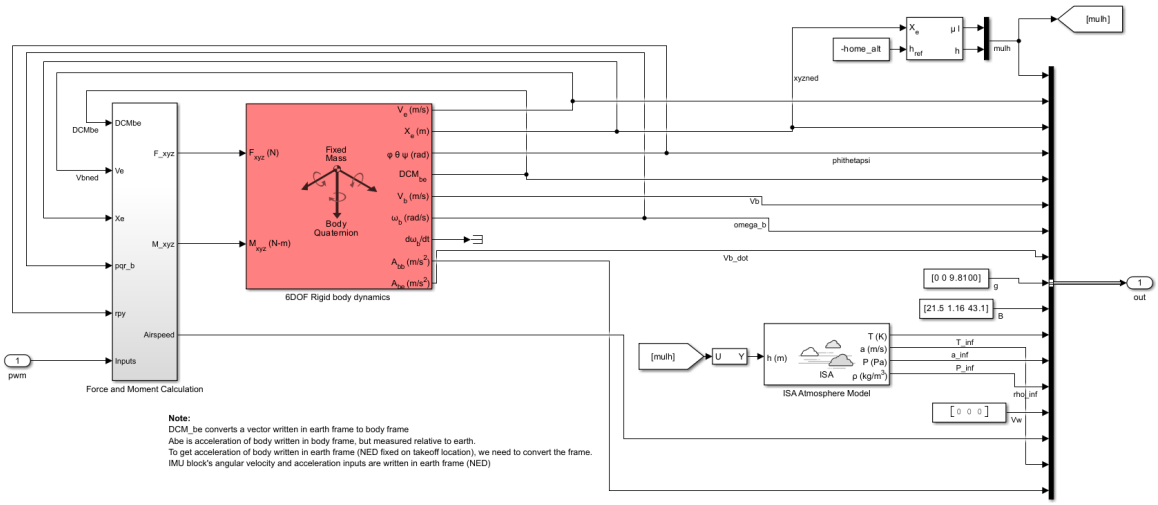

Figura 3.10: Diagrama Dinámica. Fuente: propia

3.4 Estación en tierra

QGroundControl es un software de estación terrestre de código abierto que proporciona control de vuelo completo y configuración del vehículo con controlador PX4 o Ardupilot. Posee un interfaz visual que permite un uso intuitivo de las funcionalidades de la misma. Las principales características de este software son:

- Definición de misiones: permite la creación y edición de misiones con numerosos waypoints, estableciendo en cada uno requisitos de velocidad, altitud y comportamientos específicos de la aeronave.

- Parámetros de la aeronave: permite acceder a todos los parámetros de definición del sistema de auto-piloto, permitiendo el ajuste para optimizar el vuelo de la aeronave.
- Visualización y análisis de datos: este software posee una interfaz de análisis de parámetros de vuelo obtenidos de la aeronave, pudiendo analizar las actuaciones del vehículo.
- Compatibilidad: este sistema es compatible con cualquier vehículo que utilice protocolos MAVLink.

Este software es ampliamente utilizado en proyectos de investigación y en la industria gracias a sus numerosas ventajas comentadas anteriormente [20].

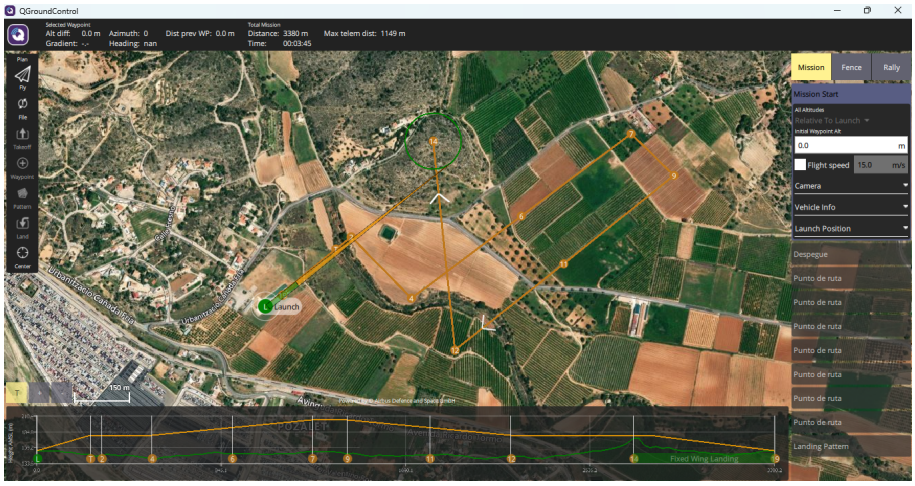

Figura 3.11: Entorno QGroundControl. Fuente: propia

3.5 Software de implementación

Para el problema a desarrollar son necesario dos software base en los que aplicar tanto el sistema de auto-piloto como el algoritmo genético, los seleccionados son los siguientes:

3.5.1 Matlab

Matlab es una plataforma de programación y cálculo numérico utilizada para analizar datos, desarrollar algoritmos, crear modelos y representación gráfica de problemas complejos. Las características del mismo son [7]:

- Posee su propio lenguaje de programación.
- Permite incorporar paquetes adicionales mediante toolboxes.
- Gran capacidad matemática a la hora de cálculos con vectores, matrices y funciones.
- Posibilidad de representaciones complejas en 2D y 3D con sistemas intuitivos.
- Posible integración con otros lenguajes de programación para extender sus funcionalidades.
- Permite cálculo en paralelo permitiendo aprovechar al máximo los núcleos de la CPU.

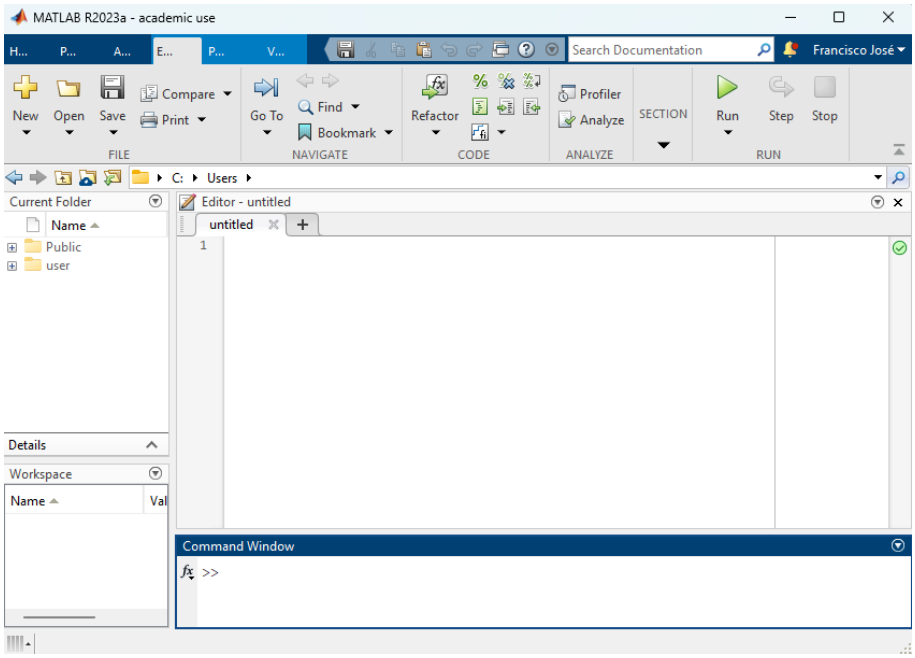

Figura 3.12: Entorno Matlab. Fuente: propia

3.5.2 Simulink

Simulink es un entorno de diagramas de bloque que es utilizado para diseñar sistemas con modelos de simulación multidominio. Centrando la atención en el campo de este proyecto Simulink ofrece numerosas herramientas para el desarrollo de sistemas de control [15]:

- Proporciona un entorno de diagrama de bloques en el que se puede modelar dinámica de planta, diseñar algoritmos de control y realizar simulaciones de lazo cerrado.
- Herramientas para análisis de sobre-impulsos, tiempos de subida, margen de fase, margen de ganancia y otras muchas características tanto en dominio de tiempo como de frecuencia.
- Permite obtener el lugar de las raíces, diagramas de Bode, LQR, LQG, control robusto y control predictivo.
- Proporciona herramientas para auto ajuste de PID.
- Algoritmos de Reinforment Learning y diversos más basados en datos e inteligencia artificial.
- Herramientas de análisis de los resultados.
- Integración con Matlab, lo que permite tanto utilizar datos definidos en Matlab en los diagramas de bloques, como realizar un pos-procesado de los resultados con las herramientas gráficas de Matlab.

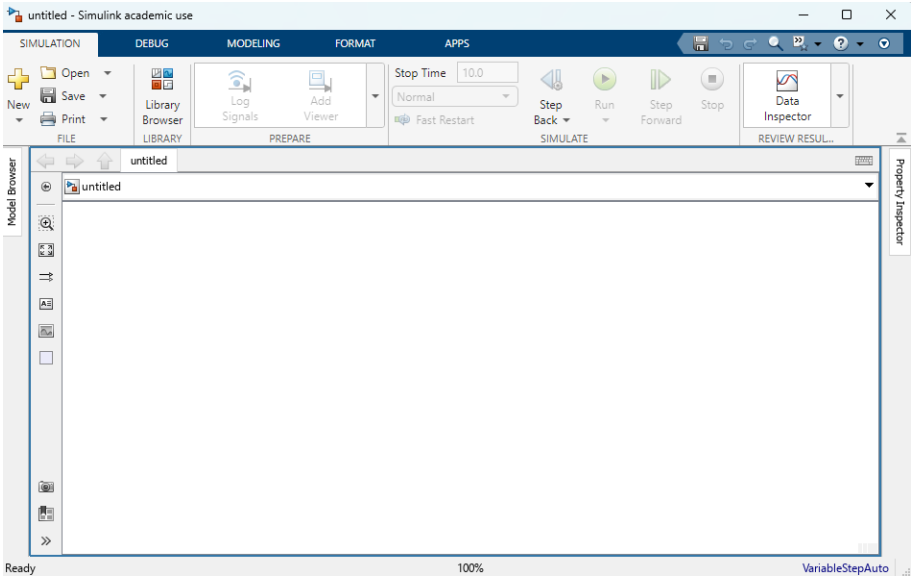

Figura 3.13: Entorno Simulink. Fuente: propia

3.6 Software in the loop

Para llevar a cabo la simulación del modelo de PX4 son necesarios tres componentes que están interconectados y permiten simular trayectorias, estos sistemas permiten determinar el sistema conocido como SITL:

- Código base de controladores de PX4 que es ejecutado en un sistema operativo Linux a través de Ubuntu 20.04.
- Estación de control en tierra para definir la misión, para este requerimiento se hace uso de QGroundControl.
- Sistema de dinámica de Simulink para obtener como actúa la aeronave.

El esquema de conexión es similar a:
El sistema implementado en este proyecto la placa Pixhawk 4 representa al controlador PX4

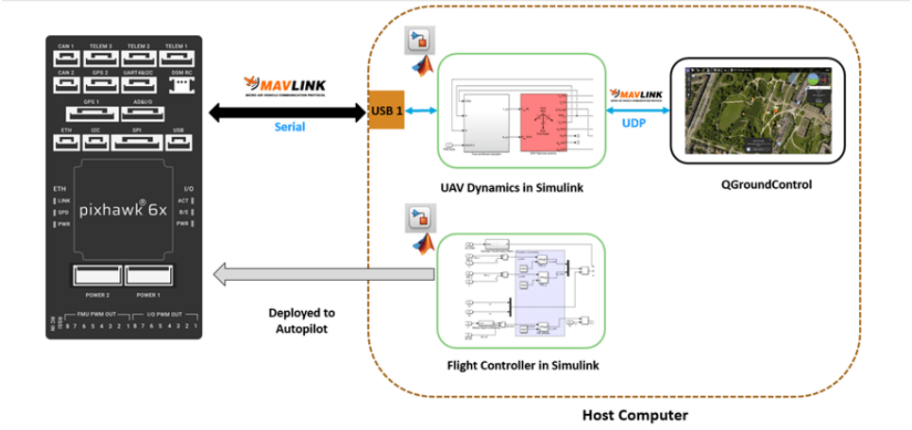

Figura 3.14: Esquema conexión PX4. Fuente: [16]

iniciado en Ubuntu, el simulador es el esquema de dinámica de Simulink y la estación en tierra se mantiene QGroundControl.
Todos estos sistemas requieren de estar conectados entre si, esa conexión se realiza mediante un protocolo de comunicación conocido como MAVlink.
Antes de proceder con la simulación es necesario conocer en que consiste un protocolo de comunicación MAVlink; MAVlink es un protocolo de comunicación ligero diseñado para la comunicación con drones desde sistemas de estación terrestre o entre sus componentes abordo, es un sistema de código abierto comenzado a desarrollar en 2009.
El protocolo de comunicación utilizado es basa en un diseño híbrido de publish-subscribe y point-to-point; los flujos de datos son publicados como topics mientras que los protocolos de misión se retransmiten por el método point-to-point. En conjunto estos mensajes se definen como archivos XML en los que se pueden recibir o enviar datos como comandos de vuelo, telemetría o parámetros de configuración de la aeronave.
Las principales características que hacen que este software sea ampliamente utilizado en el mundo de los drones son:

- Sistema muy eficiente ya que trabaja en entornos de bajo ancho de banda.
- Fiable, su utilización ha ido aumentando desde su inicio y posee métodos para determinar fallos en las comunicaciones.
- Permite una gran diversidad de lenguajes de programación y soporte multiplataforma.
- Permite una comunicación robusta tanto con estaciones terrestres como con componentes internos del drone.

Una vez conocida la definición de este protocolo ya se puede continuar con las partes del sistema de simulación.
Para comenzar es necesario instalar el controlador PX4_HyDrone en WSL2 Ubuntu 20.04, el código de dicho controlador se encuentra en [23] donde se establecen los pasos a seguir, estos pasos son:

- Abrir WSL2 Ubuntu 20.04.
- Clonar el repositorio github.
- Instalar las dependencias de Linux.
- Obtener el repositorio PX4_HyDrone con el PX4_Autopilot original.
- Definir la dirección IP de Windows donde se ejecutará Simulink.
- Compilar el modelo de ala fija para STIL.

Con esto el sistema de PX4 queda iniciado con:

Tras esto el sistema de controlador PX4 esta a la espera de conexión con el diagrama de dinámica de Simulink.
Para esta simulación se hace uso de un entorno SITL (software-in-the-loop) compatible con la comunicación MAVLink, este modulo es utilizado por PX4 para conectar con el simulador (en este caso Simulink) por el puerto local TCP 4560 e intercambiar información.

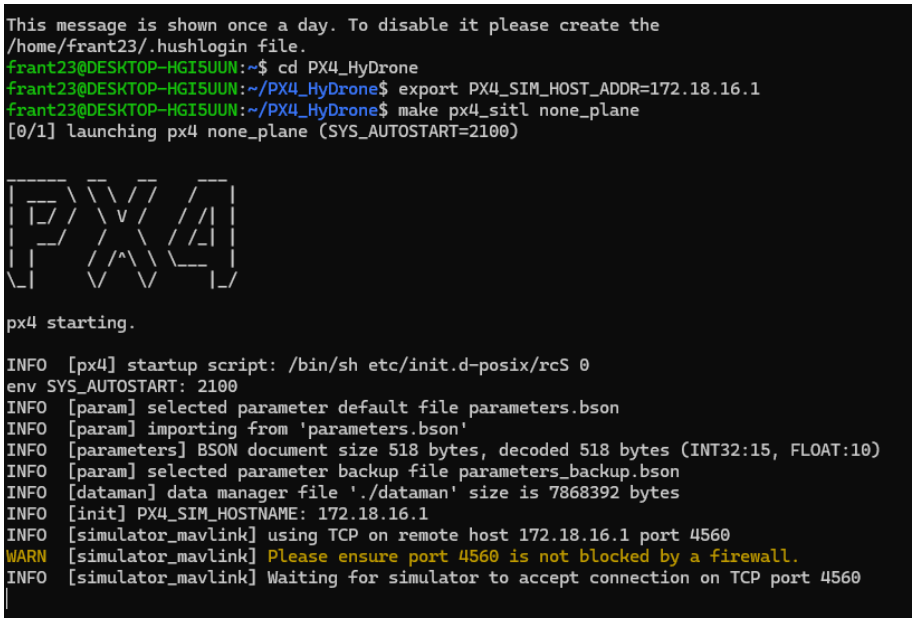

Figura 3.15: Interfaz PX4 (Ubuntu 20.04). Fuente: propia

Por otra lado, también es necesario conectar con la estación en tierra QGroundControl, para este conexión se utiliza también un puerto remoto UDP, en este caso el 18570.
Teniendo una visión global del sistema las conexiones son las siguientes:

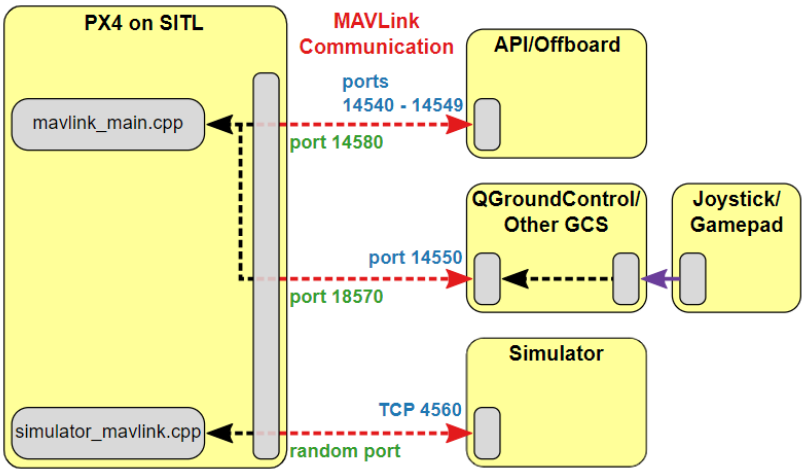

Figura 3.16: Esquema Conexión SITL. Fuente: [11]

El sistema de simulación de la dinámica de la aeronave se modela mediante el mismo bloque de dinámica del modelo implementado, pero es necesario incorporar una serie de modificaciones para poder realizar la conexión mediante procedimiento MAVLink con PX4 y QGroundControl.

El bloque que se encarga de la conexión es el *Pixhawk SITL*, las entradas es necesario transformarlas de un tiempo continuo a un tiempo discreto, al iguala que las salidas que van al sistema de dinámica se transforman a continuo.

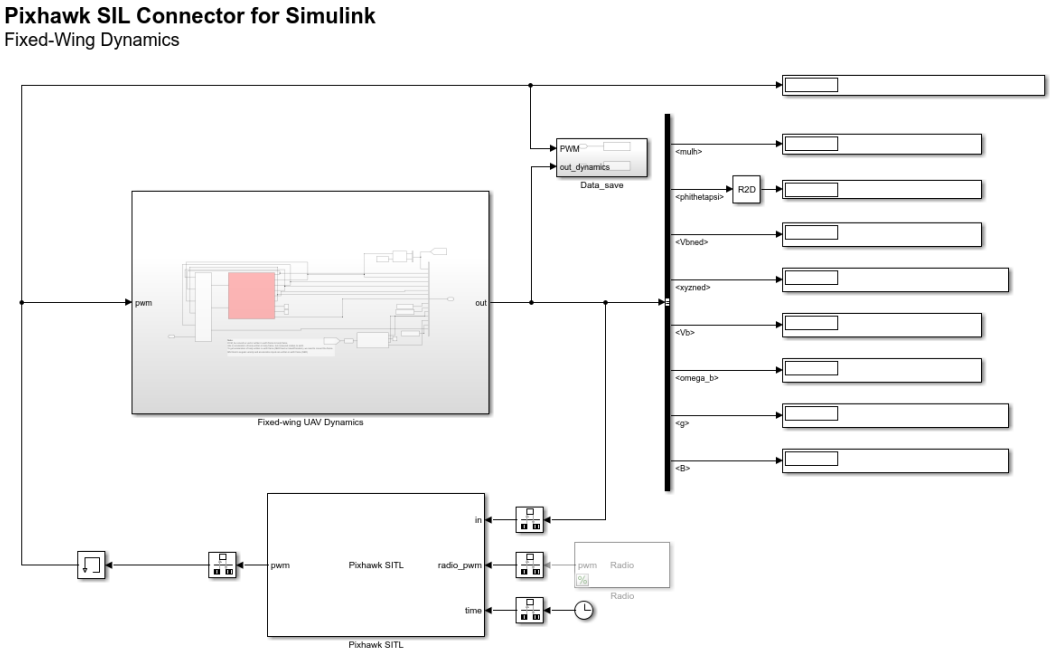

Figura 3.17: Diagrama Pixhak SIL Connector y Dinámica. Fuente: propia

El bloque *Pixhawk SITL* es un software en bucle que permite ejecutar el vuelo de una aeronave sin hacer uso de ningún hardware. Para generar el diagrama interno de dicho bloque es necesario seguir los siguientes pasos [24]:

- Instalar las librerias requeridas de Matlab-Simulink.
- Descargar "pixhawk_sil_connector.cpp", "make.mz ïncludes.zip".
- Ejecutar "make.m"para generar los archivos necesarios para la conexión .

El esquema interno de este bloque es el siguiente:

El sistema restante del diagrama es QGroundControl, este sistema es el encargado de definir la trayectoria a seguir por la aeronave y de representar visualmente la respuesta de la aeronave. La instalación de dicho sistema es muy sencilla a través de la página web del software, a la hora de realizar la configuración para la conexión es necesario configurar un *Comm Link*, estableciendo el puerto a utilizar (18570) y una dirección de servidor que se refiere a la IP definida en PX4.

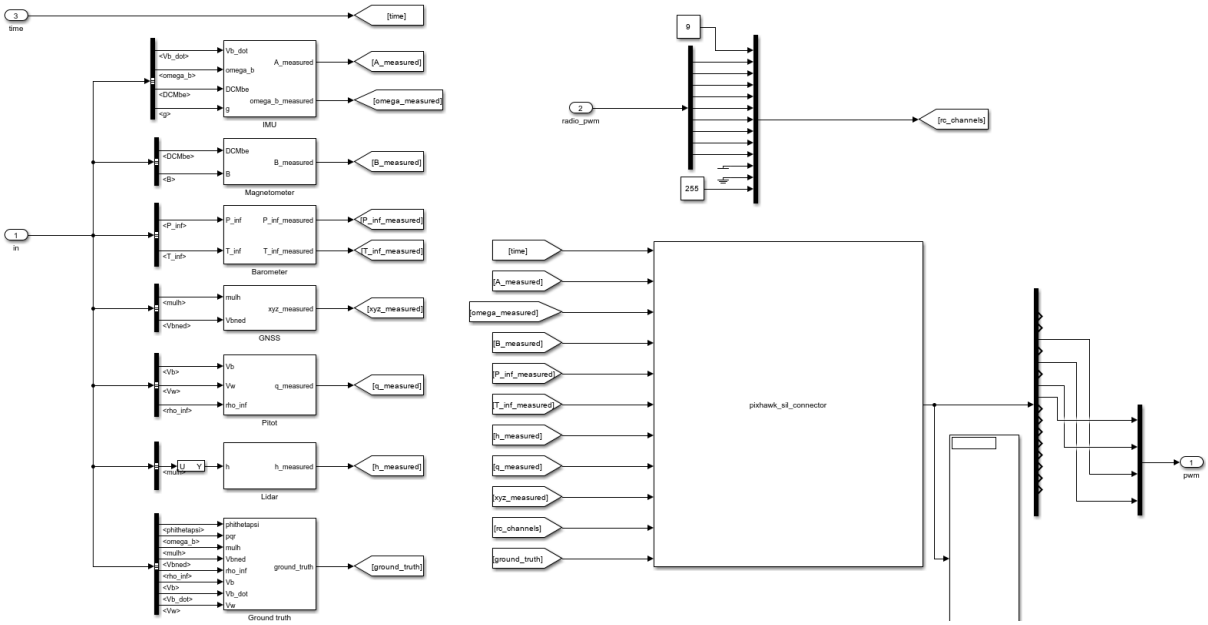

Figura 3.18: Diagrama Pixhak SIL Connector. Fuente: propia

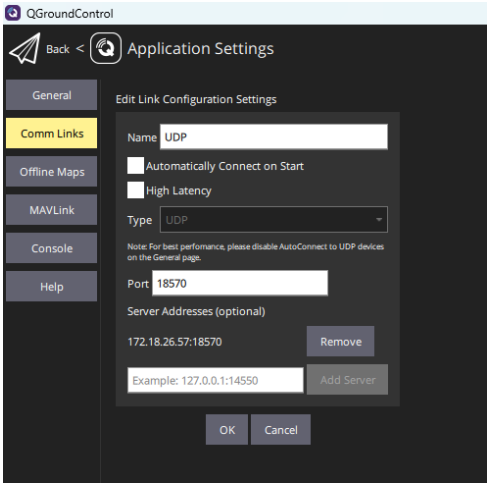

Figura 3.19: Conexión QGroundControl. Fuente: propia

Capítulo 4

Condicionantes y limitaciones del proyecto

Para la realización de este proyecto es necesario tener en cuenta una serie de limitaciones y condicionantes que caracterizan la forma de su desarrollo y los resultados obtenidos. Estos factores se pueden dividir en diversos bloques explicados a continuación.

4.1 Software

El proyecto establece su desarrollo principalmente en el software antes mencionado Simulink, esto es uno de los mayores condicionantes ya que se ha de realizar una adaptación del código del controlador PX4 a diagramas de bloques o código de Matlab. La utilización de este programa se realiza debido a su uso intuitivo, familiriarización durante el grado cursado en diferentes asignaturas y su potencia a la hora de realizar simulaciones de este tipo. Otro factor para la elección de este entorno de simulación reside en el uso de un algoritmo genético desarrollado en Matlab como sistema de auto-tuning por lo que si la simulación del control es en Simulink existe una rápida y eficaz conexión entre ambos.
Otro condicionante relacionado con software es el uso de PX4 como auto-piloto de referencia para el desarrollo del esquema de Simulink, en el mercado hay numerosos software de auto piloto como el caso de Ardupilot o Librepilot. La elección de este sistema de control reside en la gran versatilidad que proporciona este software junto con la posibilidad de escalar este proyecto y ser utilizado en un futuro en proyectos universitarios con dicho sistema.

4.2 Dinámica

El sistema de estimación de dinámica de la aeronave también establece una serie de condicionantes, estas limitaciones residen en gran medida en la estructura de simulación del sistema, estableciendo un SITL. También establece limitaciones en la aeronave a utilizar en el modelo, siendo desarrollado el diseño de la dinámica para la aeronave comentada con anterioridad.

Capítulo 5

Soluciones alternativas

Al inicio del desarrollo del proyecto, se platean diversas soluciones para afrontar el problema de estudio, en este caso las soluciones planteadas son las siguientes:

5.1 Software

Desde el punto de vista del software a utilizar como principal se plantean dos opciones para resolver el problema de estudio, estas son las siguientes:

5.1.1 Implementación en Matlab-Simulink

La primera de estas dos soluciones consiste en una implementación total en Simulink de la parte de los controladores de PX4.
Comentando en primer lugar la parte del desarrollo del control, con esta solución se plantea la transformación del código de control del sistema de PX4 a diagramas de bloques de Simulink. Este sistema de control de PX4 esta formado, como ha sido comentado en apartados anteriores, por dos subsistemas principales: el sistema de control de posición, encargado de establecer unos ángulos de actitud de la aeronave de referencia a partir de unos requerimientos específicos de la misión a desarrollar (posición waypoints y velocidad de referencia); el segundo subsistema corresponde al control de actitud que permite obtener las deflexiones de las superficies de control a partir de unos ángulos de actitud de referencia. Estos dos subsistemas se implementarían haciendo uso de las herramientas suministradas por Simulink. De esta forma se obtendría un único software de simulación al conectar el sistema implementado en Simulink de PX4 con el diagrama de estimación de dinámica, ya desarrollado en la misma plataforma. Este sistema de control implementado en Simulink requiere de una validación para verificar si es equivalente al control ofrecido por el sistema PX4, para esta verificación se comparan las soluciones obtenidas por la simulación SITL de PX4 y de la implementación en Simulink.
El esquema conceptual a seguir para la implementación en Simulink del control PX4 es el siguiente:
Comentado el desarrollo a seguir en la segunda parte del proyecto, el sistema de auto-tuning

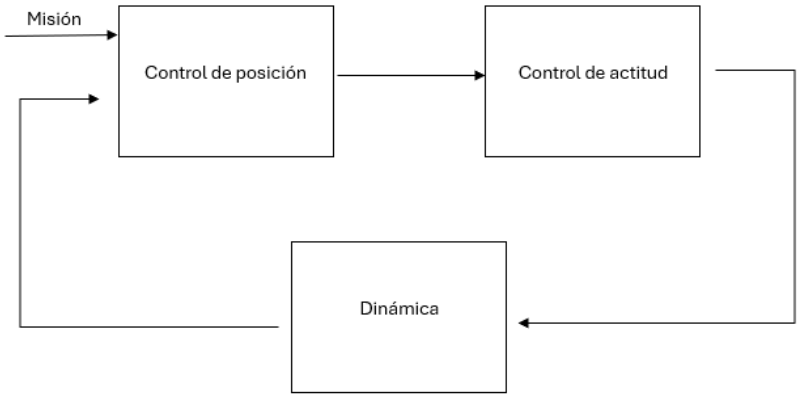

Figura 5.1: Diagrama conceptual implementacion PX4 en Simulink. Fuente: propia

para el sistema de control de actitud de la aeronave se puede llevar a cabo tanto por la herramienta Simulink como por Matlab. Para ambos casos existen numerosos Toolbox ya existentes que permiten una rápida y sencilla implementación de estos sistemas de algoritmos genéticos o Reinforment Learning, este tipo de implementación es posible de realizar tanto en diagrama de bloques de Simulink, como a través de un scrip de Matlab.

5.1.2 Modificación código PX4

La solución de modificación del código de PX4 consiste realizar una adaptación del código base de PX4 para introducir los algoritmos de auto-ajuste dentro del mismo. Para el desarrollo de esta solución es necesario eliminar del sistema de control de actitud de PX4 los parámetros de control, junto con los sistema de auto-tuning existentes en el propio sistema para evitar de esta forma interferencias que provoquen errores mayores a la hora de la optimización global. La implementación de este sistema de auto-tuning nuevo tiene que ser programado en el mismo lenguaje de programación ya existente en el controlador PX4, esto puede llegar a suponer un gran reto si la persona no esta muy familiarizada con ese tipo de lenguaje de programación o con el propio controlador PX4. Junto con todo lo desarrollado anteriormente, también es requerido comprobar si es posible realizar una desconexión de los sistemas de control dejando únicamente operativo el control de actitud (bloque a ajustar) para evitar de esta forma interferencias del control de posición en este ajuste y obtener la mejor la mejor solución posible.
Toda la implementación comentada con anterioridad requiere de su respectiva validación para verificar que el sistema sigue las referencias establecidas comprobando que las modificaciones e incorporaciones al código no producen errores de control.
Tras la correcta implementación de todos los sistemas de auto-ajuste con el código base y la certificación de su correcto funcionamiento, se prosigue con una fase de optimización global del sistema para que las nuevas funciones incorporadas realicen su propósito de obtener los parámetros de control necesarios.
En las simulaciones de PX4 (SITL) es necesario conectar el sistema con una estación en tierra (QGroundControl) que es la encarga de definir la trayectoria a recorrer y un sistema de Simulink que se encarga de la dinámica de la aeronave teniendo como entradas los valores de las superficies de control, y como salida la actitud de la aeronave. El esquema de simulación para este caso es

de mayor complejidad ya que se requieren un mayor número de sistemas para la simulación de trayectorias.

5.2 Ajuste parámetros

Por parte del ajuste de parámetros de control, las soluciones planteadas son dos, utilizar algoritmos genéticos o realizar el auto-ajuste mediante un proceso conocido como Reinforment Learning, a continuación se van a desarrollar ambos casos:

5.2.1 Algoritmos genéticos

Un algoritmo genético es un método de resolución de problemas de optimización tomando como base el proceso de evolución natural.El algoritmo genético trata de replicar las leyes de selección natural, el proceso consiste en la modificación repetida de una población de soluciones seleccionando aleatoriamente individuos para utilizarlos como padres y producir hijos de la siguiente generación, esto permite que la población evolucione hasta la solución óptima [18].
El algoritmo genético trata de replicar las leyes de selección natural. La población sobre la que opera un algoritmo genético es el conjunto de posibles soluciones del problema, esta población esta formada por individuos, siendo cada uno una posible solución; cada individuo esta definido por unas características particulares que hace que sea distinto de los demás, si dos individuos son iguales poseen las mimas características. El entorno esta definido por una función objetivo la cual será utilizada por el algoritmo genético para determinar que individuos están mejor adaptados al entorno y sobreviven.
Una vez establecidas los requerimientos del entorno y una población inicial la evolución se simula aplicando lo que se conocen como operadores genéticos, que van modificando los individuos de la población inicial para intentar conseguir la mejor adaptación al medio, los métodos más utilizados para esta adaptación son:

- Selección: se seleccionan los individuos mejor adaptados al medio, esta selección se realiza en función del valor obtenido por cada miembro para la función objetivo. Esta operación tiene como objetivo eliminar los individuos menor adaptados al medios. Hay diversos métodos de selección pero lo más utilizados están basado en una ruleta.
- Cruce: combinación de individuos de una población para obtener unos nuevos que combinan las características de los dos padres. Con esta operación se pretenden conseguir individuos con características mejores de las existentes. No todos los individuos se cruzan si no que esto depende de una probabilidad definida como probabilidad de cruce.
- Mutación: operación que genera un nuevo individuo con nuevas características a partir de modificar aleatoriamente las ya existentes. Esto permite obtener individuos con características no vistas hasta el momento en la evolución. Al igual que la operación de cruce, esta no ocurre en todos los individuos si no que viene definida por una probabilidad.

Con estos tres métodos, entre otros disponibles, se obtiene una evolución de la población hacia la mayor adaptación al entorno (solución óptima).
La iteración del sistema finaliza dependiendo de dos parámetros, o por número de iteraciones

(criterio de tiempo) o por calidad de la solución (mínimo de la función objetivo) [2]. Esta im-

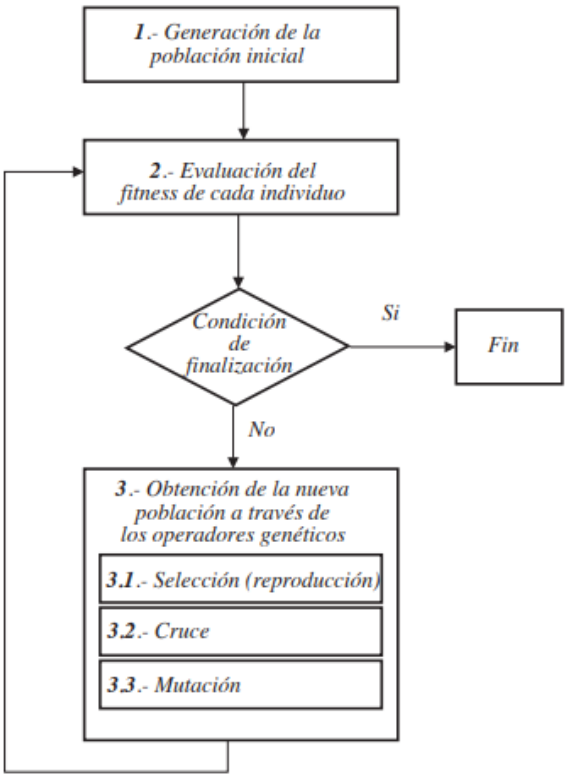

Figura 5.2: Diagrama flujo algoritmo genético. Fuente: [2]

plementación tiene tres formas de realizarla, incorporando el algoritmo genético al propio código del controlador de PX4, mediante un desarrollo a partir de diagramas de bloques de Simulink o a través de funciones de Matlab. Estas tres soluciones ya han sido comentadas con anterioridad en el apartado de Software. Dentro de Simulink existen una serie de bloques específicos que facilitan la implementación de este tipo de optimización, estos están recogidos en Global Optimization Toolbox, también existen Toolbox que establecen algoritmos genéticos en scrips de Matlab.

5.2.2 Reinforcement Learning

Al igual que en el caso anterior, se va ha comenzar con una breve explicación de en que consiste este método de optimización. Reinforcement Learning es una técnica de Machine Learning en la que un agente informático realiza numerosas iteraciones de prueba y error en un entorno dinámico con el objetivo de aprender una tarea determinada, este tipo de aprendizaje no requiere de datos estáticos ya que opera en un entorno dinámico en el que los datos y experiencias se obtienen a partir de diversa interacciones, esto permite diferenciar este tipo de aprendizaje de otros como el supervisado y el no supervisado en los que es necesario suministrar y preprocesar los datos antes de entrenar el modelo.

Esto permite que un modelo de Reinforment Learning aprenda una tarea sin necesidad de supervisión humana, si el incentivo establecido es el adecuado.

El flujo de trabajo de esta optimización consiste en los siguientes pasos:

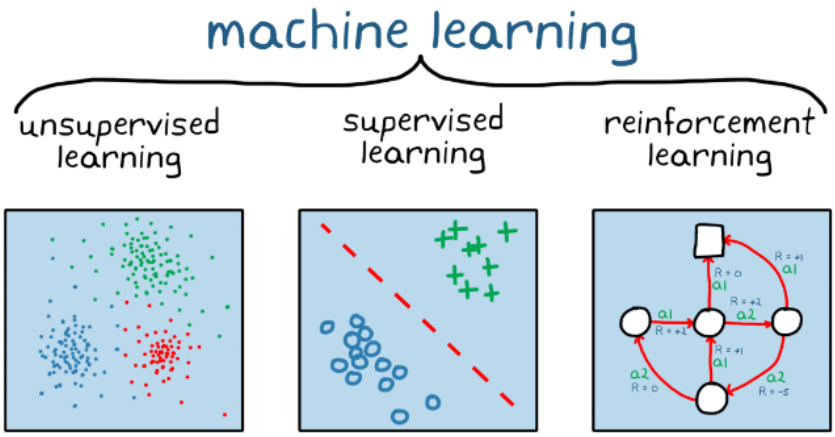

Figura 5.3: Machine learning. Fuente: [17]

1. Definir un entorno en el que el agente actué incluyendo la interfaz entre el agente y el entorno; puede ser un sistema real o un modelo de simulación, este último establece un primer paso más seguro que el sistema real.
2. Definir una señal de recompensa que permitirá guiar al agente para cumplir los objetivos establecidos, este sistema puede ser complicado y requiere de varias iteraciones para lograr el resultado deseado.
3. Crear el agente que se encarga de llevar a cabo el proceso de optimización, para ello es necesario definir su política (redes neuronales o tablas de búsqueda) y seleccionar un algoritmo de entrenamiento correcto, este ultimo es determinante ya que la solución dependerá de el tipo de algoritmo seleccionado. La mayoría de los algoritmos modernos depende de redes neuronales ya que permiten obtener buenas soluciones frente a problemas complejos.
4. Entrenamiento del agente hasta lograr ajustar la política. Es necesario validar la política entrenada revisando cada uno de los bloques definidos anteriormente. El tiempo de entrenamiento de este tipo de sistemas varía mucho en función del problema a resolver, para aplicaciones complejas se utiliza la paralelización en múltiples CPU, GPU y clusters que permiten acelerar el proceso.
5. Representación de la política entrenada.

El desarrollo de este tipo de sistemas es un proceso iterativo, ya que las decisiones y resultados de etapas anteriores condicionan las soluciones finales del sistema.
Los modelos de Reinforment Learning tienen numerosas aplicaciones, principalmente están enfocados a sistemas de control avanzados, conducción autónoma, robótica, calibración, ...
Al igual que para el caso de los algoritmos genéticos, para Reinforment Learning existe un toolbox de Simulink que permite facilitar la implementación de este tipo de optimización a través de diversos bloques; esta implementación también se puede realizar mediante código de Matlab o cualquier otro tipo de código de programación [17].

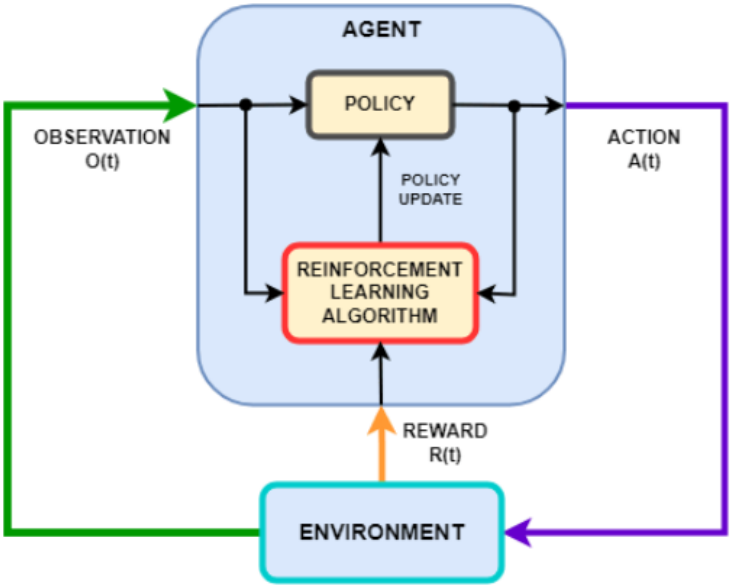

Figura 5.4: Reinforcement Learning. Fuente: [21]

Capítulo 6

Implementación PX4 en Simulink

Una vez se ha establecido un contexto inicial sobre el problema, las herramientas utilizadas y limitaciones del mismo; el siguiente paso a seguir es el desarrollo del problema.
De entre las posibles soluciones disponibles a la hora de seleccionar un software a utilizar como base de simulación del problema, se opta por la opción de Simulink, esto permite la obtención de un sistema de simulación más simple incorporado en su totalidad en el mismo software, de esta forma se facilita en futuros trabajos la implementación de los sistemas de auto-tuning. La primera parte de desarrollo del proyecto, como se ha comentado anteriormente, consiste en la implementación del sistema de auto-piloto del controlador PX4.

6.1 Controladores

Los controladores son los encargados de establecer las acciones llevadas a cabo por el auto-piloto de la aeronave, este sistema se puede dividir principalmente en dos partes que son el controlados de actitud y el controlador de posición.
Para llevar a cabo una correcta implementación de los diferentes controladores es necesario aplicarlos en un tiempo discreto, esto es debido a que en PX4 los controladores no se ejecutan durante un tiempo continuo, si no que tienen una determinada frecuencia de ejecución. Para obtener un modelo lo más similar es necesario que esta implementación sea en discreto, la frecuencia seleccionada de ejecución es: $frequency = 0{,}012[hz]$. Para llevar a cabo esta condición se hace uso de los bloques *Zero_ Order_ Hold* que permite transformar la señal de entrada del bloque de dinámica de continua a discreta, y *First_ Order_ Hold* que realiza el cambio a la inversa para que la señal de las superficies de control entre al bloque de dinámica en tiempo continuo.

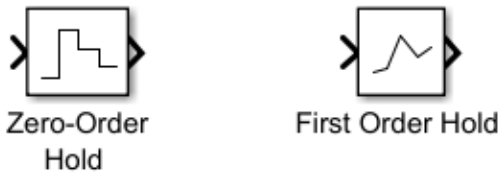

Figura 6.1: Bloques cambio a tiempo discreto y continuo. Fuente: propia

6.1.1 Controlador de posición

El controlador de posición se encarga de obtener unos ángulos de actitud de referencia para la aeronave y el empuje necesario a partir de unas condiciones predefinidas de la misión a seguir, estas condiciones viene establecidas por la definición de la trayectoria.
Este controlador de posición se divide en tres bloques:

- Seguimiento de waypoints
  Este primer subsistema, dentro del controlador de posición, es el encargado de obtener unos valores de referencia (rumbo, velocidad y altitud) a partir la definición de una trayectoria mediante una serie de waypoints, estos puntos son definidos por sus coordenadas en un sistema de referencia NED (sistema de coordenadas centrado en un punto de la superficie terrestre) el cual se encuentra centrado en el punto de origen del vuelo, para el caso de vuelo a desarrollar, este punto pertenece a la pista del Aeroclub RC Valencia en Cheste.
  Para la determinación de la trayectoria se hace uso del software QGroundControl, el cual permite determinar de manera visual una determinada trayectoria y ofrece la latitud y longitud de cada uno de los puntos, estos valores se encuentran en un sistema de referencia LLA (sistema de coordenadas Geodésicas). Para poder llevar a cabo la implementación es necesario transformar las coordenadas de los waypoints de LLA a NED, para ello se hace uso de la función *lla*2*ned* definida en Matlab, tomando como punto de referencia el comentado en el párrafo anterior.
  Una vez la trayectoria ha sido definida, se prosigue realizando la implementación del sistema en Simulink, para ello es necesario la utilización de dos bloques principales:
  El primero de ellos es el bloque *Path Manager*, este bloque es utilizado para cambiar secuencialmente entre los puntos especificados en la misión cuando se establece que se ha alcanzado el punto objetivo. Las entradas de este bloque interesantes para el problema a desarrollar son:

  - *Pose* que establece la posición de la aeronave y su rumbo.
  - *MissionData* esta entrada se define como un bus de datos compuesto por modo, posición y parámetros de cada waypoint, en función del modo que se selecciona, los parámetros de cada waypoint variaran, los modos disponibles son: Takeoff, Waypoint, Orbit, Land, Return to launch position y Custom.
  - *isModeDone* a través de esta entrada se determina cuando se realiza el cambio de waypoints.

- *MissionCmd* se utiliza en caso de ser necesario un cambio de misión, las posibles opciones son las siguientes: ejecutar la misión del primer al último punto (tipo utilizado para este problema), quedarse en el punto actuar orbitando, repetir la misión al alcanzar el punto final o ejecutar el modo Returno to laund position.
- *Home* establece la posición del punto de casa con un vector de tres coordenadas en NED.

Una vez comentadas las entradas de este bloque es necesario explicar que salida proporciona este bloque:

- *MissionParams* proporciona un bus de datos de $2x1$ en el cual la primera cadena de datos corresponde al current waypoint y la segunda cadena correspondiente al previus waypoint, cada una de las cadenas esta formada por el modo, la posición y los parámetros de cada waypoint; en función del modo de la misión el modo y los parámetros de los waypoints variara.

Para tener más información de la estructura de definición de waypoints al igual que sobre este bloque se puede encontrar en la ayuda de Matlab.

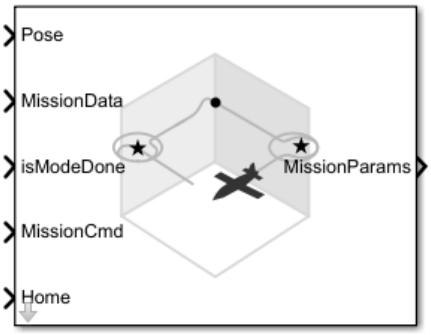

Figura 6.2: Bloque Path Manager. Fuente: propia

El segundo bloque importante de este subsistema es el *Waypoint Follower*, este bloque es el encargado de a partir de los waypoints de salida del bloque *Path Manager* y una distancia de anticipación (lookahead distance), proveniente del controlador L1, calcular unos valores de referencia para que la aeronave siga la trayectoria que une ambos puntos. Los parámetros de entrada de este bloque son los siguientes:

- *Pose* establece la posición y rumbo actual de vehículo.
- *Waypoints* establece los waypoints de la trayectoria a seguir, estos puntos están definidos por un vector de 3 componentes. Esta entrada se obtiene a partir de la salida del bloque *Path Manager*.
- *LookaheadDistance* establece la distancia de anticipación de los puntos de la trayectoria. Este valor proviene del controlador L1.

- *Transition Radius* este parámetro se define internamente en el bloque pero es determinante en el cálculo; determina un radio alrededor de los waypoints, cuando la aeronave se encuentra dentro de ese radio el bloque pasa al siguiente waypoint.

Ya comentadas las entradas se prosigue con las variables de salida del bloque:

- *LookaheadPoint* se obtiene el próximo punto de referencia a seguir por la trayectoria de la aeronave, este parámetro es utilizado solo con fines de comprobación de la trayectoria.
- *DesiredCourse* se define el rumbo de referencia a seguir para conseguir la trayectoria deseada, este valor esta comprendido entre $[-pi, pi]$.
- *DesiredYaw* obteniendo un ángulo de guiñada deseado, este valor coincide con el *DesiredCourse*.
- *Status* devuelve 0 o 1, cuando se alcanza un waypoint se devuelve 1 en un pulso para indicar que se pase al siguiente, cuando se han alcanzado todos los waypoints definidos se devuelve un 1 constante.

Las demás salidas son utilizadas para otro tipo de aplicaciones que para este caso no son de interés.
Más información sobre este bloque se puede encontrar en la ayuda de Matlab.

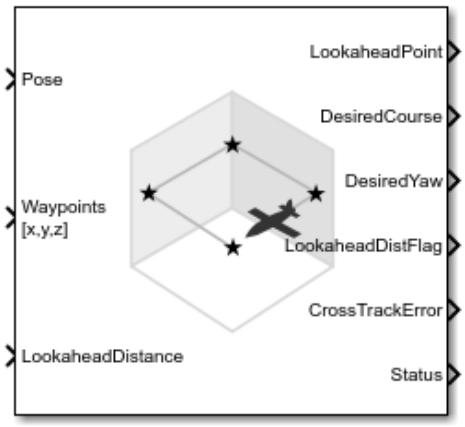

Figura 6.3: Bloque Waypoint Follower. Fuente: propia

Una vez comentados los dos bloques anteriores, se prosigue a mostrar y explicar la implementación realizada para obtener las variables de referencia necesarias:

De esta implementación es necesario explicar los bloques auxiliares a los anteriormente definidos:

- Función *waypoints_ alt*: este bloque permite realizar una asignación de la posición de los waypoints de entrada en una sola matriz de forma $wps = [fromWP', toWP']$ y definir la altitud de referencia como $alt = -toWP(3)$. Para obtener las entradas de este bloque hay que modificar las salidas del bloque *Path Manager* seleccionando datos requeridos (en este caso la posición).

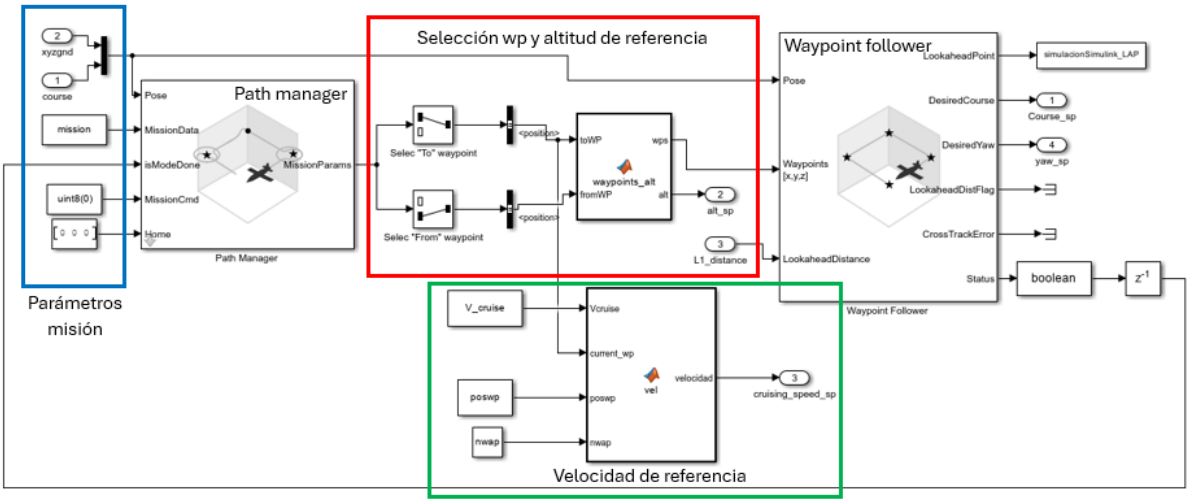

Figura 6.4: Bloque Seguimiento Waypoints. Fuente: propia

- Función *vel*: este bloque se utiliza para establecer la velocidad de referencia en cada segmento de la trayectoria, para eso se define un bucle que recorre el numero de waypoints existentes y termina cuando la posición de la aeronave sea igual a la del waypoint actual, esa posición es la utilizada para establece que la variable de salida de velocidad sea igual a la velocidad de referencia de ese punto.
- Conexión *Waypoint follower-Path Manager*: a la hora de realizar esta conexión es necesario convertir el tipo de dato de salida del bloque *Waypoint follower* para poder ser admitido por el bloque *Path Manager* a su vez para evitar un error de bucle algebraico hay que introducir un delay, para este caso el valor de este retraso es de $\frac{1}{z}$.

Una vez comentadas todas las partes de este diagrama se puede comprender correctamente su funcionamiento, para obtener el código de completo empleado en las funciones se puede encontrar en el anexo del documento.

- Controlador L1

  Una vez obtenidas las variables principales que definen la trayectoria a seguir, se comienza con la definición de los sistemas que permiten obtener los ángulos de actitud de referencia de la aeronave.

  El sistema de controlador L1 permite obtener un ángulo de alabeo de referencia en función de un rumbo a seguir, el rumbo de la aeronave estimado y la velocidad del vehículo, estas variables son dependientes del tiempo, pero a su vez el sistema depende de otras variables constantes.

  Antes de definir las ecuaciones que rigen este tipo de controlador, es interesante conocer los valores de las constantes, estas se van a expresar en la siguiente tabla:

  Los parámetros anteriores determinan: *L1_period* el tiempo de giro de la aeronave y *L1_damping* establece el amortiguamiento de la trayectoria.

  Una vez conocidas las variables constantes, se puede proceder a definir el conjunto de ecuaciones que rigen este sistema:

Variable	Valor	Unidades
L1_period	25	s
L1_damping	0.7	-

Tabla 6.1: Constantes Controlador L1

$$L1\_ratio = \frac{L1\_period * L1\_damping}{pi} \tag{6.1}$$

$$L1\_distance = L1\_ratio * groundspeed \tag{6.2}$$

La variable *L1_ distance* determina una distancia de anticipación ante el siguiente punto a seguir de la trayectoria, esto determina, utilizando el bloque anterior *Waypoint Follower*, los puntos de anticipación de la trayectoria.

$$K\_L1 = L1\_damping^2 * 4 \tag{6.3}$$

$$eta = course\_sp - course \tag{6.4}$$

Eta es el ángulo que existe entre el rumbo de referencia y el que posee la aeronave en un instante determinado, este ángulo posee un rango de valores de $[-pi, pi]$. Los valores tanto de rumbo como de rumbo de referencia tienen un rango de valores igual que el ángulo de diferencia entre ambos , para evitar un cálculo erróneo al realizar la resta entre ambos en el caso en el que tengan signos diferentes, es necesario implementar una función que calcule el resto de $\frac{eta}{2*pi}$, tras esto para cumplir con el rango de valores de eta es necesario aplicar la siguiente condición que si $eta > pi$ entonces $eta = eta - 2 * pi$ de esta forma se establece que el rango es el requerido. El valor de eta, a su vez se restringe entre $[-\frac{-pi}{2}, \frac{pi}{2}]$ evitando de esta forma aceleraciones laterales muy grandes que llevan a giros muy bruscos e inestabilidades.

$$lateral\_accel = L1\_distance * groundspeed^2 * K\_L1 * sin(eta) \tag{6.5}$$

La ecuación anterior permite obtener la aceleración lateral necesaria para realizar un giro. Por último, haciendo uso de todas las ecuaciones anteriores se obtiene que el ángulo de alabeo de referencia esta determinado de la siguiente forma:

$$roll\_sp = atan(\frac{lateral\_accel}{g}) \tag{6.6}$$

Con el objetivo de evitar ángulos de alabeo excesivos que puedan generar inestabilidades en el sistema, se establece que dicho ángulo tiene un rango de $[-50, 50](deg)$. Con todas estas ecuaciones se puede establecer el esquema de Simulink de dicho controlador.

- Total energy control system (TECS)
  Una vez es conocida la parte teórica del desarrollo, se procede con la implementación en Simulink.

  - Bucle de control de energía total
    En primer lugar se procede con la implementación del bucle que comanda la obtención de la posición de la palanca de gases de referencia.

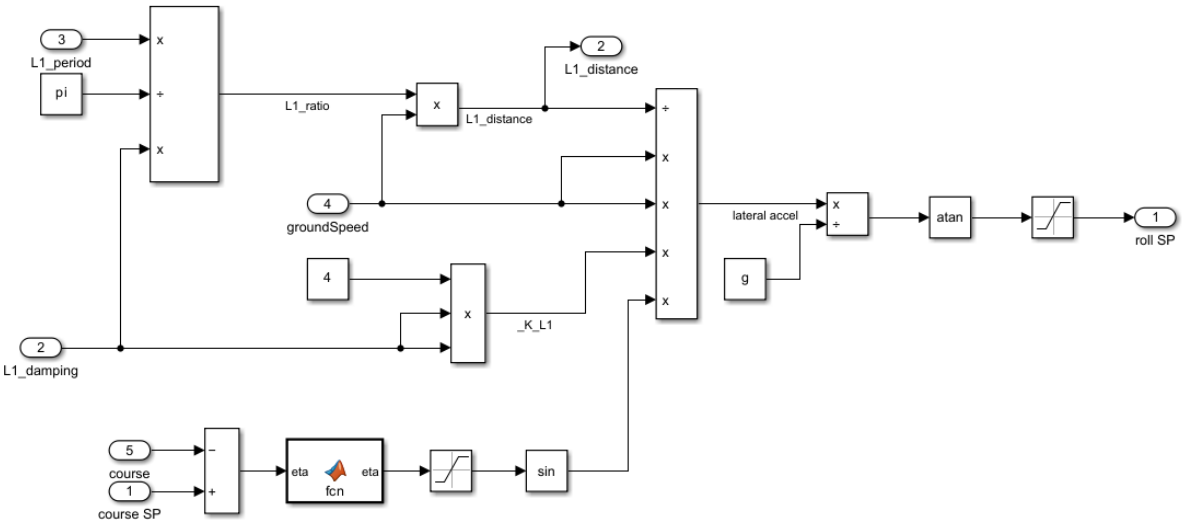

Figura 6.5: Diagrama Simulink controlador L1. Fuente: propia

El esquema global del sistema es:

La explicación de este sistema comienza por los bloques de obtención de la variación

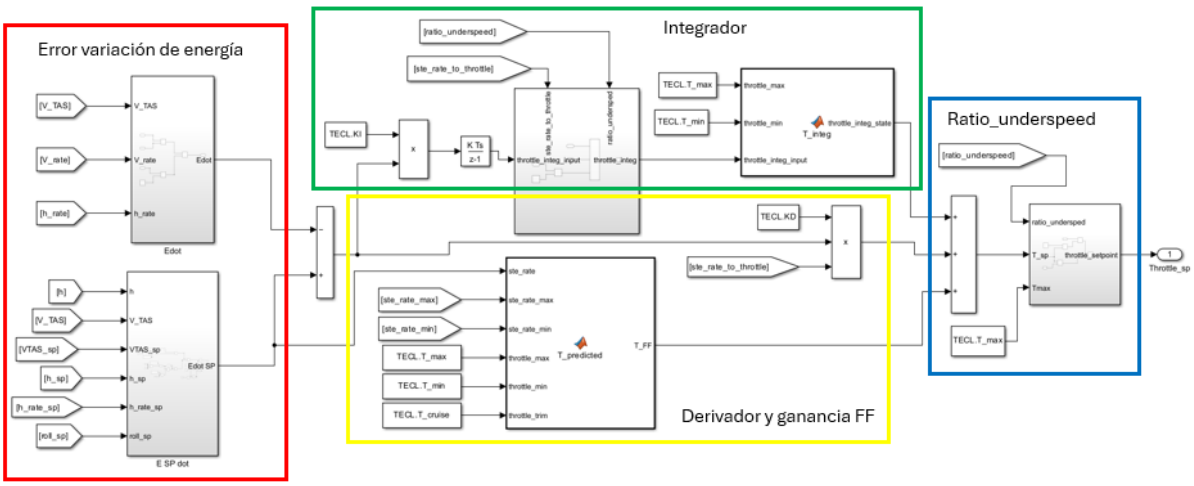

Figura 6.6: Diagrama Total Energy Control Loop. Fuente: propia

de energía específica, tanto la estimada como la de referencia.

Se observan numerosas constantes que definen el sistemas y van ha ser definidas en la siguiente tabla:

Una vez conocidas las variables constantes, se procede con las variables dependientes del tiempo que para los bloques iniciales son:

- V_TAS: esta variable se obtiene a partir del módulo de la velocidad en ejes cuerpo de la aeronave transformada de EAS a TAS aplicando la corrección de altitud.

Variable	Valor	Unidades
g	9.81	m/s^2
V_error_gain	0.2	-
h_error_gain	0.2	-
h_ff_gain	0.3	-
load_factor_correction	15	-
max_climb_rate	5	m/s
min_sink_rate	2	m/s
tas_error_percentage	0.15	-
equivalent_airspeed_trim	15	m/s
tas_min	10	m/s
T_max	1	-
T_min	0	-
T_cruise	0.6	-

Tabla 6.2: Constantes Total Energy Control Loop

- *VTAS_sp*: esta velocidad es la establecida como referencia en la salida del seguimiento de waypoints, pero transformada de EAS a TAS.
- *h*: establece la altitud de vuelo en la que se encuentra la aeronave.
- *h_sp*: es la altitud de referencia de vuelo, esta variable es obtenida como salida del seguimiento de waypoints.
- *VTAS_rate*: establece la variación de velocidad de la aeronave durante la trayectoria.
- *h_rate*: esta variable determina como varía la altitud a partir de la velocidad vertical.
- *h_rate_sp*: determina una velocidad vertical de referencia.
- *roll_sp*: proviene del bloque de seguimiento de waypoints y establece el ángulo de alabeo a seguir.

Estos parámetros son los utilizados en los bloques de establecimiento de variación de energía, pero para posteriores cálculos son necesarias otras variables:

- *ste_rate_max*: establece el límite máximo de variación de energía específica para los límites máximos de la palanca de gases.

$$ste\_rate\_max = max(max\_climb\_rate, 10^{-6}) * g \tag{6.7}$$

- *ste_rate_min*: establece el límite mínimo de variación de energía específica para los límites máximos de la palanca de gases.

$$ste\_rate\_min = max(min\_sink\_rate, 10^{-6}) * g \tag{6.8}$$

- *ste_rate_to_throttle*: esta variable determina una ganancia para pasar de variación de energía específica a posición de palanca de gases.

$$ste\_rate\_to\_throttle = \frac{1}{ste\_rate\_max - ste\_rate\_min} \tag{6.9}$$

- *ratio_ underspeed*: establece un valor entre 0 y 1 para determinar una caída de velocidad. La determinación de este parámetro es más compleja y se realiza a través de un bloque de función de Matlab en el que se implementan las siguientes ecuaciones:

$$tas\_error\_bound = tas\_error\_percentage * equivalent\_airspeed\_trim \quad (6.10)$$

$$tas\_underspeed\_soft\_bound = tas\_error\_bound \quad (6.11)$$

$$tas\_fully\_underspeed = max(tas\_min - tas\_error\_bound - tas\_underspeed\_soft\_bound, 0) \quad (6.12)$$

$$tas\_starting\_to\_underspeed = max(tas\_min - tas\_error\_bound, tas\_fully\_underspeed) \quad (6.13)$$

$$ratio\_underspeed = 1 - \frac{V\_TAS - tas\_fully\_underspeed}{max(tas\_starting\_to\_underspeed - tas\_fully\_underspeed, 10^{-6})} \quad (6.14)$$

Establecidas las entradas de estos dos bloques, se procede a mostrar los esquemas de Simulink, el bloque de variación de energía específica en el que se establece el sumatorio de la variación de energía cinética más la variación de energía potencial.

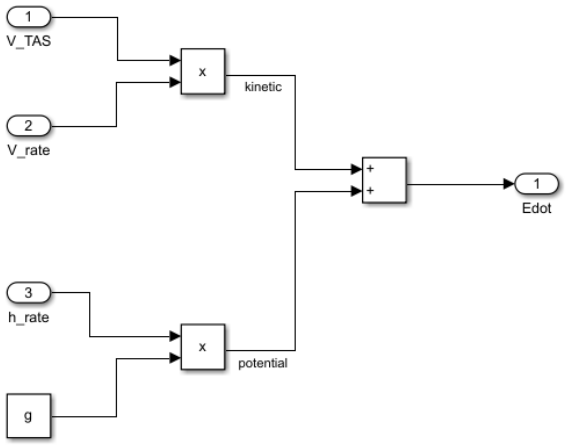

Figura 6.7: Diagrama variación energía específica estimada. Fuente: propia

La obtención de la variación de energía específica de referencia difiere ligeramente de la anterior, el esquema a seguir se establece a continuación:

Esta variación esta formada de igual manera de un parte cinética, otra potencial, pero se le incluye un termino debido a un determinado ángulo de alabeo, esta corrección es utilizada durante giros para compensar la resistencia inducida generada.
Conocidas las variaciones de energía específica se procede a la parte del controlador, en ella se pueden diferenciar una parte integral que es la encargada de eliminar el error en

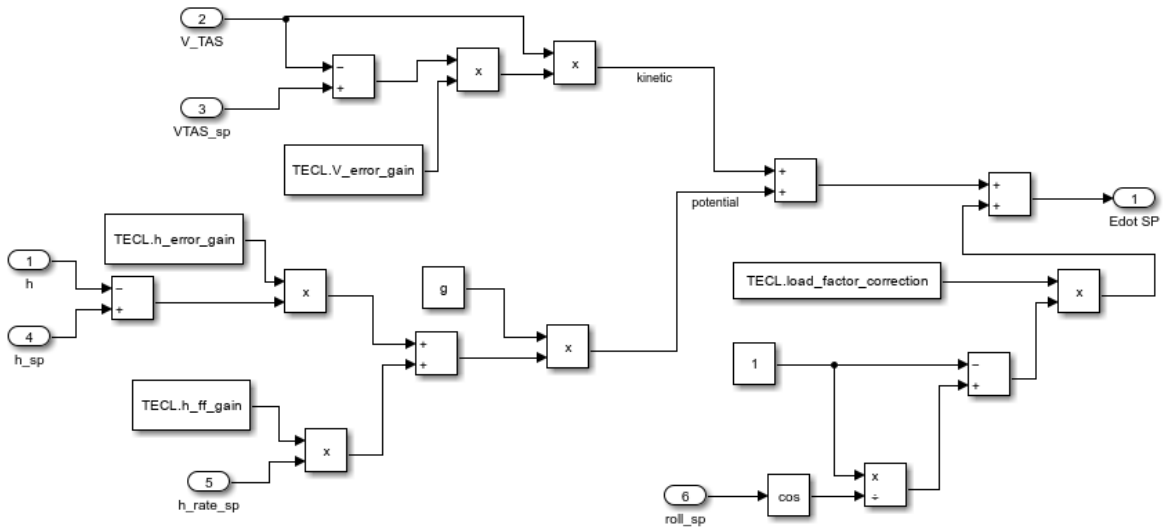

Figura 6.8: Diagrama variación energía específica referencia. Fuente: propia

el caso estacionario, una parte derivativa que es la encarga de evitar oscilaciones, y una parte proporcional estimada como una ganancia feedforward; este tipo de controlador es de lazo cerrado.
Comenzado por la parte integral, el esquema obtenido es el siguiente:

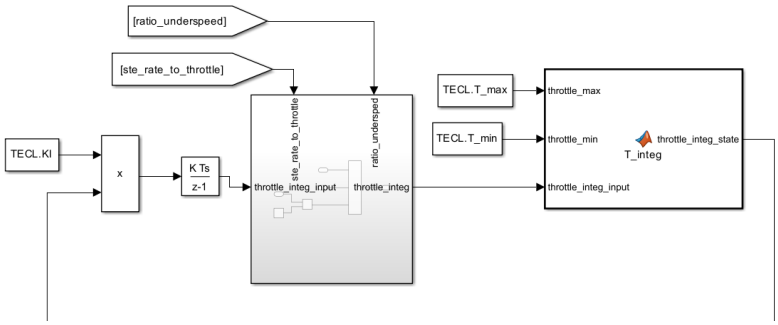

Figura 6.9: Diagrama Integrador Total Energy Control Loop. Fuente: propia

Como entrada se toma la resta de la variación de energía específica de referencia menos la variación de energía específica de la aeronave.
La entrada es multiplicada por una ganancia $KI = 0{,}05$ y por un integrador en tiempo discreto, esta señal obtenida tiene que ser multiplicada por $ste\_rate\_to\_throttle * (1 - ratio\_underspeed)$ para introducir que en condiciones de baja velocidad la integración sea 0 y transformar una variación de energía en una posición de palanca de gases. El bloque de función T_integ establece que el integrado solo se propague en la dirección en la que no satura la el throttle.
Una vez comprendida la acción integral se prosigue con la derivativa, el esquema que se implementa es el siguiente:

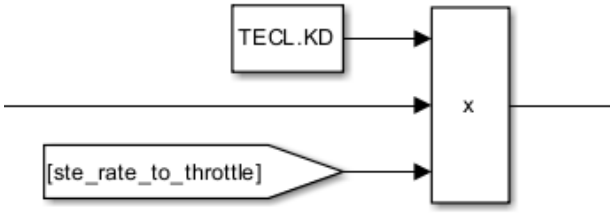

Figura 6.10: Diagrama Derivador Total Energy Control Loop. Fuente: propia

La entrada de este sistema, al igual que en el caso anterior, es el error en la variación de energía específica; esta entrada es multiplicada por la ganancia derivativa $KD = 0{,}1$ y por la variable que permite transformar de variación de energía a throttle.
La última parte del controlador que queda por explicar es la acción que corresponde con una ganancia feedforward, esta se modela con un bloque de función de Matlab a partir del cual se predice un valor de palanca de gases. A diferencia de los bloques anteriores, la entrada de este pertenece a la variación de energía específica de referencia, no al error.

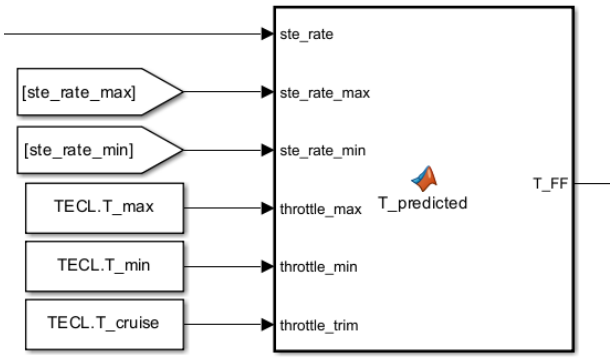

Figura 6.11: Diagrama ganancia feedforward Total Energy Control Loop. Fuente: propia

Dentro de este bloque se implementan las siguientes funciones:

$$throttle\_above\_trim\_per\_ste\_rate = \frac{throttle\_max - throttle\_trim}{ste\_rate\_max} \tag{6.15}$$

$$throttle\_below\_trim\_per\_ste\_rate = \frac{throttle\_trim - throttle\_min}{ste\_rate\_min} \tag{6.16}$$

Aplicando la condición que si la variación de energía específica de referencia es mayor o igual a 0 se aplica que:

$$T\_FF = throttle\_trim + ste\_rate * throttle\_above\_trim\_per\_ste\_rate \tag{6.17}$$

Si la condición anterior no se cumple se aplica la siguiente ecuación:

$$T\_FF = throttle\_trim - ste\_rate * throttle\_below\_trim\_per\_ste\_rate \quad (6.18)$$

La salida final del controlador se establece como la suma de las tres salidas de cada una de las partes que lo componen, obteniendo un valor de palanca de gases de referencia. Por último, es necesario aplicar una condición en la que en función del ratio underspeed se aplique una determinada posición máxima de la palanca de gases, para ello se hace uso de la siguiente ecuación:

$$throttle\_sp = ratio\_underspeed * T\_max + (1 - ratio\_underspeed) * T\_sp \quad (6.19)$$

Esta ecuación se encuentra implementada en el siguiente bloque:

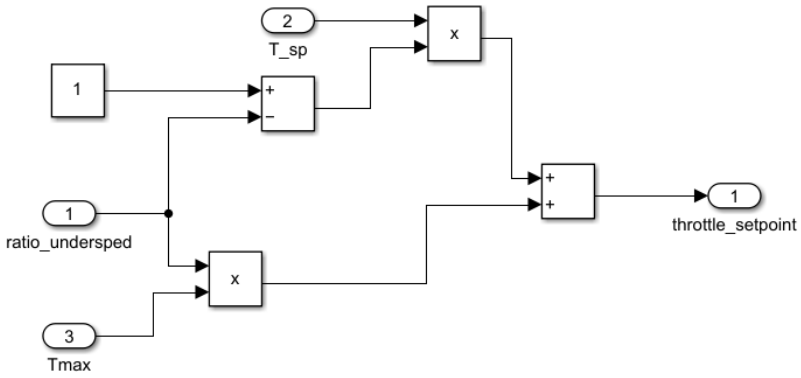

Figura 6.12: Diagrama corrección throttle con ratio underspeed. Fuente: propia

Como salida final del bucle de control de energía total se obtiene una posición de referencia de la palanca de gases, esta posición esta limitada en 0 y 1 y se suministrará como entrada de control al sistema que se encarga de definir la dinámica de la aeronave.

- Bucle de control de balance de energía total
  Al aplicar el método de energía y desacoplar el sistema, la obtención del ángulo de cabeceo de referencia de la trayectoria a seguir se realiza a través del siguiente diagrama de bloques:

  Se observa que la estructura es similar a la de obtención de palanca de gases de referencia, una primera parte donde se calcula un error de variación del balance de energía, al cual prosigue un controlador que es el encargado de ajustar la señal.
  Comenzando por las variables de entrada observadas, hay que destacar la incorporación de dos nuevas:

  - *weight_spe*: factor que indica la cantidad de ponderación que se aplica al control de la energía específica potencial, con un rango de valores de $[0, 1]$.

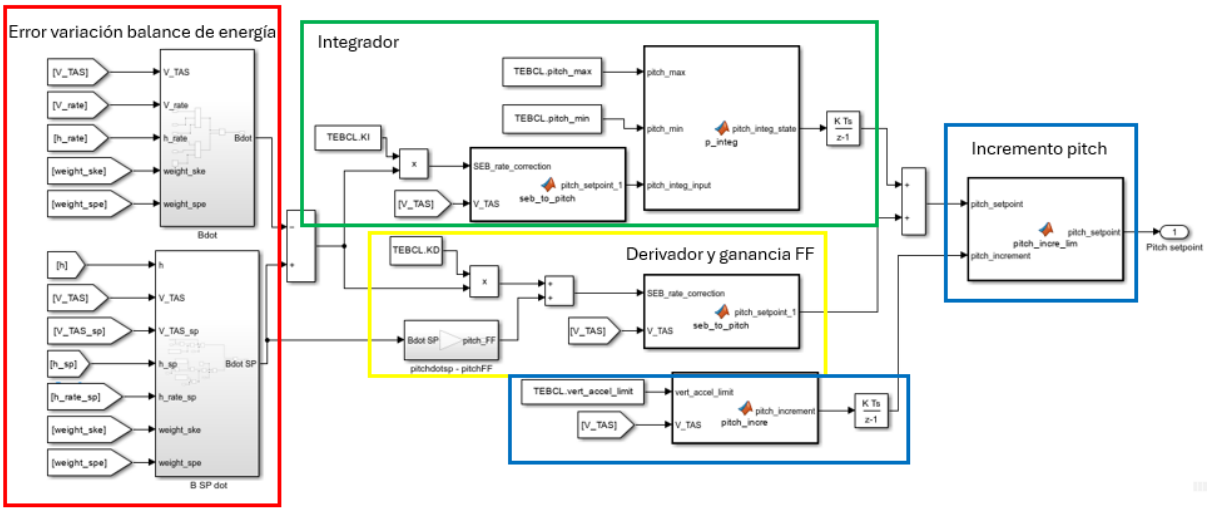

Figura 6.13: Diagrama Total Energy Balance Control Loop. Fuente: propia

- *weight_ske*: factor que indica la cantidad de ponderación que se aplica al control de la energía específica cinética, con un rango de valores de $[0, 1]$.

Estas dos ponderaciones se obtienen del bloque de función de Matlab *weight*, el sistema parte de un parámetro a definir por el usuario *pitch_speed_weight* que determina si el control de cabeceo centra su atención en eliminar los error en la altura sin importar lo de velocidad ($pitch\_speed\_weight = 0$), intenta eliminar el error en la velocidad ($pitch\_speed\_weight = 2$), o un caso equilibrado en el que se reducen la misma cantidad ambos error sin ser ninguno prioritario ($pitch\_speed\_weight = 1$), este último caso es el seleccionado para llevar a cabo las respectivas simulaciones.
Para obtener las variables es necesario tener en cuenta una condición, si el valor de $ratio\_underspeed > 10^6$ la ecuación a utilizar es:

$$pitch\_speed\_weight\_2 = 2 * ratio\_underspeed + (1 - ratio\_underspeed) * pitch\_speed\_weight \tag{6.20}$$

Si la condición anterior no se cumple se establece que: $pitch\_speed\_weight\_2 = 0$
Una vez conocido el valor anterior para calcular las variables de interés se hace uso de:

$$weight\_spe = 2 - pitch\_speed\_weight\_2 \tag{6.21}$$

$$weight\_ske = pitch\_speed\_weight\_2 \tag{6.22}$$

Al igual que aparecen variables nuevas, también se observan constantes que aun no han sido definidas:

Tras estas definiciones ya son conocidas todas las entradas del sistema, por lo que se procede con su análisis.
Se comenzará como en el caso anterior por la determinación de la variación del balance de energía específica de la aeronave.

En el se encuentran numerosas similitudes con el de variación de energía específica, la

Variables	Valor	Unidades
pitch_max	30	deg
pitch_min	-30	deg
vert_accel_limit	7	m/s^2

Tabla 6.3: Constantes Total Energy Balance Control Loop

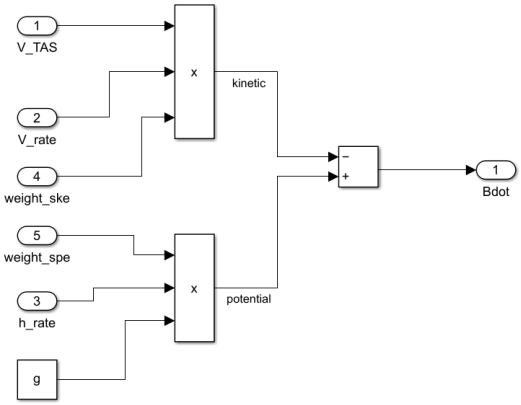

Figura 6.14: Diagrama variación balance de energía estimado. Fuente: propia

diferencia se centra en la incorporación de las ponderaciones en ambas energías y que, como se especifica en la parte teórica del control de energía, el balance se obtiene como la diferencia entre energía potencial y cinética a diferencia del caso anterior en el que se sumaban para obtener la energía total.

Prosiguiendo con el cálculo de la variación de balance de energía específica de referencia, para ello el esquema implementado es:

Al igual que en el esquema superior, las diferencias con la variación de energía específica se encuentran en la incorporación de las ponderaciones y en la resta de las energías para obtener el balance; también se observa que no se encuentra la corrección en giros incorporada en la variación de energía específica, esto se debe a que el aumento de resistencia inducida se compensa solo con la palanca de gases y no con la determinación de ángulo de cabeceo de referencia.
Ya se ha establecido la parte inicial del diagrama, lo siguiente a abordar es el controlador formado por tres partes.
La primera de ellas la acción integral esta compuesta por una ganancia, una transformación de balance a ángulo de cabeceo, una limitación de ese ángulo y una acción integral discreta.

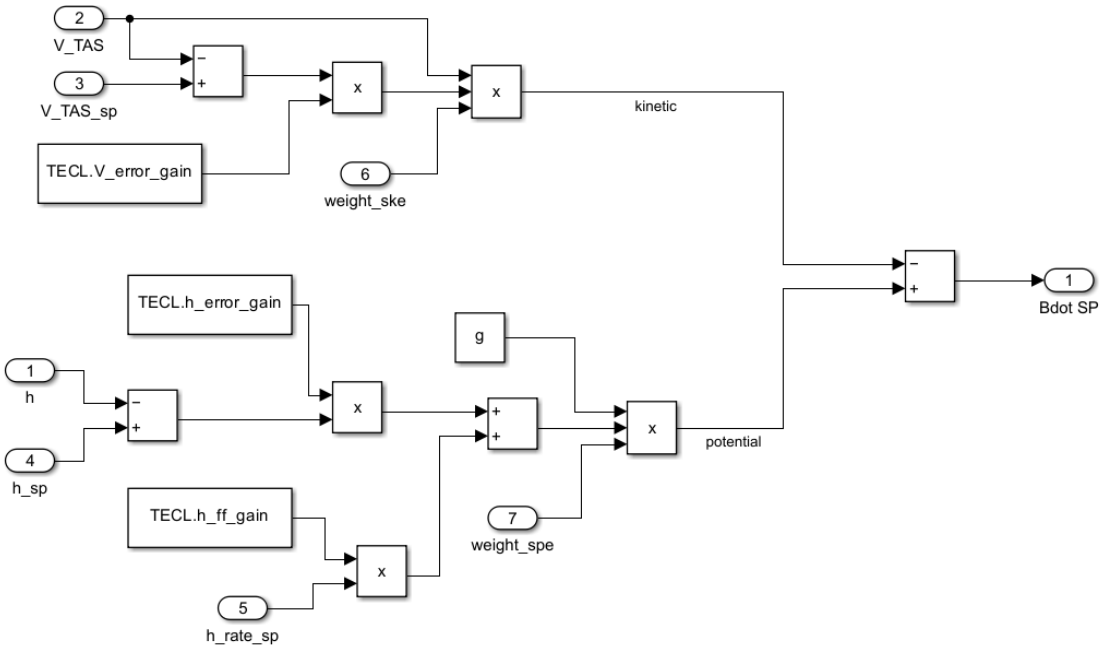

Figura 6.15: Diagrama variación balance de energía referencia. Fuente: propia

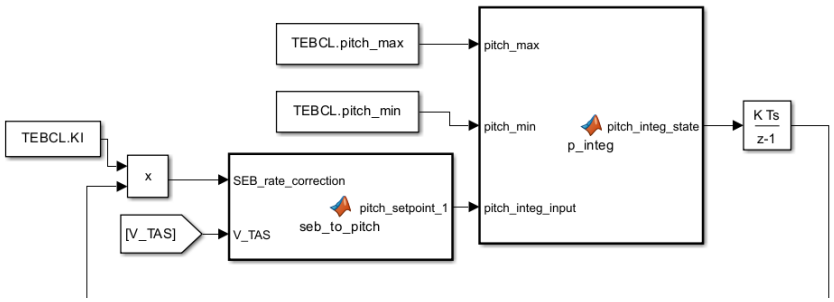

Figura 6.16: Diagrama Integrador Total Energy Balance Control Loop. Fuente: propia

La señal de entrada de esta parte del diagrama corresponde al error en la variación del balance de energía específica, este error es multiplicado por una ganancia $KI = 0{,}1$. El valor requerido de salida es un ángulo de cabeceo por lo que es necesario transformar el error de la variación de balance energético, para ello se hace uso de la función *seb_to_pitch* en la que están implementadas las ecuaciones:

$$climb\_angle\_to\_seb\_rate = V\_TAS * g \tag{6.23}$$

$$pitch\_setpoint = \frac{seb\_rate\_correction}{climb\_angle\_to\_seb\_rate} \tag{6.24}$$

Para este cálculo se asume que:

- La ángulo de ascenso sigue el ángulo de cabeceo con un retraso lo suficientemente pequeño para no desestabilizar el sistema.

- La diferencia entre el ángulo de ascenso y el ángulo de cabeceo es constante, excluyendo los efectos transitorios de cabeceo durante las acciones de control o turbulencias.

Tras esto hay que limitar el valor de la parte integral para evitar que el integrador cambien en la dirección en la que aumente la saturación del cabeceo demandado, esto se realiza por la función *p_integ*.
La explicación restante se realiza de manera conjunta por la simplicidad de la misma, el diagrama conjunto de parte derivativa y ganancia feedforward.

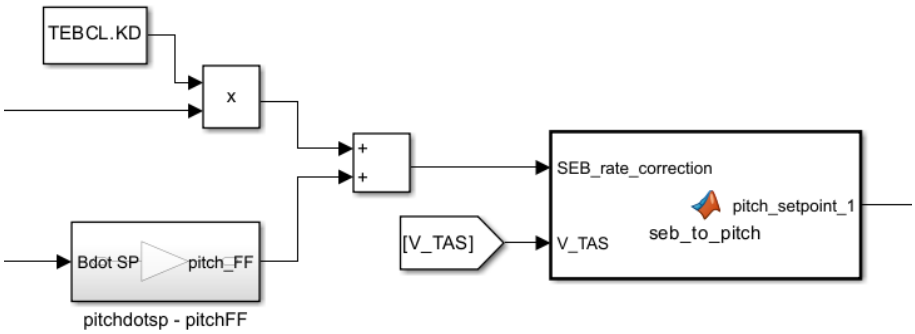

Figura 6.17: Diagrama Derivador y ganancia feedforward Total Energy Balance Control Loop. Fuente: propia

La entrada de la parte derivativa corresponde, al igual que la integral, al error en la variación del balance de energía; a esta entrada se le aplica una ganancia $KD = 0{,}1$.
La parte relacionada con la ganancia feedforward se toma como entrada la variación de balance energético de referencia y se multiplica por una ganancia $rate\_ff = 1$.
A estas dos salidas hay que aplicar una transformación para pasar a de energía a ángulo de cabeceo, para ello se hace uso de la función antes definida *seb_to_pitch*.
Obtenidas las partes del controlador, se suman obteniendo un ángulo de cabeceo de referencia, esta referencia se limita con un incremento de cabeceo, esto se establece en:

La función *pitch_incre* establece un valor del incremento de cabeceo a partir de un límite de aceleración vertical y velocidad:

$$pitch\_increment = \frac{vert\_accel\_limit}{max(V\_TAS, 10^{-6})} \tag{6.25}$$

A este incremento se le aplica un integrador en tiempo discreto. Para establecer los límites se suma y se resta este incremento al ángulo de cabeceo de referencia, esto se realiza en la función *pitch_incre_lim*.
Este ángulo final se satura entre las constantes *pitch_max* y *pitch_min*, y es el utilizado en el posterior control de actitud de la aeronave para obtener la posición de los controles.
Con la definición de este sistema se completa el bloque de control de energía obteniendo los valores de referencia de palanca de gases del motor y de ángulo de cabeceo.

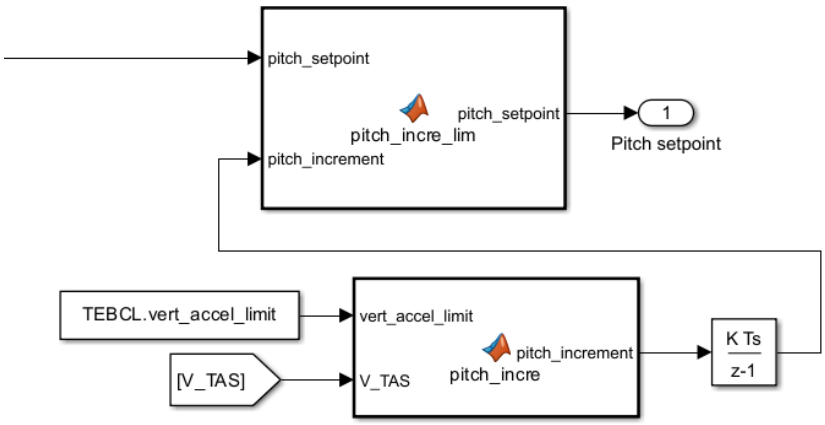

Figura 6.18: Diagrama correccion incremento pitch. Fuente: propia

Con todo esto queda definido al completo el control de posición.

6.1.2 Attitude control

Una vez es conocida la teoría que determina el esquema se prosigue con la implementación del esquema en Simulink:

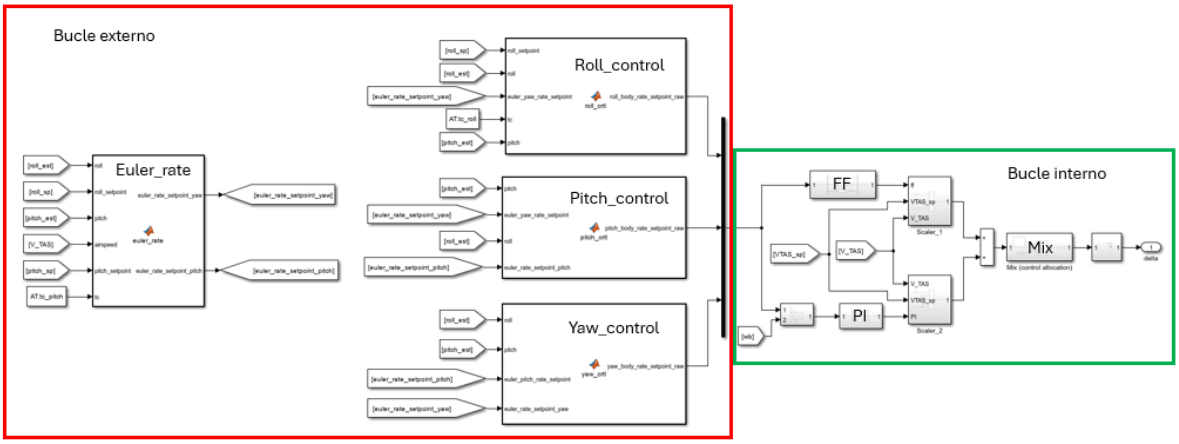

Figura 6.19: Diagrama Attitude Control. Fuente: propia

Antes de comenzar con el desarrollo de los bloques es necesario conocer las entradas de este sistema:

- *roll_sp*: ángulo de alabeo de referencia obtenido del controlador L1.
- *pitch_sp*: ángulo de cabeceo de referencia obtenido del bucle de control de balance de energía.

- *roll_ est, pitch_ est, yaw_ est*: valores de ángulos de alabeo, cabeceo, guiñada obtenidos del bloque de dinámica de la aeronave.
- *wb*: variación de ángulos de actitud en ejes cuerpo obtenidos del bloque de dinámica de la aeronave.
- *V_ TAS*: velocidad de la aeronave.
- *VTAS_ sp*: velocidad de referencia obtenida del bloque de seguimiento de waypoints.

Por otro lado existen variables constantes que determinan los valores de control del sistemas, para esta definición se divide en función del ángulo que controla:

Variable	**Valor**	**Variable**	**Valor**	**Variable**	**Valor**
tc_roll	0.45	tc_pitch	0.4	-	-
roll_ff	0.4	pitch_ff	0.4	yaw_ff	0.3
roll_P	0.085	pitch_P	0.05	yaw_P	0.05
roll_I	0.13	pitch_I	0.05	yaw_I	0.1
roll_D	0	pitch_D	0	yaw_D	0

Tabla 6.4: Valores Attitude Control

La explicación detallada de la implementación comenzara con los bloques del bucle externo formado por la parte izquierda del diagrama.

- Bucle externo
  Esta parte del controlador de actitud esta formada por diferentes bloques de función de Matlab, dicha implementación es realizada de esa forma para facilitar la utilización de las ecuaciones necesarias.
  Comenzado por la función *euler_ rate*:

  En el interior de esta función se realiza el cálculo de la variación del error de cabeceo utilizando las siguientes ecuaciones:

$$pitch\_error = pitch\_sp - pitch \tag{6.26}$$

$$euler\_rate\_setpoint\_pitch = \frac{pitch\_error}{tc\_pitch} \tag{6.27}$$

  La variable *tc_ pitch* es una ganancia que permite establecer el punto de ajuste de la velocidad de cabeceo.
  A la hora de obtener el error en guiñada se aplica la ecuación de giros coordinados comentada con anterioridad, restringiendo el alabeo entre $[-abs(roll\_sp), abs(roll\_sp)]$.
  Conocidas las salidas del bloque anterior se procede con la función *roll_ crtl*:

  Lo obtenido tras esta función es un punto de ajuste de la velocidad de alabeo transformada a ejes cuerpo, en primer lugar se procede con el cálculo del error en el ángulo de alabeo:

$$roll\_error = roll\_sp - roll \tag{6.28}$$

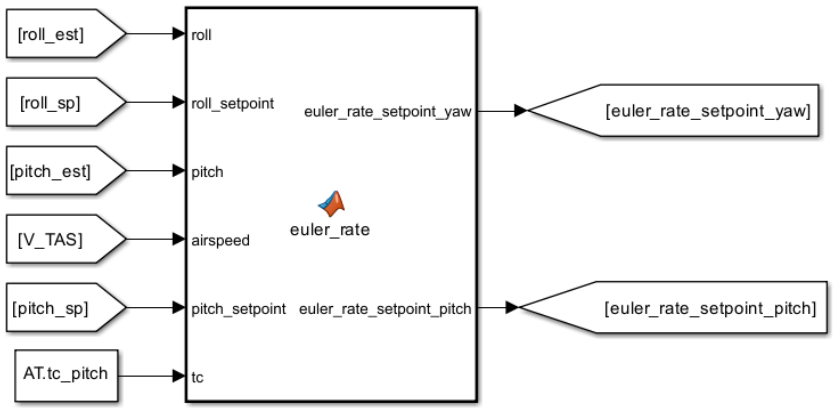

Figura 6.20: Diagrama Euler Rate Setpoint Pitch y Yaw. Fuente: propia

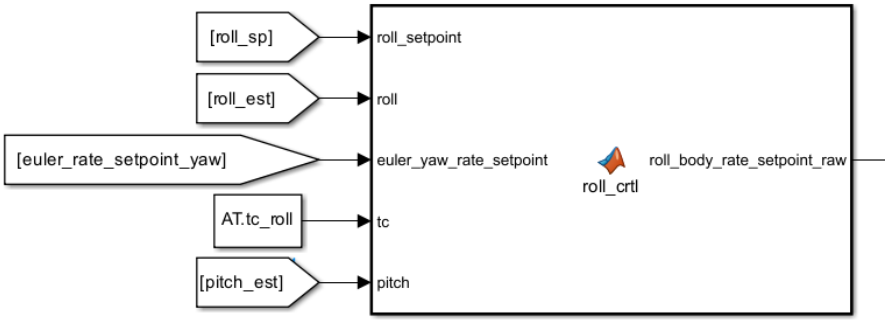

Figura 6.21: Diagrama Roll Body Rate Setpoint. Fuente: propia

A este error se le aplica una ganancia para generar un punto de ajuste de velocidad de alabeo:

$$euler\_rate\_setpoint\_roll = \frac{roll\_error}{tc\_roll} \tag{6.29}$$

Este punto de ajuste se encuentra en un sistema de referencia NED y es necesario transformarlo a coordenadas centradas en el cuerpo de la aeronave, para ello se hace uso de:

$$\begin{aligned} roll\_body\_rate\_setpoint\_raw = euler\_rate\_setpoint\_roll \\ -sin(pitch\_est) * euler\_rate\_setpoint\_yaw \end{aligned} \tag{6.30}$$

Este es el punto de ajuste de velocidad de alabeo.
La siguiente función a desarrollar es *pitch_ crtl*.

Este bloque tiene como entrada el punto de referencia de velocidad de cabeceo, dentro se realiza la transformación de coordenadas NED a cuerpo, para ello se aplica:

$$\begin{aligned} pitch\_body\_rate\_setpoint\_raw = cos(roll\_est) * euler\_rate\_setpoint\_pitch \\ +cos(pitch\_est) * sin(roll\_est) * euler\_rate\_setpoint\_yaw \end{aligned} \tag{6.31}$$

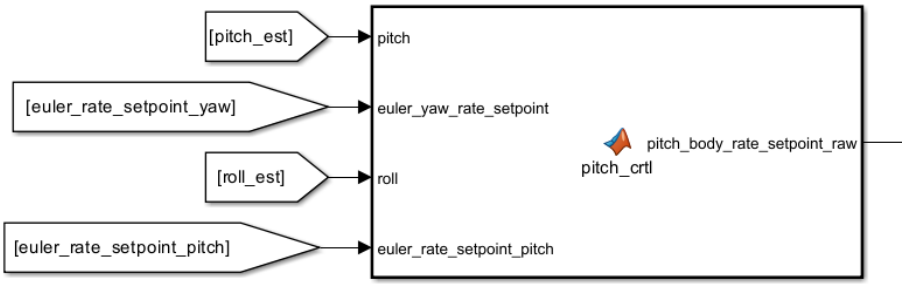

Figura 6.22: Diagrama Pitch Body Rate Setpoint. Fuente: propia

Este es el punto de ajuste de velocidad de cabeceo.
El último bloque del lazo externo es la función *yaw_ crtl*.

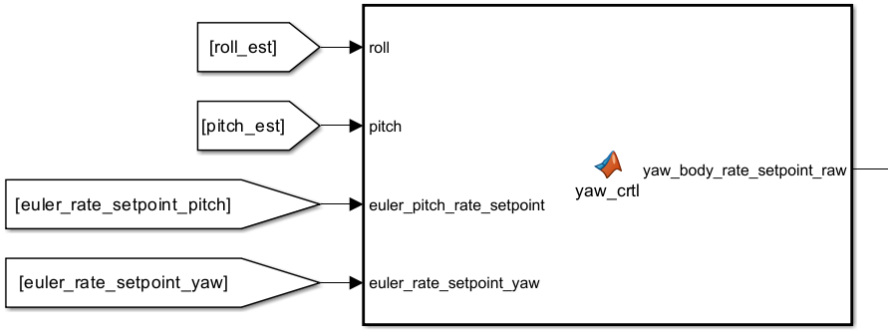

Figura 6.23: Diagrama Yaw Body Rate Setpoint. Fuente: propia

Al igual que la función anterior, el punto de referencia de velocidad de guiñada ya esta establecido en la función *euler_ rate* pero es necesario un cambio de coordenadas, para ello se hace uso de la transformación:

$$\begin{aligned} yaw\_body\_rate\_setpoint\_raw = &-sin(roll\_est) * euler\_rate\_setpoint\_pitch \\ &+cos(pitch\_est) * cos(roll\_est) * euler\_rate\_setpoint\_yaw \end{aligned} \tag{6.32}$$

Este es el punto de ajuste de velocidad de guiñada.
Con todo lo anterior queda totalmente definido el bucle externo de control de actitud obteniendo las entradas del bucle interno que se desarrolla a continuación.

- Bucle interno
  El bucle interno del diagrama es el encargado de obtener las deflexiones de las superficies de control a partir de los puntos de referencia de las velocidades de variación de los ángulos. Comenzando por la parte superior se observan dos bloques, el primero de ellos corresponde a las ganancias feedforward de cada uno de los ángulos cuya utilidad ha sido antes mencionada

  El segundo bloque, llamado *Scaler_ 1*, se utiliza para escalar la señal de salida del controlador feedforward con la velocidad para mejorar el rendimiento del control. El diagrama en este caso es bastante sencillo:

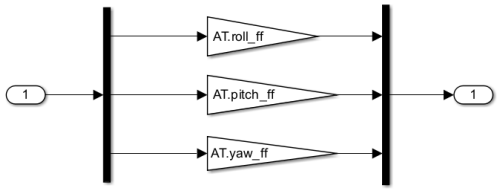

Figura 6.24: Diagrama ganancia feedforward. Fuente: propia

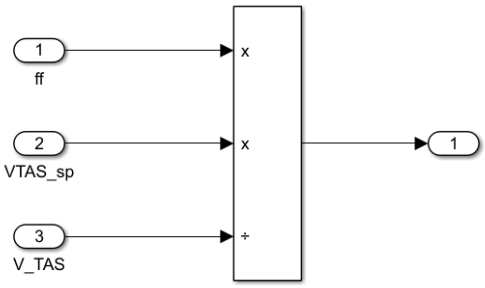

Figura 6.25: Diagrama Scaler_1 ganancia feedforward. Fuente: propia

Una vez concluida con la parte superior del diagrama se procede con la inferior en la que se implementa el controlado PI.
El primer bloque que se encuentra es el encargado de obtener el error entre las velocidades de referencia obtenidas del bucle externo y las velocidades de variación de los ángulos de la aeronave, provenientes del bloque de dinámica.
En el segundo bloque se aplican los controladores PI para cada uno de los errores:

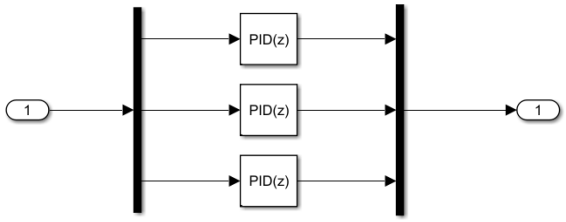

Figura 6.26: Diagrama Control PI. Fuente: propia

Los bloques aplicados corresponden a un PID en el que se ha establecido una ganancia derivativa nula para transformarlo en un PI como se establece en la teoría. Los valores de cada controlador se han especificado en tablas anteriores.
El último bloque de esta parte corresponde a *Scaler_2*, al igual que el primero, se utiliza para escalar la salida del controlador con la velocidad, en este caso es la salida del contro-

lador PI; la estructura es ligeramente diferente a la anterior
En esta caso las velocidades se encuentran al cuadrado.

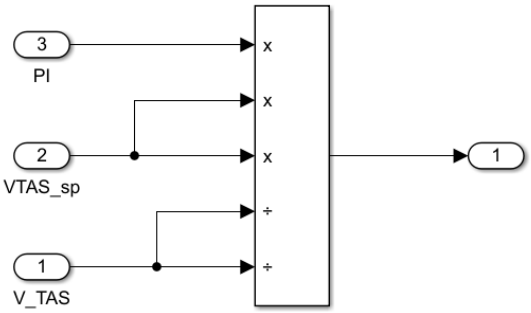

Figura 6.27: Diagrama Scaler_2 Control PI. Fuente: propia

La parte final del control de actitud esta formada por dos bloques, el primero de ellos se llama *Mix (Control allocation)*, este bloque es el encargado de con los comandos de torque deseados de los controles, traducirlos a comandos de actuador que controlan los servos; para los usos destinados a este proyecto se supone como una matriz identidad:

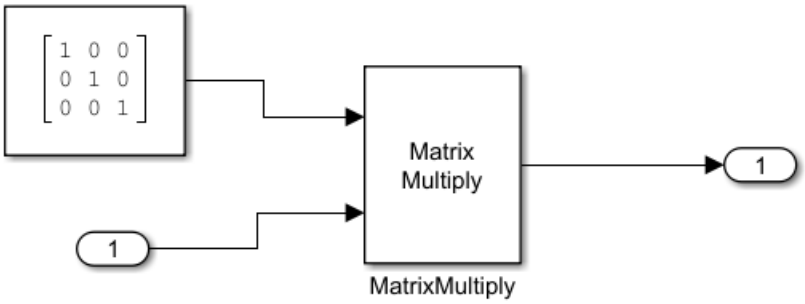

Figura 6.28: Diagrama Mix (Control Allocation). Fuente: propia

Para finalizar con este controlador se encuentra un último bloque en el que se limitan las acciones de los controles para que las deflexión no sean superiores a las máximas establecidas en el bloque de dinámica.
Con esta última definición, el control de actitud queda totalmente definido.

Con todo lo desarrollado anteriormente, el diagrama de controladores esta implementado al completo en Simulink; resumiendo el sistema, a partir de una trayectoria definida por la posición de waypoints y por la dinámica de la aeronave es posible obtener los valores de los controles necesarios para que el sistema siga la referencia establecida.

6.2 Validación Simulink

Tras el desarrollo de la implementación es necesario realizar una validación del modelo realizado para comprobar que el grado de similitud entre ambos es grande y que el modelo de Simulink pueda ser utilizado para remplazar al de PX4 resolviendo determinados problemas. Este punto es de especial importancia ya que sin él el modelo implementado no tiene validez ni puede ser utilizado.
Para realizar esta validación se requiere de simular ambos modelos y comparar las respuestas obtenidas de una serie de valores críticos del modelo.
Para la validación se establece una trayectoria común a ambos sistemas, esta trayectoria se define en QGroundControl, esta trayectoria incluye giros a derecha e izquierda, de mayor y menor radio, cambios de altitud y cambios de velocidad, para asegurar que la referencia establecida de vuelo se cumple correctamente en todos los ámbitos. La trayectoria definida es la siguiente:

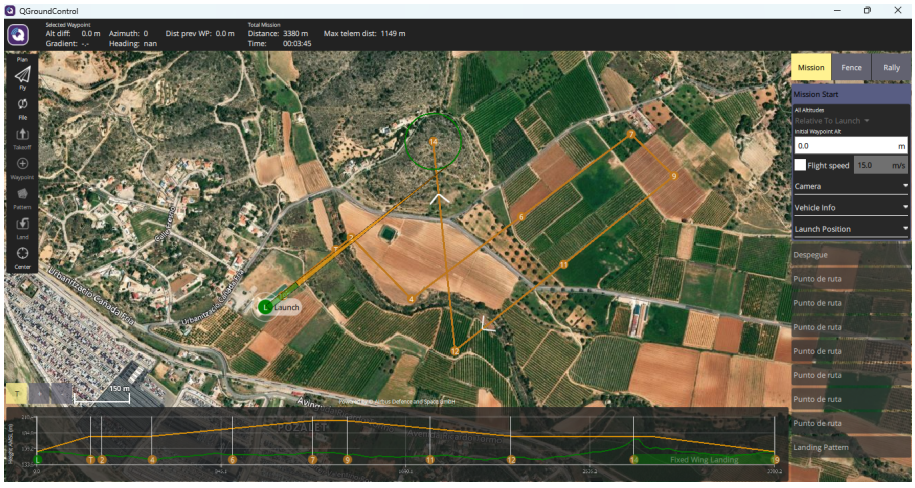

Figura 6.29: Trayectoria QGroundControl. Fuente: propia

Las altitudes y velocidades de referencia de esta trayectoria son:

Waypoint	**Latitud NED** [m]	**Longitud NED** [m]	**Altitud** [m]	**Velocidad** [m/s]
1	0	0	25	15
2	203.7	253.2	25	15
3	39.9	414.7	25	15
4	259.8	709.6	37.5	20
5	479.6	1004.6	50	25
6	368.1	1119.7	50	25
7	131.9	823.5	37.5	20
8	-98.7	531.5	25	15
9	459.9	473.1	25	15

Tabla 6.5: Waypoints Trayectoria Validación

Con todo esto la trayectoria queda totalmente definida.

6.2.1 Comparación resultados modelo implementado Simulink vs PX4

Como ha sido antes mencionado, la validación del modelo implementado tiene gran importancia a la hora de uso, para realizar esta validación se comparan diferentes variables importantes del sistema a la hora de realizar una trayectoria y se observan si las señales que se obtienen de salida coinciden y si los valores se encuentran en un rango aceptable.
Una observación a tener en cuenta a la hora de comparar las gráficas es que para la señal de PX4 se parte de una posición en tierra y se realiza un despegue mientras que en Simulink la aeronave se parte de un vuelo a una altitud y velocidad establecidas, por ese motivo las señales iniciales de algunas variables difieren.
La primera comprobación a realizar es la más visual, rápida y fácil de observar, consiste en la representación de la trayectoria en ejes x-y:

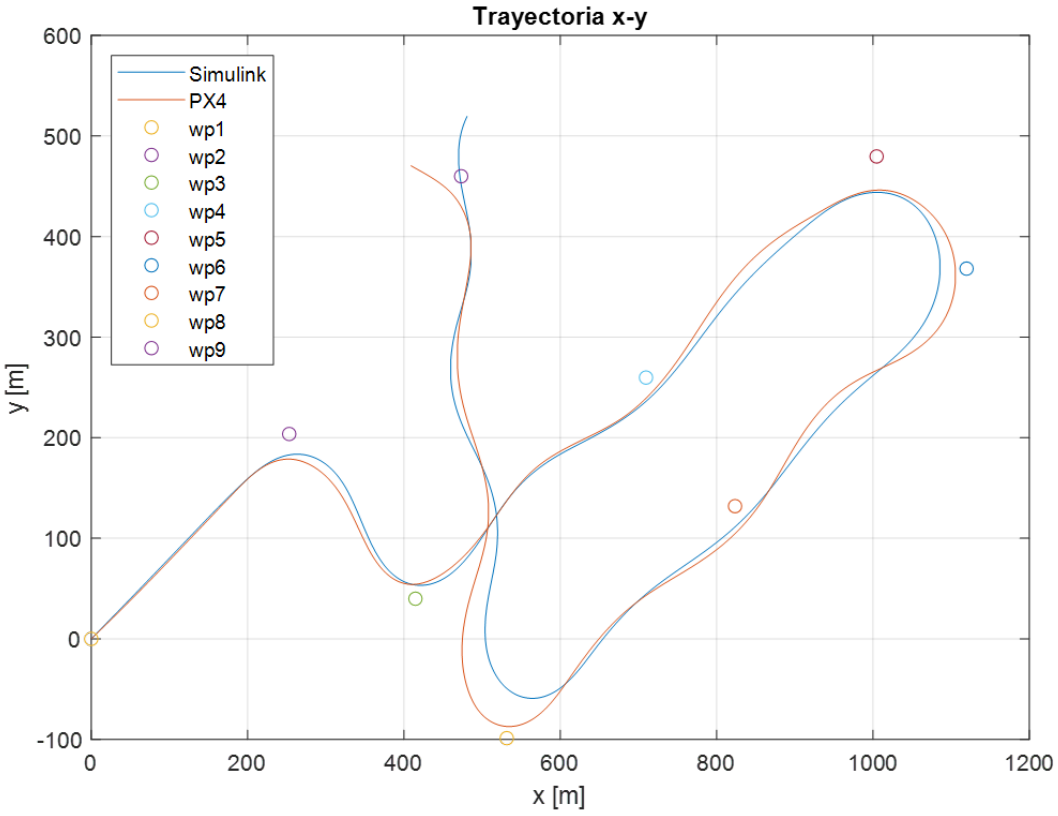

Figura 6.30: Validación Trayectoria x-y. Fuente: propia

A primera vista se observa que el seguimiento de waypoints se realiza correctamente, la trayectoria también es significativamente similar sin producirse sobre-oscilaciones tras los giros o desplazándose en gran medida de la trayectoria de referencia, aunque si que se observan ligeras desviación con respecto a la trayectoria estimada por PX4.

Los siguientes parámetros importantes a observa son el seguimiento de referencia en altitud y velocidad:

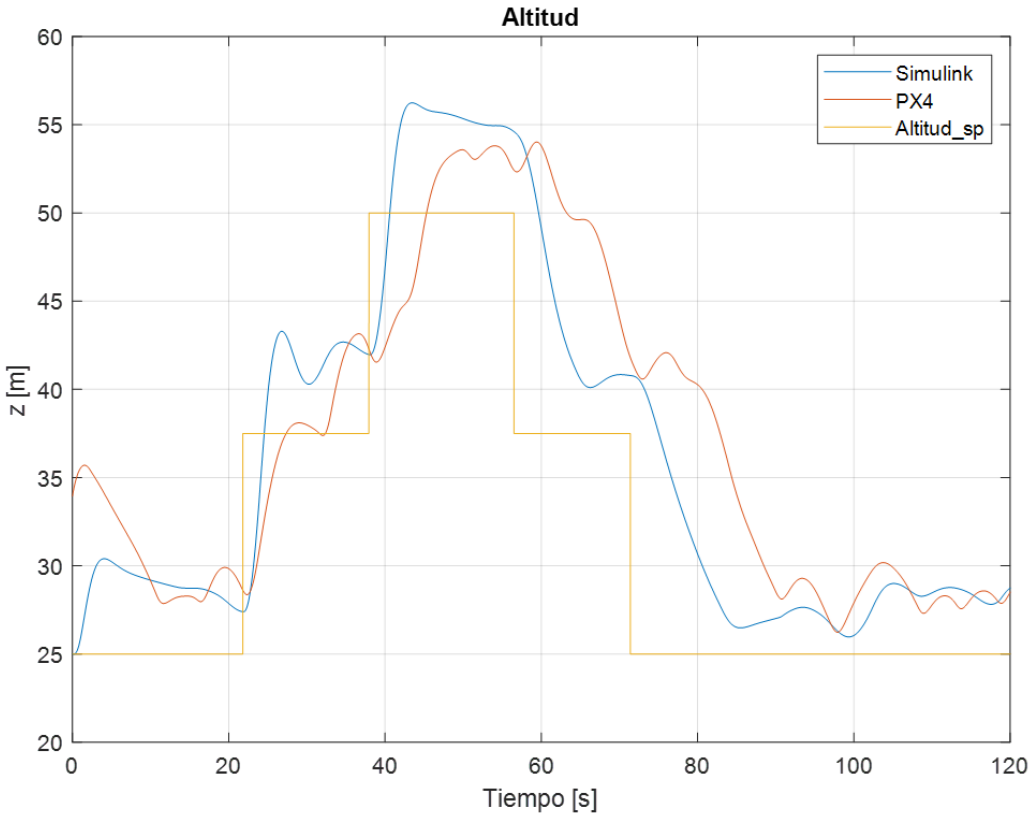

Figura 6.31: Validación Altitud. Fuente: propia

Observando la gráfica de altitud se puede ver que en general la forma obtenida por ambas respuesta es similar pero existen zonas en las que la diferencia notable por lo que requieren de una explicación. En la fase inicial del vuelo [0,30] segundos, el modelo de PX4 tiene un pico de altitud y el modelo de Simulink no, esto es debido a que con PX4 se realiza el despegue de la aeronave y un ascenso hasta la altitud de referencia mientras que con Simulink se parte de una condición en vuelo a una cierta altitud, estas dos definiciones iniciales de misión provocan la diferencia inicial de las señales. Posteriormente, otras diferencias observables son los ascenso y descensos, en el caso del modelo de PX4 las altitudes son proporcionadas por QGroundControl, sistema en el que al establecer una diferencia de altitud entre dos waypoints divide este segmento en un numero x de subsegmentos para realizar un ascenso gradual que es lo que se observa, en el caso de Simulink para simplificar la definición de waypoints los cambios de altitud se dividen solo en dos segmentos, lo que produce cambios más pronunciados. A parte de dichas diferencias, lo importante del modelo reside en un buen seguimiento de la referencia que en este caso es correcto.

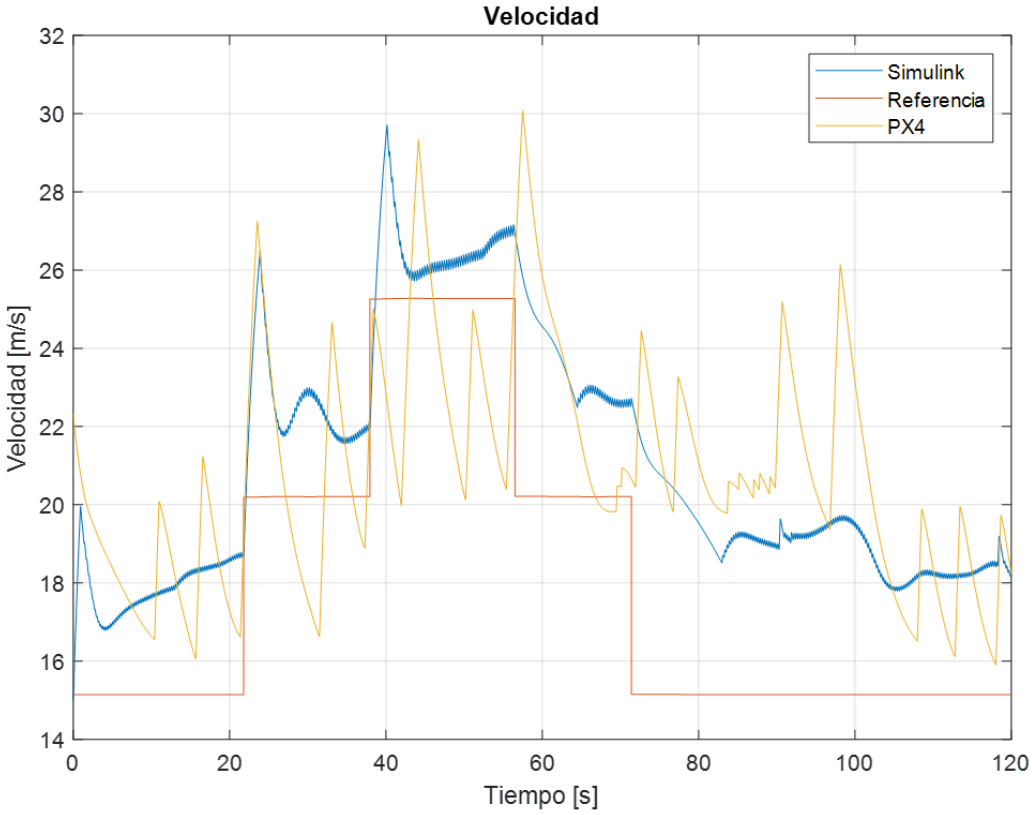

Figura 6.32: Validación Velocidad. Fuente: propia

La gráfica de velocidad es más confusa que las vistas anteriormente, se observa una gran oscilación de la velocidad obtenida de PX4 aun así se observa que la tendencia a seguir es similar a la señal de Simulink y las velocidades de referencia se establecen correctas sin obtener valores anómalos de velocidades excesivamente grandes o pequeña; el error con respecto a la referencia es asumible teniendo en cuenta el caso en el que esta configurada la parte de navegación.
El modo de seguimiento de la referencia establecido en el bloque de navegación pretender seguir con el mínimo error tanto la velocidad como la altitud, por ese motivo la respuesta difiere ligeramente de la referencia, en el caso de establecer que se siga fielmente una de las variables, el sistema se centra en seguir ese valor y se olvida de corregir lo demás.
Ya conocidas las variables que se encargan de definir la trayectoria de la aeronave y comprobando que se ajustan tanto a la misión establecida como al modelo de PX4 se puede proceder con los valores ángulos de actitud de la aeronave.

Como se observa las referencias de control lateral de la aeronave se siguen de manera bastante precisa, lo que comprueba el buen seguimiento obtenido en la gráfica de la trayectoria x-y, el control lateral se encuentra definido por el alabeo y la guiñada; por otro lado, centrando la atención en el control longitudinal, el seguimiento de la referencia no se realiza tan fielmente como en los otros dos casos, esto se debe a las diferencias antes mencionadas en las gráficas de altitud y velocidad entre el sistema PX4 y Simulink en el control de posición de la aeronave; aun con estas diferencias se puede observar que la tendencia en cada fase del vuelo es similar,

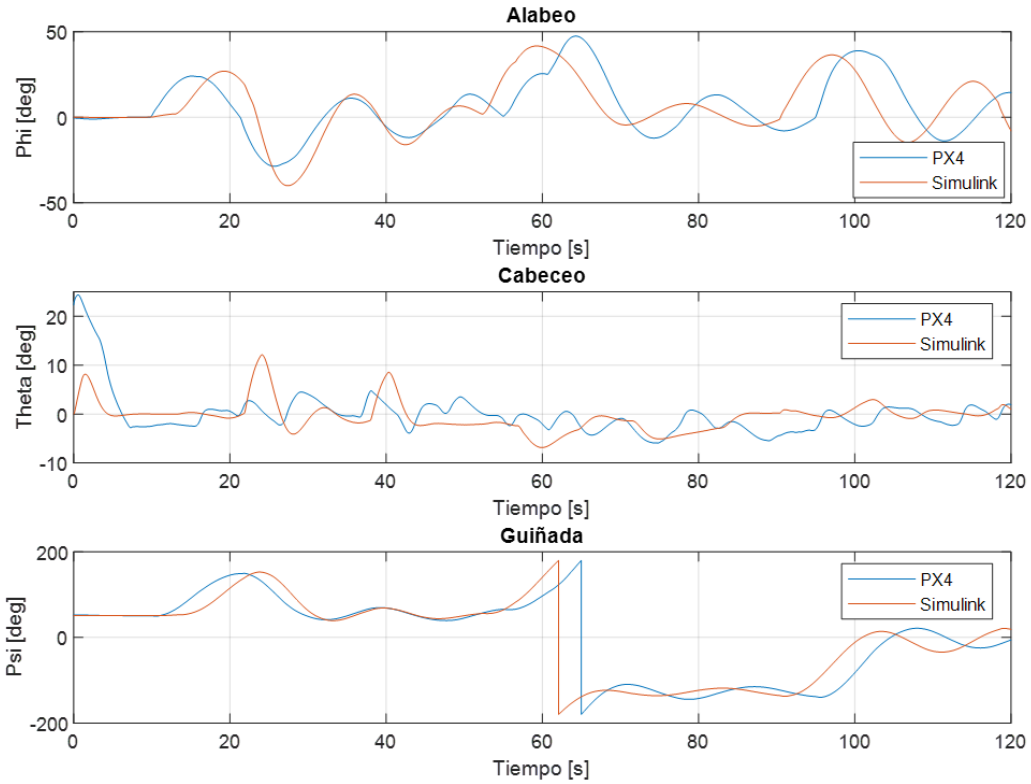

Figura 6.33: Validación Actitud. Fuente: propia

para tiempos menores de 60 se producen los ascenso que como se ve para el caso de Simulink se realizan con dos escalones de ángulo de cabeceo positivos mientas que en el caso de PX4 los escalones son mayores en número pero de menor amplitud; lo mismo se puede observar en la zona central, entre 60 y 100 segundos, donde para el descenso donde para ambos casos los ángulos de cabeceo son negativos pero el procedimiento seguido de descenso es ligeramente diferente. Un punto muy notable a primera vista es el desfase entre las señales, este efecto es igualmente observable en las gráficas de trayectoria, altitud y velocidad, y es debido a ligeras diferencias en el modulo de navegación comentadas previamente, aun así los ángulos obtenidos por ambos modelos son iguales. Como condición global se puede observa que para ninguno de los ángulos antes mostrados la señal de salida obtiene valores anómalos, entendiendo por estos ángulos muy elevados tanto positivos como negativos.

La última comprobación a realizar son las señales de salida del sistema de controladores, esta parte es de gran importancia ya que es la encargada de suministras al sistemas de dinámica las actuaciones que tiene que llevar a cabo. Las comparaciones de resultados es la siguiente:

A primera vista se puede ver que los seguimientos de todas las señales de salida se encuentran entre unos valores aceptables de palancas de control, estableciendo que un valor de 1 o -1 corresponden con las máximas deflexiones de los controles. Para comentar los resultados se procede de igual forma que anteriormente, diferenciando entre componentes que determinan fuerzas latera-

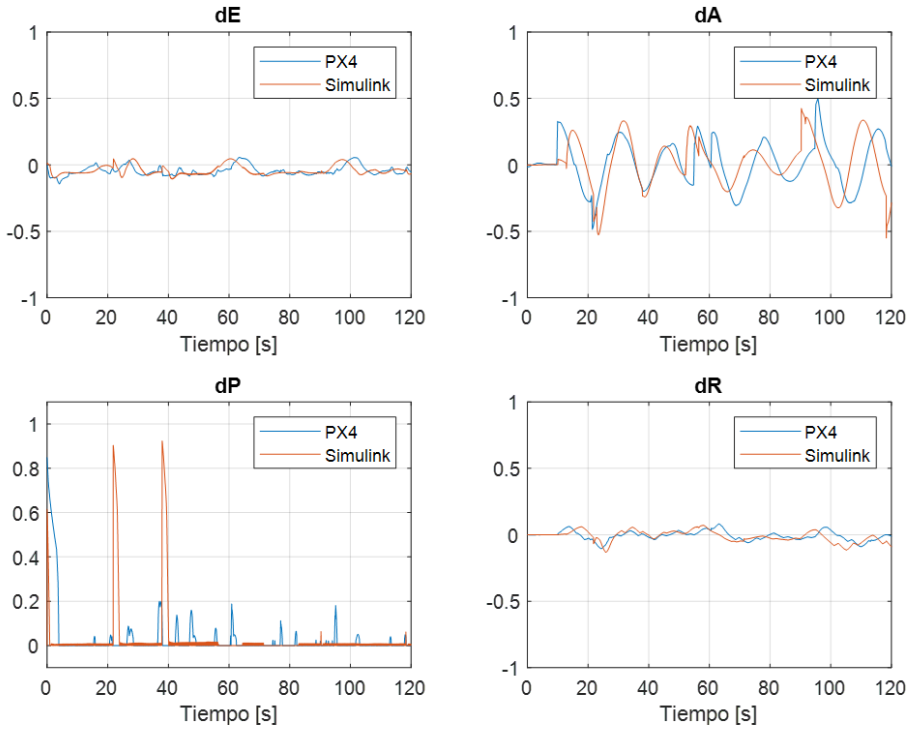

Figura 6.34: Validación Controles. Fuente: propia

les y los que determinan fuerzas longitudinales; comenzado por la parte longitudinal se pueden encontrar la deflexión del estabilizador horizontal y la posición de la palanca de gases, para la primera variable las deflexiones son muy pequeñas y similares a las de referencia, sin embargo fijando la atención en la palanca del motor se observan una serie de picos diferentes a los del sistema de PX4, las anomalías de esta variable se deben a lo antes mencionado en el error de la altitud y velocidad ya que esta es la encargada, en parte del seguimiento de esas referencias. Pasando a las componentes longitudinales, el error entre ambas respuestas es pequeño, siendo el modelo de Simulink fiel al de PX4, la conclusión extraíble de estas dos gráficas es que a la hora de realizar giros o equilibrar componentes laterales, la superficie que más actúa son los alerones, muy por encima del timón de cola en el que las deflexiones son mucho menores. Al observar globalmente las cuatro gráficas se ve también el mismo desfase comentando en el punto de ángulos de actitud.

Con todo lo anterior comentado sobre las gráfica se llega a la conclusión de que el modelo implementado no es totalmente idéntico al modelo de PX4 pero puede ser utilizado para simular vuelos en su lugar ya que las referencias se siguen de manera correcta y las diferencias entre modelos son despreciables para casos generales.

Capítulo 7

Implementación auto-ajuste

La segunda parte del proyecto consiste en el desarrollo de una herramienta que permita ajustar los valores de los controladores de antes de realizar el primer vuelo de la aeronave, de esta forma es posible realizar un ajuste inicial de dichos valores en tierra a través de simulaciones y luego terminar el ajuste en vuelo reduciendo los riesgos y el tiempo necesario de pruebas de vuelo.
Antes de comenzar a desarrollar el proceso de auto-tuning es necesario definir como se va a abordar el problema, en este caso debido a numerosas limitaciones computacionales y de tiempo la elección se ha decantado por realizar una implementación de un algoritmo genético para realizar el ajuste de los valores de control. El ajuste también a sido limitado a un ajuste únicamente del control de actitud de la aeronave.
Una vez realizada una introducción del procedimiento a realizar se comienza con el desarrollo de esta implementación.

7.1 Algoritmo genético

La definición de algoritmo genético ha sido mencionada con anterioridad, pero aun así se va ha volver a comentar brevemente; un algoritmo genético es una técnica de optimización basada en la evolución natural de las especies, esta formado por una población de individuos con una serie de capacidades, estos individuos se relacionan entre ellos o mutan para pasar a la siguiente generación, lo que se busca es obtener el individuo que más adaptado al medio este (los valores que minimicen la función objetivo).
El algoritmo genético a utilizar en este proyecto es el desarrollado por F. Xavier Blasco Ferragud en *"Control predictivo basado en modelos mediante técnicas de optimización heurística. Aplicación a procesos no lineales y multivariables"* [3].
Para comenzar con la explicación de la función es necesario comentar las variables de entrada que se requieren:

- *gaDat.FieldD* define un vector en el que se encuentran los valores máximos y mínimo de las variables, esto define el espacio de búsqueda de los valores óptimos. Este parámetro es de obligada definición.

- *gaDat.Objfun* constituye la función objetivo a minimizar, esta función varia dependiendo del problema.
- *gaDat.NVAR* no es definida por el usuario pero establece el número de variables a optimizar (número de características de cada individuo).
- *gaDat.MAXGEN* define el número máximo de generaciones que realiza el algoritmo, si no esta establece un valor se define por: $gaDat.MAXGEN = gaDat.NVAR * 20 + 10$.
- *gaDat.NIND* establece el número de individuos de cada población, si no se define se establece por defecto: $daGat.NIND = gaDat.NVAR * 50$.
- *gaDat.alfa* establece un parámetro de cruce lineal de individuos, por defecto se establece a 0.
- *gaDat.Pc* se refiere a la probabilidad de cruce de los individuos, por defecto es 0.9.
- *gaDat.Pm* es la probabilidad que tienen los individuos de mutar, se establece por defecto a 0.1.
- *gaDat.Objfunpar* define parámetros adicionales de la función objetivo, no es necesario su definición.
- *gaDat.indini* inicializa unos individuos iniciales para la primera población, si no se define se toman individuos con características aleatorias.

Una vez definidas las variables se procede comentando rápidamente el procedimiento seguido para obtener los valores óptimos; la parte más importante de este algoritmo se encuentra desarrollada por la función *gaevolution*, inicialmente en esta función se evalúa la función objetivo con los valores de todos los individuos de la población, tras eso se ordenan de menor valor de función objetivo a mayor con la función *ranking*, una vez ordenados se proceden con 3 modificaciones de estos individuos que permiten generar una nueva generación:

- Selección (función *select*): es la encargada de seleccionar los individuos que pasan a la siguiente generación, esta selección puede ser realizada por dos métodos:
  - Mecanismo de ruleta simple (función *rws*): simula una ruleta dividida en porciones con los individuos, en función del valor de la función objetivo la porción será más grande o más pequeña para cada individuo; finalemente se hace girar la ruleta tantas veces como individuos se quieran seleccionar para la siguiente generación.
  - Stochastic Universal Sampling (función *sus*): el mecanismos de ruleta es similar al anterior, pero la diferencia reside en que ahora no existe un único puntero que selecciona si no tantos como individuos se quieran elegir para la siguiente simulación por lo que de un solo giro se seleccionan todos.
- Cruce (función *lxov*): esta función es la encargada de mezclar las características de los individuos creando de esta forma otros nuevos diferentes. Esta operación no se realiza a todos los individuos si no cada uno tiene una probabilidad de cruce definida anteriormente.
- Mutación (función *mutbga*): realiza una variación aleatoria de las características de los individuos de una población, esta variación se rige por la probabilidad de mutación definida

con anterioridad. Esta operación es complementaria al cruce ya que permite generar nuevas características para los individuos.

Con estas variaciones se produce una nueva generación de individuos mejor adaptados y se vuelve a repetir el proceso hasta obtener un solución lo más óptima posible. [2]
Con esto queda explicado brevemente el funcionamiento de evolución de este algoritmo.

7.2 Planteamiento del problema

Pasando ahora a centrar la atención en el problema a desarrollar en este proyecto se comienza por definir como se va ha plantear el problema, para ello es necesario conocer cuales son las variables a ajustar.
El auto-tuning planteado centra su atención solamente en el control de actitud de la aeronave, este control esta compuesto por 14 variables, un número muy grande para realizar una única simulación. Partiendo de la premisa anterior y del conocimiento de la poca influencia de la dinámica lateral en la longitudinal se divide el problema en una primera resolución solo de los controles longitudinales, con estos valores se realiza un primer ajuste de la dinámica lateral y posteriormente tomando los valores obtenidos por en el ajuste de cada dinámica por separado como iniciales se ajusta el conjunto de todo el sistema para tener en cuenta las influencias entre ellas.
Esta división permite reducir tiempos de simulación del algoritmo de auto-tuning ya que es posible simular trayectorias más simples que reduce el tiempo de simulación de cada individuo y tener que ajustar un menor número de parámetros que permite reducir el número de iteraciones necesarias.
El sistema completo se implementa en un único scrip de Matlab que permite la ejecución total de las tres partes y el muestreo de los resultados obtenidos intermedios y finales.

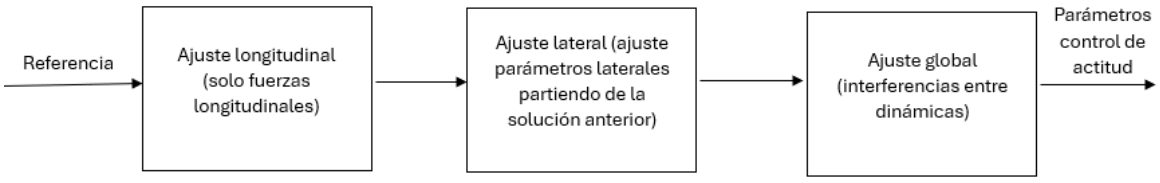

Figura 7.1: Diagrama planteamiento algoritmo genético. Fuente: propia

La estructura global del sistema viene definida por:

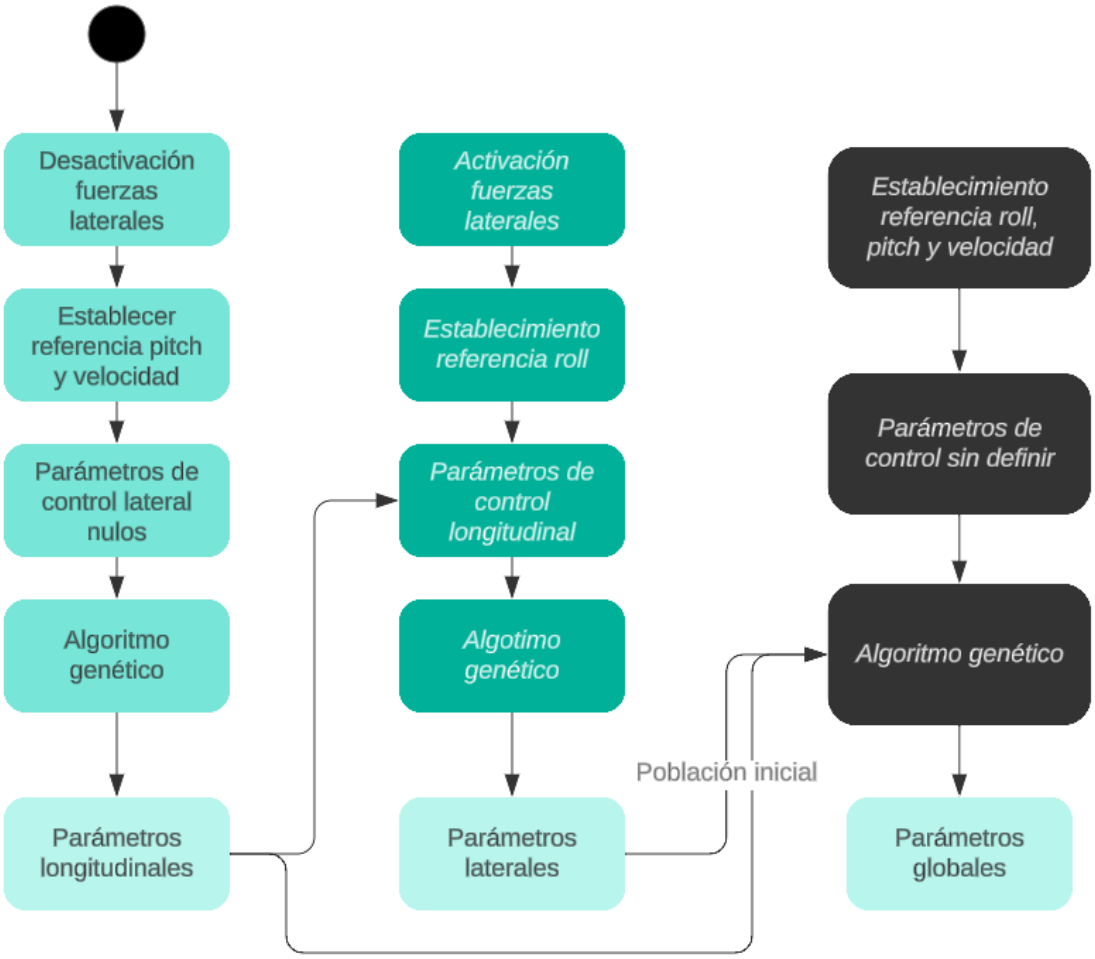

Figura 7.2: Flujograma sistema algoritmo genético. Fuente: propia

7.3 Función objetivo

Una de las partes más importantes de esta implementación es la función objetivo, esta es la condición a minimizar por el algoritmo genético y permitiendo obtener los mejores valores para las variables.

Para el caso de estudio el objetivo de la función es obtener el error medio cuadrático entre unos ángulos de actitud de la aeronave y su velocidad de variación con respecto a los resultados obtenidos simulando los valores de controlador establecidos por el algoritmo genético.

Antes de comenzar con el desarrollo es necesario conocer en que consiste una función que calcula el error medio cuadrático, el error medio cuadrático mide el promedio de los errores al cuadrado, es utilizado para medir la cantidad de error que hay entre dos conjuntos de datos, su valor es siempre positivo siendo el mínimo 0 que significa que los dos conjuntos de datos son iguales. Este error se define por la siguiente fórmula:

$$ECM = \frac{1}{n} * \sum_{i=1}^{n} (Y\_est_i - Y\_ref_i)^2 \tag{7.1}$$

Siendo n el numero de datos a comparar [13].
El código desarrollado para esta función se puede dividir en dos parte claramente diferenciadas. La primera de ellas es la encargada de definir las dos matrices de valores a comparar, en primer lugar se asignan los valores de entrada de la función a cada una de las variables del sistema a utilizar en la simulación y se guardan en el workspace
Posteriormente simula el modelo con las nuevas variables de entrada del algoritmo genético y se cargan los resultados obtenidos obteniendo de esta forma los valores de referencia y los estimados.
Una vez se tienen todos los parámetros obtenidos se procede con el cálculo del error medio cuadrático, aplicando la formula antes mencionada que lo define, para establecer el sumatorio se establece un bucle for que calcula el error de cada una de las variables, finalmente los suma todos obteniendo un error total, este error es el valor que devuelve la función objetivo y el que se debe minimizar por el algoritmo genético.
Para cada parte del problema (longitudinal, lateral y total) se establece una función objetivo en la que se varían las variables a comparar.

7.4 Ajuste dinámica longitudinal

Como se ha comentado en el planteamiento del problema, el ajuste se comienza por las componentes longitudinales, esta dinámica es gobernada por el ángulo de cabeceo por lo que establece que parámetros de control serán los necesarios a ajustar en este caso. El resto de parámetros de control correspondientes a ángulo de alabeo y guiñada se establecen a un valor inicial de 0.
Los parámetros a ajustar son los siguientes definidos junto con su rango:
Conocidos los parámetros se define la referencia a seguir, esta es establecida como una señal

Parámetro	**Rango**
pitch_P	[0,10]
pitch_I	[0,10]
pitch_D	[0,10]
tc_pitch	[0.2,1]
pitch_ff	[0,10]

Tabla 7.1: Rango valores control pitch

escalón con diferentes cambios de referencia tanto positivos como negativos y de diferentes amplitudes, esto permite un buen ajuste ante diferentes referencias; esta entrada puede ser modificada a gusto del usuario para modificar la trayectoria a seguir.
También se establece en este caso una señal de entrada de velocidad variable para ajustar el comportamiento ante este tipo de variaciones.
La posición de la palanca de gases de establece a un valor de crucero de 0.6 ya que su establecimiento depende del modulo de navegación y para estas simulaciones se encuentra desactivada para evitar influencias en el ajuste y simplificar el cálculo.
Con las referencias establecidas, es necesario tener en cuenta un último detalle, para evitar inestabilidades laterales, en el modelo de estimación de la dinámica de la aeronave se desactivan las componentes de fuerzas y momentos laterales dejando únicamente las longitudinales.

Como parámetros del algoritmo genético a definir se establece un número de individuos de cada población de 175 (valor intermedio entre 150-200), y un número de iteraciones de 50, este valor de iteraciones se establece por experiencia para evitar un tiempo excesivo de simulación pero a la vez obtener un buen resultado. Si se requiere de unos resultados más precisos sin limitación temporal se aumentará tanto el número de iteraciones como de individuos.
Con todo lo anterior establecido queda totalmente definida el ajuste longitudinal del sistema.

7.5 Ajuste dinámica lateral

La segunda parte del ajuste de los parámetros del controlador de actitud corresponde a la dinámica lateral de la aeronave gobernada por los ángulos de alabeo y guiñada.
Los parámetros de control a ajustar y sus rangos se determinan en la siguiente tabla:
Para este caso no se puede desacoplar el sistema ya que la influencia de la dinámica longitudi-

Parámetro	**Rango**	**Parámetro**	**Rango**
roll_P	[0,10]	yaw_P	[0,10]
roll_I	[0,10]	yaw_I	[0,10]
roll_D	[0,10]	yaw_D	[0,10]
tc_roll	[0.2,1]	-	-
roll_ff	[0,10]	yaw_ff	[0,10]

Tabla 7.2: Rango valores control roll y yaw

nal en la lateral es elevada, en el modelo de estimación de dinámica se vuelven a conectar las influencias de las fuerzas y momento laterales. Para los parámetros de control de la dinámica longitudinal se toman los obtenidos en el ajuste anterior.
Para ajustar los parámetros se requiere de una referencia a seguir, para este caso se establece un señal escalón para el ángulo de alabeo, esta referencia esta compuesta por ángulos de alabeo positivos, negativos y de diversas amplitudes, para que el ajuste sea lo mejor posible. Las demás variables (ángulo de cabeceo, velocidad y posición de la palanca de gases) se establecen constantes en el tiempo, en el caso del ángulo de cabeceo nulo.
Para el algoritmo genético de este ajuste se establecen los mismos valores de las variables de entrada que para el caso anterior; para este ajuste el número de variables es mayor por lo que serían necesarias más iteraciones que en el ajuste longitudinal, pero para evitar un tiempo excesivo se establece igual. Otro cambio con respecto al caso longitudinal se encuentra en la función objetivo del sistema en la que ahora se han modificado las entradas correspondiendo a los parámetros de control de la dinámica lateral y las componentes del error..

7.6 Ajuste global

El último paso del proceso del auto-tuning consiste en un ajuste conjunto de dinámica lateral y longitudinal partiendo como población inicial los valores de los parámetros obtenidos en las simulaciones anteriores, con este ajuste total se pretende tener en cuenta las influencias entre dinámicas para obtener los parámetros finales del sistema.
En este ajuste final se establecen variaciones en las referencias tanto en el ángulo de alabeo como en el de cabeceo y en la velocidad, tomando conjuntamente las señales establecidas para cada caso, el valor de la palanca de gases se mantiene constante, ya que como se ha explicado en el ajuste longitudinal esta parámetro se define en el control de posición que para el caso de estudio se encuentra desactivado.
Con las referencias establecidas, la siguiente modificación se encuentra en la función objetivo en la que se toman las 14 variables de los controladores como entrada y las variables con las que se calculan el error son ángulos de cabeceo y alabeo y velocidades de variación de alabeo, cabeceo y guiñada.
El número de iteraciones e individuos por población es el mismo que en los casos anteriores al tratarse de un caso final, estos números aumentaran o disminuirán en función del grado de influencia entre la dinámicas.
El código completo de la función se muestra en el anexo del documento.

7.7 Interfaz gráfica

Una vez el sistema implementado es el correcto, se puede realizar una interfaz gráfica que permita obtener un planteamiento y visualización de los datos más visual que el observable a través de un scrip de Matlab.
La interfaz gráfica esta compuesta por seis apartados:

1. Definición de variables de entrada: se establecen como entrada las referencias a seguir como señales escalón definiendo el tiempo de cambio y la amplitud de la señal. También se definen las iteraciones y el número de individuos de cada ajuste.
2. Resultados ajuste longitudinal: gráficas y tablas con los parámetros de control longitudinales y seguimiento de referencias.
3. Resultados ajuste lateral: gráficas y tablas con los parámetros de control laterales y seguimiento de referencias.
4. Resultados numéricos de ajuste global: tablas con los resultados de los parámetros de control totales del sistema de actitud.
5. Resultados gráficos de los parámetros de control de ajuste global: gráficas con la evolución de los parámetros de control con las iteraciones.
6. Resultados gráficos seguimiento de referencia: gráficas de seguimiento de referencias con los parámetros obtenidos del ajuste global.

Figura 7.3: Interfaz: Definición variables. Fuente: propia

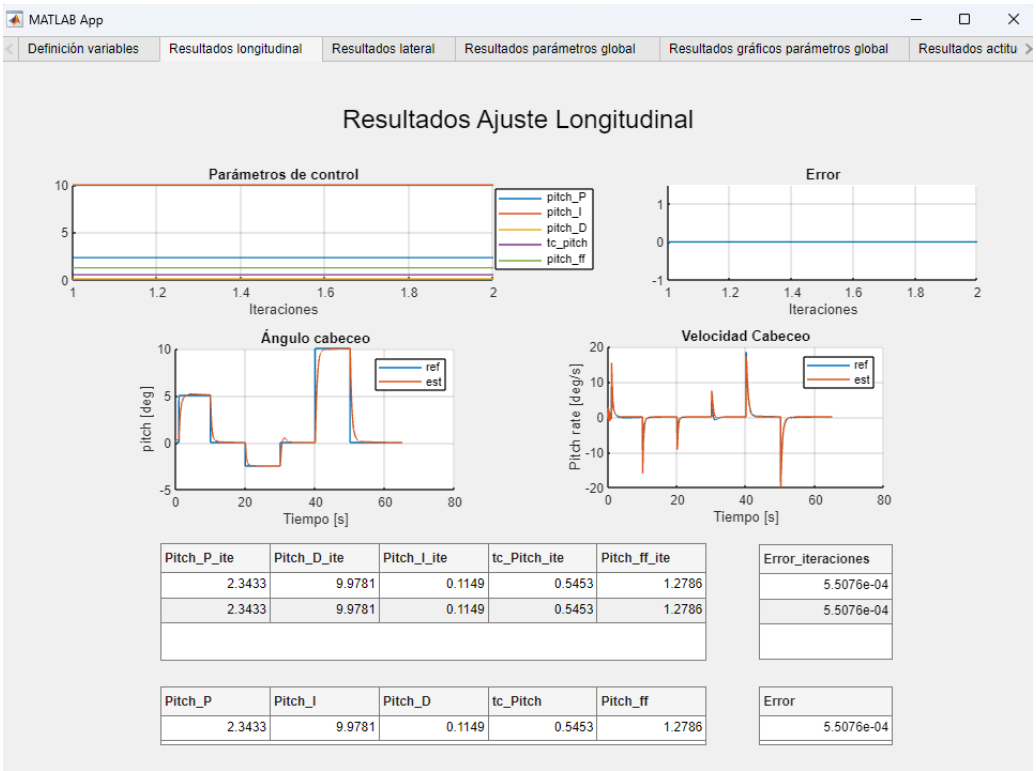

Figura 7.4: Interfaz: Resultados ajuste longitudinal. Fuente: propia

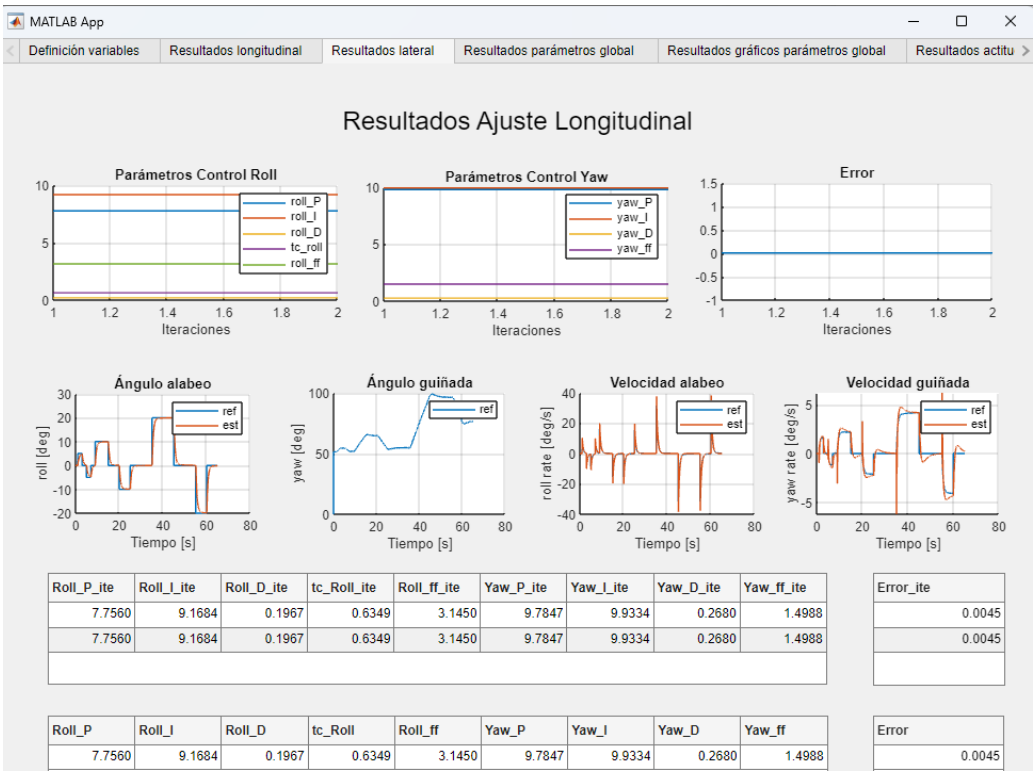

Figura 7.5: Interfaz: Resultados ajuste lateral. Fuente: propia

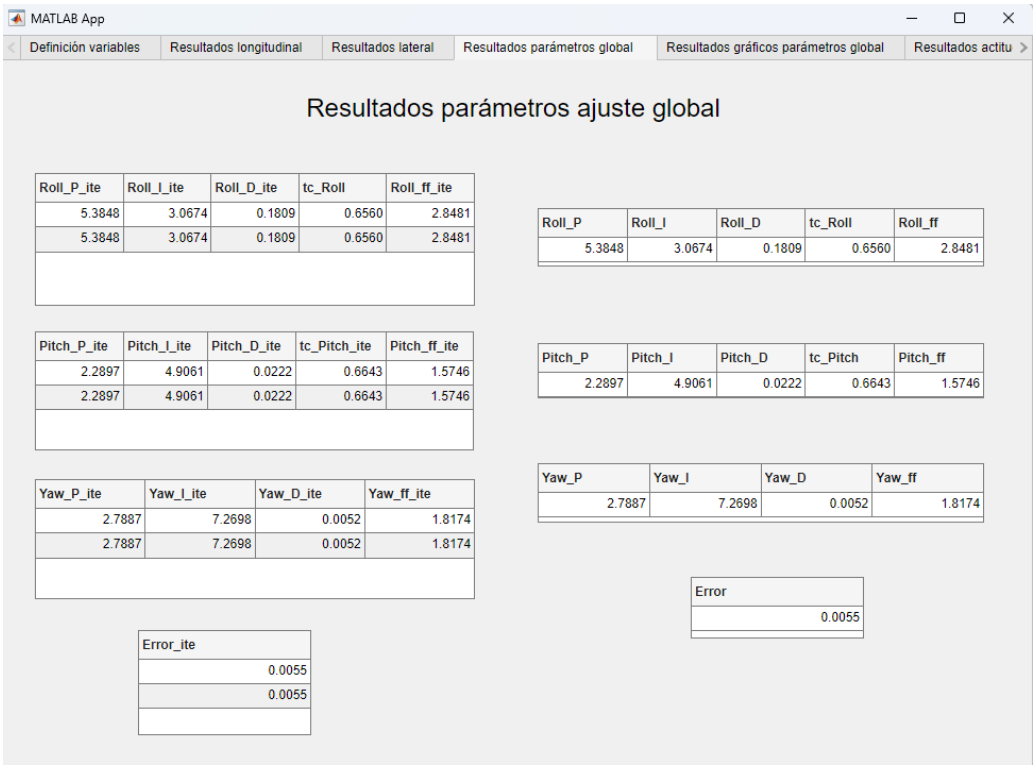

Figura 7.6: Interfaz: Resultados parámetros ajuste global. Fuente: propia

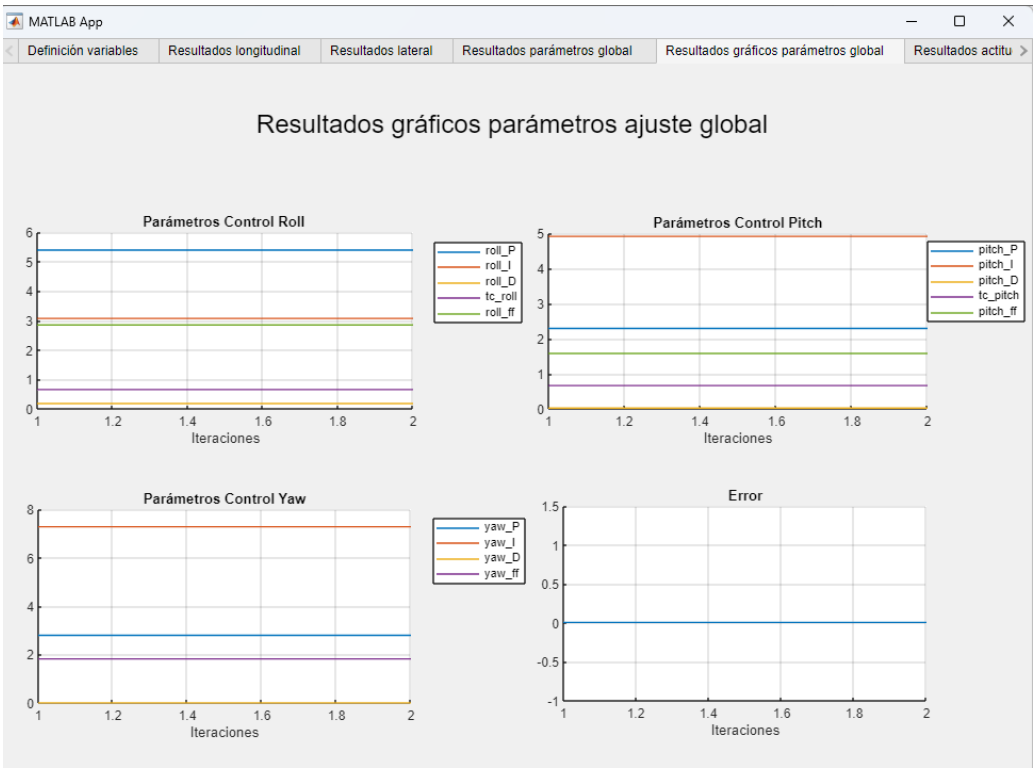

Figura 7.7: Interfaz: Resultados gráficos parámetros ajuste global. Fuente: propia

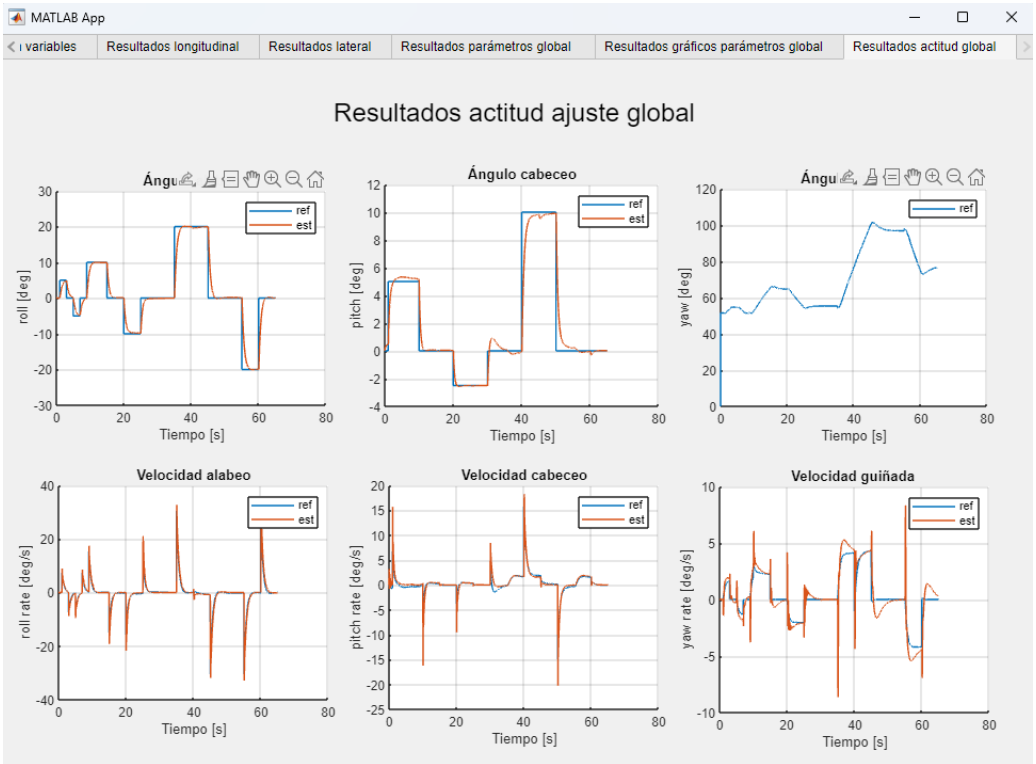

Figura 7.8: Interfaz: Resultados actitud ajuste global. Fuente: propia

Capítulo 8

Resultados

Como apartado final del proyecto desarrollado se exponen los resultados obtenidos del proceso, principalmente en esta sección se van a comentar las soluciones obtenidas del auto-tuning de valores para los diferentes controladores.
Partiendo del primer caso, el ajuste de la dinámica longitudinal ha resultado en los siguientes parámetros de control:

Parámetro	**Valor estimado**
pitch_P	2.3433
pitch_I	9.9781
pitch_D	0.1149
tc_pitch	0.5453
pitch_ff	1.2786
Error	$5{,}5076 * 10^{-}4$

Tabla 8.1: Valores estimados control pitch

Al igual que los valores final también es interesante observar si el modelo se encuentra estabilizado a los valores finales para ello es necesario conocer su evolución con a lo largo de las iteraciones.

De las representaciones de estas variables se observa una buena convergencia de los resultados a un valor final junto con una disminución del error hasta un valor significativamente pequeño. Con esto se comprueba que el número de iteraciones e individuos seleccionados es correcto para obtener un buen resultado.

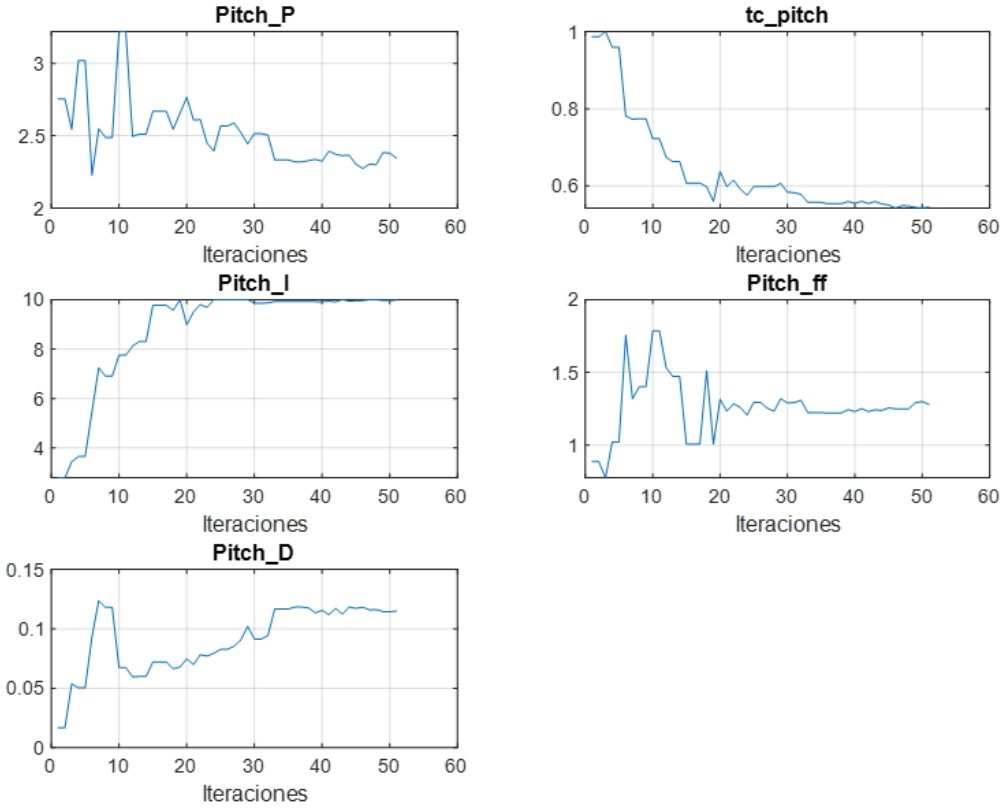

Figura 8.1: Iteraciones parámetros pitch. Fuente: propia

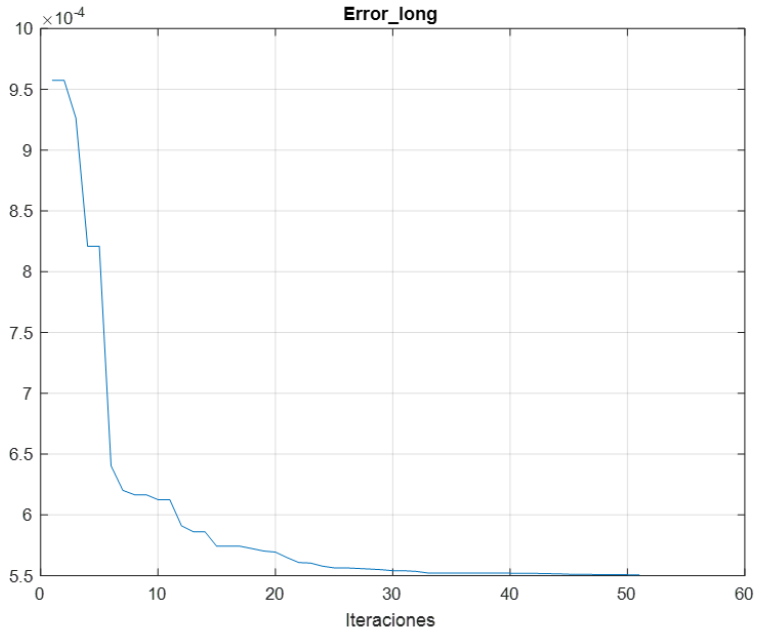

Figura 8.2: Error longitudinal. Fuente: propia

Otra buena comprobación a realizar es si se realiza un buen seguimiento de la referencia establecida por parte de las señales estimadas.

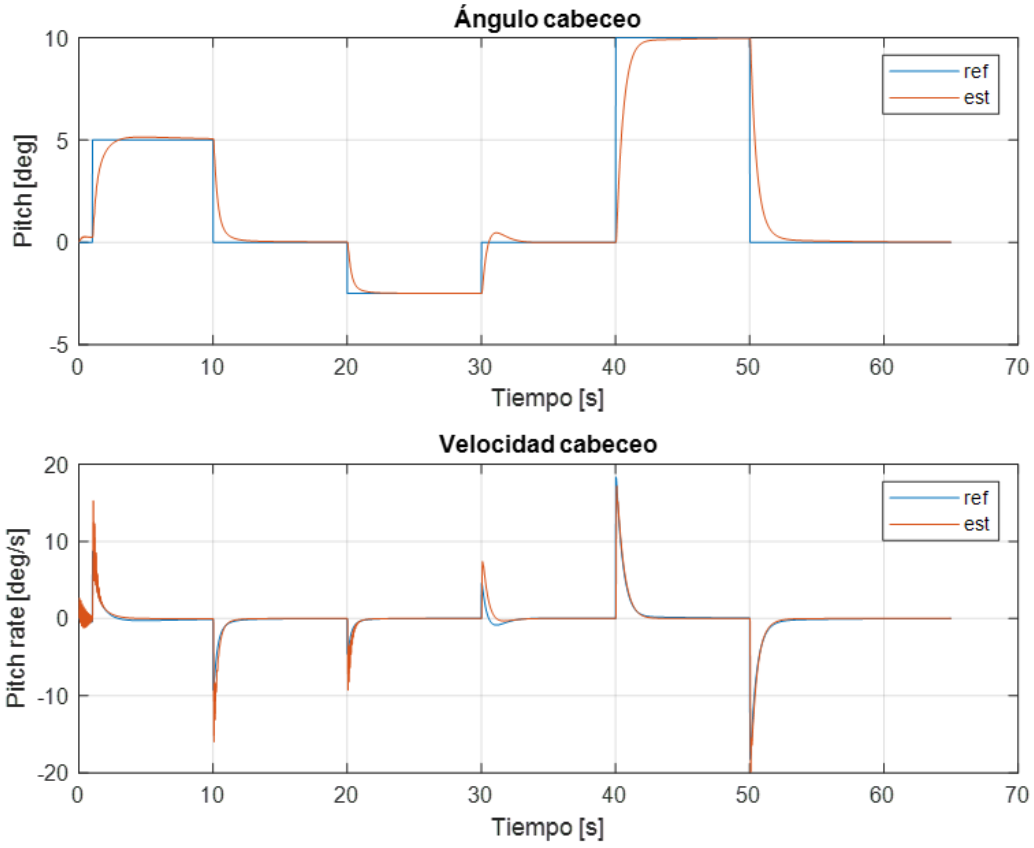

Figura 8.3: Comparación referencias longitudinales. Fuente: propia

Estas dos gráficas son de gran importancia porque permiten observar como se va ha realizar el seguimiento de la referencia, en este caso para el ángulo de cabeceo. En ambas representaciones se observa un fiel seguimiento de la señal de referencia, sin producirse sobre-oscilaciones en la señal, con un error de establecimiento aproximadamente nulo y un tiempo de establecimiento pequeño. Estas características determinan como de brusco es control aplicado, en función de las aplicaciones del vehículos serán necesarios unos requerimientos de control u otros.

Con esto el ajuste longitudinal queda comentado y se proceden con los resultados del ajuste lateral.

Los valores de control obtenidos para este ajuste son los siguientes:

Parámetro	Valor estimado	Parámetro	Valor estimado
roll_P	7.7560	yaw_P	9.7847
roll_I	9.1684	yaw_I	9.9334
roll_D	0.1967	yaw_D	0.2680
tc_roll	0.6349	-	-
roll_ff	3.1454	yaw_ff	1.4988
Error		0.0045	

Tabla 8.2: Valores estimados control roll y yaw

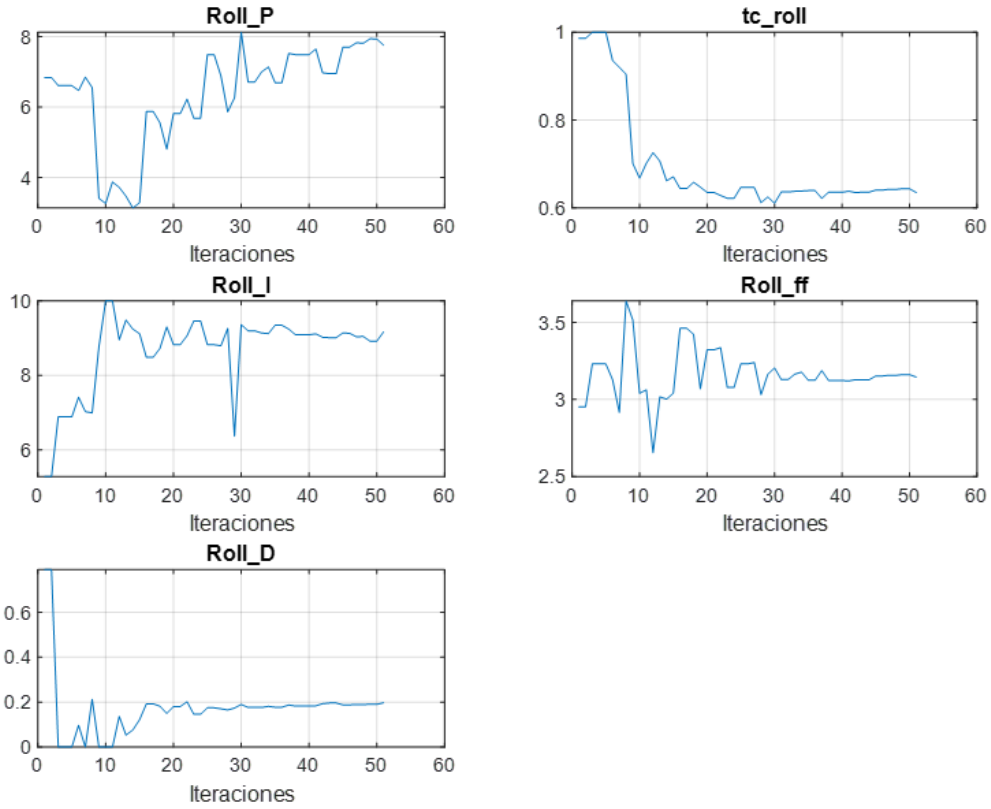

Figura 8.4: Iteraciones parámetros roll. Fuente: propia

Las representaciones de las parámetros de control sirven como comprobación de convergencia de las soluciones, observando las gráficas se puede ver que esta comprobación es correcta obteniendo un valor para cada control aproximadamente convergido. Otro parámetro a destacar reside en la aparición de un error mayor que en el caso longitudinal, esto se debe a un mayor número de variables que contribuyen al error lo que hace que la suma total sea más elevada.

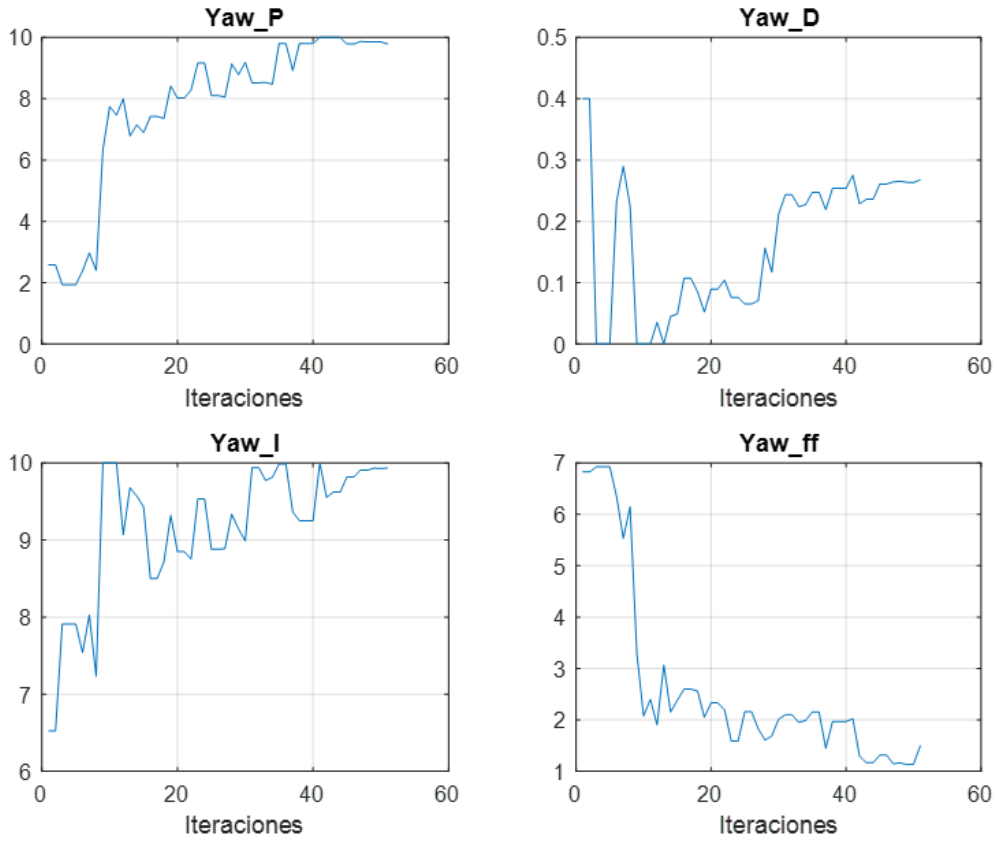

Figura 8.5: Iteraciones parámetros yaw. Fuente: propia

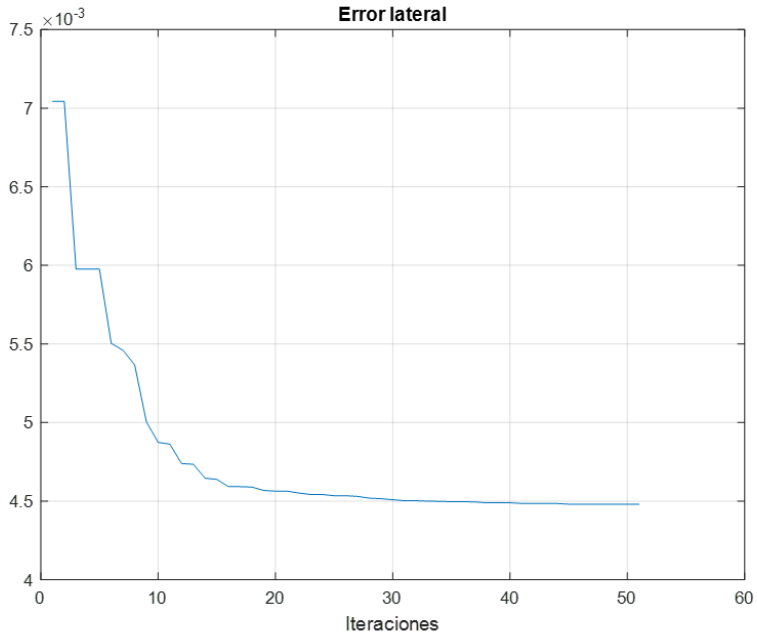

Figura 8.6: Error lateral. Fuente: propia

También es interesante observar como se realiza el seguimiento de la referencia.

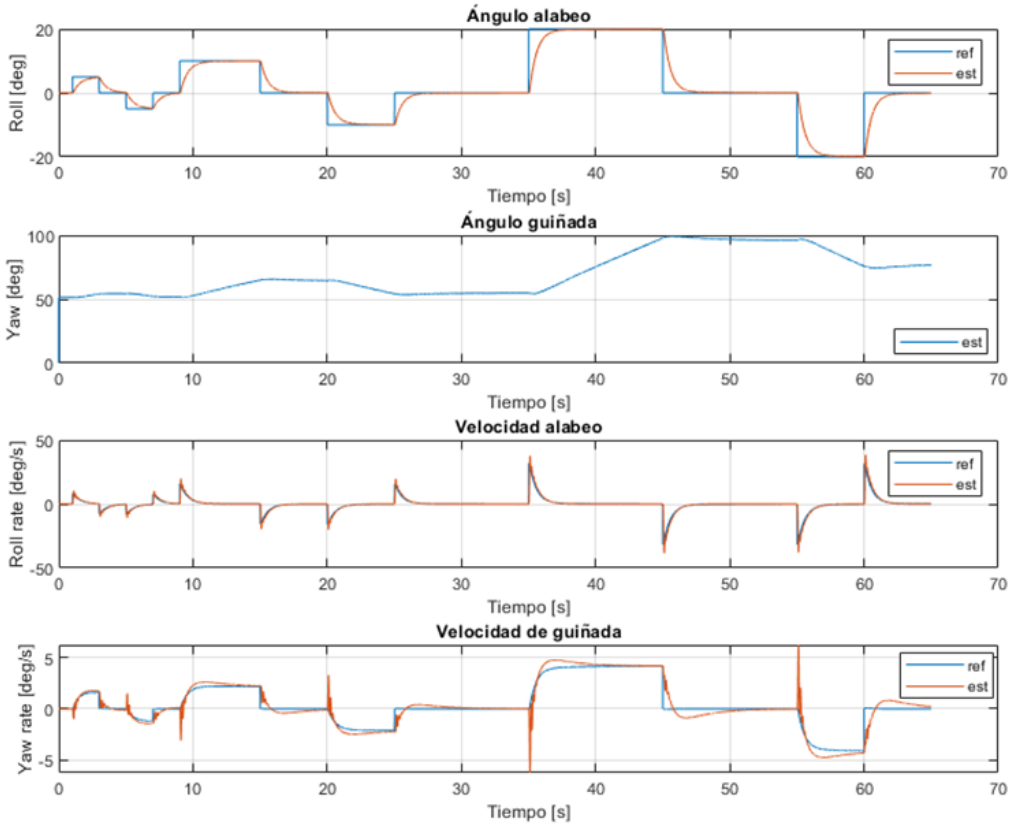

Figura 8.7: Comparación referencias laterales. Fuente: propia

Las conclusiones obtenidas de estas gráficas son similares a las del caso longitudinal, el seguimiento de la referencia es correcto sin sobre-oscilaciones y con un error de establecimiento muy pequeño; por lo que se puede concluir que el ajuste realizado es correcto. De este esquema puede llamar la atención la gráfica del ángulo de guiñada ya que no existe referencia, esto se debe a que la velocidad de guiñada se obtiene por la fórmula de giro coordinado y no por el error entre ángulo de guiñada de referencia y el estimado.

Una vez realizas las primeras estimaciones de los parámetros de control por separado se ajustan conjuntamente para observar influencias entre dinámicas.

Parámetro	**Valor est.**	**Parámetro**	**Valor est.**	**Parámetro**	**Valor est.**
roll_P	5.3848	pitch_P	2.2897	yaw_P	2.7887
roll_I	3.0674	pitch_I	4.9061	yaw_I	7.2698
roll_D	0.1809	pitch_D	0.0222	yaw_D	0.0052
tc_roll	0.6560	tc_pitch	0.6643	-	-
roll_ff	2.8481	pitch_ff	1.5746	yaw_ff	1.8174
	Error			0.0055	

Tabla 8.3: Valores estimados control final

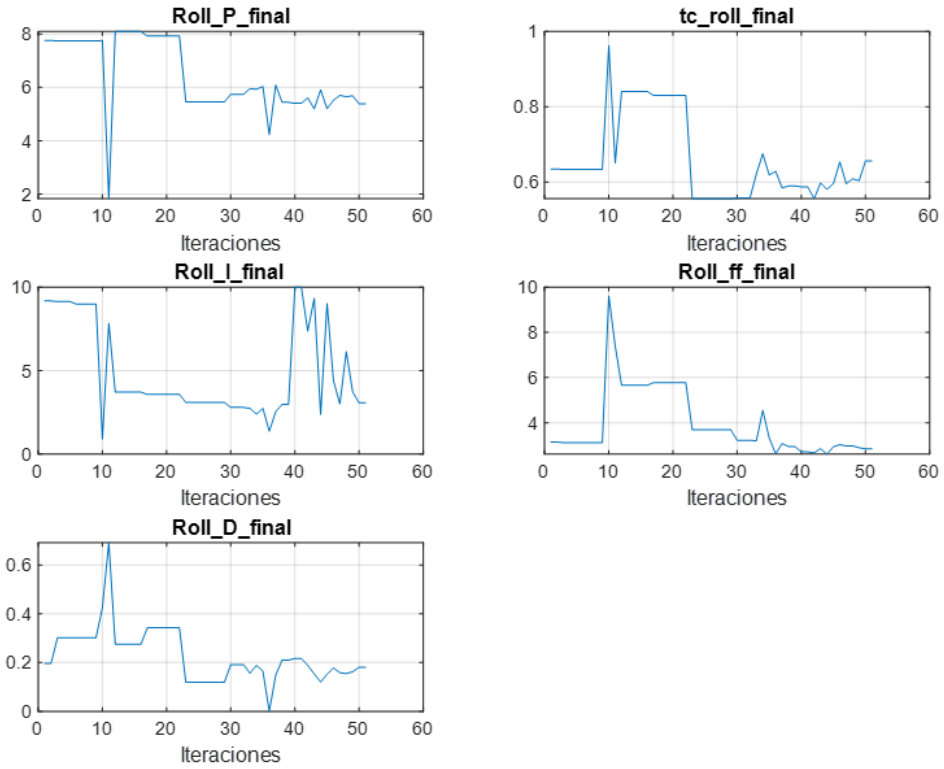

Figura 8.8: Iteraciones parámetros roll finales. Fuente: propia

Como en los casos anteriores se estudia la convergencia, para este ajuste final la estabilidad de alguno de los controles no es tan clara como en los casos anteriores, para observar una mejor convergencia serían necesarias un mayor número de iteraciones para terminar de ajustar más fielmente el sistema. Aun así la respuesta obtenida se puede tomar como correcta ya que el error es pequeño y aproximadamente estabilizado, el valor es mayor a los anteriores porque tiene un mayor número de contribuciones.

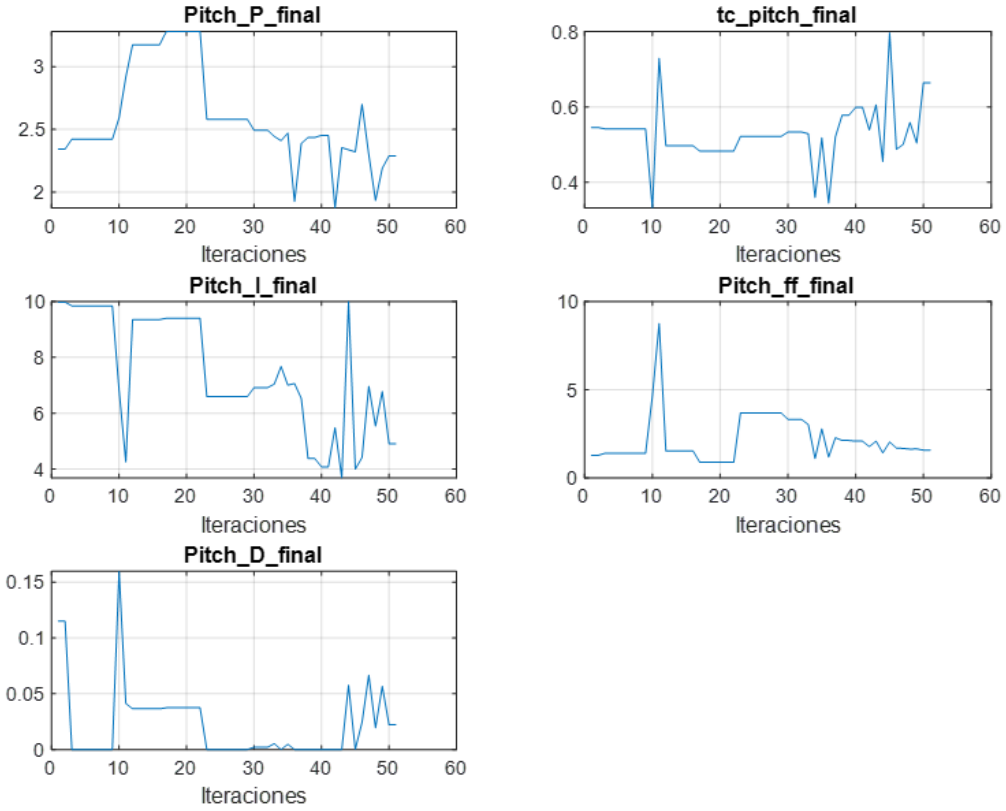

Figura 8.9: Iteraciones parámetros pitch finales. Fuente: propia

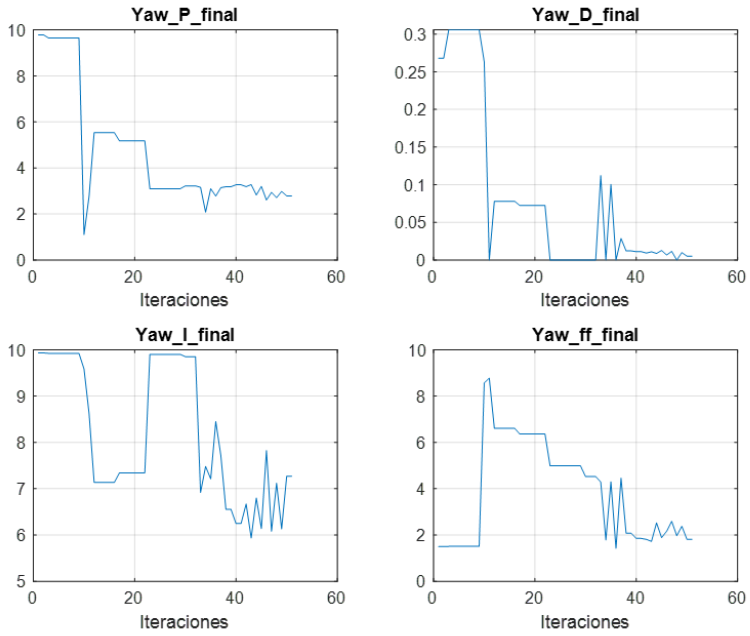

Figura 8.10: Iteraciones parámetros yaw finales. Fuente: propia

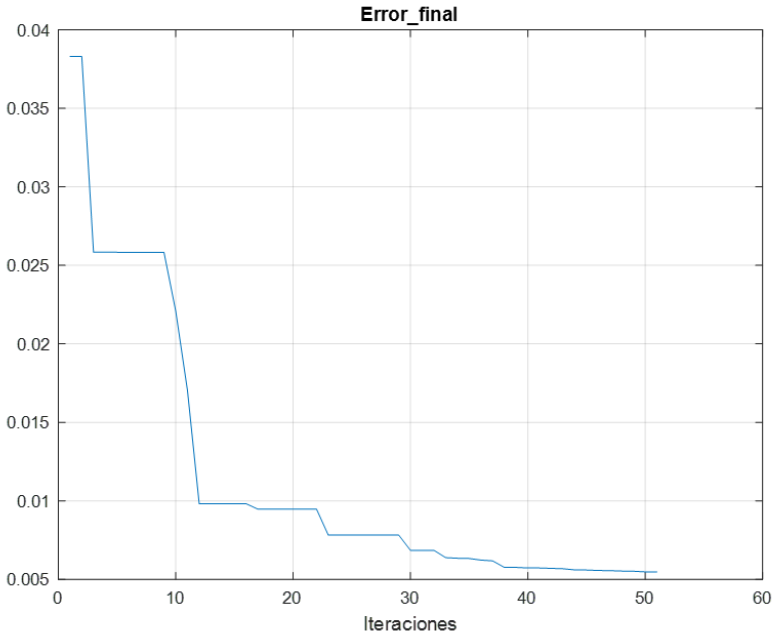

Figura 8.11: Error global. Fuente: propia

Observando finalmente el seguimiento de las referencias con los parámetros globales:

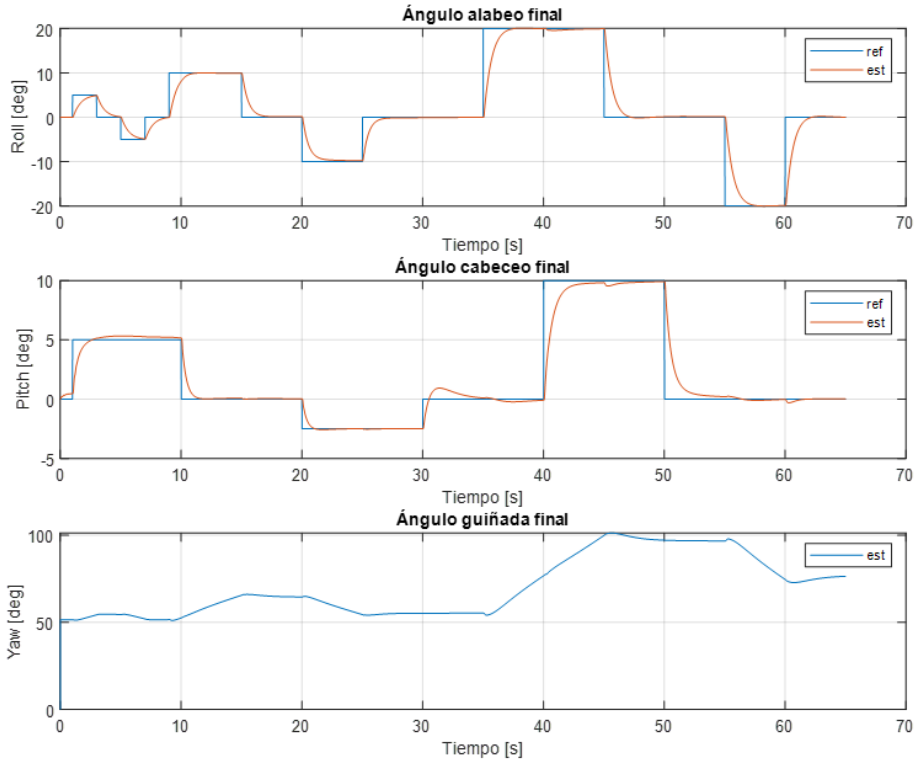

Figura 8.12: Comparación referencias en ángulos. Fuente: propia

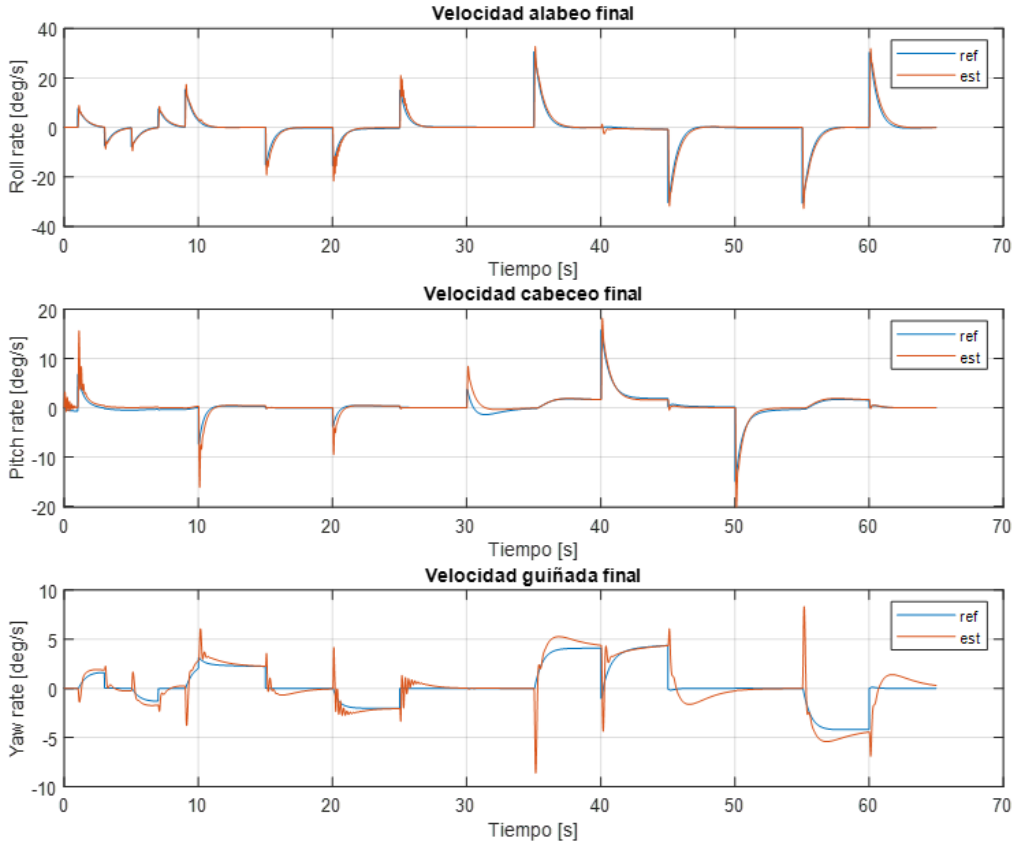

Figura 8.13: Comparación referencias en velocidades. Fuente: propia

Para finalizar, el seguimiento de las referencias globales, al incorporar variaciones tanto en ángulo de alabeo como de cabeceo, se realiza de forma fiel al igual que en los casos de desacople del sistema. Una conclusión global del sistema ya desarrollada en el comentario longitudinal es el tipo de respuesta obtenida, en este caso es una respuesta rápida que pretende alcanzar lo más rápido posible el nuevo valor de referencia, esto provoca que las acciones sean muy bruscas, dependiendo de la aplicación esto será bueno o malo.

Con todo lo desarrollado anteriormente se verifica el correcto funcionamiento de ajuste del sistema implementado de auto-tuning.

El código completo de este sistema se puede encontrar en el anexo del documento.

Para terminar de verificar los datos anteriores se realizan varias simulaciones con los resultados anteriores.
En la primera de ellas se compara el modelo de PX4 con ambos valores de controladores para observar como varía la trayectoria con el nuevo ajuste y verificar el seguimiento de las referencias en el modelo global:

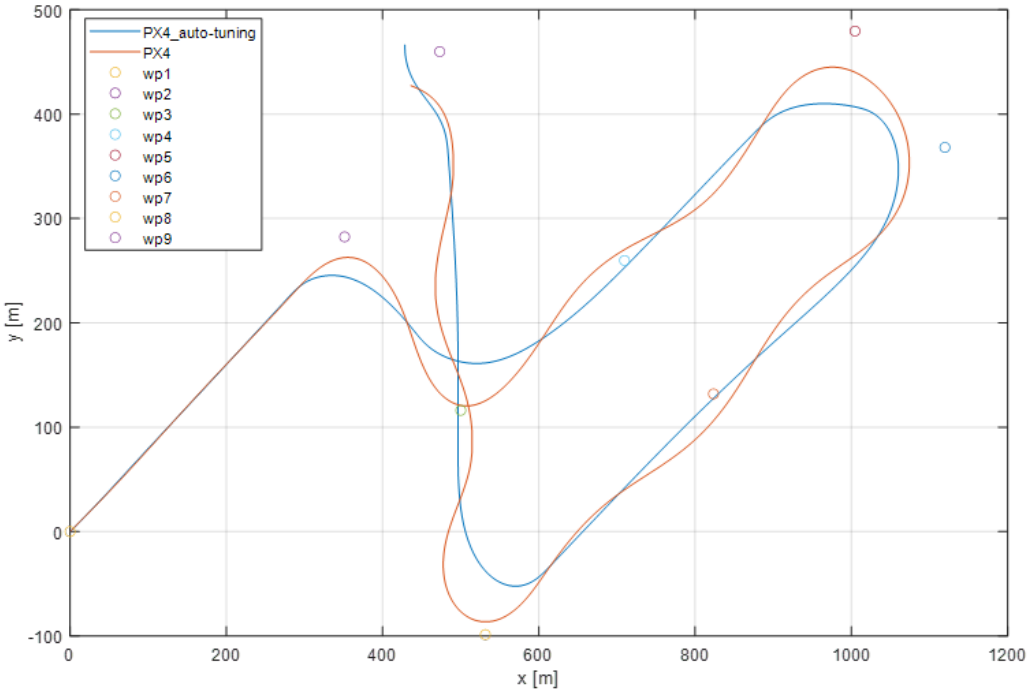

Figura 8.14: Trayectoria PX4 vs PX4 auto-tuning. Fuente: propia

Observando los resultados, se puede ver una gran relación entre ambas trayectorias, las diferencias se deben, a como ha sido antes mencionado, el modelo ajustado por el sistema de auto-tuning establece un control más brusco ante respuestas por ese motivo los giros se realizan con anterioridad ya que en el instante en el que se cambia de waypoint intenta seguir lo antes posible al siguiente, al igual que en los ascensos y descensos la pendiente es ligeramente más pronunciada. Aun así se observa que en el modelo de auto-tuning se producen menos oscilaciones lo que lleva a un mejor seguimiento de la referencia y un control más robusto.

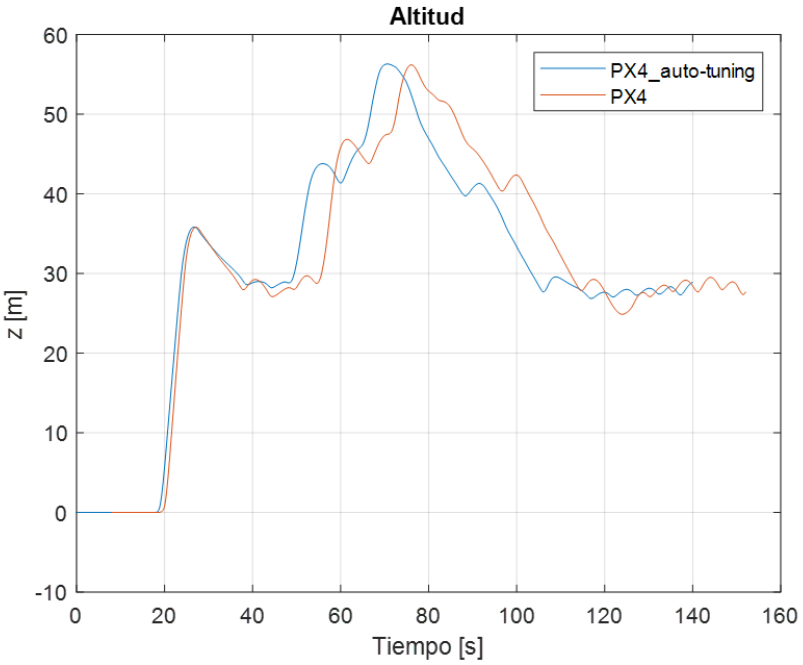

Figura 8.15: Altitud PX4 vs PX4 auto-tuning. Fuente: propia

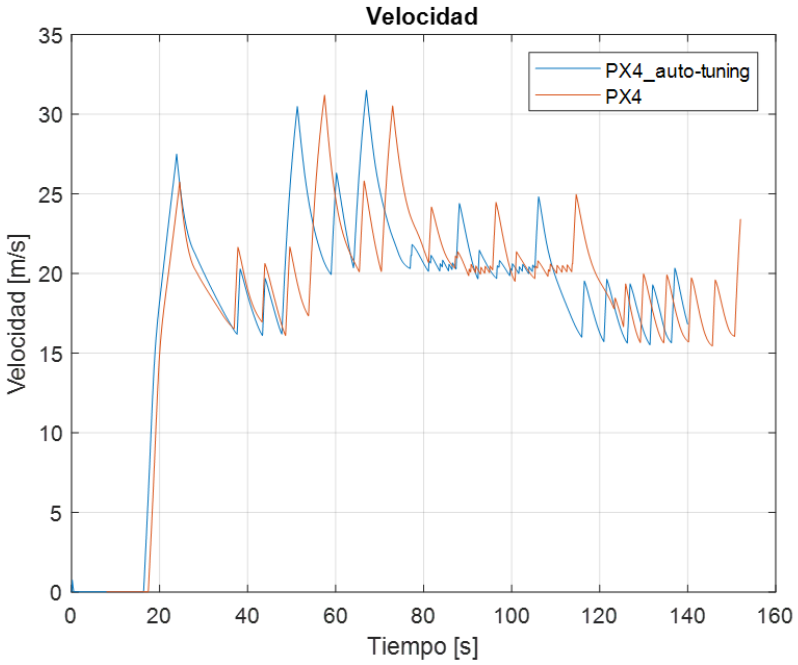

Figura 8.16: Velocidad PX4 vs PX4 auto-tuning. Fuente: propia

La segunda comprobación ha realizar se centra en una comparación entre los resultados obtenidos por el modelo implementado de Simulink y el modelo de referencia de PX4 estableciendo en ambos los parámetros de control obtenidos mediante el proceso de auto-tuning. Las gráficas obtenidas son:

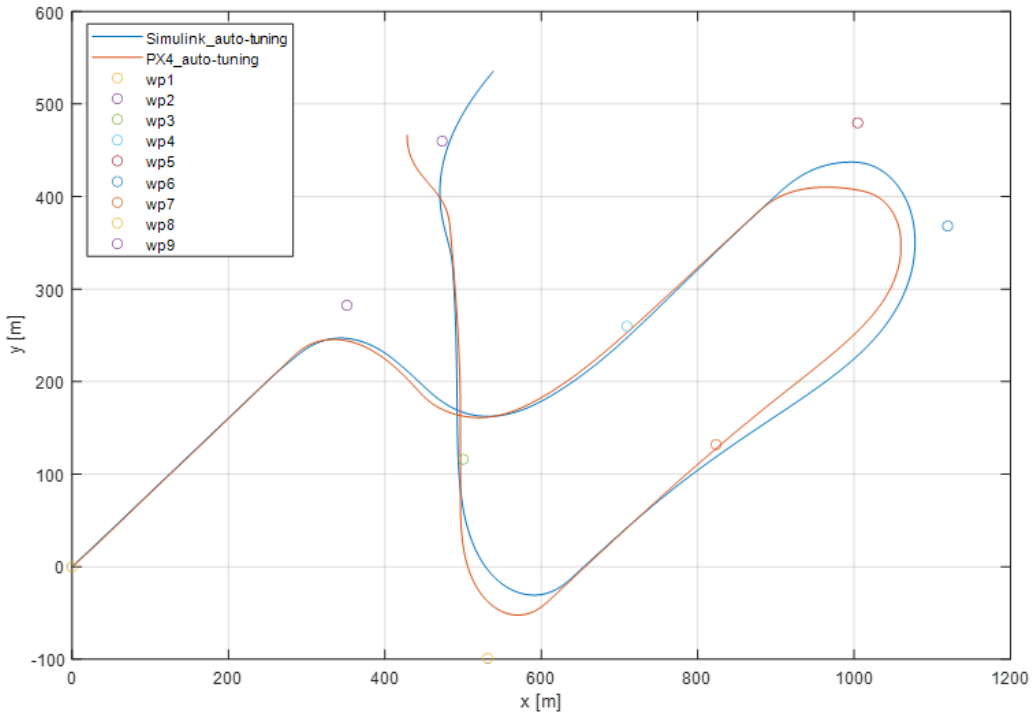

Figura 8.17: Trayectoria Simulink auto-tuning vs PX4 auto-tuning. Fuente: propia

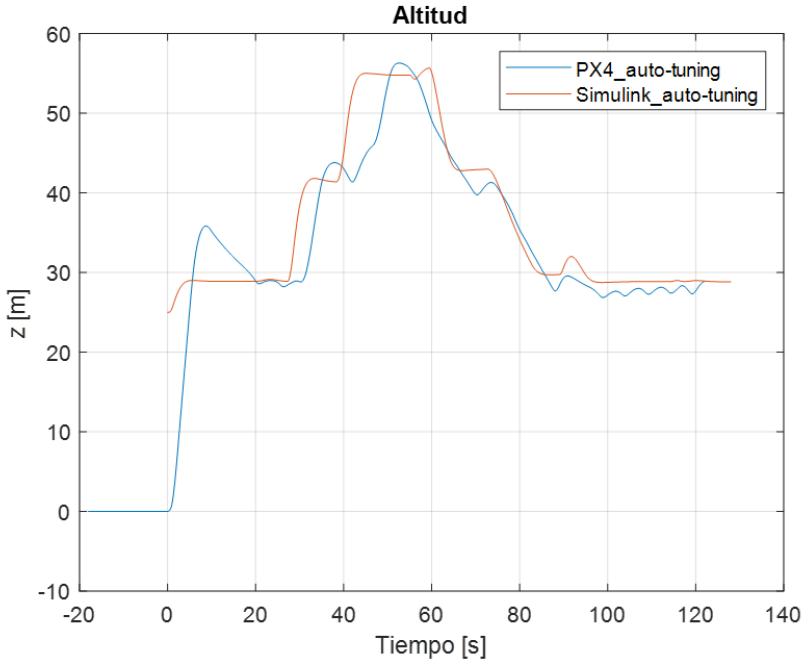

Figura 8.18: Altitud Simulink auto-tuning vs PX4 auto-tuning. Fuente: propia

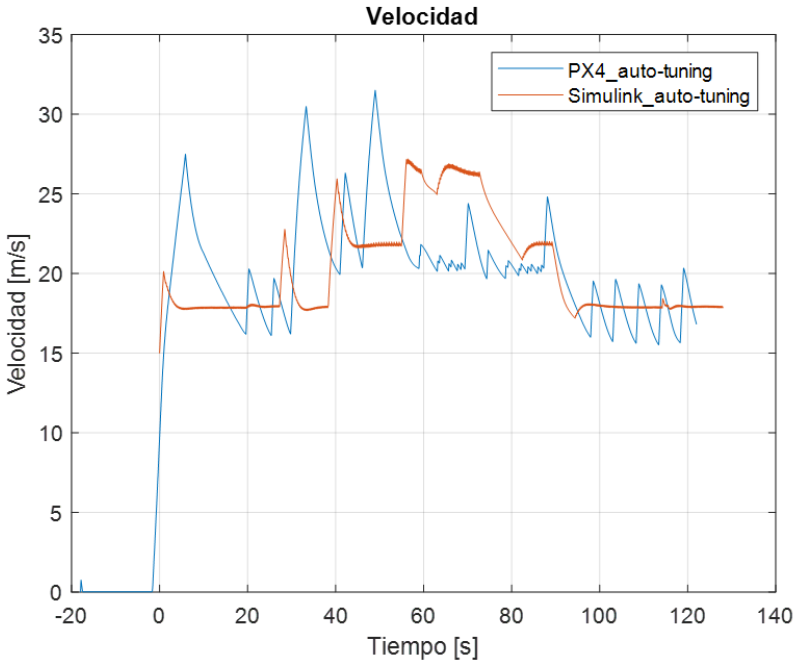

Figura 8.19: Velocidad Simulink auto-tuning vs PX4 auto-tuning. Fuente: propia

Obviando las diferencias entre los modelos establecidas en el punto de validación del sistema implementado en Simulink, se observan que las referencias se siguen correctamente en todos los casos obteniendo aproximadamente los mismos resultados con ambos modelos.

Como conclusión global de este apartado, se ha logrado obtener un sistema que permite estimar los parámetros de control de actitud de una determinada aeronave con un sistema de auto-piloto PX4 mediante optimización global. Este proceso es realizado en tres fases permitiendo disminuir el tiempo necesario y se establece una validación comparando las soluciones con diferentes casos.

Capítulo 9

Conclusiones

9.1 Conclusiones

El objeto principal del proyecto era el desarrollo de un sistema de auto-tuning del control de actitud de un auto-piloto PX4 mediante optimización global. Para llevar a cabo este desarrollo se opto por la implementación del sistema de control al completo (control de posición y de actitud) en un esquema de Simulink permitiendo obtener un único sistema se simulación de trayectorias al incorporar el diagrama de estimación de la dinámica. Con esto se obtiene un sistema de control equivalente a PX4 pero implementado en Simulink facilitando la tarea de aplicación de métodos de auto-tuning.

Para cumplir el objetivo principal del proyecto de obtener un sistema de auto-tuning se hace uso de una optimización global a partir de algoritmos genéticos, este sistema de optimización se encuentra definido en Matlab por lo que gracias a la implementación del sistema de control en Simulink la conexión entre ambos se realiza de manera rápida y sencilla, obtenido en sistema robusto.

Todos los sistemas implementados requieren de sus respectivas validaciones para comprobar que en todos los casos los modelos son correctos y equivalentes a los de referencia, por este motivo se comprueba tanto la implementación en Simulink de PX4 como las soluciones del auto-ajuste de los parámetros de control, concluyendo en ambos casos que las soluciones obtenidas se pueden tomar como correctas.

Para finalizar el proyecto, se destaca que se han cumplido todos los objetivos establecidos al inicio del proyecto, concluyendo con un sistema de auto-tuning que es capaz de ajustar el control de actitud de una determinada aeronave.

Capítulo 10

Anexo

10.1 Objetivos del desarrollo sostenible agenda 2030

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

Tabla 10.1: Caption

Este proyecto se relaciona en mayor medida con el ODS 9. Industria, innovación e infraestructuras, ya que supone un desarrollo en el mundo de los UAVs que pueden ser implementados para diversas aplicaciones tecnológicas optimizando los trabajos.

10.2 Funciones variables de entrada

```
function [ste_rate_to_throttle,ste_rate_max,ste_rate_min]  = ste(max_climb_rate,min_sink_rate)
% Cambio de ste_rate a throttle
 g = 9.81;
ste_rate_max = max(max_climb_rate,10^-6)*g;
ste_rate_min = -max(min_sink_rate,10^-6)*g;

ste_rate_to_throttle = 1/(ste_rate_max-ste_rate_min);
end
```

Figura 10.1: Función ste

```
function airspeed_rate_output = V_rate(ste_rate_max,ste_rate_min,V_TAS,airspeed_rate)
% Limitación variación velocidad (aceleración)
max_tas_rate_sp = 0.5*ste_rate_max/max(V_TAS,10^-6);
min_tas_rate_sp = 0.5*ste_rate_min/max(V_TAS,10^-6);

airspeed_rate_output = min(max(airspeed_rate,min_tas_rate_sp),max_tas_rate_sp);
end
```

Figura 10.2: Función V_rate

```
function [weight_spe,weight_ske] = weight(ratio_undersped,pitch_speed_weight)
% Factores ponderación balance de energía
pitch_speed_weight = min(max(pitch_speed_weight,0),2);

if ratio_undersped>10^-6
    pitch_speed_weight = 2*ratio_undersped+(1-ratio_undersped)*pitch_speed_weight;
end

weight_spe = min(max(2-pitch_speed_weight,0),1);
weight_ske = min(max(pitch_speed_weight,0),1);
end
```

Figura 10.3: Función weight

```
function ratio_undersped = ratio_underspeed(tas_error_percentage,equivalent_airspeed_trim,tas_min,tas)
% Ratio underspeed
tas_error_bound = tas_error_percentage*equivalent_airspeed_trim;
tas_undersped_soft_bound = tas_error_bound*equivalent_airspeed_trim;

tas_fully_undersped = max(tas_min-tas_error_bound-tas_undersped_soft_bound,0);
tas_starting_to_undersped = max(tas_min-tas_error_bound,tas_fully_undersped);

ratio_undersped = 1-((norm(tas)-tas_fully_undersped)/max(tas_starting_to_undersped-tas_fully_undersped,10^-6));
end
```

Figura 10.4: Función ratio_underspeed

```
function [course, gndspeed] = cour_gnds(vxyz)
% Rumbo estimado y groundspeed
vx = vxyz(1);
vy = vxyz(2);

course = atan2(vy,vx);
gndspeed = norm([vx,vy]);
end
```

Figura 10.5: Función cour_gnds

10.3 Funciones Control de posición

10.3.1 Seguimiento waypoints

```
function [wps, alt]= waypoints_alt(toWP, fromWP)
% Waypoints y altitud de referencia
    wps = zeros(2,3);
    wps = [fromWP'; toWP'];
    alt = -toWP(3);
end
```

Figura 10.6: Función waypoints_alt

```
function velocidad  = vel(Vcruise, current_wp,poswp,nwap)
% Velocidad de referencia
velocidad = Vcruise(1);
for i = 1:nwap
    if current_wp(1) == poswp(i,1)
        velocidad = Vcruise(i);
    end
end
```

Figura 10.7: Función vel

10.3.2 Total Energy Control Loop

```
function T_FF  = T_predicted(ste_rate,ste_rate_max, ste_rate_min, throttle_max, throttle_min, throttle_trim)
% Throttle predecido
throttle_above_trim_per_ste_rate = (throttle_max - throttle_trim)/ste_rate_max;
throttle_below_trim_per_ste_rate = (throttle_trim - throttle_min)/ste_rate_min;

if ste_rate >= 10^-6
    T_FF = throttle_trim + ste_rate*throttle_above_trim_per_ste_rate;
else
    T_FF = throttle_trim - ste_rate*throttle_below_trim_per_ste_rate;
end
```

Figura 10.8: Función T_predicted

```
function throttle_integ_state = T_integ(throttle_max,throttle_min, throttle_integ_input)
% Limitacion integrador throttle
if throttle_integ_input >= throttle_max
    throttle_integ_input = min(0,throttle_integ_input);
elseif throttle_integ_input<= throttle_min
    throttle_integ_input = max(0,throttle_integ_input);
end
    throttle_integ_state = throttle_integ_input;
end
```

Figura 10.9: Función T_integ

10.3.3 Total Energy Balance Control Loop

```
function pitch_setpoint_1  = seb_to_pitch(SEB_rate_correction,V_TAS)
% Balance de energía a pitch
g = 9.81;
airspeed_for_seb_rate = 15;

if V_TAS > 10^-6

airspeed_for_seb_rate = V_TAS;
end

climb_angle_to_SEB_rate = airspeed_for_seb_rate*g;

pitch_setpoint_1 = SEB_rate_correction/climb_angle_to_SEB_rate;
end
```

Figura 10.10: Función seb_to_pitch

```
function pitch_integ_state = p_integ(pitch_max,pitch_min,pitch_integ_input)
% Limitación integrador pitch
if pitch_integ_input >= pitch_max
    pitch_integ_input = min(pitch_integ_input,0);
elseif pitch_integ_input <= pitch_min
    pitch_integ_input = max(pitch_integ_input,0);
end
pitch_integ_state = pitch_integ_input;
end
```

Figura 10.11: Función p_integ

```
function pitch_increment = pitch_incre(vert_accel_limit,V_TAS)
% Incremento de pitch
pitch_increment = vert_accel_limit / max(V_TAS,10^-6);
end
```

Figura 10.12: Función pitch_incre

```
function pitch_setpoint = pitch_incre_lim(pitch_setpoint,pitch_increment)
% Limitación pitch setpoint
pitch_setpoint = min(max(pitch_setpoint,pitch_setpoint-pitch_increment),pitch_setpoint+pitch_increment);
end
```

Figura 10.13: Función pitch_incre_lim

10.3.4 L1 Control

```
function eta = fcn(eta)
% Corrección eta
eta = mod(eta,2*pi);
if eta>pi
    eta = eta-2*pi;
end
```

Figura 10.14: Función eta

10.4 Funciones Control Actitud

```
function [euler_rate_setpoint_yaw,euler_rate_setpoint_pitch] = euler_rate(roll,roll_setpoint,pitch,airspeed,pitch_setpoint,tc)
g = 9.81;
% Euler rate yaw
contrained_roll = min(max(roll,-abs(roll_setpoint)),abs(roll_setpoint));    % Limitación roll
euler_rate_setpoint_yaw = tan(contrained_roll)*cos(pitch)*g/airspeed;       % Turn coordination

% Euler rate pitch
pitch_error = pitch_setpoint-pitch;             % Error pitch
euler_rate_setpoint_pitch = pitch_error/tc;     % Control proporcional
end
```

Figura 10.15: Función euler_rate

```
function roll_body_rate_setpoint_raw = roll_crtl(roll_setpoint,roll,euler_yaw_rate_setpoint,tc,pitch)
% Control roll
roll_error = roll_setpoint-roll;        % Error roll
euler_rate_setpoint = roll_error /tc;   % Control proporcional
roll_body_rate_setpoint_raw = euler_rate_setpoint - sin(pitch)*euler_yaw_rate_setpoint;     % Roll rate en ejes cuerpo
end
```

Figura 10.16: Función roll_crtl

```
function pitch_body_rate_setpoint_raw  = pitch_crtl(pitch,euler_yaw_rate_setpoint,roll,euler_rate_setpoint_pitch)
% Cambio pitch rate a ejes cuerpo
pitch_body_rate_setpoint_raw = cos(roll)*euler_rate_setpoint_pitch+cos(pitch)*sin(roll)*euler_yaw_rate_setpoint;
end
```

Figura 10.17: Función pitch_crtl

```
function yaw_body_rate_setpoint_raw    = yaw_crtl(roll,pitch,euler_pitch_rate_setpoint,euler_rate_setpoint_yaw)
% Cambio yaw rate a ejes cuerpo
yaw_body_rate_setpoint_raw = -sin(roll)*euler_pitch_rate_setpoint+cos(roll)*cos(pitch)*euler_rate_setpoint_yaw;
end
```

Figura 10.18: Función yaw_crtl

10.5 Funciones auto-tuning

10.5.1 Algoritmo genético

Se va ha exponer el código del sistema longitudinal, sin embargo el auto-tuning de la dinámica lateral y del modelo global se realizar siguiendo la misma estructura pero con las modificaciones comentadas anteriormente.

```
%% Código auto-tuning
% En este código se realiza una implementación de un sistema de auto-tuning
% para parámetros de control de un auto-piloto PX4.

timestop = 65;      % Define el tiempo final de la simulación
%% Ajuste dinámica longitudinal
    % Se desactivan las fuerzas laterales del modelo, un valor de 1 indica
    % que se activan y un valor de 0 que se desactivan.
    F_lon = 1;
    F_lat = 0;

    % Define referencia a seguir
        nav = 0;        % Si nav = 0 se desactiva el control de posición, si nav = 0↙
se activa el control de posición

    % Pitch ref: se establece como una entrada escalón
tiempo_pitch = [0, 1, 1, 10, 10, 20, 20, 30, 30, 40, 40, 50, 50, 60];              %↙
Tiempos de cambio de referencia
amplitud_pitch = deg2rad([0, 0, 5, 5, 0, 0, -2.5, -2.5, 0, 0, 10, 10, 0, 0]);      %↙
Amplitudes correspondientes de ángulo de cabeceo en grados
pitch_sp =timeseries(amplitud_pitch,tiempo_pitch);

    % Roll ref: se establece como un valor constante y 0 para el caso
    % longitudinal
tiempo_roll = [0];                                      % Tiempos de cambio
amplitud_roll= deg2rad([0]);                            % Amplitudes correspondientes↙
de ángulo de alabeo en grados
roll_sp = timeseries(amplitud_roll,tiempo_roll);
```

```
    % Velocidad ref: se establece como una entrada escalón
tiempo_vel = [0, 10, 10, 20, 20, 30, 30, 40];          % Tiempos de cambio de
referencia
amplitud_vel= [15, 15, 20, 20, 25, 25, 15, 15];        % Amplitudes correspondientes
de velocidad en m/s
velocidad_sp = timeseries(amplitud_vel,tiempo_vel);

    % Throttle ref: se establece como un valor constante ya que este es
    % calculado por el control de posición.
throttle_sp = 0.6;

% Sistema simulado
    % Controles laterales: se establecen a un valor inicial de 0
    roll_P = 0;
    roll_I= 0;
    roll_D = 0;
    tc_roll = 1;
    roll_ff = 0;
    yaw_P = 0;
    yaw_I = 0;
    yaw_D = 0;
    yaw_ff = 0;

    % Controles longitudinales
    clear ("pitch_P","pitch_I","pitch_D","tc_pitch","pitch_ff");
    lb_pitch = [0 0 0 0.2 0];       % Límite inferior de los parámetros de control



longitudinales
    ub_pitch = [10 10 10 1 10];     % Límite superior de los parámetros de control
longitudinales

    % Algoritmo genético
    InitSimulinkPlantVars      %Inicializa los parámetros del modelo de Simulink
    gaDat.Objfun = 'Error_Medio_Cuadratico_lon';       % Función objetivo dinámica
longitudinal
    lb = lb_pitch;
    ub = ub_pitch;
    gaDat.FieldD = [lb; ub];
    gaDat.indini = [];
    gaDat.MAXGEN = 50;     % Máximo número de iteraciones
    gaDat.NIND = 175;      % Número de individuos por población, su valor estandar
entre 150-200
    gaDat = ga(gaDat);
    gaDat.xmin;
    gaDat.fxmin;
```

Figura 10.19: Auto-tuning longitudinal

10.5.2 Funciones objetivo

Las funciones objetivo del sistema longitudinal y lateral se exponen a continuación, la función del modelo global se obtiene como la combinación de la longitudinal y la lateral.

```
function ECMt = Error_Medio_Cuadratico_lon(p)

% Dinámica longitudinal
    % Valores controladores
pitch_P = p(1);
pitch_I = p(2);
pitch_D = p(3);
tc_pitch = p(4);
pitch_ff = p(5);
    % Guardar variables en workspace para ser utilizadas en la simulación
assignin("base","pitch_P",pitch_P);
assignin("base","pitch_I",pitch_I);
assignin("base","pitch_D",pitch_D);
assignin("base","tc_pitch",tc_pitch);
assignin("base","pitch_ff",pitch_ff);
    % Simular el sistema
sim('FixedWing_Dynamic_Model_SIL_simulink');
    % Cargar los resultados
Simulink_pitch_sp = load('simulacionSimulink_pitch_sp');
Simulink_RPY = load('simulacionSimulink_RPY');
Simulink_rates_sp = load('simulacionSimulink_ratesRPY_sp');
Simulink_rates = load('simulacionSimulink_ratesRPY');
Ysp = [Simulink_pitch_sp.ans.Data(:),Simulink_rates_sp.ans.Data(:,2)];
Yest = [Simulink_RPY.ans.Data(:,2),Simulink_rates.ans.Data(:,2)];
%Obtener error cuadrático medio
[dimYrow,dimYcol] = size(Ysp);
ECM = zeros(dimYrow,dimYcol);
n = dimYrow;
for i = 1:dimYrow
ECM(i,:) = (Yest(i,:)-Ysp(i,:)).*(Yest(i,:)-Ysp(i,:));
end
ECMp = 1/n*sum(ECM);

ECMt = 0;
for i = 1:dimYcol
    ECMt = ECMt+ECMp(:,i);
end
end
```

Figura 10.20: Función objetivo longitudinal

```
function ECMt = Error_Medio_Cuadratico_lat(p)
 % Valores controladores
roll_P = p(1);
roll_I = p(2);
roll_D = p(3);
tc_roll = p(4);
roll_ff = p(5);
yaw_P = p(6);
yaw_I = p(7);
yaw_D = p(8);
yaw_ff = p(9);
    % Guardar variables en workspace para ser utilizadas en la simulación
assignin("base","roll_P",roll_P);
assignin("base","roll_I",roll_I);
assignin("base","roll_D",roll_D);
assignin("base","tc_roll",tc_roll);
assignin("base","roll_ff",roll_ff);
assignin("base","yaw_P",yaw_P);
assignin("base","yaw_I",yaw_I);
assignin("base","yaw_D",yaw_D);
assignin("base","yaw_ff",yaw_ff);
    % Simular el sistema
sim('FixedWing_Dynamic_Model_SIL_simulink');
    % Cargar los resultados
Simulink_roll_sp = load('simulacionSimulink_roll_sp');
Simulink_RPY = load('simulacionSimulink_RPY');
Simulink_rates_sp = load('simulacionSimulink_ratesRPY_sp');
Simulink_rates = load('simulacionSimulink_ratesRPY');
Ysp = [Simulink_roll_sp.ans.Data(:),Simulink_rates_sp.ans.Data(:,1),Simulink_rates_sp.↙
ans.Data(:,3)];
Yest = [Simulink_RPY.ans.Data(:,1),Simulink_rates.ans.Data(:,1),Simulink_rates.ans.↙
Data(:,3)];
%Obtener error cuadrático medio
[dimYrow,dimYcol] = size(Ysp);
ECM = zeros(dimYrow,dimYcol);
n = dimYrow;
for i = 1:dimYrow
ECM(i,:) = (Yest(i,:)-Ysp(i,:)).*(Yest(i,:)-Ysp(i,:));
end
ECMp = 1/n*sum(ECM);

ECMt = 0;
for i = 1:dimYcol
    ECMt = ECMt+ECMp(:,i);
end
end
```

Figura 10.21: Función objetivo lateral

Bibliografía

[1] *6DoF Quaternion*. MathWorks Documentation. Último acceso: 2024-06-25 (vid. pág. 21).

[2] Francesc Xavier Blasco Ferragud. "Control predictivo basado en modelos mediante técnicas de optimización heurística. Aplicación a procesos no lineales y multivariables". Tesis doct. Universitat Politècnica de València, 2012 (vid. págs. 34, 67).

[3] MATLAB Central. *Basic Genetic Algorithm*. Último acceso: 2024-06-25 (vid. pág. 65).

[4] V Delgado. *Que son los Drones*. 2019 (vid. pág. 4).

[5] PX4 Documentation. *Controller Diagrams of PX4*. Último acceso: 2024-06-25 (vid. págs. 14, 17-19).

[6] *Dron de Ala Fija*. Umiles Group. Último acceso: 2024-06-25 (vid. pág. 11).

[7] Economipedia. *MATLAB - Definición*. Último acceso: 2024-06-25 (vid. pág. 22).

[8] Javier Fernández-Lozano y Gabriel Gutiérrez-Alonso. "Aplicaciones geológicas de los drones". En: *Revista de la Sociedad Geológica de España* 29.1 (2016), págs. 89-105 (vid. pág. 5).

[9] Flaps5. *¿Qué son el Dutch Roll y el Yaw Damper?* `https://flaps5.blogspot.com/2019/04/que-son-el-dutch-roll-y-el-yaw-damper.html`. Último acceso: 2024-06-25. 2019 (vid. pág. 19).

[10] Fundación Aena. *Drones: Aplicaciones en aviación*. `https://fundacionenaire.es/conocimiento/drones-aplicaciones-aviacion`. Último acceso: 2024-07-04 (vid. pág. 7).

[11] *Getting Started with PX4 Basic Concepts*. Último acceso: 2024-06-25. PX4 (vid. págs. 14, 26).

[12] UMILES Group. *RPAS, UAS, UAV: ¿Cuáles son las diferencias?* `https://umilesgroup.com/rpas-uas-uav-diferencias/`. Último acceso: 2024-07-04 (vid. págs. 3, 4).

[13] Pablo Andrés Muñoz Gutiérrez, Eduardo Giraldo Suárez y Jhon Abraham Bonilla Becerra. "Identificación y control de un Vehículo Aéreo no Tripulado tipo Quadcopter". En: *Ingenierías USBMed* 7.1 (2016), págs. 5-10 (vid. pág. 69).

[14] Mariano I Lizarraga Fernandez. "Design, implementation and flight verification of a versatile and rapidly reconfigurable UAV GNC research platform". En: *Ph. D. Thesis* (2009) (vid. págs. 11, 12, 21).

[15] MathWorks. *Control Systems with MATLAB*. Último acceso: 2024-06-25 (vid. pág. 23).

[16] MathWorks. *HITL Simulink Fixed-Wing Plant Example*. `https://es.mathworks.com/help/uav/px4/ref/hitl-simulink-fixed-wing-plant-example.html`. Último acceso: 2024-07-04 (vid. pág. 24).

[17] MathWorks. *Reinforcement Learning in MATLAB*. Último acceso: 2024-06-25 (vid. pág. 35).

[18] MathWorks. *What is a Genetic Algorithm*. Último acceso: 2024-06-25 (vid. pág. 33).

[19] Ministerio de Transportes, Movilidad y Agenda Urbana. *Operaciones con UAS (drones)*. `https://www.seguridadaerea.gob.es/es/ambitos/drones/operaciones-con-uas-drones`. Último acceso: 2024-07-04 (vid. págs. 5-7).

[20] *QGroundControl User Guide*. Último acceso: 2024-06-25. QGroundControl (vid. pág. 22).

[21] *Reinforcement Learning for Control Systems Applications*. `https://es.mathworks.com/help/reinforcement-learning/ug/reinforcement-learning-for-control-systems-applications.html`. Último acceso: 2024-07-04 (vid. pág. 36).

[22] Wendy Rodríguez Rivero. "Estudio del algoritmo de control de rumbo L1 implementado en el autopiloto Ardupilot". Universidad Central "Marta Abreu"de Las Villas, jun. de 2015 (vid. pág. 16).

[23] Sergarro. *PX4 HyDrone Project on GitHub*. Último acceso: 2024-06-25 (vid. págs. 13, 25).

[24] Avium Technologies. *Pixhawk SIL Connector*. Último acceso: 2024-06-25 (vid. pág. 27).

[25] José Lino Carrillo Villalobos et al. "El Mundo de los drones: tipos de drones y sus principales usos". En: *FINGUACH. Revista De Investigación Científica De La Facultad De Ingeniería De La Universidad Autónoma De Chihuahua* 4.14 (2018), págs. 3-5 (vid. pág. 5).

[26] William Von Hagen. *Ubuntu Linux Bible*. Vol. 352. John Wiley & Sons, 2007 (vid. pág. 20).

[27] Manual de Vuelo. *Adyacencias Yaw*. `https://www.manualvuelo.es/1pbav/19_adyaw.html`. Último acceso: 2024-06-25 (vid. pág. 19).

Parte I

Pliego de condiciones

Índice general

Resumen I

I Pliego de condiciones I

Índice general III

1 Introducción 1

2 Descripción del proyecto 3

3 Pliego de condiciones generales 5

3.1 Documentación . . . 5
3.2 Contratante . . . 6
3.3 Obligaciones y derechos del contratista . . . 6
3.4 Ingeniero Director . . . 6
3.5 Condiciones generales económicas . . . 7
3.6 Condiciones generales de ejecución . . . 7

4 Pliego de prescripciones técnicas 9

4.1 Software . . . 9
4.2 Hardware . . . 10

Capítulo 1

Introducción

En este documento se desarrolla el pliego de condiciones correspondiente al trabajo de fin de grado de titulo "Diseño e Implementación de un sistema de auto-tuning mediante optimización numérica para auto-pilotos basados en el stack de control PX4".

El objetivo de este documento es recopilar las condiciones técnicas y legales que se deben cumplir para el correcto desarrollo del proyecto.

Capítulo 2

Descripción del proyecto

El proyecto establece el desarrollo de un sistema de auto-tuning de parámetros de control mediante el uso de un algoritmo genético implementado en Matlab, el sistema a optimizar es un auto-piloto implementado en Simulink, basado en el stack de control de PX4. El objetivo es obtener un sistema implementado equivalente al auto-piloto PX4 por lo que para ello es necesario la validación del esquema de Simulink, esto es realizado mediante la simulación SITL del sistema de referencia: PX4 como auto-piloto con conexión con QGroundControl encargado de la definición de la trayectoria a seguir y con el diagrama de modelización de la dinámica de un UAV en Simulink; esta señal es comparada con la obtenida por el sistema implementado. Con el sistema validado se aplica una optimización mediante algoritmo genético de los parámetros de control del sistema que establece el seguimiento de referencia de la actitud de la aeronave; para este ajuste se establece que la función objetivo a optimizar es la del error medio cuadrático de los ángulos de actitud y sus velocidades.

Capítulo 3

Pliego de condiciones generales

Este apartado tiene como objetivo describir las condiciones legales y administrativas del proyecto.

3.1 Documentación

La estructura del documento esta definida por las siguientes partes:

- Memoria: dedicada al desarrollo del trabajo. Tiene que contener como mínimo los siguientes puntos:
  - Objeto
  - Estudio de necesidades, factores a considerar: limitaciones y condicionantes.
  - Planteamiento de soluciones alternativas y justificación de la solución adoptada.
  - Descripción detallada de la solución adoptada.
  - Justificación detallada de los elementos o componentes de la solución adoptada
  - Anexos
- Pliego de condiciones: condiciones técnicas y legales. La estructura a seguir es la siguiente:
  - Objeto
  - Condiciones de los materiales: Descripción y Control de calidad.
  - Condiciones de la ejecución: Descripción y control de calidad.
  - Pruebas y ajustes finales o de servicio.
- Presupuesto: establece el importe de realización del proyecto. Se divide en:
  - Costes según naturaleza.
  - Precios descompuestos.

- Secciones homogéneas.

Todos estos documentos son los requeridos para la presentación correcta del proyecto.

3.2 Contratante

El contratante del proyecto debe presentar al Ingeniero Superior los siguientes documentos para la realización del proyecto:

- Modelo de la aeronave: geometría, aerodinámica y planta propulsora.

3.3 Obligaciones y derechos del contratista

- Será responsable de la organización del proyecto.
- Será conocedor de las leyes que estipulan el desarrollo del proyecto.
- Será responsable del cumplimiento de las leyes en los documentos del proyecto.
- Será encargado de gestionar y coordinar actividades con empresas subcontratadas.
- Será encargado de establecer a un delegado suyo en caso oportuno.
- Será encargado de comunicar la designación del delegado suyo.
- Deberá presentar todos los documentos expuestos en el apartado de Documentación.
- Será responsable del buen aspecto y desarrollo de los trabajos realizados, estableciéndose dentro del presupuesto estipulado.
- Será responsable de las revisiones del proyecto junto con el Ingeniero Director.
- Será responsable de comunicar al Ingeniero Director las modificaciones o aclaraciones realizadas sobre el documento de pliego de condiciones.
- Será responsable de suministrar el Ingeniero Director las instrucciones o aclaraciones requeridas por este durante el desarrollo del proyecto.

3.4 Ingeniero Director

El Ingeniero Director es el responsable del cumplimiento de los objetivos del proyecto. Para ello sus derechos y deberes son:

- Será responsable de la supervisión del proyecto.
- Será responsable de suministras ayuda en las tarea de mayor grado de dificultad durante el desarrollo del proyecto.
- Será responsable de organizar y dirigir la realización del proyecto.
- Será responsable de certificar la documentación del proyecto.

3.5 Condiciones generales económicas

En este apartado se establecen las condiciones económicas.

3.5.1 Precios

Los precios se pueden dividir en:

- Costes directos: relacionados con la mano de obra, componentes técnicos y coste de sistemas y dispositivos.
- Costes indirectos: se establecen como gastos generales en el presupuesto y representan en 13 % de los costes directos.
- Beneficio industrial: se establece como el 6 % de los costes directos.

Cada uno de estos precios se especifica en el presupuesto del proyecto.

3.6 Condiciones generales de ejecución

3.6.1 Extensión del proyecto

En caso de ser necesaria una extensión del proyecto, los trabajos continuarán según las directrices del Ingeniero Director mientras se realiza la reformulación del proyecto.
En caso de un retraso en el inicio del proyecto será posible el inicio de una prórroga si ambas partes están de acuerdo.

3.6.2 Recepción del proyecto

- El certificado de entrega de los documentos y códigos se expide con la firma entre contratante y contratista.
- Tras la entrega del proyecto comienza el periodo de garantía.
- El contratista deberá presenta toda la documentación del proyecto junto con lo códigos de desarrollo al contratante.
- Los código se encuentras desarrollados en las plataformas Matlab y Simulink.
- El contratista no se hará responsable del funcionamiento del código en otras aplicaciones no estipuladas para la correcta ejecución.

Capítulo 4

Pliego de prescripciones técnicas

Este apartado tiene como objetivo el desarrollo de los sistemas utilizados para la realización del proyecto, tanto software como hardware.

4.1 Software

Para el desarrollo de este trabajo se requiere del uso de diferentes softwares:

- *Matlab*: la versión utilizada en la R2023a junto con las herramientas *Simulink*, *Aerospace Toolbox* y *UAV Toolbox* para realizar las implementaciones del sistema. La licencia es suministrada por la institución en la que se desarrolla el proyecto (UPV).
- *QGroundControl*: software de estación en tierra bajo una doble licencia como Apache 2.0 y GPLv3, las contribuciones deben realizarse bajo ambas licencias.
- *PX4*: software de código libre de auto-piloto bajo la licencia BSD-3-clause. Es licencia establece los términos bajo los que es posible realizar modificaciones en el código del sistema PX4.
- *GitHub*: plataforma responsable de generar documentación para QGroundControl y PX4. Las especificaciones técnicas de cada uno se encuentran en sus repositorios.
- *Overleaf*: editor de texto online utilizado para desarrollar los documentos del trabajo. Sus términos y condiciones se encuentran en el apartado legal de su página web.

4.2 Hardware

El hardware necesario para este proyecto varia en función de la parte a desarrollar, para la primera parte del desarrollo (implementación PX4 en Simulink) no se requiere muchos recursos computacionales; sin embargo para la parte de auto-tuning mediante optimización global, se requiere de una gran potencia de cálculo por lo que se necesita un ordenador con mayor capacidad. Para la primera parte se emplea:

- Ordenador Portatil *HP 15s-fq1xxx*
- Procesador: *intel core i5 1.0GHz/1.2GHz*
- Memoria RAM: 8 GB

Para la segunda parte el coste computacional es mayor y se emplea:

- Ordenador de mesa
- Procesador: *intel core i5 2.5 GHz/4.6GHz*
- Memoria RAM: 16 GB
- Disco duro SSD: 1 T
- Tarjeta gráfica: GeForec RTX 4060

Parte I

Presupuesto

Índice general

Resumen I

I Presupuesto I

Índice general III

1 Introducción 1

2 Presupuesto parcial 3

2.1 Presupuesto software 3

2.2 Presupuesto hardware 4

3 Presupuesto Total 5

Capítulo 1

Introducción

En este documento se desarrolla el pliego de condiciones correspondiente al trabajo de fin de grado de titulo "Diseño e Implementación de un sistema de auto-tuning mediante optimización numérica para auto-pilotos basados en el stack de control PX4".

El desarrollo de este apartado se centra en el desglose del presupuesto requerido para la realización del proyecto.

Capítulo 2

Presupuesto parcial

2.1 Presupuesto software

En esta sección se desarrolla el presupuesto necesario para realizar el proyecto. Gran parte del software utilizado es de código abierto y no se requiere de licencia, para el desarrollo del proyecto se estima un tiempo de cuatro meses en los que es necesario el uso de los programas. Las licencias de cada software son necesarias durante la totalidad del desarrollo del proyecto ya

Programa	**Tipo de licencia**	**Coste anual (€)**	**Coste proyecto (€)**
QGroundControl	Permanente	0	0
PX4	Permanente	0	0
Ubuntu 20.04	Permanente	0	0
Matlab	Anual	900	300
Simulink	Anual	1360	453.33
UAV toolbox	Anual	900	300
Aerospace Blockset	Anual	780	260
Aerospace toolbox	Anual	560	186.67
Coste total		4500	1500

Tabla 2.1: Presupuesto Software

que el desarrollo requiere de diversas combinaciones entre ellos. Una estimación del tiempo de uso de cada software es:

	QGroundControl	**PX4**	**Matlab-Simulink**
Tiempo de uso (h)	16	50	319

Tabla 2.2: Tiempo de uso de software

2.2 Presupuesto hardware

En este apartado se establece el coste utilizado en hardware para el desarrollo del proyecto de estudio. Para este proyecto se ha hecho uso de dos ordenadores diferentes, el primero de ellos un portatil *HP 15s-fq1xxx* con un procesador *intel core i5* con una memoria RAM de 8 GB, con este se ha realizado la mayor parte del trabajo a excepción de las simulaciones del sistema de optimización que se ha hecho uso de un ordenador de mesa con un procesador *intel core i5* con una memoria RAM de 16 GB y un disco duro SSD de 1 T, este segundo ordenador solo ha sido utilizado en el último mes del proyecto. Se supone una amortización de ambos ordenadores de 5 años.

Ordenador	**Adquisición (€)**	**Coste mensual (€/mes)**	**Coste proyecto (€)**
Portatil HP	700	11.67	46.68
Ordenador de mesa	1231.22	20.52	20.52
Coste total del material			67.2

Tabla 2.3: Presupuesto Hardware

2.2.1 Presupuesto personal

Para el cálculo de los costes personales se estima un sueldo promedio de 22 €/h que pertenece a un ingeniero aeroespacial novato, el requerimientos de horas de cada proceso se establece a continuación:

Tarea	**Tiempo empleado (h)**	**Coste (€)**
Instalación y primeras simulaciones PX4	20	440
Implementación Simulink de los sistemas de control	200	4400
Implementar algoritmo genético	15	330
Optimización de parámetros de control	150	3300
Redacción documento	40	880
Total	425	9350

Tabla 2.4: Presupuesto Personal

Capítulo 3

Presupuesto Total

En este apartado se resume el coste total bruto de la realización del proyecto como la suma de los costes de software, de hardware y de personal; a esto se le suma el beneficio industrial (6 %) y la adición de gastos generales (13 %). El resumen se observa en la siguiente tabla:

Concepto	**Coste total (€)**
Presupuesto software	1500
Presupuesto hardware	67.2
Presupuesto personal	9350
Coste total bruto	10917.2
Beneficio industrial	655.03
Gastos generales	1419.24
Coste total bruto + BI + GG	12991.47

Tabla 3.1: Presupuesto total bruto

Para obtener el coste total neto del proyecto es necesario incluir el IVA en el coste total bruto, este impuesto es del 21 %. El presupuesto total se desarrolla a continuación:

Concepto	**Coste total (€)**
Coste total bruto	11474.22
IVA (21 %)	2409.59
Presupuesto total neto	13883.81

Tabla 3.2: Presupuesto total neto

El presupuesto total neto del proyecto es: TRECE MIL OCHOCIENTOS OCHENTA Y TRES EUROS Y OCHENTA Y UN CÉNTIMOS.