UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Escuela Técnica Superior de Ingeniería del Diseño
Universitat Politècnica de València

Diseño de estrategias de control basadas en observadores de estados para el seguimiento de trayectorias en vehículos aéreos no tripulados tipo quadricóptero

TRABAJO DE FINAL DE GRADO

28 de junio de 2020, València

Autor: Martínez Ibáñez, Josep Balbino

Tutor: González Sorribes, Antonio
Segundo Tutor: García Gil, Pedro José

4° Curso de Ingeniería Aeroespacial: Especialidad Aeronaves

Agradecimientos

Después de cuatro años de carrera que llegan a su fin, me gustaría agradecerles todo lo que han hecho por mí a mis padres, Juan Balbino y Adela. En todas las etapas de mi vida me han apoyado en cualquier decisión que tuviera que tomar, me han aconsejado y se han sacrificado de la forma que nadie lo haría para darme una buena educación que me ha hecho evolucionar como persona hasta llegar a ser quien soy ahora mismo. Todos mis actos se han hecho pensando en hacerles sentirse orgullosos de mí y espero haberlo cumplido porque se lo merecen después de todo lo pasado. Por eso me gustaría dedicarles este trabajo, símbolo de tantos años de esfuerzo, a ellos.

Por otro lado también me gustaría agradecerle a mi tutor, Antonio, todo el tiempo dedicado y la ayuda que me ha proporcionado para la realización de este trabajo.

Resumen

El objetivo del proyecto es diseñar y validar mediante entornos de realidad virtual un algoritmo de control multivariable, centrado en el control de seguimiento de trayectorias de un quadricóptero en vuelo libre. Para ello se desarrolla el modelo dinámico no lineal que gobierna el movimiento de un quadrotor, posteriormente se presenta una estrategia de control basado en realimentación de estado y un control con observador sobre el modelo desarrollado. Para validar tanto el modelo dinámico como el sistema de control se hace uso de AirSim, un simulador de código libre de Microsoft que permite la comunicación con el lenguaje de programación de Python para realizar simulaciones en SITL. Los resultados mediante simulación corroboran el cumplimiento de las especificaciones estáticas y dinámicas de comportamiento establecidas.

Palabras clave: Diseño de sistema de control, Control multivariable, Quadricóptero, Modelado de sistemas, Control de trayectoria, AirSim

Abstract

The objective of the project is to design and validate by means of virtual reality environments a multivariable control algorithm for the tracking control of the trajectories of a quadcopter in free flight. For this, the dynamic non-linear model that governs the movement of a quadrotor is developed and later a control strategy based on state feedback and an observer control on the developed model are presented. To validate both the dynamic model and the control system, AirSim is used, a free source simulator from Microsoft that allows communication with the Python programming language to perform simulations in SITL. The results through simulation corroborate compliance with the established static and dynamic behavioral specifications.

Key words: Control system design, Multivariable control, Quadrotor, System modelling, Trajectory control, AirSim

Resum

L'objectiu d'el projecte és dissenyar i validar mitjançant entorns de realitat virtual un algoritme de control multivariable, centrat en el control de seguiment d'trayectories d'un quadricóptero en vol lliure. Per a això es desenvolupa el model dinàmic no lineal que governa el moviment d'un quadrotor, posteriorment es presenta una estratègia de control basat en realimentació d'estat i un control amb observador sobre el model desenvolupat. Per validar tant el model dinàmic com el sistema de control es fa ús de AirSim, un simulador de codi lliure de Microsoft que permet la comunicació amb el llenguatge de programació de Python per a realitzar simulacions en SITL. Els resultats mitjançant simulació corroboren el compliment de les especificacions estàtiques i dinàmiques de comportament establertes.

Paraules clau: Disseny de sistema de control, Control multivariable, Quadrocòpter, Modelatge de sistemes, Control de trajectòria, AirSim

Índice general

Índice de figuras

Índice de tablas

0. Índice de algoritmos

Índice de códigos

0. Índice de símbolos

Abreviaturas

cdg	Centro de gravedad
gdl	Grado de libertad
IGE	*In Ground Effect*
LQE	*Lineal Quadratic Estimator*
LQG	*Lineal Quadratic Gaussian*
LQR	*Lineal Quadratic Regulator*
OGE	*Out of Ground Effect*
VTOL	*Vetical Take Off and Landing*

Subíndices

0	Valor en el punto de equilibrio
1	Hace referencia al bucle de control interno
1	Hace referencia al rotor 1
2	Hace referencia al bucle de control externo
2	Hace referencia al rotor 2
3	Hace referencia al rotor 3
3	Hace referencia al rotor 4
A	Fuerza aerodinámica
a_i	Hace referencia al apoyo con el suelo i
B	Vector unitario del sistema de referencia ejes cuerpo
G	Fuerza debida a la gravedad
G	Vector unitario del sistema de referencia global
GR	Fuerza o momento debido a la reacción del suelo
H	Vector unitario del sistema de referencia horizonte local
i	Iterador

ss Estado estacionario

T Fuerza o momentos debido a la tracción de los rotores

w Hace referencia al viento

x Componente de un vector expresado en el eje unitario x

y Componente de un vector expresado en el eje unitario y

z Componente de un vector expresado en el eje unitario z

Constantes

μ Coeficiente de fricción de Coulomb del contacto $[-]$

ρ Densidad del aire $[kg\,m^{-3}]$

A Matriz del sistema [-]

B Matriz de entradas [-]

b Distancia entre dos rotores opuestos $[m]$

b Factor de resistencia $[kg\,m^2]$

C Matriz de salidas [-]

c_a Amortiguamiento de contacto $[N\,s\,m^{-1}]$

CTR Matriz de controlabilidad del sistema [-]

D Matriz de *feedforward* [-]

d Distancia entre dos rotores sucesivos $[m]$

I Matriz de inercias del quadrotor $[kg\,m^2]$

I_R Momento de inercia de las palas alrededor de su eje de giro $[kg\,m^2]$

I_{xx} Momento de inercia en el eje X del quadrotor $[kg\,m^2]$

I_{yy} Momento de inercia en el eje Y del quadrotor $[kg\,m^2]$

I_{zz} Momento de inercia en el eje Z del quadrotor $[kg\,m^2]$

K Matriz de realimentación de estados [-]

k Factor de tracción $[kg\,m]$

k_a Rigidez de contacto $[N\,m^{-1}]$

K_b Coeficiente de tracción debida al cuerpo $[-]$

K_i Ganancia del integrador [-]

K_r Ganancia de la referencia [-]

l Longitud del brazo del quadrotor $[m]$

m Masa del quadrotor $[kg]$

m Número de estados internos del sistema [-]

n Número de acciones de control del sistema [-]

p Número salidas del sistema [-]

p Polos del sistema en bucle cerrado [-]

Q Matriz de ponderación de los estados para el diseño de un controlador LQR [-]

R Matriz de ponderación de las acciones de control para el diseño de un controlador LQR [-]

R Radio de los rotores [m]

T_m Tiempo característico de la dinámica de giro de un rotor [s^{-1}]

Operadores

$\dot{}$ Derivada temporal

$\bar{}$ Sistema ampliado

∂ Derivada parcial

$diag$ Generación de una matriz diagonal a partir de un vector

f Función no lineal de estados

h Función no lineal de salidas

Superíndices

B Vector expresado en sistema de referencia ejes cuerpo

G Vector expresado en sistema de referencia global

H Vector expresado en sistema de referencia horizonte local

Prefijos

δ Variación pequeña de una variable

Variables

β Ángulo de batimiento de la pala [-]

ω Velocidad de giro de un rotor [s^{-1}]

ϕ Estado interno del sistema del quadrotor. Ángulo de asiento lateral o alabeo [$-$]

ψ Estado interno del sistema del quadrotor. Ángulo de rumbo o guiñada [$-$]

ψ Ángulo de azimut de la pala [$-$]

θ Estado interno del sistema del quadrotor. Ángulo de asiento longitudinal o cabeceo [$-$]

$\vec{\Gamma}$ Vector que contiene los momentos giroscópicos debido al giro de los rotores[Nm]

$\vec{\omega}$ Vector que contiene los tres estados internos del quadrotor correspondientes a la velocidad angular, puede estar expresado en ejes globales o cuerpo [s^{-1}]

$\vec{v}$ Vector que contiene los tres estados internos del quadrotor correspondientes a la velocidad, puede estar expresado en ejes globales o cuerpo $[ms^{-1}]$

$\vec{\xi}^G$ Vector que contiene los tres estados internos del quadrotor correspondientes a la posición, medidos en ejes globales $[m]$

$\vec{A}$ Fuerzas de perturbación $[N]$

$\vec{D}$ Resistencia aerodinámica, en ejes globales o cuerpo $[N]$

$\vec{F}$ Vector de fuerzas externas que actúan sobre el sistema $[N]$

$\vec{M}$ Vector de momentos externos que actúan sobre el sistema $[Nm]$

$\vec{M}_w$ Momentos de perturbación $[Nm]$

$\vec{r}$ Vector que contiene las referencias a seguir [-]

$\vec{T}$ Vector tracción del quadrotor, expresado en ejes cuerpo o globales [-]

$\vec{u}$ Vector que contiene las 4 acciones de control del quadrotor [-]

$\vec{W}$ Velocidad del aire, en ejes globales o cuerpo $[ms^{-1}]$

$\vec{x}$ Vector que contiene los 12 estados internos del quadrotor [-]

$\vec{y}$ Vector que contiene las salidas o estados controlados del quadrotor [-]

ξ Vector de errores de entrada [-]

p Estado interno del quadrotor. Velocidad angular en eje x_B $[s^{-1}]$

q Estado interno del quadrotor. Velocidad angular en eje y_B $[s^{-1}]$

r Estado interno del quadrotor. Velocidad angular en eje z_B $[s^{-1}]$

R_X Matriz de rotación alrededor del eje X $[-]$

R_Y Matriz de rotación alrededor del eje Y $[-]$

R_Z Matriz de rotación alrededor del eje Z $[-]$

T Matriz de transformaciones angulares, de ejes cuerpo a ejes globales $[-]$

T Tracción del quadrotor en el eje z_B [-]

t Tiempo [s]

u Estado interno del quadrotor. Velocidad en eje x_B $[ms^{-1}]$

v Estado interno del quadrotor. Velocidad en eje y_B $[ms^{-1}]$

w Estado interno del quadrotor. Velocidad en eje z_B $[ms^{-1}]$

x Estado interno del quadrotor. Posición en eje x_G $[m]$

y Estado interno del quadrotor. Posición en eje y_G $[m]$

z Estado interno del quadrotor. Posición en eje z_G $[m]$

${}^BR^G$ Matriz de rotación de ejes globales a ejes cuerpo $[-]$

${}^GR^B$ Matriz de rotación de ejes cuerpo a ejes globales $[-]$

Parte I

Memoria

1. Objetivos y alcance del trabajo

Antes de comenzar con la memoria se van a expresar cuáles son los objetivos que se han impuesto para el correcto desarrollo del trabajo de final de grado. Se explicitarán los objetivos principales y los secundarios que deben de ser cubiertos para el correcto cumplimiento de los primarios. Posteriormente se analiza cuál es al alcance del trabajo, es decir, hasta que fase del diseño se espera llegar al final del documento y bajo qué condiciones.

1.1 Objetivos primarios

Los objetivos primarios son aquellos que definen el trabajo. Se han dividido en el diseño de la ley de control que garantice el seguimiento de una determinada referencia de posición o velocidad mediante el uso del control de orientación y en el diseño del generador de trayectorias que se encarga de generar las referencias suministradas al primer controlador mencionado anteriormente.

- Diseño de una ley de control basada en espacio de estados para un quadrotor. Se debe garantizar la aplicabilidad de la ley de control para todo el rango de estados que se puedan dar en un vuelo normal.

  Esta ley de control debe de ser capaz de seguir las referencias que sean impuestas tanto por el piloto automático como por el usuario, además se debe garantizar el error de posición nulo en este seguimiento, la estabilidad del sistema y el rechazo de las posibles perturbaciones que afecten al sistema dinámico.

  Además se deben de considerar los diferentes tipos de controlador que pueden ser implementados en un quadrotor, entre ellos el control de orientación, control de posición y control de velocidad, con la posibilidad de que las referencias del control de orientación sean generadas por el control de posición o velocidad en función de las necesidades.

- Diseño del controlador de seguimiento de trayectoria. Para ello se debe de considerar una serie de *waypoints* a través de los cuales debe de pasar el quadrotor con la mínima desviación posible y generando la trayectoria óptima entre ellos sin cambios bruscos de dirección.

  Este controlador debe ser capaz de controlar la velocidad y guiñada con la que se pasa por cada uno de los puntos de interés de tal forma que sea aplicable a un rango de misiones elevado.

  El diseño del controlador de trayectoria debe de hacerse de tal forma que permita generar unas referencias de posición o velocidad en función de los estados del quadrotor en la iteración de evaluación.

1.2 Objetivos secundarios

- Desarrollo del modelo cinemático y dinámico del movimiento de un quadrotor con 6 gdl. Este modelo debe de ser lo más realista posible, incluyendo las reacciones del suelo, efecto suelo y resistencia aerodinámica, para así crear un simulador que pueda ser utilizado para el diseño de las leyes de control.

  El desarrollo del modelo utilizado para simular el movimiento del quadrotor debe hacerse de la forma más sencilla posible y permitiendo su fácil implementación en diagrama de bloques de Simulink. Esta plataforma se usará para simular de forma muy rápida el comportamiento de las leyes de control hasta obtener las especificaciones deseadas y posteriormente podrán ser exportadas a otros *softwares* de simulación externos donde validar los resultados. En consecuencia, el diseño preliminar se hará en el entorno de Matlab que permite realizar esta tarea de forma sencilla.

- Puesto que el tipo de control que se va a realizar es lineal, se debe linealizar el modelo dinámico del sistema entorno a un punto de equilibrio mediante el desarrollo en serie de Taylor para obtener un sistema sobre el que se pueda aplicar el tipo de control deseado. Puesto que solamente es utilizado para el diseño de las leyes de control, en este proceso de linealización se deben simplificar las ecuaciones no lineales completas eliminando algunos de los términos que puedan ser despreciados o considerados como perturbaciones.

  El proceso de linealización y la obtención de las matrices del sistema debe de hacerse de la forma más genérica posible para permitir la evaluación en diferentes puntos de equilibrio en función de la necesidad.

- Diseño de un observador de estados que permita obtener los estados del sistema a partir de mediciones ruidosas de los sensores abordo y de las acciones de control suministradas al sistema. Este observador debe de proporcionar buenos resultados incluso en puntos alejados del equilibrio alrededor del cual se ha linealizado el sistema.

- Validación del sistema dinámico desarrollado en el presente documento mediante el uso de simuladores externos. También se deben validar con el simulador externo las leyes de control que hayan sido diseñadas, siendo deseable que se obtengan resultados similares con el modelo dinámico desarrollado y con el simulador.

- Extensión de las leyes de control al dominio discreto para que puedan ser utilizadas en un piloto automático ejecutado en el microprocesador de un quadrotor real. Para ello se deberá hacer uso del *software* de simulación y validar su control en este dominio.

1.3 Alcance del trabajo

Una vez se hayan cumplido todos y cada unos de los objetivos de forma satisfactoria, se debe de haber obtenido una plataforma sólida que permita simular la evolución de la dinámica de un quadrotor a la que se le han suministrado unas entradas a lo largo del tiempo. Este simulador dinámico se puede utilizar para el diseño de leyes de control de un quadrotor en la plataforma de Simulink, utilizando todas las facilidades del lenguaje de programación de Matlab para la ingeniería.

Además se debe de obtener una ley de control que pueda ser implementada de forma sencilla en un piloto automático de un quadrotor. Esta ley debe de ser capaz de controlar la posición y velocidad del quadrotor mediante el uso del control de orientación y altura. También se habrán considerado los sensores de los que dispondrá un quadrotor real para diseñar un observador de estados que permita estimar todos los estados internos del sistema.

Finalmente, la validación de la ley de control y el modelo dinámico mediante el uso de un simulador permitirá examinar el trabajo con un modelo más preciso donde se consideran fenómenos no modelados y de esta forma ver si el controlador es lo suficientemente robusto. También se alcanzarán los conocimientos suficientes para trabajar con este *software* externo y utilizarlo en futuros trabajos con aplicaciones destinadas a vehículos autónomos.

2. Introducción y evolución histórica

Un vehículo aéreo no tripulado o UAV (*Unmanned Aerial Vehicle* en sus siglas en inglés) es una aeronave que no está pilotada por una persona abordo, es decir los controles necesarios para seguir una determinada referencia o cumplir una determinada misión pueden ser proporcionados de forma autónoma por el propio piloto automático del vehículo o por un control remoto desde una estación en tierra que puede estar operada o no por un humano. Por tanto, en este tipo de vehículos toma una gran importancia la estación de control y el sistema de comunicación entre el vehículo y el control, considerados como parte del UAV. El sistema de comunicación debe de ser lo suficientemente rápido como para permitir el vuelo en tiempo real, por un lado el quadrotor debe enviar toda la información proveniente de los sensores y necesaria para que el piloto en tierra pueda evaluar los controles que se le deben suministrar y por otro lado se deben enviar estos controles de vuelta al vehículo.

Cabe destacar que un UAV es un vehículo que puede ser utilizado para diferentes misiones, entre ellas el transporte de cualquier tipo de mercancía o de reconocimiento entre ellas. Es por esta razón por la que los misiles, aunque no estén tripulados por personal a bordo, no son considerados como UAV, ya que son principalmente un arma. Además un UAV puede ser un vehículo reutilizable o de un solo uso.

Aparte del término UAV, a lo largo del tiempo también se han utilizado nuevos nombres para referirse a ellos. Por un lado aparece el término *drone*, aunque esté usado extensamente por el público para referirse a los quadrotor, en realidad un *drone* es un sinónimo de UAV. Otros términos similares utilizados pueden ser UAS (*Unmanned aerial system*) o RPAS (*Remotely piloted aircraft system*).

2.1 Clasificación de los UAVs

Según [1] existen diferentes criterios con los que clasificar un UAV. Entre ellos se encuentra la clasificación en función de la aerodinámica del vehículo que considera cómo son y de dónde provienen las fuerzas de sustentación y de resistencia del vehículo. Por otro lado se encuentra la clasificación en base al tipo de despegue y aterrizaje que pueden hacer, (VTOL, STOL o HTOL). Finalmente la última de las clasificaciones es en base al peso y al rango del vehículo. En la figura 2.1 se muestran, en función del tipo de clasificación que se realice, los diferentes tipos de UAV existentes.

En la clasificación con respecto al tipo de despegue que puede realizar el vehículo se encuentran aquellos que tienen capacidades de despegue, aterrizaje y vuelo en vertical (VTOL) y los que necesitan de velocidad en avance para estas tareas (HTOL y STOL). Los multirotores y entre ellos los quadrotor estarían englobados dentro de los primeros dándoles una gran atractivo por el rango amplio de tareas que podrían realizar al no necesitar de velocidad relativa con el aire que lo rodea para mantenerse en vuelo.

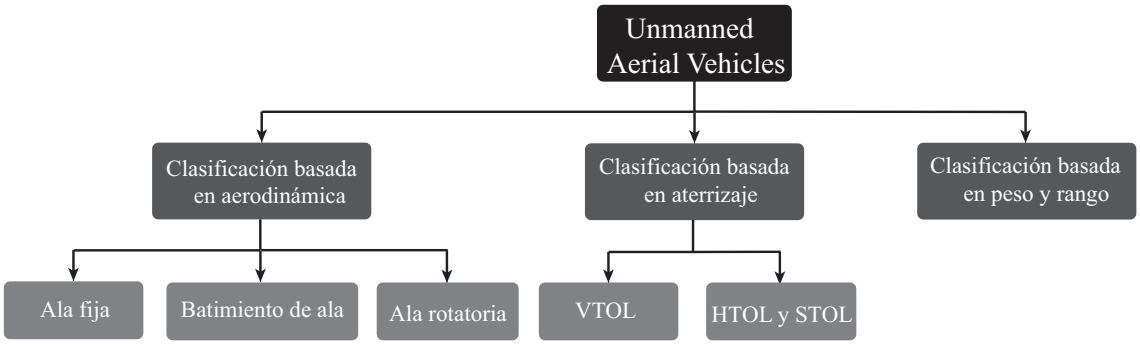


Figura 2.1: Clasificación de los UAV en función de aerodinámica, aterrizaje o peso y rango [1]

En cuanto a la aerodinámica se encuentran aquellas aeronaves que presentan ala fija, estas serían un tipo de UAV con despegue horizontal, a no ser que contasen con un sistema que les permita realizar el despegue vertical, generalmente estos serán del primer tipo de despegue. Los UAV que pueden volar gracias al batimiento de sus alas imitando el comportamiento de los insectos y las aves, presentan una aerodinámica compleja, sin embargo pueden conseguir vuelos más estables que aquellos de ala fija. Finalmente las aeronaves de ala rotatoria incluyen aquellas que utilizan una o varias palas cuyo giro produce la fuerza de sustentación, aquí estarían englobados los helicópteros y los multirotores. Los quadrotor, objeto de estudio del presente documento son un tipo de multirotor con cuatro rotores. En la sección 2.3 son analizadas las diferentes configuraciones de los multirotores en función del número de rotores y su disposición, posteriormente, en la sección 2.4 son analizadas las diferentes configuraciones de los quadrotor.

La clasificación en función del peso y rango de de los UAV se ha resumido en la tabla 2.1 donde se considera si cada tipo de UAV hace referencia a ala fija, a multirotores o ambos.

Tipo	Peso máximo al despegue	Máximo Rango	Categoría
Nano	0.2 kg	5 km	Ala fija, Multirotor
Micro	2 kg	25 km	Ala fija, Multirotor
Mini	20 kg	40 km	Ala fija, Multirotor
Light	50 kg	70 km	Ala fija, Multirotor
Small	150 kg	150 km	Ala fija
Tactical	600 kg	150 km	Ala fija
MALE	1000 kg	200 km	Ala fija
HALE	1000 kg	250 km	Ala fija
Heavy	2000 kg	1000 km	Ala fija
Super Heavy	2500 kg	1500 km	Ala fija

Tabla 2.1: Clasificación de los UAV en función de su peso y su rango [1]

Como se puede observar, los multirotores generalmente presentan un peso considerablemente menor que las aeronaves de ala fija. Esto es debido a la naturaleza de la fuerza de sustentación ya que es más sencillo levantar altas cargas con la sustentación debida al avance de la aeronave puesto que la tracción no debe ser tan grande como la carga. Sin embargo, en los multirotores la tracción debe de ser capaz de contrarrestar el peso de la carga y además producir la fuerza de avance, por ello cuando aumenta el tamaño se vuelve cada vez más difícil escalar la geometría del quadrotor. Se produce lo que es conocido como un cuello de botella.

Para aumentar la carga que es capaz de llevar un multirotor es necesario aumentar la tracción que producen los rotores, por lo que se debe aumentar su radio de giro. Este aumento hace que la res-

Figura 2.2: Multicóptero a escala completa: VC200

puesta de la dinámica de los motores ante cambios en la velocidad de giro sea considerablemente más lenta debido al aumento de la inercia en el eje de giro de la pala, por lo que se dificulta la rapidez de respuesta en las actuaciones del quadrotor. Si se compara con los helicópteros, estos tienen palas muy largas pero que giran a velocidad constante y cambian su ángulo de paso para incrementar la sustentación, a diferencia de los multirotores que aumentan la velocidad de giro manteniendo constante el paso [2].

Además, si se incrementa el radio de los rotores aumenta en gran medida el momento flector en el encastre de la pala con el rotor que tiende a aumentar el ángulo de batimiento de la pala, causando fatiga que puede llegar a provocar la rotura. Por esta razón en helicópteros se suele optar por unir las palas con el eje del rotor utilizando una articulación que permita el movimiento de batimiento.

La solución que se ha utilizado en multirotores de grandes escalas es el incremento del número de pequeños rotores de tal forma que se permite aumentar la tracción para contrarrestar el peso, por otro lado se aumenta la maniobrabilidad y la seguridad en vuelo ya que existe redundancia. Sin embargo, aumentan las posibilidades de que uno de los motores falle y las tareas de mantenimiento que se le deben realizar, también se incrementa el consumo energético disminuyendo el tiempo de vuelo total. Un ejemplo donde puede observarse esta solución es en el multicóptero alemán VC200 mostrado en la figura 2.2[1].

Si se compara este tipo de multirotores con uno del mismo tamaño es posible ver que no existen mejoras en cuanto a tiempo de vuelo, mantenimiento o fiabilidad del sistema de propulsión. La aparición de este tipo de multirotores altamente actuados es únicamente debido a la necesidad de aplicar el concepto a gran escala por lo que los multicópteros de pequeño tamaño seguirán manteniendo la configuración original [2].

2.2 Evolución histórica de los quadrotor

En la historia reciente, los vehículos del tipo quadrotor han experimentado un incremento notable de su popularidad dándoles muchas aplicaciones que hasta hace poco tiempo eran realizadas mediante otras soluciones. Sin embargo, los primeros vehículos de este tipo tienen su aparición a principios del siglo XX, concretamente en 1907 cuando los hermanos Bréguet, guiados por Charles Richet construyeron uno de los primeros intentos de vehículo de ala rotatoria, concretamente este era un quadrotor con el nombre de Bréguet-Richet Gyroplane No.1, mostrado en la figura

[1] `https://www.thedrive.com/news/3862/this-drone-inspired-multicopter-could-be-the-flying\-car-youve-dreamed-of`

2.3[2]. Aunque despegó del suelo hasta una altura de 0.6*m*, este vuelo no fue controlable ya que necesitó de ayuda externa para soportar la estructura [3].

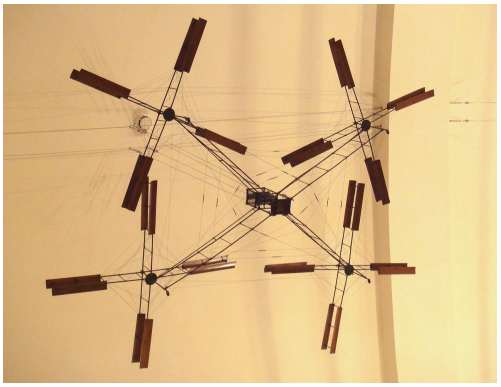


Figura 2.3: Bréguet-Richet Gyroplane No.1

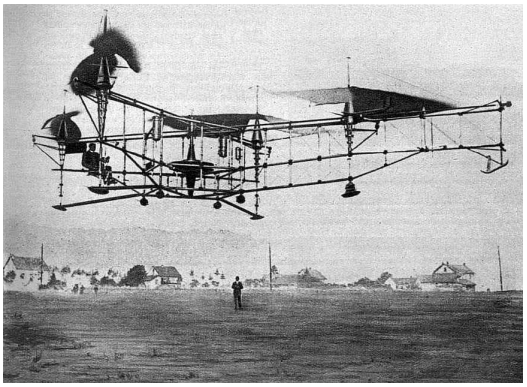


Figura 2.4: Oehmichen Helicopter No.2

tras numerosos cálculos y prediseños, en el año 1920 el ingeniero Etienne Oehmichen mostró el Oehmichen Helicopter No. 2, figura 2.4, que fue capaz en 1923 de permanecer 14 minutos en vuelo [3].

Otros ingenieros también mostraron interés en este tipo de vehículos, como pueden ser George De Bothezat, Ivan Jerome o incluso el ejército de Estados Unidos, lo que hicieron aparecer numerosos prototipos. Sin embargo, debido a que la tecnología todavía no era lo suficiente avanzada como para permitir el desarrollo, no se pudieron mejorar los resultados obtenidos lo suficiente durante tres décadas [2]. En los años 50 aparecieron modelos más sofisticados que consiguieron volar en estado estacionario y avance, como el Convertawings Model 'A' o 'Curtis-Wright VZ-7' que no tuvieron demasiado interés debido a la poca capacidad de carga que tenían, además sus capacidades en cuanto a velocidad de vuelo, rango o tiempo máximo de vuelo eran considerablemente menores que las de otros vehículos aéreos como el avión. Otro de los problemas con los que contaban era que la carga de trabajo era considerablemente elevada. Además estos vehículos a diferencia de los quadrotor actuales estaban a escala completa, con motores y sensores demasiado grandes y pesados como para permitir un vuelo eficiente. Por esta razón, ninguno de estos prototipos siguió adelante con el desarrollo de la tecnología y quedaron congelados.

Sin embargo, alrededor de los años 90 con el desarrollo de los sistemas micro electro mecánicos o MEMS se abrió la posibilidad de volver a desarrollar los vehículos del tipo multirotor con la diferencia que su escala sería mucho más pequeña. Estos dispositivos, que constituyen una unidad de medidas inerciales o IMU, tendría la característica de pesar y ocupar muy poco espacio, por lo que se podría solucionar uno de los problemas principales que tuvieron los anteriores diseños. Sin embargo, los MEMS tienen como principal desventaja que sus mediciones son bastante ruidosas por lo que no pueden ser utilizadas directamente. Por esta razón se comenzó a investigar diferentes técnicas de filtrado de las señales y además con los procesadores en los que se deben hacer los diferentes cálculos del control y de los sensores. Durante los años posteriores se empezaron a diseñar las diferentes leyes de control, además se comenzaron a dar a los quadrotor de numerosas aplicaciones y posibilidades que los hicieron muy populares. Entre ellas se encuentran la fotografía, la investigación, el sector militar o las actividades humanitarias.

[2] https://en.wikipedia.org/wiki/Br%C3%A9guet-Richet_Gyroplane#/media/File:Breguet_Gyroplane_1907.jpg

[3] https://commons.wikimedia.org/wiki/File:%C3%89tienne_%C5%92hmichen_sur_l%27Oehmichen_Helicopter_No._2,_le_4_mai_1924_%C3%A0_Valentigney_(vol_d%271_km_en_circuit_ferm%C3%A9).jpg

2.3 Partes de un quadrotor

Una vez analizado el tipo de UAV que es un quadrotor, en esta sección se detallan los componentes por los que está compuesto un quadrotor. Por norma general, un quadrotor tiene una configuración bastante simple y está compuesta por elementos bien diferenciados entre sí como pueden ser la estructura, los diferentes sistemas de propulsión y el sistema de control [2].

- Estructura

Generalmente, la estructura de un quadrotor está compuesta por el fuselaje y el tren de aterrizaje. La principal función del fuselaje es la de servir como soporte y unión del resto de componentes del sistema. Existen ciertos parámetros que van asociados al tipo de fuselaje que tenga el quadrotor como pueden ser el tamaño, el peso, el material o la resistencia que determinan la durabilidad, seguridad y actuación del vehículo. Puesto que el peso es uno de los parámetros más restrictivos de un vehículo aéreo es necesario que sea el mínimo posible mientras que se asegure la seguridad del vuelo. Otro de los parámetros más importantes de un quadrotor es la longitud de su brazo que determina de forma directa la maniobrabilidad del vehículo, como contrapartida presenta que a mayor longitud de brazo, mayor será el peso de la aeronave, por tanto se debe buscar un balance entre ambos criterios.

La configuración del fuselaje es uno de los criterios que pueden ser utilizados para clasificar los multirotores, algunas de las configuraciones más populares son el quadrotor, objeto de estudio del presente documento, el tricopter o hexacopter. En la figura 2.5 se muestran de forma gráfica la estructura de algunos de los multirotores más populares. En la sección 2.4 se muestran las configuraciones concretas que un quadrotor puede presentar.

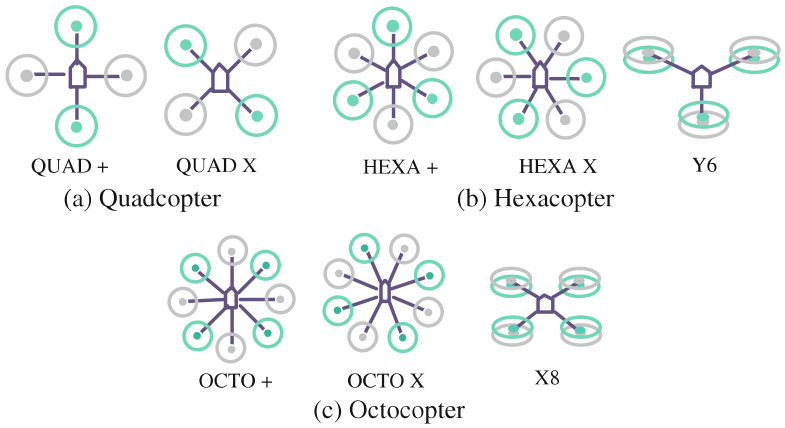


Figura 2.5: Configuraciones populares de UAV multirotores [2]

Por otro lado, el tren de aterrizaje tiene como principales funciones la de servir como soporte del vehículo en las etapas de despegue y aterrizaje, también permite proteger el cuerpo y los rotores ante posibles contactos con el suelo y amortiguar los impactos.

- Sistema de propulsión

El sistema de propulsión está compuesto principalmente por las palas, motores, controladores de velocidad electrónicos o ESC y las baterías. La fuente de energía de un quadrotor es generalmente eléctrica y está almacenada en baterías de ion o polímero de litio y es la que determina, junto con el resto de componentes del sistema de propulsión como serán las actuaciones del vehículo, entre ellas el tiempo de *hovering*, máxima velocidad, máximo rango o máximo tiempo de vuelo.

En cuanto a los motores que son utilizados en los multirotores estos son generalmente los motores eléctricos sin escobillas o BLDC. La razón por la que se utilizan estos motores es debido a la alta eficiencia y alta relación potencia-peso. También, debido a que la conmutación de las fases es electrónica y no mecánica, se reduce en gran medida el ruido y los costes de mantenimiento. Como desventajas presentan que es necesario un ESC para controlar la velocidad de giro del motor mediante la conmutación de las fases. El controlador de velocidad de los motores también presenta otras funciones como son el BEC o *Battery Eliminator Circuit* que permite sacar una salida de 5 V a partir de las baterías para alimentar la controladora de vuelo.

- Sistema de control

El sistema de control de un quadrotor está compuesto principalmente por el piloto automático, la estación de control en tierra y el comando o control manual. El piloto automático tiene como funciones principales el control de la orientación, posición y trayectoria que puede recibir referencias externas por parte del piloto o ser generadas por el mismo controlador. El primer caso constituye un piloto parcialmente automático, mientras que el segundo de ellos es completamente automático.

El *software* del piloto automático que se encarga de seguir las referencias necesita un cierto apoyo físico que se encarga de correr los bucles de percepción y control. Este apoyo esta constituido por el *hardware* compuesto por diferentes componentes entre los que se encuentran los sensores y los microprocesadores.

Los sensores son los encargados de percibir el entorno que rodea al vehículo y le permiten estimar al sistema de control cuáles son los estados internos del quadrotor. Los sensores principales con los que suele contar un quadrotor son un GPS, IMU y sensor de altura.

El primero de ellos es utilizado para estimar la posición del vehículo en el espacio tridimensional, sin embargo solamente tiene precisión y aplicabilidad en misiones en un espacio exterior, para otras misiones en interiores se recomienda otro tipo de sensores como podría ser el reconocimiento de imagen, [4].

El *Inertial Measurement Unit* está compuesto por un acelerómetro de tres ejes y giroscopio de tres ejes, con la posibilidad de añadir un magnetómetro de tres ejes y un barómetro. Con la combinación de los tres primeros es posible estimar la orientación del quadrotor en el espacio y mediante el último es posible estimar la altura con respecto a una referencia que ha sido previamente calibrada.

El sensor de altura se puede corresponder con un barómetro como el que podría estar presente en la IMU o en un sensor de ultrasonidos. El primero de ellos permite calcular tanto la altura absoluta (con respecto al nivel del mar) o altura relativa (con respecto al suelo), en función como haya sido escogida la presión de referencia. El segundo de ellos solamente permite obtener una altura relativa con respecto al suelo aunque con mayor precisión que el primero. Por ello para vuelo cercano al suelo se recomienda el uso del sensor ultrasónico puesto que sus mediciones son más precisas. Para aplicaciones que requieran un vuelo alejado del suelo, las mediciones ultrasónicas dejan de ser válidas y por tanto se debería utilizar el barómetro.

Finalmente, en el microprocesador se ejecuta la lectura de cada uno de los sensores y se hace uso de estas mediciones para estimar los estados internos del vehículo, esta primera parte se conoce con el nombre de percepción. Es importante considerar que el periodo de actualización de los diferentes sensores es diferente. Posteriormente, en la parte de control se hace uso de las mediciones obtenidas para determinar que acciones de control se deben utilizar para conseguir el seguimiento de las referencias. Las acciones de control de un quadrotor son generalmente las velocidades de giro de cada uno de los rotores. En el caso de que se trate de un piloto completamente automático, también se genera, mediante el software del sistema de control que corre en el microprocesador, cuáles son las referencias de orientación, posición y velocidad que se deben suministrar a los bucles de control.

En la figura 2.6 se muestran las placas utilizadas por algunos de los pilotos automáticos de código abierto para quadrotor más populares que pueden encontrarse en el mercado. Estas placas tienen implementado el *software* del sistema de control en el microprocesador abordo, también tienen incluidos en el circuito impreso todos los sensores esenciales que permiten el control de orientación del quadrotor, acelerómetro, giroscopio, magnetómetro y barómetro. Estas placas permiten de forma sencilla incluir el control de vuelo a cualquier quadrotor con su estructura y sistema de propulsión siempre y cuando se ajusten sus parámetros físicos y las ganancias de los controladores. Estos pilotos automáticos también se han hecho muy populares gracias a las diferentes posibilidades que existen en cuanto a personalización de la plataforma de vuelo y del control de vuelo, de tal forma que mediante la misma placa es posible programar un control suave que permita hacer tareas de reconocimiento y otro control que permita a un quadrotor hacer maniobras más rápidas e incluso acrobáticas.

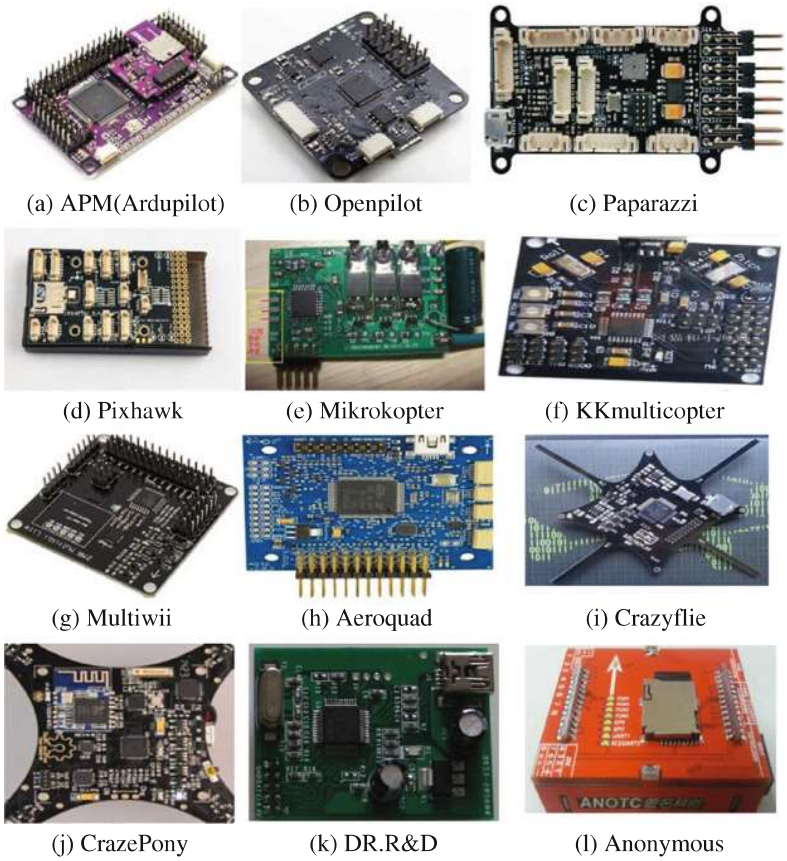


Figura 2.6: Placas de pilotos automáticos de código abierto para quadrotor [2]

Por otro lado, la estación de control en tierra o GCS (*Ground Control Station*) permite planificar los *waypoints* por los que debe pasar un vehículo y permite la comunicación con el control remoto de tal forma que se monitorizan todos los parámetros del vuelo e incluso permiten cambiar la sucesión de *waypoints* por las que va a pasar el vehículo. De igual forma que sucedía con los pilotos automáticos de código abierto, existen numerosos GCS de código abierto disponibles para ser utilizados. Es importante destacar que muchos de ellos están asociados a un determinado piloto automático de los que ya han sido introducidos antes. En la página de Ardupilot[4] se hace una revisión de todas las alternativas disponibles de estaciones de control en tierra donde se puede comparar cuales son las características y actuaciones que se puede esperar de cada uno de ellos. En la figura 2.7[5] se muestra una de las GCS más populares entre los usuarios de quadrotor.

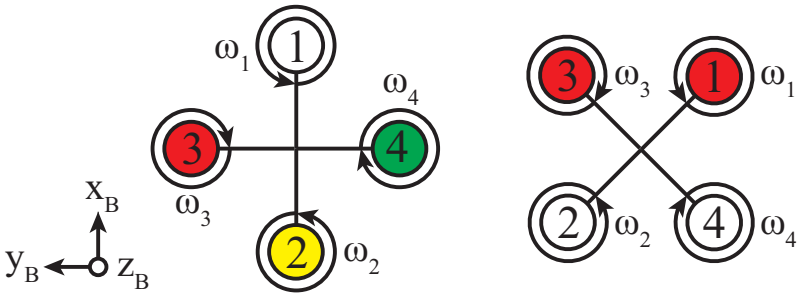
[4] `https://ardupilot.org/copter/docs/common-choosing-a-ground-station.html`
[5] `https://ardupilot.org/planner/index.html#home`

Figura 2.7: Estación de control en tierra: Mission Planner

Finalmente, la comunicación entre la GCS y el vehículo debe hacerse a través de telemetría con un determinado protocolo de comunicación. Uno de los más populares protocolos utilizados en comunicación de vehículos aéreos de pequeño tamaño es el MAVLink[6]. Algunos de los controladores que hacen uso de él son PX4, APM o Parrot Drone [2].

2.4 Configuraciones de un quadrotor

Existen dos configuraciones típicas a las que se ajustan la mayoría de los quadrotor. Tal y como se muestra en la figura 2.8 estas son la configuración en 'X' y en '+'. Estas configuraciones son idénticas pero una es la otra rotada 45°, por tanto las especificaciones dinámicas que se esperarán de ellas serán idénticas pero también rotadas. En el caso de la configuración en '+', esta presenta un rotor en la nariz y otro en la parte trasera, mientras que los otros dos estarían ubicados en los lados. La configuración en 'X' tendría dos rotores delante y otros dos detrás.



a) Configuración + b) Configuración X

Figura 2.8: Configuraciones geométricas más populares de un quadrotor

Cabe destacar que en la figura 2.8 se están representando los quadrotor en vista azimutal desde arriba. El número asociado a cada uno de los rotores será asumido como una convención y utilizado en futuras secciones en el desarrollo matemático del modelo dinámico. Además, se incluyen los sentidos de giro de cada uno de los rotores.

La configuración en 'X' presenta algunas ventajas sobre la configuración en '+' que hace que ses elegida preferiblemente por los usuarios y diseñadores. Asumiendo que las longitudes de los brazos del quadrotor tienen la misma longitud para ambas configuraciones, es posible observar

[6] `https://mavlink.io/en/`

cómo en la configuración en 'X' los rotores están más cerca de los ejes de giro, asumiendo que son giros en torno al eje x_B o y_B, lo que hace que el momento generado sea menor. Sin embargo, para la realización de un giro los cuatro rotores estarán actuando en favor de él a diferencia de la configuración en '+' donde solamente dos de ellos lo harán, por tanto la configuración en 'X' tiene una respuesta más rápida ante cambios en las referencias. La respuesta ante un giro en el eje z_B es idéntica para ambas geometrías.

En la configuración en 'X' es más fácil para un usuario determinar cuál es la parte delantera del quadrotor debido a que dos rotores están marcando esta zona mediante dos luces del mismo color, tal y como puede observarse en la figura 2.8.

La última de las razones por las que es preferible la configuración en 'X' es para tener la zona delantera despejada ya que la cámara estará orientada en esta dirección, en el caso de un quadrotor con la configuración en '+', uno de los brazos estará ubicado delante de la cámara obstruyendo parcialmente la visión.

La configuración para la cual se diseñará la ley de control será la de tipo 'X', aunque como se verá, es posible cambiar la aplicación de la ley a la configuración tipo '+' cambiando únicamente la matriz de mezclado introducida en la subsección 4.5.2.

2.5 Movimientos de un quadrotor

En esta sección se pretende realizar una introducción al principio de vuelo de un quadrotor, para ello se asumirá la configuración en 'X' mostrada en la sección 2.4 con la numeración y sentido de giro de los rotores explicitados.

Mediante la variación de la velocidad de giro de cada uno de los rotores se podrá modificar las tracciones en dirección vertical al quadrotor que generan cada uno de ellos, que tendrá asociado unos momentos en los ejes x_B y y_B debido a que los puntos de aplicación de las fuerzas no son el cdg del quadrotor. La tracción total del quadrotor se calculará como la suma de la de cada rotor y se aplicará en el cdg, posteriormente los momentos generados alrededor de cada eje por cada rotor también son sumados y aplicados en el cdg. Por otro lado, también aparecerá un momento de reacción en el eje z_B debido al giro de los rotores y que tendrá signo opuesto al de la velocidad angular de cada uno de ellos.

Para mantenerse en vuelo en punto fijo todos los rotores deberán proporcionar la misma tracción cuya suma debe ser igual a la fuerza gravitatoria, de esta forma, los momentos generados por cada uno de ellos se verán cancelados entre sí. Para conseguir vuelo en avance, puesto que los rotores solamente pueden proporcionar tracción en dirección del eje z_B, será necesario orientar el quadrotor en esa dirección. Si se quiere mover en dirección del eje x_B será necesaria proporcionar mayor velocidad de giro a los rotores 2 y 4 generando un momento que tienda a bajar la nariz del quadrotor. Los momentos de reacción en el eje z_B quedarían cancelados. En el caso que se quiera mover en dirección del eje y_B, los rotores a los que habrían que dar mayor velocidad de giro serían el 1 y 4.

El último de los movimientos sería el cambio de orientación con respecto al eje z_B, para ello, considerando los momentos de reacción, en sentido contrario al de giro, para girar con respecto al eje positivo será necesario darle mayor velocidad de giro a los rotores 3 y 4 y menor a los 1 y 2. Para conseguir el giro negativo será la operación contraria. Cabe destacar que mediante la combinación de cada uno de los movimientos es posible conseguir movimientos más complejos y elaborados que requieran cambios simultáneos en varios ejes.

3. Estado del arte

Uno de los objetivos que tiene el control automático en un vehículo es el de reducir la carga de trabajo que tiene el piloto. En un quadrotor el control es capaz de transformar unas referencias que han sido suministradas por un piloto en unas acciones de control sin intervención externa alguna, además, mediante el uso de este tipo de algoritmos es posible estabilizar el vehículo ante perturbaciones e incluso automatizar tareas que antes requerían de supervisión constante por el piloto como puede ser el control de la velocidad de vuelo o su altura.

Algunas de las estrategias que han sido utilizadas en los controladores que para conseguir la estabilización del vehículo y el seguimiento de referencias de orientación son el *Backstepping* [5], *Feedback Linearization* [6], *Sliding Mode Control* [7], PID [8], control robusto [9], control óptimo [10] y el control basado en aprendizaje [11].

Visto el número de alternativas diferentes que hay para conseguir la estabilización del vehículo y conseguir una determinada orientación es posible concluir que este problema ha sido ya ampliamente estudiado, obteniendo unos resultados satisfactorios. El interés de la investigación actual se centra en la obtención de algoritmos que permitan el seguimiento de trayectorias, la evasión de obstáculos, la robustez ante fallos y la planificación de trayectorias [12].

Existen dos aproximaciones principales al control de una trayectoria de un quadrotor, la primera de ellas contiene una dependencia con el tiempo, es decir, se impone que el quadrotor debe de pasar por ciertos puntos en un tiempo determinado lo que es una nueva restricción para el problema, suele recibir el nombre de *trajectory tracking control*. La segunda de las aproximaciones permite quitar la dependencia temporal con la posibilidad de explicitar a qué velocidad se debe mover el quadrotor por el camino, recibiendo el nombre de *path following* o PF. Además, los algoritmos empleados permiten imponer el ángulo de guiñada con el que un quadrotor se debe mover a lo largo de la trayectoria.

Generalmente, los controladores PF presentan ciertas ventajas con respecto a la primera alternativa. Puesto que no existen dependencias con el tiempo, estos son más fáciles de diseñar [13], la convergencia a la trayectoria deseada es más suave y requiere de menor coste en las acciones de control [14]. Ademas, mediante este tipo de seguimiento, el error en el transitorio es menor y su robustez mayor [12].

Cabe destacar que el controlador de trayectoria necesita de un conjunto de puntos que definen el camino para poder generar la referencia que debe de seguir el quadrotor. Es por ello por lo que generalmente el controlador de trayectoria recibe un camino que ha sido calculado como la trayectoria a seguir óptima entre dos puntos de tal forma que se minimiza el coste o se evitan algunos obstáculos. Además, en muchas ocasiones el sistema de trayectoria también está compuesto por un suavizador de trayectoria que se encarga de redondear las esquinas para que haya continuidad en las primeras derivadas del camino a seguir y no bruscos cambios de dirección.

A continuación se muestra un estudio de las referencias en las que se han empleado algoritmos para el control de la trayectoria.

En la referencia [14] se hace uso de un controlador del tipo *Backstepping* para garantizar la estabilidad del sistema y controlar las fuerzas y momentos debido a los rotores. Además, se usa el mismo tipo de controlador para garantizar una convergencia asintótica a una determinada trayectoria en el espacio tridimensional.

Por otro lado, en la referencia [15] se hace uso de la estrategia de control de *feedback linearization* para diseñar el controlador que garantice el seguimiento de una determinada trayectoria manteniendo una determinada velocidad o aceleración. Los resultados obtenidos demuestran que el sistema es capaz de converger de forma asintótica a la trayectoria previamente establecida.

En [16] se hace uso de una estrategia geométrica para conseguir el seguimiento de una determinada trayectoria. Concretamente se hace uso del NLGL, algoritmo que permite ajustarse a una trayectoria con una determinada velocidad especificada y una determinada guiñada. Cabe destacar que este algoritmo es geométrico y está basado en el concepto VTP o *Virtual Target Point*. También se muestran otras alternativas geométricas como podrían ser el *Carrotchase*, muy similar al NLGL, o el LOS y PLOS.

Cabe destacar que existen numerosos tipos de controladores de trayectoria similares como pueden ser los basados en Lyapunov, los *model predictive control* o los basados en aprendizaje reforzado. En la referencia [12] se hace un análisis extenso de todas las diferentes alternativas e incluso se compara su actuación para determinar la ley más efectiva.

4. Modelado del sistema

A lo largo del presente capítulo se exponen las consideraciones tomadas y el desarrollo matemático que permite obtener las ecuaciones que gobiernan el movimiento de un quadrotor con 6 gdl. También se muestra el proceso seguido en el modelado de las fuerzas y momentos que actúan sobre el sistema, el análisis del modelo obtenido y finalmente su implementación mediante el uso de diagrama de bloques en Simulink de Matlab.

4.1 Hipótesis

El primer paso que debe ser tomado para correctamente definir el comportamiento dinámico y modelar las ecuaciones que rigen el movimiento de un sistema es partir de unas hipótesis. Mediante este procedimiento se podrá simplificar el proceso de modelado obteniendo un resultado que se ajuste a la realidad de forma razonable y sin un coste computacional o técnico excesivo. Evidentemente, el resultado final que se alcance dependerá en gran medida del punto de partida escogido, siendo posible que en el caso que se ha simplificado en exceso el sistema mediante la suposición de hipótesis indebidas, los resultados no se ajusten con la precisión requerida a la realidad. En tal caso, sería necesario eliminar algunas de las suposiciones asumidas. Por tanto, todo proceso de modelado de un sistema requiere de una posterior validación con la realidad o con otros modelos que ya han sido validados para poder así asegurar que la simplificación del sistema ha sido realizada con la suficiente rigurosidad.

Las hipótesis que han sido tomadas en el desarrollo quedan listadas a continuación:

1. El quadrotor es un cuerpo rígido y que no tiene partes móviles excepto los rotores. El movimiento puede ser descrito mediante la traslación del cdg y una rotación alrededor de él.

2. La masa y los momentos de inercia y productos de inercia alrededor de los tres ejes son asumidos constantes.

3. Los planos Z-X y Z-Y son dos planos de simetría de tal forma que los productos de inercia son nulos, $I_{xy} = I_{xz} = I_{yz} = 0$, y los ejes locales del quadrotor son ejes principales de inercia. Además, se considera despreciable el momento de inercia de las palas de los rotores frente al del quadrotor.

4. El cdg de gravedad se encuentra en la intersección de los dos planos de simetría, que se corresponde con el centro geométrico.

5. Las únicas fuerzas que actúan sobre el sistema son la gravedad, la tracción de los rotores, las aerodinámicas y las debidas a la reacción del suelo. Los momentos que actúan sobre el sistema son los generados por la tracción de los rotores, los aerodinámicos y los debidos a

la reacción del suelo. La fuerza gravitatoria no genera ningún momento ya que se expresa el movimiento de traslación con el cdg.

6. Se desprecia la rotación de la tierra de tal forma que se puede considerar un eje fijo al suelo como inercial.

7. Se considera la densidad del aire constante e igual a la del nivel del mar, $\rho = 1.225\,kg/m^3$. Por tanto se asume que el rango de alturas sobre las que va a operar el quadrotor son reducidas.

4.2 Sistemas de referencia

Antes de definir los principios físicos que permiten la obtención de las ecuaciones que rigen el movimiento del quadrotor es necesario definir los ejes de referencia que se utilizan para medir la posición y orientación del quadrotor, así como las velocidades y aceleraciones lineales y angulares.

4.2.1. Sistema de referencia global

El sistema de referencia global o inercial (G) se encuentra fijo a la superficie del suelo. El eje x_G positivo se encuentra orientado hacia el norte, el eje z_G positivo está orientado en dirección vertical de tal forma que la gravedad tiene signo negativo, por último el eje y_G positivo está orientado formando un triedro a derechas, es decir, hacia el oeste. Para poder desarrollar el modelo matemático y aplicar las ecuaciones de Newton es necesario asumir que se trata de un sistema de referencia inercial, es decir, que no presenta aceleración lineal ni velocidad o aceleración angular.

4.2.2. Sistema de referencia horizonte local

El sistema de referencia horizonte local (H) tiene su origen en el cdg del quadrotor y son en todo momento paralelos a los ejes globales. Es decir, el sistema de referencia horizonte local ha sido obtenido a partir de una transformación de traslación a partir de los ejes globales. Por tanto, su eje x_H apuntará también al norte, el eje z_H en dirección vertical hacia arriba y el eje y_H hacia el oeste.

4.2.3. Sistema de referencia cuerpo

El sistema de referencia local o en ejes cuerpo (B) tiene su origen, al igual que los ejes horizonte local, en el cdg del quadrotor pero presentan como diferencia que son fijos a la estructura y se mueven con él. Para la configuración en X, mostrada en la sección 2.4, el eje x_B positivo está orientado formando 45° con los dos rotores delanteros, el eje z_B positivo está orientado en dirección vertical, hacia arriba, finalmente el eje y_B positivo está orientado para formar un triedro a derechas. Como se ha indicado en la sección 4.1, esta combinación de ejes son principales de inercia debido a la doble simetría del quadrotor.

En la figura 4.1 se representan los sistemas de referencia global y local. De esta figura es posible establecer los tres primeros estados del quadrotor, x, y, z. Estos son la posición que tiene su cdg con respecto al origen del sistema de referencia global y medido en ejes globales. Se define la variable $\vec{\xi}^G$ como el vector posición del quadrotor en ejes globales.

En la figura 4.2 se representan los ejes de horizonte local y los cuerpo aunque el quadrotor no se ha representado para facilitar la comprensión y reducir la carga visual. Partiendo de los ejes horizonte local es posible realizar tres rotaciones sucesivas alrededor de tres ejes que permiten obtener los

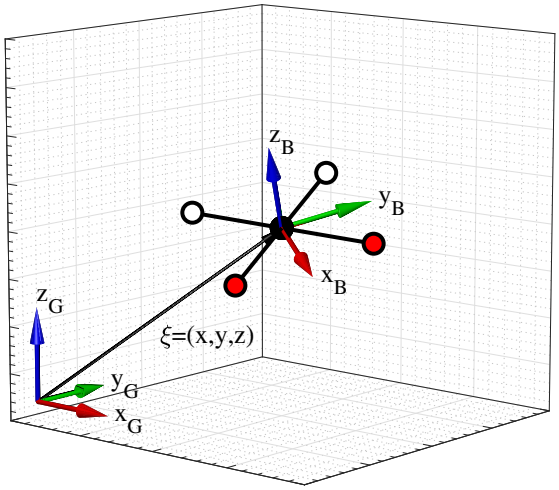


Figura 4.1: Sistemas de referencia global (G) y local (B)

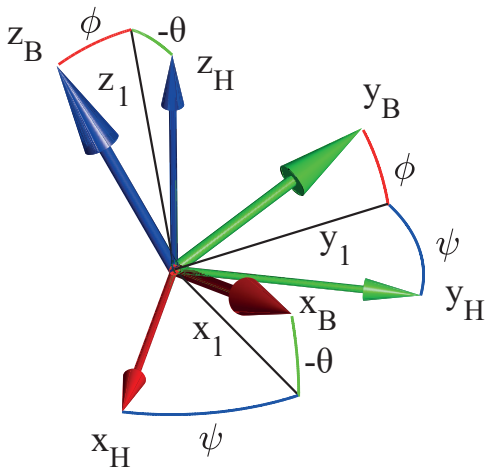


Figura 4.2: Sistemas de referencia horizonte local (H) y local (B)

ejes cuerpo, obteniendo así la orientación del quadrotor en el espacio. Este procedimiento se conoce como el de los ángulos de Euler. La secuencia de rotación seguida puede ser principalmente de dos tipos, extrínseca si el giro se realiza siempre alrededor de los ejes originales que en este caso serían los ejes horizonte local o intrínseca si el giro se realiza alrededor de los ejes solidarios al cuerpo del quadrotor.

Además, el orden y los ejes en que se realizan los giros pueden ser diferentes. Sin considerar si la naturaleza es extrínseca o intrínseca se pueden escoger diferentes secuencias.

1. Ángulos de Euler: (Z-X-Z, X-Y-X, Y-Z-Y, Z-Y-Z, X-Z-X, Y-X-Y)
2. Ángulos de Tait-Bryant: (X-Y-Z, Y-Z-X, Z-X-Y, X-Z-Y, Z-Y-X, Y-X-Z)

La convención que ha sido escogida para la caracterización de la orientación del quadrotor es una secuencia intrínseca Z-Y-X de Tait-Bryant. En la figura 4.2 se puede observar cómo, partiendo de los ejes horizonte local, se realiza en primer lugar un giro ψ alrededor del eje z_B, posteriormente se realiza un giro θ alrededor del eje y_B y finalmente un giro ϕ alrededor del eje x_B. De esta figura es posible observar cómo cambia la orientación de cada uno de los ejes locales.

Concretamente el ángulo $\psi \in [-180°, 180°]$ recibe el nombre de ángulo de rumbo o de guiñada, $\theta \in [-90°, 90°]$ es el ángulo de asiento longitudinal o cabeceo y $\phi \in [-180°, 180°]$ es el ángulo de asiento lateral o alabeo. De la figura 4.2 y considerando que se está representando el giro positivo de ϕ y ψ y el giro negativo de θ, es posible establecer que una guiñada positiva gira la nariz del quadrotor hacia la izquierda, un giro positivo del cabeceo mueve la nariz del quadrotor hacia abajo y finalmente un giro positivo del alabeo levanta la parte izquierda y baja la derecha del quadrotor.

Mediante $\vec{\xi}^G$ y los tres ángulos de Euler, (ϕ, θ, ψ), es posible determinar la posición y orientación del quadrotor en el espacio y son los estados base para determinar el modelo cinemático y dinámico del sistema.

4.3 Ecuaciones cinemáticas

El primer conjunto de ecuaciones necesarias para expresar el movimiento de un objeto con 6 gdl son las cinemáticas. La integración de estas ecuaciones permite obtener la trayectoria y la orientación a lo largo del tiempo. Se encuentran compuestas por 6 ecuaciones, 3 debidas al movimiento de traslación y tres debidas al movimiento de rotación.

4.3.1. Ecuaciones cinemáticas de traslación

En la subsección 4.2.1 se introdujo cuáles eran los tres primeros estados del quadrotor, los correspondientes a su posición en ejes globales, $\vec{\xi}^G = \begin{bmatrix} x & y & z \end{bmatrix}^T$.

$$R_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{4.1a}$$

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{4.1b}$$

$$R_Z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1c}$$

Por otro lado, la velocidad del quadrotor puede ser expresada en los dos sistemas de referencia de interés, ejes globales y locales. Para el primer caso la velocidad se define como $\vec{v}^G = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$, mientras que para el segundo caso, $\vec{v}^B = \begin{bmatrix} u & v & w \end{bmatrix}^T$. La relación existente entre ambos es deducible a partir de los ángulos de Euler, para ello se ha considerado la secuencia de giros intrínseca Z-Y-X.

$$\begin{gathered} {}^G R^B = R_{ZYX}(\psi,\theta,\phi) = R_Z(\psi)R_Y(\theta)R_X(\phi) = \\ \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \end{gathered} \tag{4.2}$$

En la ecuación (4.1) se representan las matrices de rotación sobre cada uno de los ejes. Es importante destacar que se tratan de matrices ortogonales donde el giro contrario está representado por sus matrices inversas que son iguales a las traspuestas. La ecuación (4.2) muestra la matriz de rotación que permite pasar de ejes locales a ejes globales, que se obtiene mediante una combinación de las matrices de rotación sobre cada eje. Esta matriz también es ortogonal y su inversa se corresponde con la traspuesta que representa el giro contrario, ${}^B R^G = \left({}^G R^B\right)^{-1} = \left({}^G R^B\right)^T$. Para reducir el tamaño de las ecuaciones se ha sustituido cada una de las funciones trigonométricas que aparecen por su primera letra.

Finalmente en la ecuación (4.3) se muestran las tres primeras ecuaciones cinemáticas del movimiento del quadrotor en forma matricial.

$$\vec{v}^G = {}^G R^B \vec{v}^B \tag{4.3}$$

4.3.2. Ecuaciones cinemáticas de rotación

De forma similar que se ha hecho para la traslación del quadrotor es necesario establecer una relación entre las velocidades angulares $\vec{\omega} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$ y las velocidades angulares en ejes locales $\vec{\omega}^B = \begin{bmatrix} p & q & r \end{bmatrix}^T$.

Para la obtención de la siguiente relación es necesario considerar que el vector $\vec{\omega}$ no forma un conjunto mutuamente ortogonal y por tanto no se podrá aplicar la transformación mostrada en la ecuación (4.3). El principal motivo por el que no se trata de un conjunto ortogonal es debido a la

sucesión de giros intrínsecos de los ángulos de Euler. Como se puede observar en la figura 4.2, los giros se realizan siempre sobre los ejes locales del cuerpo, que serían unos ejes móviles y no sobre los ejes globales. En esta figura es posible observar como las velocidades angulares en ejes cuerpo, (p,q,r), se medirían sobre los ejes $\{B:(x_B,y_B,z_B)\}$ pero la velocidad angular $\dot{\psi}$ se mediría sobre el eje z_H, $\dot{\theta}$ sobre el eje auxiliar y_1 y $\dot{\phi}$ sobre el eje x_B.

Por tanto es necesario rotar cada una de las derivadas temporales de los ángulos de Euler, $\dot{\psi}$, $\dot{\theta}$ y $\dot{\phi}$, para hacerlas coincidir con los ejes cuerpo. Es importante tener en cuenta que el proceso de giro que se está haciendo, pasando a ejes locales, es el inverso al caso de la traslación, donde se pasaba de ejes locales a globales y por ello se trabajará con las traspuestas de las matrices de rotación. La conversión de $\vec{\omega}$ a $\vec{\omega}^B$ se muestra en las ecuaciones (4.4), donde T es la matriz de transformaciones angulares que en este caso no es ortogonal. La expresión que se encuentra en esta ecuación se corresponden con las tres ecuaciones cinemáticas de rotación en forma matricial.

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + R_X(\phi)^T \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + (R_Y(\theta)R_X(\phi))^T \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} = T^{-1} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \tag{4.4a}$$

$$T = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix} \tag{4.4b}$$

Para finalizar esta sección, en las ecuaciones (4.5) se muestra el conjunto de ecuaciones cinemáticas que gobiernan el movimiento del quadrotor, es importante notar que estas ecuaciones son iguales para cualquier sistema con 6 gdl siempre que se asuma la misma sucesión de los ángulos de Euler.

$$\dot{x} = u[c(\theta)c(\psi)] + v[s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi)] + w[c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi)] \tag{4.5a}$$

$$\dot{y} = u[c(\theta)s(\psi)] + v[s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi)] + w[c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi)] \tag{4.5b}$$

$$\dot{z} = u[-s(\theta)] + v[s(\phi)c(\theta)] + w[c(\phi)c(\theta)] \tag{4.5c}$$

$$\dot{\phi} = p + q[s(\phi)t(\theta)] + r[c(\phi)t(\theta)] \tag{4.5d}$$

$$\dot{\theta} = q[c(\phi)] + r[-s(\phi)] \tag{4.5e}$$

$$\dot{\psi} = q\left[\frac{s(\phi)}{c(\theta)}\right] + r\left[\frac{c(\phi)}{c(\theta)}\right] \tag{4.5f}$$

Si se analizan las ecuaciones cinemáticas es posible observar que existe una singularidad para $\theta = \pm 90^\circ$, con este ángulo de cabeceo, las velocidades angulares $\dot{\phi}$ y $\dot{\psi}$ se hacen infinitas. En el caso de aplicación de un quadrotor este problema no es excesivamente preocupante ya que esta orientación no es habitual en una misión donde hay seguimiento de trayectorias, por lo que puede ser ignorado. En el caso de que la misión requiera de ángulo de cabeceo cercanos a la singularidad, se puede sustituir el empleo de los ángulos de Euler por cosenos directores o quaterniones que no presentan este problema.

Esta singularidad conocida con el nombre de 'Gimbal Lock', es debida a que la orientación no está definida de forma unívoca. En función de la secuencia de giros elegida sucede para una orientación determinada, como se ha indicado anteriormente en este caso sucede para un cabeceo de $\pm 90^\circ$. Con esta orientación, y como se muestra en la figura 4.3, los ejes z_H y x_B se encuentran alineados y por tanto un cambio en el ángulo de guiñada, ψ, o alabeo ϕ tiene el mismo resultado. Además, es posible llegar a esta orientación mediante dos combinaciones de los ángulos de Euler. La primera de ellas es mediante la guiñada y el cabeceo y la segunda mediante el cabeceo y el alabeo.

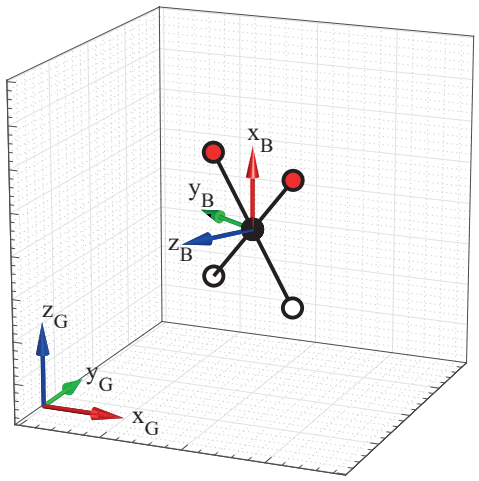


Figura 4.3: Singularidad en las velocidades angulares debido al 'Gimbal Lock'

4.4 Ecuaciones dinámicas

Al igual que sucedía con las ecuaciones cinemáticas, las ecuaciones dinámicas también están compuestas por 6 ecuaciones (3 de traslación y 3 de rotación). En estas ecuaciones se relacionan las derivadas temporales de los estados medidos en ejes cuerpo, $\dot{\vec{v}}^B = \begin{bmatrix} \dot{u} & \dot{v} & \dot{w} \end{bmatrix}^T$ y $\dot{\vec{\omega}}^B = \begin{bmatrix} \dot{p} & \dot{q} & \dot{r} \end{bmatrix}^T$ con ellos mismos y las fuerzas y momentos externos aplicados. Estas 6 nuevas variables se corresponden con los últimos estados que forman el conjunto de 12 estados que describen el sistema.

La obtención del conjunto de ecuaciones dinámicas se hará en base de los principios básicos de la mecánica.

4.4.1. Ecuaciones dinámicas de traslación

En el caso de las ecuaciones dinámicas de traslación, en la sección 4.1 se introdujeron las hipótesis de cuerpo rígido con masa constante y por tanto es posible emplear las ecuaciones de Newton-Euler para describir la dinámica del quadrotor. Estas ecuaciones establecen que, en eje de referencia local, las fuerzas externas, $\vec{F}^B = \begin{bmatrix} F_x^B & F_y^B & F_z^B \end{bmatrix}^T$, son iguales a la suma de los términos de aceleración de la masa, $m\dot{\vec{v}}^B$, y la fuerza centrífuga, $\vec{\omega}^B \times \left(m\vec{v}^B\right)$.

En el caso de que se decidan expresar las ecuaciones en el eje inercial, los términos correspondientes a las fuerzas centrífugas se verían cancelados, simplificándose así la tarea de obtención de las ecuaciones. Sin embargo, se va desarrollar la formulación en los ejes de referencia locales para evitar segundas derivadas de las variables x, y, z y medir las velocidades sobre los ejes cuerpo. De esta forma, para cada uno de los estados del sistema aparecerá su primera derivada.

En la ecuación (4.6) se muestran las tres ecuaciones dinámicas de traslación en forma matricial.

$$m\dot{\vec{v}}^B + \vec{\omega}^B \times \left(m\vec{v}^B\right) = \vec{F}^B \tag{4.6}$$

El vector de fuerzas $\vec{F}^B$ contiene todas las fuerzas externas que actúan sobre el quadrotor, entre ellas se encuentran las gravitacionales, las de tracción debidas al giro de los rotores, las aerodinámicas y la reacción del suelo. Estas se encuentran desarrolladas en la sección 4.5.

4.4.2. Ecuaciones dinámicas de rotación

La obtención de las tres últimas ecuaciones del modelo dinámico del sistema se hará en base a las hipótesis listadas en la sección 4.1. En la ecuación (4.7) se muestra la matriz de inercias del quadrotor donde se ha asumido que son constantes en el tiempo y que los productos de inercias son nulos, es decir, los ejes del quadrotor asumidos son ejes principales de inercia.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{4.7}$$

La aplicación de la segunda ley de la mecánica en ejes de referencia locales establece que el momento externo aplicado, $\vec{M}^B = \begin{bmatrix} M_x^B & M_y^B & M_z^B \end{bmatrix}^T$ es igual a la derivada temporal absoluta del momento angular del sistema. En la ecuación (4.8) se muestran las tres ecuaciones dinámicas de rotación en forma matricial. Cada uno de los términos que forman el vector $\vec{M}^B$ son analizados en la sección 4.5.

$$I\dot{\vec{\omega}}^B + \vec{\omega}^B \times \left(I\vec{\omega}^B\right) + \vec{\Gamma}^B = \vec{M}^B \tag{4.8}$$

El término $\vec{\Gamma}^B$ se corresponde con el término giroscópico que depende de $\vec{\omega}^B$, de las velocidades de giro de los rotores, $\omega_1, \omega_2, \omega_3, \omega_4$, y del momento de inercia de las palas alrededor de su eje de giro, I_R. Además, su expresión depende del sentido de giro de cada rotor, asumiendo un rotor en configuración X como el descrito en 2.4, el valor de este término queda mostrado en las ecuaciones (4.9). Usualmente este término puede ser despreciado frente al resto de componentes de la ecuación ya que la inercia del rotor es considerablemente menor que la del quadrotor en su conjunto y el término ω_G suele ser bastante pequeño puesto que la velocidad de giro de los rotores son similares, sin embargo en este modelo no se va a realizar esta simplificación.

El término giroscópico aparece cuando existe una velocidad de alabeo o cabeceo no nula que hace que el giro de los rotores cambie de plano y las velocidades de giro de los rotores 1 y 2 no se cancelan con la del 3 y 4, esta reacción sería una oposición a este cambio.

$$\vec{\Gamma}^B = \begin{pmatrix} \Gamma_x^B \\ \Gamma_y^B \\ \Gamma_z^B \end{pmatrix} = \vec{\omega}^B \times \begin{pmatrix} 0 \\ 0 \\ I_R\omega_G \end{pmatrix} = I_R\omega_G \begin{pmatrix} q \\ -p \\ 0 \end{pmatrix} \tag{4.9a}$$

$$\omega_G = \omega_1 + \omega_2 - \omega_3 - \omega_4 \tag{4.9b}$$

En las ecuaciones (4.10) se muestran el conjunto de ecuaciones dinámicas que gobiernan el movimiento del quadrotor. Estas ecuaciones a diferencia de las cinemáticas, dependen de la geometría y de las fuerzas y momentos externos que afectan al sistema que se ha modelado y por tanto varían para cada uno de los sistemas con 6 gdl que se analicen.

$$\dot{u} = rv - qw + \frac{F_x^B}{m} \tag{4.10a}$$

$$\dot{v} = pw - ru + \frac{F_y^B}{m} \tag{4.10b}$$

$$\dot{w} = qu - pv + \frac{F_z^B}{m} \tag{4.10c}$$

$$\dot{p} = \frac{(I_{yy} - I_{zz})}{I_{xx}} qr + \frac{M_x^B - \Gamma_x^B}{I_{xx}} \tag{4.10d}$$

$$\dot{q} = \frac{(I_{zz} - I_{xx})}{I_{yy}} pr + \frac{M_y^B - \Gamma_y^B}{I_{yy}} \tag{4.10e}$$

$$\dot{r} = \frac{(I_{xx} - I_{yy})}{I_{zz}} pq + \frac{M_y^B - \Gamma_z^B}{I_{zz}} \tag{4.10f}$$

Si se comparan las ecuaciones dinámicas con las cinemáticas es posible observar ciertas diferencias entre ellas. Las ecuaciones cinemáticas mostradas en (4.5) únicamente dependen de los estados del sistema y son utilizadas para describir el movimiento de traslación y rotación del quadrotor. Por otro lado, las ecuaciones dinámicas solamente dependen de los estados del sistema que están expresados en ejes cuerpo, (u, v, w, p, q, r) y mediante unas entradas que son las fuerzas y los momentos se calculan las derivadas temporales de estos estados.

4.5 Fuerzas y momentos externos

Las únicas variables que se deben definir para completamente establecer el modelo del quadrotor son las fuerzas, $\vec{F}^B$, y momentos, $\vec{M}^B$, externos que actúan sobre él. Estas variables dependen en gran medida de que tipo de sistema se esté modelando, es decir, en función de sí se está modelando un avión o un quadrotor tendremos unas fuerzas u otras.

Una de las características principales que hay que considerar a la hora de modelar estos términos es en que sistema de referencia se debería hacer. Puesto que las ecuaciones dinámicas se han modelado en el sistema de referencia local del quadrotor, las fuerzas y los momentos deben estar expresados en este sistema también.

En las ecuaciones (4.11) se muestran las diferentes componentes que conforman las fuerzas y momentos externos. El subíndice 'G' hace referencia a los términos gravitatorios, 'T' a los términos debidos a la tracción de los rotores, 'A' a los aerodinámicos y 'GR' a la reacción del suelo. Se puede observar que, como se introdujo en la sección 4.1, no existen momentos debidos a los términos gravitatorios ya que la fuerza debida a la gravedad se aplica en el cdg del quadrotor.

$$\vec{F}^B = \vec{F}_G^B + \vec{F}_T^B + \vec{F}_A^B + \vec{F}_{GR}^B \tag{4.11a}$$

$$\vec{M}^B = \vec{M}_T^B + \vec{M}_A^B + \vec{M}_{GR}^B \tag{4.11b}$$

4.5.1. Términos gravitatorios

Para obtener los vector que contiene los términos correspondientes a las fuerzas gravitatorias es necesario rotar el vector gravedad en ejes globales a ejes locales mediante la matriz de rotación mostrada en la ecuación (4.2). Puesto que se está pasando de ejes globales a ejes locales se utilizará la traspuesta de esta matrix. En la ecuación (4.12) se muestra las fuerza gravitatoria expresada en

ejes locales, como se puede observar, la dirección que tenga este vector depende de los ángulos de Euler.

$$\vec{F}_G^B = \begin{pmatrix} F_{G_x}^B \\ F_{G_y}^B \\ F_{G_z}^B \end{pmatrix} = {}^B R^G \vec{F}_G^G = {}^B R^G \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} = \begin{pmatrix} mgs(\theta) \\ -mgc(\theta)s(\phi) \\ -mgc(\theta)c(\phi) \end{pmatrix} \tag{4.12}$$

4.5.2. Términos debidos a la tracción de los rotores

Generalmente y para el caso de estudio actual, la tracción producida por un rotor tiene exclusivamente dirección vertical, es decir, la tracción de cada uno de los rotores es paralela al eje z_B. Por tanto, las componentes en los ejes x_B y y_B son nulas tal y como se muestra en la ecuación (4.13). La componente T simboliza la tracción total de los 4 rotores del quadrotor y es la suma de la tracción proporcionada por cada uno de los 'n' rotores que forman el quadrotor.

$$\vec{F}_T^B = \vec{T}^B = \begin{pmatrix} T_x^B \\ T_y^B \\ T_z^B \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \sum_1^n T_i \end{pmatrix} \tag{4.13}$$

En el caso en el que un quadrotor pudiera orientar el vector tracción correspondiente a cada uno de sus rotores, los términos en los ejes x_B y y_B no serían nulos y serían dependientes de los ángulos de Euler que se utilicen para orientar el rotor en el espacio con respecto al cuerpo. Este tipo de quadrotor recibe el nombre de 'Tilting Rotors'. Las principales ventajas que aporta esta configuración con respecto a la convencional es que la adición de más gdl al sistema permite separar los movimientos del quadrotor de tal forma que se pueden seguir las referencias de 12 estados independientes, por ello, se incrementa la maniobrabilidad y robustez en las actuaciones, [17].

Otra configuración de los rotores de un quadrotor que hace que los términos en los ejes x_B y y_B no se cancelen es aquella en la que los rotores presentan una pequeña inclinación hacia adentro. Esto permite aumentar la estabilidad del sistema, [18].

Para el cálculo de la tracción proporcionada por un rotor se hace uso de la combinación de la teoría de elemento de pala y de la teoría de la cantidad de movimiento. En la referencia [19], se muestra la obtención de las expresiones para un rotor que se asume rígido y donde se han despreciado los movimientos de batimiento y arrastre de las palas. En cuanto al ángulo de paso se asume constante para una determinada sección de la pala. Esta configuración para el rotor de un quadrotor es bastante cercana a la realidad ya que por norma general el único gdl de libertad que presentan es la variación de la velocidad de giro. De esta forma se puede considerar que la tracción es proporcional al cuadrado de la velocidad de giro del rotor tal y como se muestra en la ecuación (4.14), donde k se conoce como el factor de tracción. Cabe destacar una de las hipótesis en la obtención de las fuerzas y momentos de reacción debidos a la tracción de los rotores y es que se ha asumido que se encuentran cerca del vuelo en punto fijo.

Si se analizan los momentos generados en los tres ejes debido a la tracción de los motores es posible observar que ninguno de los términos es nulo.

Los momentos en cuanto a los ejes x_B y y_B son debidos directamente a la tracción de cada uno de los rotores que al encontrarse separados del centro de gravedad por una distancia generan un momento. Considerando la configuración en 'X', expuesta en la sección 2.4, la distancia de cada uno de los rotores a cada uno de los ejes x_B y y_B es $\sqrt{2}/2l$, siendo l la longitud del brazo.

Por otro lado, el momento de reacción debido al giro del rotor alrededor de su eje y que tiene sentido contrario a su dirección de giro, haciendo al quadrotor girar sobre su eje z_B, también puede ser modelado como proporcional al cuadrado de la velocidad de giro tal y como se muestra en

la ecuación (4.14), donde b se conoce como el factor de resistencia. Para definir correctamente el signo de los momentos de reacción inducidos en el eje z_B, se han considerado los sentidos de giro expuestos para la configuración 'X' en la sección 2.4. Visto desde arriba, los rotores que giran en sentido antihorario, rotores 1 y 2, inducirán un momento de reacción con sentido contrario y negativo. Para los rotores que giran en sentido horario, rotores 3 y 4, el momento inducido tendrá signo positivo.

Finalmente, en las ecuaciones (4.14) se muestran las expresiones de la tracción y los momentos en los tres ejes en función de la velocidad de giro de cada uno de los rotores.

$$T = k\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right) \tag{4.14a}$$

$$M_{T_x}^B = kl\frac{\sqrt{2}}{2}\left(\omega_2^2 + \omega_3^2 - \omega_1^2 - \omega_4^2\right) \tag{4.14b}$$

$$M_{T_y}^B = kl\frac{\sqrt{2}}{2}\left(\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2\right) \tag{4.14c}$$

$$M_{T_z}^B = b\left(\omega_3^2 + \omega_4^2 - \omega_1^2 - \omega_2^2\right) \tag{4.14d}$$

Expresando las ecuaciones 4.14 en forma matricial es posible obtener una conversión de velocidades angulares de los rotores a fuerzas y momentos resultantes, y además, mediante la operación inversa, es posible el cálculo de qué velocidades angulares necesitan ser mandadas a los motores para conseguir unas determinadas fuerzas y momentos. La principal ventaja que aporta este procedimiento es que, como se podrá observar en la sección 4.6, se considerarán que las acciones de control son directamente la fuerza de tracción y los 3 momentos resultantes, ya que el sistema ya es lineal con respecto a ellas.

La conversión de fuerzas y momentos a velocidades angulares y viceversa, a través de la matriz 'Mixer', se muestra en las ecuaciones 4.15

$$\begin{pmatrix} T \\ M_{T_x}^B \\ M_{T_y}^B \\ M_{T_z}^B \end{pmatrix} = \overbrace{\begin{bmatrix} k & k & k & k \\ -kl\frac{\sqrt{2}}{2} & kl\frac{\sqrt{2}}{2} & kl\frac{\sqrt{2}}{2} & -kl\frac{\sqrt{2}}{2} \\ -kl\frac{\sqrt{2}}{2} & kl\frac{\sqrt{2}}{2} & -kl\frac{\sqrt{2}}{2} & kl\frac{\sqrt{2}}{2} \\ -b & -b & b & b \end{bmatrix}}^{Mixer^{-1}} \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} \tag{4.15a}$$

$$\begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} = Mixer \begin{pmatrix} T \\ M_{T_x}^B \\ M_{T_y}^B \\ M_{T_z}^B \end{pmatrix} \tag{4.15b}$$

Finalmente, para correctamente modelar el comportamiento de la planta tractora es necesario añadir dos fenómenos que no han sido tenido en cuenta en el desarrollo hasta este momento.

El primero de ellos es el rango de velocidades admisibles de rotación de cada uno de los motores que conforman el sistema, y es que cada motor presenta una velocidad máxima que no puede ser sobrepasada. De esta forma, cuando el sistema de control mande una velocidad de giro superior a la máxima, el motor saturará. Por otro lado, aunque los motores eléctricos sean capaces de invertir su sentido de giro, si se considera la aerodinámica de las palas de los rotores, esta inversión no producirá tracción negativa y por tanto la velocidad mínima del rotor está limitada a 0.

Una vez que el sistema de control del quadrotor ha calculado las correspondientes acciones de control, $\left(T, M_{T_x}^B, M_{T_y}^B, M_{T_z}^B\right)$ para seguir una determinada referencia, mediante la matrix 'Mixer' se

calculan las velocidades de giro que deben ser mandadas a cada uno de los motores y estas son pasadas por una función que satura los valores superiores a la velocidad máxima e inferiores a la mínima.

El segundo de los fenómenos que debe ser considerado es que la actuación de los rotores no es inmediata. Cuando una determinada velocidad de giro es mandada a cada uno de los motores, estos necesitan un cierto tiempo característico, T_m, para pasar el transitorio y alcanzar el valor estacionario. Esta dinámica de los motores eléctricos sin escobillas (BLDC) puede ser simplificada mediante un filtro paso bajo de primer orden, [2], cuya expresión se muestra en la ecuación (4.16) en el dominio de Laplace, donde ω_i representa la velocidad de giro del motor número i y $\omega_{i_{ss}}$ representa la velocidad mandada a la que debe girar el mismo motor. Sin embargo, aunque exista este transitorio, por lo general, la dinámica de la respuesta de los motores de los quadrotor suele ser lo suficientemente rápida como para despreciarse, [20]. En el modelo desarrollado en este documento no se tendrá en cuenta esta simplificación.

$$\omega_i = \frac{1}{T_m s + 1} \omega_{i_{ss}} \tag{4.16}$$

4.5.3. Términos aerodinámicos

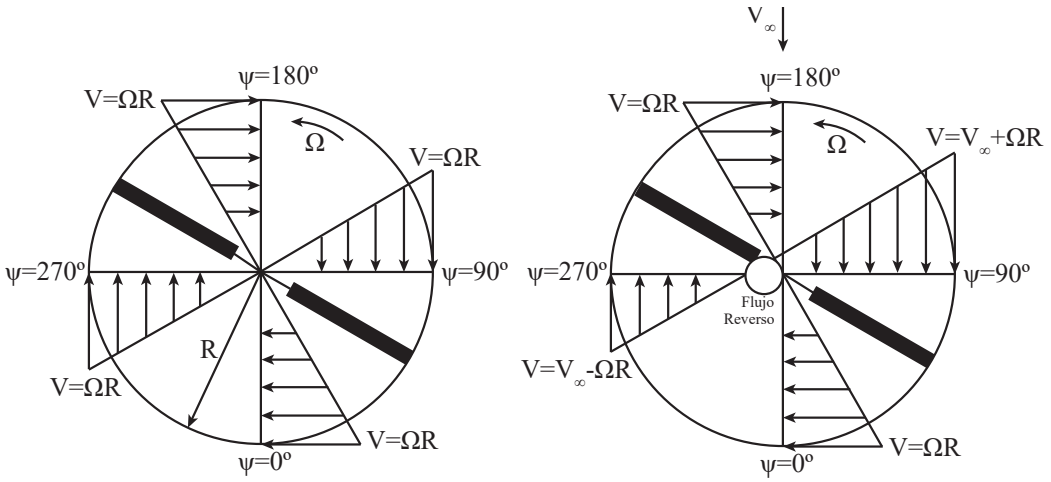
Los principales términos aerodinámicos que actúan sobre el modelo dinámico de un quadrotor son la resistencia aerodinámica que aparece cuando empieza a moverse relativamente al fluido que lo rodea y perturbaciones que pueden ser debidas a rachas de viento. En el modelado de estos términos aerodinámicos se ha asumido que la resistencia aerodinámica es una fuerza que actúa en dirección contraria a la velocidad del quadrotor y se encuentra aplicada en su cdg por lo que no habrán momentos debido a la aerodinámica. Los términos debido a perturbaciones pueden ser tanto momentos como fuerzas de cualquier tipo, escalón, impulso o armónico. Las fuerzas y momentos de perturbación también pueden ser debidos a naturalezas que no sean aerodinámicas.

La resistencia aerodinámica que afecta a un quadrotor tiene diferentes componentes, entre ellos, resistencia parásita, inducida, batimiento de pala, de perfil de pala y traslacional, [19].

La resistencia parásita se debe a las superficies no sustentadoras y la viscosidad del aire, las principales contribuciones vienen de la estructura del quadrotor y de los motores. Aunque a velocidades altas esta contribución pueda ser elevada, a las velocidades a las que se suele mover un quadrotor suele ser despreciable frente al resto de componentes.

En la figura 4.4 se puede observar que cuando el quadrotor se encuentra volando en estacionario, la distribución de la velocidad incidente a lo largo de las palas es axisimétrica, por tanto, la resistencia inducida que se produce debido a la sustentación es también axisimétrica, de tal forma que la componente resultante de resistencia inducida es nula. Sin embargo, cuando el quadrotor comienza a desplazarse con velocidad en su plano horizontal, la sección de avance, ángulos de azimut comprendidos entre 0° y 180°, experimenta velocidades de incidencia superiores, mientras que la sección de retroceso, ángulos de azimut comprendidos entre 180° y 360°, experimenta velocidades de incidencia inferiores, llegando incluso a haber flujo reverso en el llamado 'círculo de inversión'. En esta situación la resistencia inducida de la zona de avance es mayor que la de retroceso y, por tanto, aparece una componente de resistencia inducida resultante en sentido opuesto al de movimiento que será mayor cuanto mayor sea la velocidad en el plano horizontal y la tracción del rotor. Este fenómeno es despreciable en rotores de gran tamaño donde sus palas tienen poca rigidez y se permite el movimiento de batimiento, sin embargo, en rotores de quadrotor puede llegar a ser un término importante.

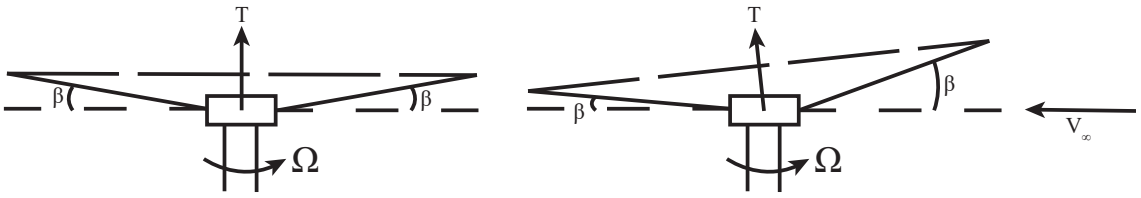
En la figura 4.5 se muestra el ángulo de batimiento de la pala, β, en estacionario y avance, este parámetro se define como el ángulo que forma la pala con el plano horizontal. Cuando la pala se



a) Vuelo estacionario b) Vuelo en avance

Figura 4.4: Distribución de velocidades de un rotor en estacionario y avance

encuentra en la zona de avance, la sustentación se ve incrementada y se genera un momento que tiende a levantar la punta de la pala, en la zona de retroceso sucede lo contrario. Por tanto, la punta de la pala sigue una trayectoria aumentando su ángulo de batimiento y disminuyéndolo. En el caso de la figura 4.4, el máximo batimiento se encontraría en el ángulo de azimut de 180°, mientras que el mínimo se encontraría en 0°, tal y como se muestra en la figura 4.5. También se puede observar que la dinámica de batimiento se encuentra desfasada 90° con respecto a la máxima velocidad de incidencia de la corriente. Por otro lado, la máxima velocidad de batimiento se encuentra en 90°, mientras que la mínima se encuentra en 270°.



a) Vuelo estacionario b) Vuelo en avance

Figura 4.5: Ángulo de batimiento en estacionario y avance

El plano que une las puntas de la pala en todas sus coordenadas azimutales se conoce como 'plano de puntas' y tiene la característica que el vector tracción es perpendicular a él. Por tanto, en vuelo en avance aparece una componente que se opone al movimiento que es proporcional a la velocidad en el plano horizontal y al vector tracción. En [21] se utiliza el fenómeno del batimiento de la pala para estimar la velocidad del quadrotor. En rotores rígidos, suele ser el caso de los quadrotor donde el ángulo de batimiento es casi nulo, este término resistivo es despreciable y por ello en la subsección 4.5.2 se asumió que el vector tracción era paralelo al eje z_B. En rotores flexibles, el efecto del batimiento es estabilizador ya que aparece una componente en la tracción que se opone al movimiento.

Es posible considerar los términos de resistencia inducida y de batimiento de las palas conjuntamente como un término de resistencia aerodinámica que depende de la velocidad de vuelo en los ejes x_B y y_B.

La resistencia traslacional es debida a la redirección del flujo incidente sobre un rotor en el movimiento de avance tal y como puede observarse en a figura 4.6. Parte de la tracción que se produce

en un rotor en su movimiento de avance es debido al cambio de dirección de la corriente de aire que como consecuencia tiene este término de resistencia asociado.

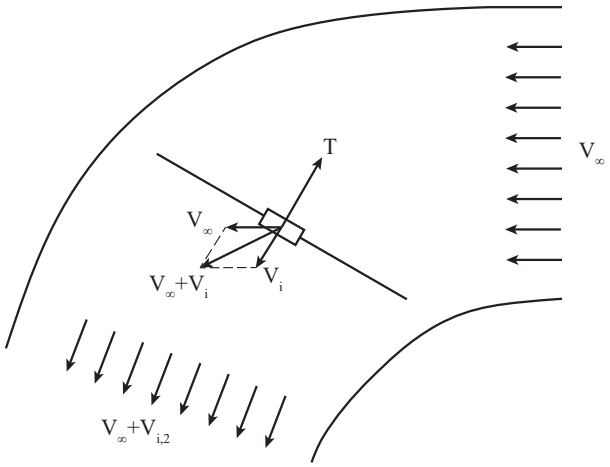


Figura 4.6: Cambio de dirección de la corriente de aire debido a la tracción

El último de los términos de resistencia aerodinámica es el debido a la resistencia parasítica de las superficies tractoras que son los perfiles de las palas. De forma similar a como ocurría con el término de resistencia inducida de las palas, cuando el quadrotor se encuentra en vuelo estacionario la resistencia debida a la viscosidad del aire es axisimétrica respecto al eje de giro, sin embargo en el momento que existe velocidad en el plano horizontal del quadrotor la simetría deja de existir ya que la sección de avance experimenta una velocidad incidente mayor que la de retroceso. Por tanto, en movimiento en avance, existe una componente de resistencia de perfil en sentido opuesto al movimiento.

En [19] se desarrolla un modelo que agrupa todos las contribuciones en función de la velocidad de movimiento y la tracción generada por los rotores, (4.17). En este modelo se ha incluido la velocidad como una velocidad relativa con el viento, $\vec{W}^B$, de tal forma que se pueden introducir perturbaciones debidas a rachas o viento constante. Tanto el vector $\vec{v}^B$ como $\vec{W}^B$ se encuentran medidos en el sistema de referencia en ejes cuerpo.

$$\vec{F}_A^B = \vec{D}^B = \begin{pmatrix} D_x^B \\ D_y^B \\ D_z^B \end{pmatrix} = -T \begin{bmatrix} \overline{c} & 0 & 0 \\ 0 & \overline{c} & 0 \\ 0 & 0 & 0 \end{bmatrix} \left(\vec{v}^B - \vec{W}^B \right) = -T \begin{pmatrix} \overline{c}\,(u - u_w) \\ \overline{c}\,(v - v_w) \\ 0 \end{pmatrix} \tag{4.17}$$

Las hipótesis que han sido tomadas en el desarrollo de este modelo es que la velocidad de vuelo es lo suficientemente baja como para despreciar la resistencia debida al término parásito de los elementos no tractores. Además, se desprecian los términos debidos al batimiento de las palas. La constante $\overline{c}$ para los quadrotor típicos tiene un valor de 0.04 ± 0.0035.

El único término que induce resistencia en el eje z_B del quadrotor es el término parasítico y puesto que se ha despreciado, en este modelo no habrá resistencia en este eje. Sin embargo, la velocidad de ascenso estará limitada por la máxima potencia de los rotores.

En este modelo no se han tenido en cuenta las interacciones que pueden existir entre las diferentes estelas de cada uno de los rotores, ni entre las estelas y la estructura del quadrotor debido a la complejidad de estos fenómenos. Para modelar correctamente estos efectos e introducirlos en el modelado, se pueden utilizar herramientas numéricas como el CFD, tal y como se muestra en [22].

4.5.4. Efecto suelo y su reacción

En esta subsección se tratan las fuerzas que aparecen sobre el quadrotor debidas al contacto con el suelo o su presencia cercana. Principalmente se tratan dos fuerzas, la primera de ellas es conocida como el efecto suelo y se define como el incremento de la tracción proporcionada por un rotor manteniendo constante su velocidad de giro y su potencia consumida por la presencia del suelo cerca del rotor, mientras que la segunda es una fuerza de naturaleza mecánica y es el contacto del quadrotor mediante sus patas en el suelo. En esta última fuerza también se considera el momento que se genera sobre el quadrotor.

El incremento de la tracción de cada uno de los rotores por el efecto suelo, estudiado ampliamente en [20], se debe a la interacción de la estela con el suelo, figura 4.7. Este efecto se ve incrementado cuanto menor es la distancia al suelo y puesto que en un quadrotor existen diferentes rotores que pueden estar bajo la influencia de este efecto, las interacciones que hay entre sus estelas hace que el comportamiento sea diferente que el que tendría un solo rotor. La interacción aerodinámica entre las estelas de los diferentes rotores y el suelo hacen que este fenómeno sea mayor para un quadrotor que para un helicóptero. A diferencia de los vehículos monorotores donde los efectos del suelo pueden llegar a notarse hasta alturas de 2 veces el radio del rotor, en un quadrotor estas contribuciones llegan hasta alturas del orden de 5 veces el radio del rotor.

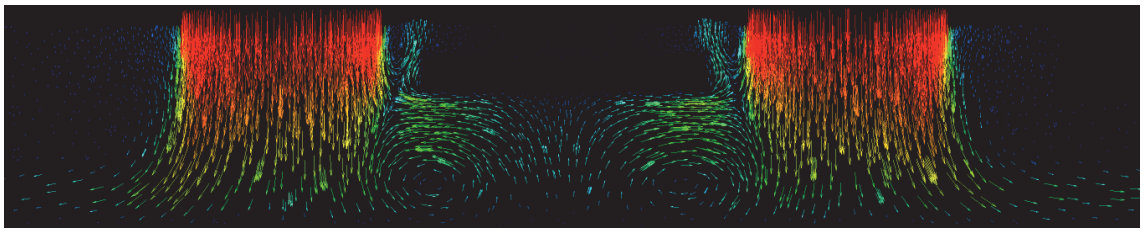


Figura 4.7: Efecto suelo en un quadrotor, [20].

Para modelar el efecto suelo se hace uso de la teoría potencial y del método de las imágenes especulares donde cada rotor es modelado como una fuente tridimensional de flujo. Sin embargo, mediante la teoría potencial no se tiene en cuenta el llamado ‘fountain effect’ que debe ser añadido de forma empírica. Este efecto aparece cuando dos estelas de diferentes rotores en proximidad con el suelo interactúan entre ellas en la parte central del cuerpo del quadrotor formando vórtices que incrementan la tracción de los rotores, también es conocido este efecto como sustentación debida al cuerpo. En la figura 4.7 puede observarse la formación de estas zonas de recirculación bajo de la estructura central. Este efecto no es único de los quadrotor, también había sido descrito anteriormente en helicópteros tándem.

La ecuación (4.18), muestra la relación existente entre la tracción proporcionada por un rotor que se encuentra en efecto suelo, ‘IGE’ (‘In Ground Effect’), y el mismo rotor operando en el mismo régimen de giro fuera de efecto suelo, ‘OGE’ (‘Out Ground Effect’), en función de sus parámetros geométricos y de su distancia al suelo. En esta ecuación ‘R’ es el radio de los rotores, ‘z’ la altura del rotor con respecto al suelo, ‘d’ la distancia entre dos rotores sucesivos, ‘b’ la distancia entre dos rotores opuestos y ‘K_b’ es un parámetro empírico utilizado para modelar la sustentación debida al cuerpo del quadrotor que tiene un valor típico de 2, aunque debe ser determinado para cada geometría.

$$f_{GE} = \frac{T_{IGE}}{T_{OGE}} = \frac{1}{1-(R/4z)^2 - R^2 z/\sqrt{(d^2+4z^2)^3} - (R^2/2)\left(z/\sqrt{(2d^2+4z^2)^3}\right) - 2R^2\left(z/\sqrt{(b^2+4z^2)^3}\right)K_b} \tag{4.18}$$

La aplicación de esta ecuación está restringida a quadrotor cuyos cuatro rotores se encuentran bajo el efecto suelo. En el caso donde únicamente uno de los rotores se encuentre bajo el efecto suelo será posible utilizar la ecuación (4.19), este será el caso de efecto suelo parcial. También puede darse el caso donde 2 o 3 rotores se encuentren bajo el efecto suelo, sin embargo, en la aplicación del presente documento que es el seguimiento de trayectorias, el efecto suelo parcial no tiene influencia y por tanto no se incluyen sus modelos.

$$f_{GE} = \frac{T_{IGE}}{T_{OGE}} = \frac{1}{1-(R/4z)^2} \tag{4.19}$$

El efecto suelo parcial necesita ser analizado con cuidado puesto que induce momentos de cabeceo y alabeo sobre el rotor que puede llegar desestabilizarlo, por esta razón debe considerarse este efecto en las leyes de control de aquellos quadrotor que vayan a realizar misiones con estas características, [20]. El origen de estos momentos que tienden a desestabilizar el quadrotor se debe a que el incremento de la fuerza de tracción sólo aparece en aquellos rotores que están bajo efecto suelo.

El efecto completo no tiene tantas implicaciones en el control de la estabilidad del quadrotor ya que en este caso, el incremento de tracción se produce en todos los rotores por igual si estos se encuentran a la misma distancia del suelo. En el caso que el quadrotor se encuentre con un ligero ángulo de alabeo o cabeceo, el rotor que se encuentre más cerca del suelo verá incrementada su tracción, tendiendo a volver a ángulos de alabeo y cabeceo nulos. El efecto suelo completo se trata por tanto de un efecto estabilizador.

Por otro lado, la reacción del suelo se modela con un contacto de una rigidez, k_a, y amortiguamiento, c_a, determinados para cada una de las superficies de contacto del quadrotor. Se va a asumir que este contacto se produce siempre en todas o parte de las patas que este tiene bajo cada uno de los rotores. La rigidez del contacto se ha seleccionado para que el equilibrio con los cuatro apoyos en contacto se produzca con $1\,mm$ de deformación. La amortiguación del contacto se ha calculado para obtener una relación de amortiguación de 0.95, sería por tanto un sistema subamortiguado muy cercano al críticamente amortiguado.

Al contacto de cada pata con el suelo se le añaden las fuerzas y los momentos derivados de la fricción seca o de Coulomb. Para modelar esta fuerza en dirección contraria al movimiento en el plano de contacto se utiliza el coeficiente de fricción, μ, que se ha asumido igual para el caso estático y dinámico para simplificar el problema.

Para cada uno de los apoyos del quadrotor, partiendo de sus coordenadas en ejes cuerpo, $\vec{\xi}^{B}_{a_i}$, y mediante la matriz de rotación (4.2) que permite pasar de coordenadas en ejes cuerpo a ejes globales, $\vec{\xi}^{G}_{a_i}$, se calculan las posiciones verticales con respecto al suelo, $z^{G}_{a_i}$, tal y como se muestra en la ecuación (4.20).

$$\vec{\xi}_{a_i}^{G} = \begin{pmatrix} x_{a_i}^{G} \\ y_{a_i}^{G} \\ z_{a_i}^{G} \end{pmatrix} = \vec{\xi}^{G} + {}^{G}R^{B}\xi_{a_i}^{B} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + {}^{G}R^{B}\begin{pmatrix} d_{ax_i} \\ d_{ay_i} \\ d_{az_i} \end{pmatrix} \tag{4.20a}$$

$$\begin{aligned} x_{a_i}^{G} = x + d_{ax_i}\cos(\theta)\cos(\psi) + d_{ay_i}(\sin(\theta)\cos(\psi)\sin(\phi) - \sin(\psi)\cos(\phi)) + \\ d_{az_i}(\sin(\theta)\cos(\psi)\cos(\phi) + \sin(\psi)\sin(\phi)) \end{aligned} \tag{4.20b}$$

$$\begin{aligned} y_{a_i}^{G} = y + d_{ax_i}\cos(\theta)\sin(\psi) + d_{ay_i}(\sin(\theta)\sin(\psi)\sin(\phi) + \cos(\psi)\cos(\phi)) + \\ d_{az_i}(\sin(\theta)\sin(\psi)\cos(\phi) - \cos(\psi)\sin(\phi)) \end{aligned} \tag{4.20c}$$

$$z_{a_i}^{G} = z - d_{ax_i}\sin(\theta) + d_{ay_i}\cos(\theta)\sin(\phi) + d_{az_i}\cos(\theta)\cos(\phi) \tag{4.20d}$$

Para incluir las fuerzas y momentos de reacción debidas a la amortiguación es necesario calcular la velocidad en el eje z_G de cada uno de los apoyos. En la ecuación (4.21) se muestran las expresiones de las velocidades verticales para cada uno de los apoyos. Las velocidades de los apoyos en los ejes x_G y y_G son necesarias para el cálculo de las fuerzas y momentos debidas a la fricción de Coulomb, estas no se incluyen en la ecuación (4.21) debido a su longitud, sin embargo, se pueden calcular derivando las ecuaciones (4.20b) y (4.20c) con respecto al tiempo.

$$\begin{aligned} \dot{z}_{a_i}^{G} = \dot{z} - d_{ax_i}\cos(\theta)\dot{\theta} + d_{ay_i}\left(-\sin(\theta)\sin(\phi)\dot{\theta} + \cos(\theta)\cos(\phi)\dot{\phi}\right) \\ + d_{az_i}\left(-\sin(\theta)\cos(\phi)\dot{\theta} - \cos(\theta)\sin(\phi)\dot{\phi}\right) \end{aligned} \tag{4.21}$$

La Expresión (4.20d) también puede utilizarse para calcular la altura de cada uno de los rotores con respecto al suelo para la estimación del coeficiente f_{GE} del efecto suelo. En este caso, las variables d_{ax_i} y d_{ay_i} para cada rotor serán iguales que las de cada pata, sin embargo, d_{az_i} serán diferentes puesto que el contacto y el rotor no se encuentran a la misma altura en ejes cuerpo.

En la ecuación (4.22) se muestra el valor de las fuerzas de reacción debidas al contacto y en función de la altura de cada apoyo y su velocidad de movimiento. Es importante considerar que sólo es válida cuando cada z_{a_i} sea menor que la altura del suelo, en caso contrario serán nulas las reacciones mostradas para el apoyo en cuestión. Es decir, la fuerza de reacción debida a la rigidez sólo puede ser en dirección positiva del eje z_G. En cuanto a la debida a la amortiguación puede ser en cualquier sentido del eje z_G. Por otro lado, las fuerzas debidas a la fricción de Coulomb tienen sentido opuesto al del movimiento en los ejes x_G y y_G.

$$\vec{F}_{GR}^{G} = \begin{pmatrix} F_{GR_x}^{G} \\ F_{GR_y}^{G} \\ F_{GR_z}^{G} \end{pmatrix} = \begin{pmatrix} -\mu\sum_{i=1}^{4}\left|-k_a z_{a_i}^{G} - c_a\dot{z}_{a_i}^{G}\right| sign(\dot{x}_{a_i}) \\ -\mu\sum_{i=1}^{4}\left|-k_a z_{a_i}^{G} - c_a\dot{z}_{a_i}^{G}\right| sign(\dot{y}_{a_i}) \\ \sum_{i=1}^{4} -k_a z_{a_i}^{G} - c_a\dot{z}_{a_i}^{G} \end{pmatrix} \tag{4.22}$$

Se definen los momentos que las fuerzas de reacción ejercen con respecto al cdg del quadrotor. Para ello es necesario expresar la distancia entre el cdg y cada punto de apoyo en ejes globales. En la ecuación (4.23), se expresa el momento resultante de uno de los apoyos, siendo el total la suma de todos los apoyos en contacto.

$$\vec{M}_{GR_i}^{G} = \begin{pmatrix} M_{GR_{x_i}}^{G} \\ M_{GR_{y_i}}^{G} \\ M_{GR_{z_i}}^{G} \end{pmatrix} = \left({}^{G}R^{B}\begin{pmatrix} d_{ax_i} \\ d_{ay_i} \\ d_{az_i} \end{pmatrix}\right) \times \begin{pmatrix} F_{GR_{x_i}}^{G} \\ F_{GR_{y_i}}^{G} \\ F_{GR_{z_i}}^{G} \end{pmatrix} \tag{4.23}$$

Finalmente, las ecuaciones (4.10) fueron desarrolladas en ejes locales y por tanto las fuerzas y momentos deben de ser expresadas en este sistema de referencia, (4.24).

$$\begin{bmatrix} \vec{F}_{GR}^{B} & \vec{M}_{GR}^{B} \end{bmatrix} = {}^{B}R^{G}\begin{bmatrix} \vec{F}_{GR}^{G} & \vec{M}_{GR}^{G} \end{bmatrix} \tag{4.24}$$

4.6 Modelo no lineal del sistema cinemático y dinámico

Una vez que se han definido todos y cada uno de los términos de fuerzas y momentos que afectan al quadrotor, se muestran las ecuaciones completas que gobiernan el movimiento de un quadrotor en el espacio.

$$\dot{x} = u\left[c(\theta)c(\psi)\right] + v\left[s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi)\right] + w\left[c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi)\right] \tag{4.25a}$$

$$\dot{y} = u\left[c(\theta)s(\psi)\right] + v\left[s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi)\right] + w\left[c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi)\right] \tag{4.25b}$$

$$\dot{z} = u\left[-s(\theta)\right] + v\left[s(\phi)c(\theta)\right] + w\left[c(\phi)c(\theta)\right] \tag{4.25c}$$

$$\dot{\phi} = p + q\left[s(\phi)t(\theta)\right] + r\left[c(\phi)t(\theta)\right] \tag{4.25d}$$

$$\dot{\theta} = q\left[c(\phi)\right] + r\left[-s(\phi)\right] \tag{4.25e}$$

$$\dot{\psi} = q\left[\frac{s(\phi)}{c(\theta)}\right] + r\left[\frac{c(\phi)}{c(\theta)}\right] \tag{4.25f}$$

$$\dot{u} = rv - qw + g\sin\theta + \frac{A_x^B + D_x^B + F_{GR_x}^B}{m} \tag{4.25g}$$

$$\dot{v} = pw - ru - g\cos\theta\sin\phi + \frac{A_y^B + D_y^B + F_{GR_y}^B}{m} \tag{4.25h}$$

$$\dot{w} = qu - pv - g\cos\theta\cos\phi + \frac{T + A_z^B + D_z^B + F_{GR_z}^B}{m} \tag{4.25i}$$

$$\dot{p} = \frac{(I_{yy} - I_{zz})}{I_{xx}}qr + \frac{M_{T_x}^B + M_{wx}^B + M_{GR_x}^B - \Gamma_x^B}{I_{xx}} \tag{4.25j}$$

$$\dot{q} = \frac{(I_{zz} - I_{xx})}{I_{yy}}pr + \frac{M_{T_y}^B + M_{wy}^B + M_{GR_y}^B - \Gamma_y^B}{I_{yy}} \tag{4.25k}$$

$$\dot{r} = \frac{(I_{xx} - I_{yy})}{I_{zz}}pq + \frac{M_{T_z}^B + M_{wz}^B + M_{GR_z}^B - \Gamma_z^B}{I_{zz}} \tag{4.25l}$$

En las ecuaciones (4.25), los únicos vectores que quedan por definir son las fuerzas y momentos de perturbación, $\vec{A}^B$ y $\vec{M}_w^B$ respectivamente. De forma externa al sistema, estas variables son utilizadas para evaluar su respuesta ante diferentes perturbaciones y serán utilizadas en futuros capítulos con la finalidad de valorar las diferentes leyes de control y ver si estas son capaces de rechazar las diferentes perturbaciones que aparecen. Mediante estos dos vectores es posible simular la dinámica y el control de un quadrotor de forma más precisa y cercana a la realidad, puesto que en un vuelo real existen numerosos factores que no se han tenido en cuenta y que pueden hacer que una buena ley de control que en simulaciones funciona bien, en un ensayo experimental falle.

Como una perturbación más, se añaden las velocidades del viento en cada instante temporal que se encuentran definiendo la resistencia aerodinámica del quadrotor. En seguimiento de trayectorias, la aparición de fuertes rachas de viento variables pueden hacer que un quadrotor deje de seguir la referencia. Por tanto, mediante esta variable se podrá simular este fenómeno.

En el modelo existen 12 estados, 3 de los cuales son la posición del cdg del quadrotor en el sistema de referencia global, 3 son la orientación del quadrotor en el espacio, 3 son su velocidad lineal en ejes cuerpo y los últimos 3 son la velocidad angular en ejes cuerpo. Se pueden agrupar en forma vectorial tal y como se muestran en la ecuación 4.26.

$$\vec{x} = \left[x, y, z, \phi, \theta, \psi, u, v, w, p, q, r\right]^T \tag{4.26}$$

En cuanto a las acciones de control del sistema, este se encuentra formado por 4. La fuerza de tracción total de los rotores en el eje z_B y los momentos generados por las fuerzas de tracción

individuales de cada rotor en cada uno de los ejes cuerpo, x_B, y_B y z_B. El resto de fuerzas y momentos que actúan sobre el sistema han sido consideradas como perturbaciones de tal forma que sólo se puede actuar sobre los rotores. La razón por la que se han seleccionado las fuerzas y momentos debidas a la tracción como acciones de control en vez de las velocidades angulares de cada rotor es debido a que el modelo del sistema es lineal con respecto a ellas, mientras que con respecto a las velocidades angulares este es cuadrático. De esta forma, haciendo uso de la matriz de mezclado descrita en la ecuación 4.15 de la sección 4.5.2, esta consideración facilita la implementación del control lineal. En la ecuación (4.27) se muestran en forma vectorial las acciones del sistema.

$$\vec{u} = \left[T, M_{T_x}^B, M_{T_y}^B, M_{T_z}^B\right]^T \tag{4.27}$$

Además, la relación existente entre los estados es fuertemente no lineal donde aparecen acoplamientos entre las diferentes variables. De esta forma es posible definir el sistema como en la ecuación (4.28) que relaciona las derivadas de primer orden de cada estado con ellos mismos y las acciones de control. En esta misma ecuación, la función 'f', como se ha indicado anteriormente, es no lineal y para poder realizar un control lineal será linealizada en torno al punto de funcionamiento en la sección 5.1. Aunque no se ha indicado, la ecuación 4.28 también depende de las perturbaciones.

$$\dot{\vec{x}}(t) = f(t, \vec{x}(t), \vec{u}(t)) \tag{4.28}$$

Si se analiza el tipo de sistema en cuanto al número de acciones de control o actuadores que tiene, es posible hacer una clasificación. En este caso se trata de un sistema subactuado, es decir, tiene un menor número de acciones de control que gdl a ser controlados. El modelo de quadrotor presentado en este documento tiene un total de 6 gdl, 3 de ellos traslacionales y 3 rotacionales, medidos en cada uno de los ejes x, y, z. Además, en cada uno de los ejes y para cada gdl traslacional o rotacional a controlar, es posible definirlo como la velocidad o la posición independientemente del resto, sin embargo, no se puede controlar el grado de libertad de posición y velocidad en un mismo eje a la vez.

Las principales implicaciones derivadas del tipo de sistema subactuado vienen del lado de la actuación que se puede esperar del sistema. Las incertidumbres de modelado, el fallo de uno o varios motores, el fuerte acoplamiento entre los diferentes estados, el ruido de medida y las perturbaciones pueden afectar en gran medida la eficacia con la que actúa el sistema de control. En el caso de los quadrotor se complican en gran medida el seguimiento de trayectorias y la estabilización, [23].

Puesto que el sistema cuenta con 4 acciones de control y con 6 gdl a controlar, el grado de subactuación es 2. Esto es debido a que un quadrotor que se encuentra volando en 'hover' es incapaz de conseguir un movimiento en avance o lateral a no ser que cambie su ángulo de cabeceo o alabeo.

Para conseguir movimiento en avance es necesario orientar el vector tracción en esta dirección, y puesto que la tracción de los rotores son en todo momento paralelas al eje z_B hace que para conseguir este tipo de movimientos sea necesario reorientar el cuerpo del quadrotor, cambiando su cabeceo o alabeo según sea necesario. Por esta razón, se hace imposible definir una posición de referencia en el espacio y una orientación del quadrotor con respecto a él. Para conseguir una posición estacionaria en el tiempo solamente es necesaria una componente del vector tracción perpendicular al suelo y que contrarreste la gravedad y por tanto el cabeceo y el alabeo serán nulos. Por otro lado, en vuelo en avance es necesaria una componente del vector tracción que contrarreste la fuerza del peso y la resistencia aerodinámica y por esta razón el cabeceo o el alabeo no serán nulos.

Sin embargo, un sistema subactuado presenta numerosas ventajas que pueden contrarrestar las partes negativas. Por norma general, los actuadores son pesados, necesitan un suministro de ener-

gía elevado y un mantenimiento constante, por tanto, el hecho de que el número de actuadores es reducido permite incrementar la eficiencia con el que se usa la energía, reduciendo los pesos de la aeronave y su tamaño.

Un sistema subactuado es, en algunas situaciones, una buena elección, como puede ser el sector aeronáutico y concretamente el caso de los quadrotor, donde el peso y el uso de la energía eléctrica son una de los aspectos más importantes a minimizar. Además, como se ha comentado anteriormente, incluso un sistema actuado completamente puede convertirse en subactuado en el caso que un actuador experimente un fallo.

En cuanto algunos de los problemas que presentan estos sistemas, se pueden solucionar mediante un conocimiento y modelado más preciso, mejores sistemas de control o mediante la mejora de los sensores utilizados. Para compensar la problemática de las perturbaciones se pueden implementar estimadores de fuerzas y momentos externos como el mostrado en [24] que está basado en el filtro de Kalman y diseñado para vehículos 'VTOL'.

4.7 Implementación del modelo del sistema

La implementación del modelo dinámico del quadrotor junto con todas sus fuerzas y momentos internos y externos se ha hecho mediante diagrama de bloques en el entorno de Simulink de Matlab que permite su fácil integración numérica.

Siguiendo la estructura del bloque de Simulink 'State-Space Model' que permite simular la dinámica de un sistema lineal, se ha creado un bloque similar que contiene el modelo no lineal del sistema, 4.8. La principal finalidad que tiene este procedimiento es que, en cualquier sistema de control que se utilice el bloque lineal 'State-Space Model' para simular la dinámica no lineal de un quadrotor entorno a un punto de equilibrio, será posible sustituir este bloque por el bloque no lineal de forma equivalente en cuanto a la forma de operar con ellos. Solamente habrá que tener en cuenta que las perturbaciones, en el caso del bloque mostrado en la figura 4.8, son añadidas como componentes concatenadas en el vector de acciones de control.

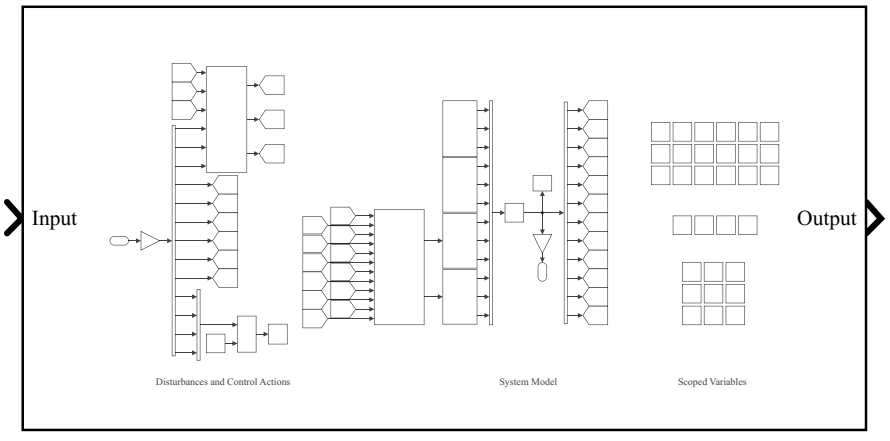


Figura 4.8: Bloque del modelo no lineal de un quadrotor en Simulink

Este bloque de Simulink se encarga de calcular la evolución con el tiempo de los estados del sistema al que se le han aplicado unas acciones de control o unas perturbaciones. Este proceso se realiza mediante la integración de las ecuaciones del modelo no lineal expuestas en 4.6. El sistema comienza con unas condiciones iniciales y debido a la modificación de su estado de equilibrio

mediante las entradas, se producirá una evolución temporal de los estados que es lo que se obtiene para cada instante temporal en la salida. Por tanto, las salidas varían con el tiempo en función de que entradas se han suministrado y estas también pueden variar con el tiempo.

La entrada de este sistema, para cada instante de tiempo, es un vector que contiene las perturbaciones y las acciones de control de este sistema. En la ecuación (4.29) se muestran cada uno de las componentes de este vector.

$$Input = \begin{bmatrix} u_w & v_w & w_w & A_x & A_y & A_z & M_{wx} & M_{wy} & M_{wz} & T & M_x & M_y & M_z \end{bmatrix} \tag{4.29}$$

En total, el número de entradas son 13, donde las 9 primeras son perturbaciones y las 4 últimas son las acciones de control. Las 3 primeras perturbaciones se definen como la velocidad del aire que rodea al quadrotor, medidas en ejes gloables, u_w en el eje x_G, v_w en el eje y_G y w_w en el eje z_G. Las 3 siguientes son fuerzas de perturbación y las 3 siguientes son momentos de perturbación, todos ellos sobre los ejes locales del quadrotor. Finalmente, las acciones de control son las ya definidas en (4.14).

En cuanto a la salida del bloque mostrado en la figura 4.8, para cada instante de tiempo, se trata de un vector que contiene los 12 estados del quadrotor, ecuación (4.30), y que ya ha sido introducido en la sección 4.6.

$$Output = \begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r \end{bmatrix} \tag{4.30}$$

En la figura 4.9, se muestra la estructura interna del bloque 4.8. De izquierda a derecha de la imagen puede seguirse el orden con el que se realiza la actualización de los estados para cada instante temporal en función de las entradas. Principalmente, el primer nivel del bloque no lineal de la dinámica del quadrotor se encuentra dividido en 3 estructuras, que están diferenciadas ya que no hay unión mediante lazos entre ellas, 'Disturbances and Control Actions', 'System Model' y 'Scoped Variables'.

La tercera estructura, 'Scoped Variables', se corresponde con un conjunto de bloques utilizados para la definición de las variables con alcance restringido al conjunto del bloque no lineal, es decir, a estas variables solo se pueden acceder desde dentro del modelo. De esta forma se asegura que cada uno de los subsistemas del interior están operando con la misma definición de cada una de las variables. A continuación se analizarán las dos estructuras restantes en función del orden.

La primera operación que se realiza y que se muestra en la primera estructura de la izquierda, 'Disturbances and Control Actions', es la asignación de cada componente del vector columna de la entrada en una variable con un nombre determinado. Sin embargo, antes de esto se realiza una operación matricial, para explicar esta operación se va a hacer un pequeño análisis de la aplicabilidad de este bloque.

Una de los principales requisitos en la implementación del modelo es conseguir la suficiente versatilidad como para que se pueda utilizar en diferentes aplicaciones del control de un quadrotor. Por ejemplo, en el caso donde se desea controlar únicamente la altura del quadrotor, la única acción de control necesaria, si se desprecian las perturbaciones en el resto de ejes que no sean el z_B, es la tracción total T y por tanto se desea que la única entrada del sistema sea esta variable, aparte de las perturbaciones. Sin embargo, el modelo dinámico hace uso del resto de acciones de control y necesitan estar definidas, por tanto, mediante esta operación matricial se consigue pasar de un vector de entrada donde no se incluyen los momentos M_x, M_y, M_z a un vector que si que los incluye y además son nulos. Otro ejemplo sería el control de orientación, en este caso se desea que las únicas entradas de control sean los momentos M_x, M_y, M_z y mediante la operación matricial descrita es posible pasar de un vector que no contiene la acción de control T a uno que si que la contiene y además es nula.

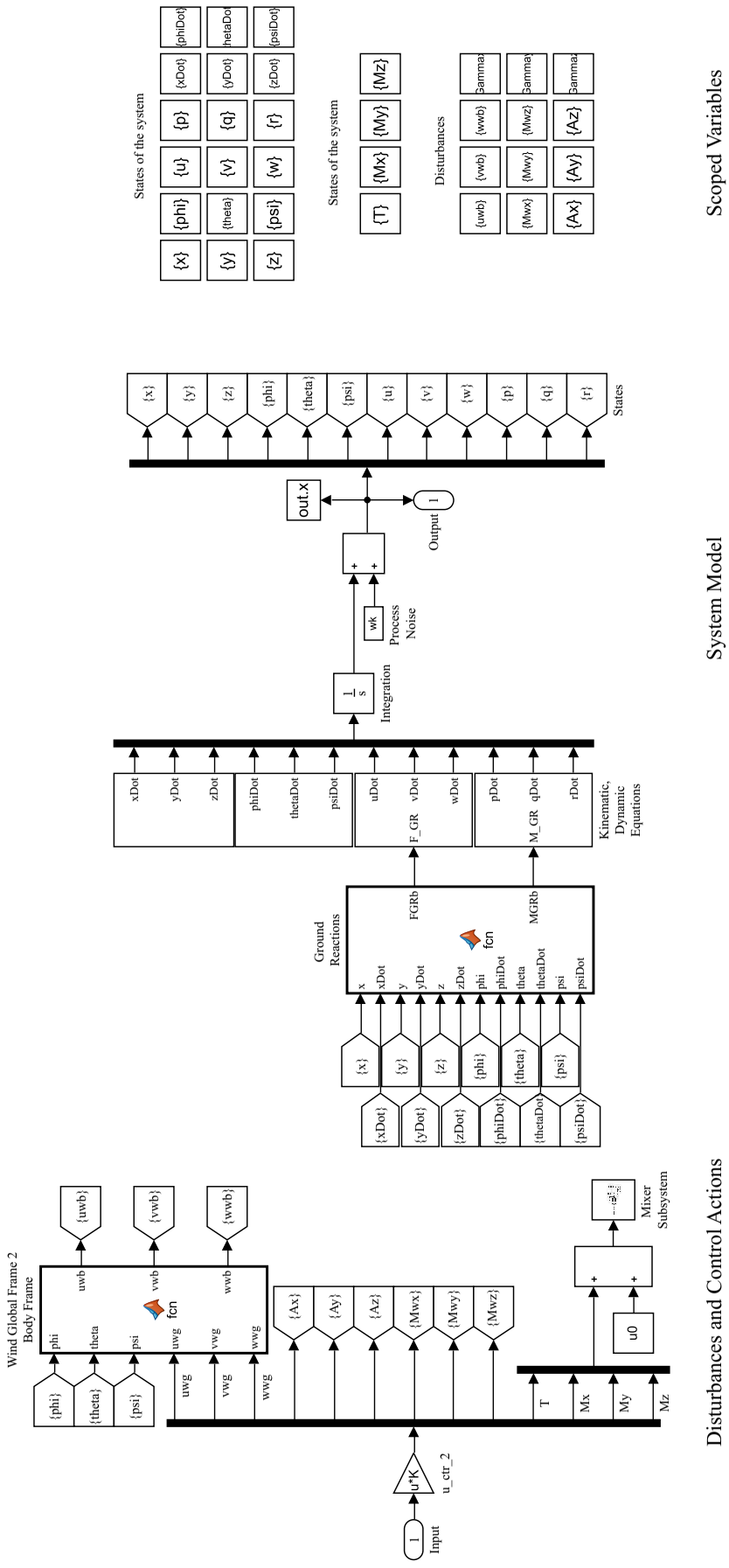


Figura 4.9: Primer nivel del bloque del modelo no lineal de un quadrotor en Simulink

Para el caso descrito del control de altura, las entradas del sistema serían las mostradas en la ecuación (4.31a). Una vez que se ha completado el vector entrada con las acciones de control que faltan se obtendría la ecuación (4.31b). Para cada uno de los casos de control que se deseen implementar con el modelo dinámico mostrado en 4.9 será por tanto necesario definir la matriz de conversión, u_{ctr_2}, mostrada en la ecuación (4.31c).

$$Input_1 = \begin{bmatrix} u_w & v_w & w_w & A_x & A_y & A_z & M_{wx} & M_{wy} & M_{wz} & T \end{bmatrix} \tag{4.31a}$$

$$Input_2 = \begin{bmatrix} u_w & v_w & w_w & A_x & A_y & A_z & M_{wx} & M_{wy} & M_{wz} & T & 0 & 0 & 0 \end{bmatrix} \tag{4.31b}$$

$$Input_2 = Input_1\left(u_{ctr_2}\right) \tag{4.31c}$$

Una vez que se ha completado el vector columna con las posibles acciones de control que falten en el sistema, se produce la asignación de cada variable a una variable de alcance restringido que pueda ser utilizada por el resto de bloques y estructuras del sistema. Esta asignación para las perturbaciones A_x, A_y, A_z, M_{wx}, M_{wy}, M_{wz} es inmediata tal y como se puede observar en la figura 4.9.

Sin embargo, es necesaria una preparación de las componentes del viento y de las acciones de control antes de ser asignadas a una variable de alcance restringido. Comenzando por la velocidad del aire que rodea al quadrotor, estas variables aparecen en el modelo de resistencia mostrado en la ecuación (4.17). Las velocidades en este modelo se encuentran medidas en ejes locales o cuerpo y por tanto se necesita convertirlas mediante el bloque de Simulink 'Matlab Function' llamado 'Wind Global Frame 2 Body Frame'. Las entradas de este bloque son las velocidades del viento en ejes globales y los 3 ángulos de Euler, mientras que las salidas son las velocidades del viento en ejes cuerpo que finalmente son asignados a las correspondientes variables de alcance restringido. En la subsección 4.7.1 se muestra el contenido de este bloque 'Matlab Function'.

En cuanto a las acciones de control, se forma el vector $\begin{bmatrix} T & M_x & M_y & M_z \end{bmatrix}$, para su procesado y obtención de las velocidades de giro según se expuso en 4.5.2, mediante la matriz de mezclado.

Siguiendo la anlogía del bloque 'State-Space Model' donde las entradas del sistema se miden sobre el estado de equilibrio, en el modelo no lineal se ha hecho esta misma definición. Es decir, las entradas correspondientes a las acciones de control son en realidad incrementos con respecto a las acciones de control en el equilibrio. Como se verá en la sección 5.1, para la obtención del modelo lineal del sistema, se ha asumido que el punto de equilibrio es el estacionario donde el quadrotor se encuentra volando en un punto fijo, también llamado 'hover'. Las acciones de control en el equilibrio se muestran en la ecuación (4.32). Por esta razón, una vez que se tiene el vector que forman los incrementos de las acciones de control, será necesario sumar las acciones de control en el punto de equilibrio.

$$u_0 = \begin{bmatrix} T_0 & M_{x0} & M_{y0} & M_{z0} \end{bmatrix} = \begin{bmatrix} mg & 0 & 0 & 0 \end{bmatrix} \tag{4.32}$$

En cuanto al valor de las perturbaciones en el punto de equilibrio del sistema se ha asumido que este valor es nulo, además puesto que se encuentran medidas en ejes locales no necesitan de ningún procesado para poder ser utilizadas.

El resto del procesado sobre las acciones de control absolutas se realiza en el bloque 'Mixer subsystem' que puede ser observado en detalle en la imagen 4.10. Dentro de él se mezclan estos datos mediante la matriz 'Mixer' para obtener las velocidades de giro que deben ser mandadas a cada rotor, (4.15). Puesto que las velocidades de giro admisibles están limitadas se saturan entre 0 y la máxima velocidad de giro. Posteriormente se añade a cada uno de los rotores un filtro paso bajo con una constante de tiempo determinada que simboliza el transitorio de aceleración de un rotor hasta alcanzar un estacionario, (4.16).

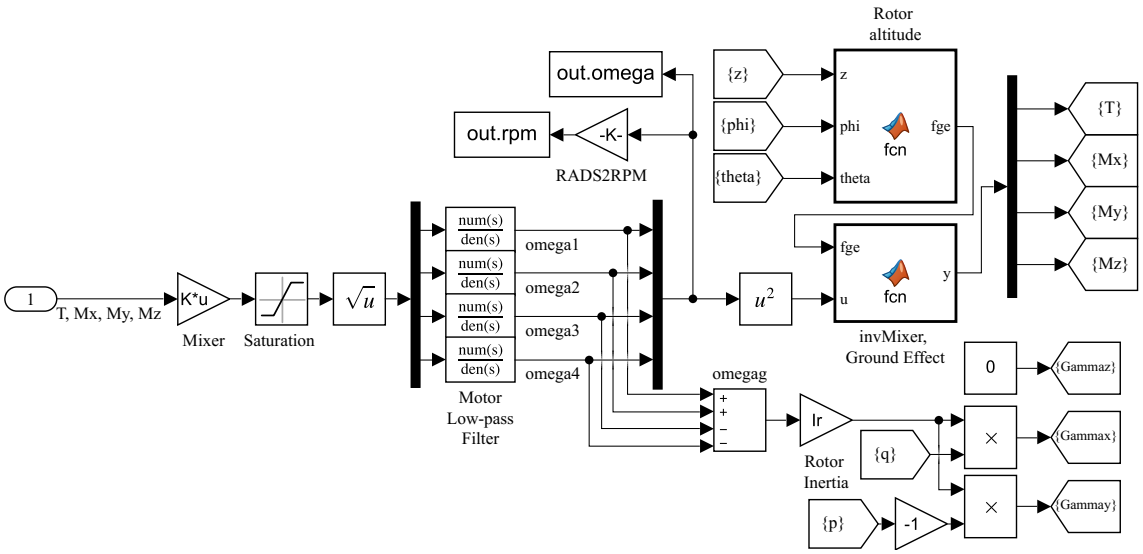
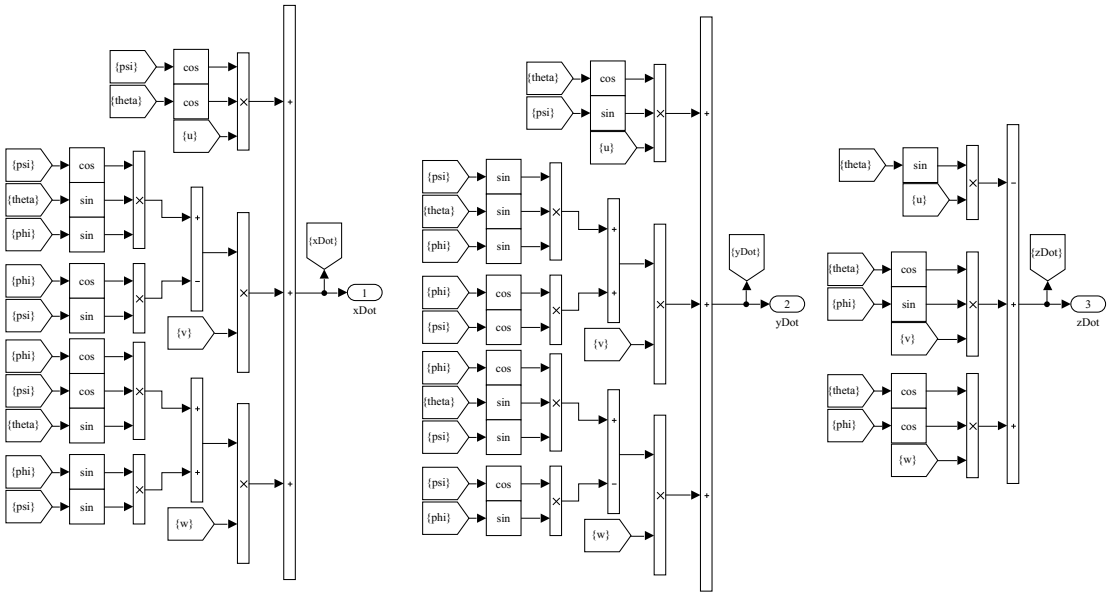


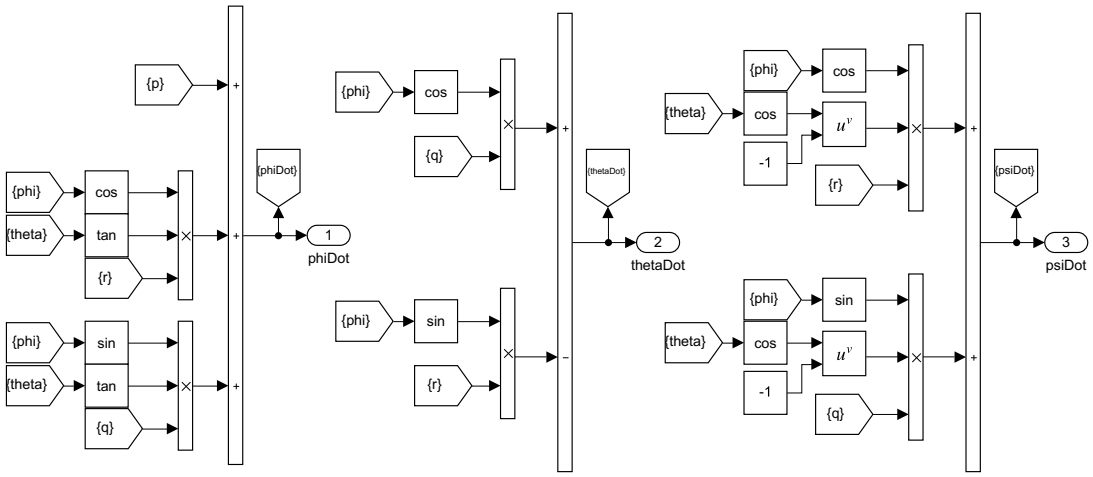
Figura 4.10: Subsistema de mezclado de fuerzas y momentos a velocidades de giro

El resultado de este proceso son las cuatro velocidades de giro actuales de los cuatro rotores y que serán diferentes de las mandadas por las acciones de control debido a que se han tenido en cuenta los efectos de la saturación y la dinámica de los motores. Estas velocidades de giro reales son utilizadas para calcular la fuerza de tracción y los momentos en los tres ejes que actúan sobre el quadrotor debido a los rotores. En este proceso inverso se tiene en cuenta la altura a la que se encuentra cada rotor para añadirles el incremento de tracción debido al efecto suelo, (4.18). En el bloque de mezclado, aprovechando que se calculan las velocidades de giro de cada rotor, también se calculan los momentos giroscópicos que actúan sobre el sistema para posteriormente introducirlos en las correspondientes ecuaciones. El contenido de los dos bloques 'Matlab Function se incluyen en la subsección 4.7.1.



$$\dot{x} = uCos(\theta)Cos(\psi) + v(Cos(\psi)Sin(\theta)Sin(\phi) - Cos(\phi)Sin(\psi)) + w(Cos(\phi)Cos(\psi)Sin(\theta) + Sin(\phi)Sin(\psi))$$
$$\dot{y} = uCos(\theta)Sin(\psi) + v(Cos(\phi)Cos(\psi) + Sin(\theta)Sin(\phi)Sin(\psi)) + w(-Cos(\psi)Sin(\phi) + Cos(\phi)Sin(\theta)Sin(\psi))$$
$$\dot{z} = -uSin(\theta) + vCos(\theta)Sin(\phi) + wCos(\theta)Cos(\phi)$$

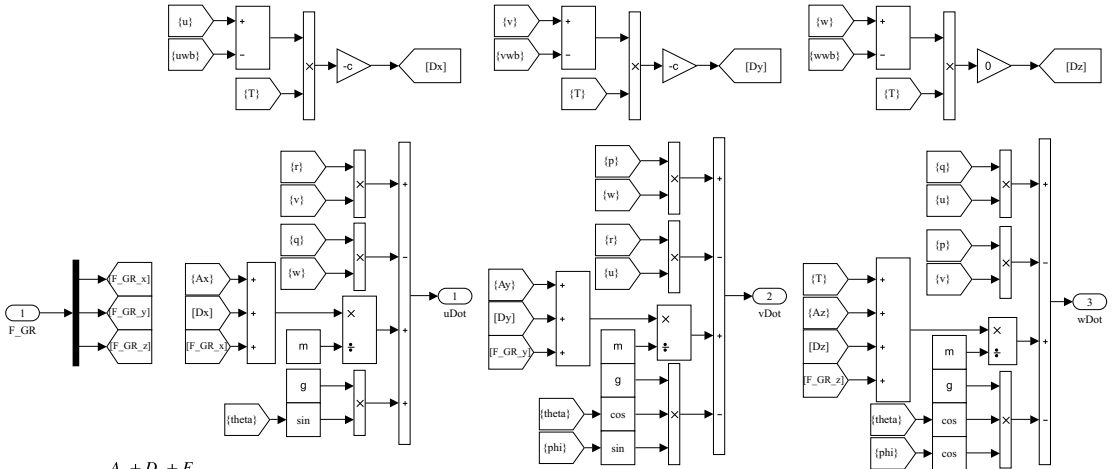
Figura 4.11: Ecuaciones cinemáticas de traslación



$$\dot{\phi} = p + rCos(\phi)Tan(\theta) + qSin(\phi)Tan(\theta)$$
$$\dot{\psi} = rCos(\phi)Sec(\theta) + qSec(\theta)Sin(\phi)$$
$$\dot{\theta} = qCos(\phi) - rSin(\phi)$$

Figura 4.12: Ecuaciones cinemáticas de rotación

Tras finalizar de procesar y asignar todas las variables del bloque 'Disturbances and Control Actions' en las correspondientes variables de alcance restringido se presentan las ecuaciones cinemáticas y dinámicas del modelo del sistema. En el bloque no lineal de Simulink mostrado en la figura 4.9 se han separado siguiendo el orden con el que se han introducido en el presente documento. El primero de los cuatro bloques comenzando por arriba se corresponde con las ecuaciones cinemáticas de traslación, cuyo interior se muestra en la figura 4.11. Como se puede observar en ella, la salida de este bloque se corresponde con las derivadas temporales de los tres primeros estados del sistema, x, y, z.

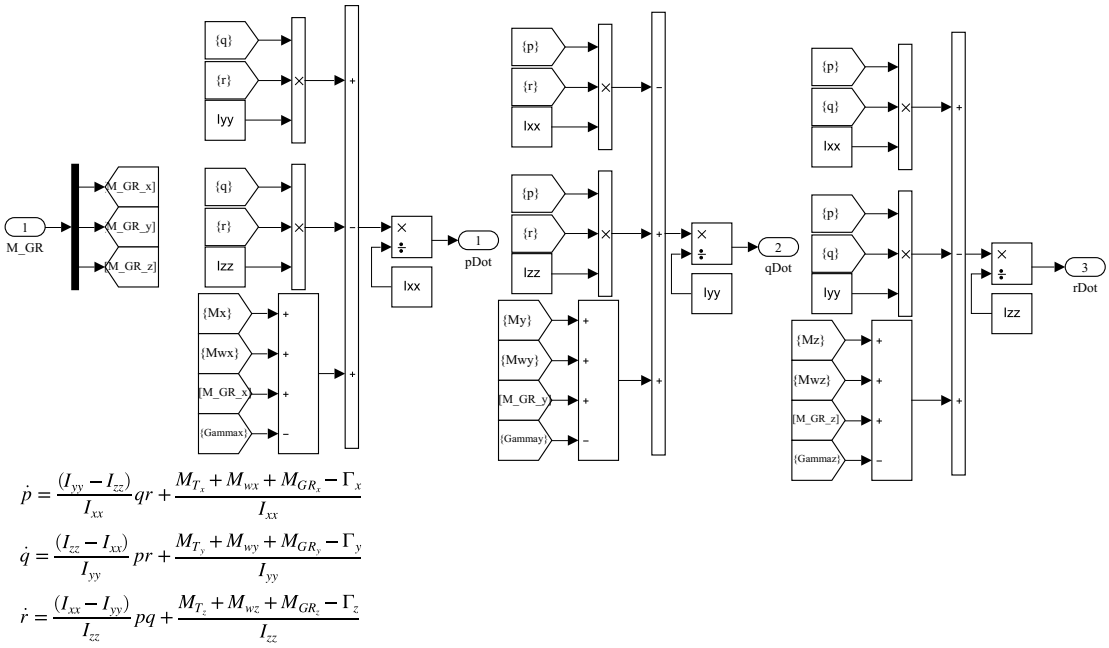


$$\dot{u} = rv - qw + \frac{A_x + D_x + F_{GR_x}}{m} + gSin(\theta)$$
$$\dot{v} = pw - ru + \frac{A_y + D_y + F_{GR_y}}{m} - gCos(\theta)Sin(\phi)$$
$$\dot{w} = qu - pv + \frac{T + A_z + D_z + F_{GR_z}}{m} - gCos(\theta)Cos(\phi)$$

Figura 4.13: Ecuaciones dinámicas de traslación

El segundo de los bloques que contienen el modelo cinemático y dinámico del sistema se corresponde con las ecuaciones cinemáticas de rotación. En la figura 4.12 se muestra el interior del bloque y se puede observar que sus tres salidas se corresponden con las derivadas temporales de

los tres grados de libertad que definen la orientación del quadrotor en el espacio, ϕ, θ, ψ, los tres ángulos de Euler.



$$\dot{p} = \frac{(I_{yy} - I_{zz})}{I_{xx}} qr + \frac{M_{T_x} + M_{wx} + M_{GR_x} - \Gamma_x}{I_{xx}}$$

$$\dot{q} = \frac{(I_{zz} - I_{xx})}{I_{yy}} pr + \frac{M_{T_y} + M_{wy} + M_{GR_y} - \Gamma_y}{I_{yy}}$$

$$\dot{r} = \frac{(I_{xx} - I_{yy})}{I_{zz}} pq + \frac{M_{T_z} + M_{wz} + M_{GR_z} - \Gamma_z}{I_{zz}}$$

Figura 4.14: Ecuaciones dinámicas de rotación

El tercero y cuarto de los bloques que contienen las ecuaciones dinámicas y cinemáticas, presentan como entradas las fuerzas y momentos de reacción del suelo que son calculadas mediante un bloque 'Matlab Function' que es mostrado en detalle en la subsección 4.7.1. Por un lado, el tercer bloque contiene las ecuaciones dinámicas de traslación y cuyo interior se muestra en la figura 4.13. Como se puede observar, la salida de este bloque se corresponde con las derivadas temporales de las velocidades lineales del quadrotor medidas en ejes cuerpo, u, v, w. Es importante destacar que en este bloque es donde se consideran la acción de control debida a la tracción, la resistencia aerodinámica que es calculada dentro de él y las perturbaciones.

Finalmente, el cuarto bloque contiene las ecuaciones dinámicas de rotación tal y como se muestra en la figura 4.14. La salida de este bloque son las derivadas temporales de las velocidades angulares en ejes cuerpo, p, q, r. Además, en este bloque es donde se incluyen los momentos en los tres ejes debidos a la tracción, los términos giroscópicos y los momentos de perturbación.

Una vez que se tienen las derivadas temporales de cada uno de los estados se produce el proceso de integración partiendo de unas condiciones iniciales que pueden ser modificadas para que el usuario sea capaz de simular el caso deseado. La modificación de las condiciones iniciales se hace dentro del bloque 'Integration' y debe tener tantas columnas como el número de estados del sistema.

Tal y como se puede observar en la figura 4.9, a continuación del proceso de integración se produce la adición del ruido de proceso que será utilizado en la sección 5.3 en el observador de estados. Tras esta adición se produce una trifurcación en el camino, si se continúa recto se produce la asignación de cada uno de los estados en una variable de alcance restringido de tal forma que puede ser utilizada por cualquier bloque dentro del modelo no lineal de la dinámica de un quadrotor. Si se continúa hacia arriba en la trifurcación se puede encontrar un bloque donde se guarda el vector de estados a lo largo del tiempo en el 'Worksapce' para poder ser utilizado por el usuario posteriormente. La estructura donde se guarda el vector de estados, $\vec{x}$, para cada instante temporal en forma matricial se corresponde con una variable del tipo 'Struct' y será donde se guarden todas las variables de cada simulación como pueden ser el vector de tiempos donde se ha guardado cada vector de estados del sistema o el vector de acciones de control.

Finalmente, continuando hacia abajo en la trifurcación, se realiza una operación matricial que permite elegir cuales serán los estados que serán la salida del bloque no lineal utilizados para el control. Esta operación es similar a la descrita anteriormente para el control de altura con la matriz u_{ctr_2}. Volviendo al caso del control de altura, la mayoría de los estados no serían necesarios para realizar esta tarea y por tanto deberían de ser eliminados del vector de estados. Los únicos estados necesarios para realizar el control serían la posición vertical, z y la velocidad lineal vertical w, tal y como se muestra en las ecuaciones (4.33) mediante el uso de esta operación matricial se puede pasar de un vector de estados completo a uno que sólo contiene los estados necesarios. Como sucedía para el caso de la matriz u_{ctr_2}, la matriz x_{ctr_2} debe de ser elegida por el usuario y definida en el 'Worksapce' de Matlab.

$$Output_1 = \left[x, y, z, \phi, \theta, \psi, u, v, w, p, q, r\right] \tag{4.33a}$$
$$Output_2 = \left[z, w\right] \tag{4.33b}$$
$$Output_2 = x_{ctr_2} Output_1 \tag{4.33c}$$

Aunque puede parecer complejo el interior y el funcionamiento del bloque no lineal utilizado para describir la cinemática y dinámica del quadrotor, en realidad, puede ser tratado como una caja negra a la que se le introducen unas acciones de control y unas perturbaciones y partiendo de unas condiciones iniciales se obtiene la evolución de los estados del sistema.

4.7.1. Bloques de Simulink 'Matlab Function'

Wind Global Frame 2 Body Frame

En el código 4.1 se muestra la conversión de las componentes de velocidad del aire que rodea al quadrotor en el sistema de referencia global al sistema de referencia local. En un primer lugar, considerando la orientación del quadrotor en el espacio, se calcula la matriz de rotación que permite hacer este cambio y posteriormente se realiza la conversión.

```
function [uwb,vwb,wwb] = fcn(phi,theta,psi,uwg,vwg,wwg)

% Inputs (Attitude, Wind in Global Frame) -> Rotation Matrix -> Output ...
% (Wind in Body Frame)

% Rotation Matrix GlobaL Frame 2 Body
RBI = [cos(theta).*cos(psi),cos(theta).*sin(psi),(-1).*sin(theta);...
       cos(psi).*sin(theta).*sin(phi)+(-1).*cos(phi).*sin(psi),...
       cos(phi).*cos(psi)+sin(theta).*sin(phi).*sin(psi),...
       cos(theta).*sin(phi);...
       cos(phi).*cos(psi).*sin(theta)+sin(phi).*sin(psi),...
       (-1).*cos(psi).*sin(phi)+cos(phi).*sin(theta).*sin(psi),...
       cos(theta).*cos(phi)];

% Wind in Body Frame
bodywind = RBI*[uwg; vwg; wwg];
uwb = bodywind(1);
vwb = bodywind(2);
wwb = bodywind(3);
```

Código 4.1: Wind Global Frame 2 Body Frame

Rotor altitude

El código 4.2 muestra el cálculo de la matriz por la que debe de ser multiplicada de forma elemento a elemento la matriz inversa de mezclado mostrada en la ecuación 4.15 para considerar el incremento debido al efecto suelo. Para ello es necesario el cálculo de la altura de cada rotor con respecto al suelo y su introducción en la relación que permite calcular el efecto suelo completo para un quadrotor mostrada en la ecuación 4.18.

Si se comparan las entradas mostradas en el código 4.2 y en el bloque 'Matlab Function' de la figura 4.10 se puede observar que se incluyen parámetros geométricos que en el diagrama de la figura no se mostraban. Estas variables dentro del bloque 'Matlab Function' se han definido como parámetros y son leídos del 'Workspace' de Matlab. Esto se repite para los códigos 4.3 y 4.4.

```
function fge = fcn(z,phi,theta,Rad,l,l_arm_x,l_arm_y,l_arm_z,KB)

% Input (Altitude,Attitude,Dimensions) -> Output (Traction Ratio IGE/OGE)

% Rotor altitude
z1 = z-(l_arm_x)*sin(theta)+(-l_arm_y)*cos(theta)*sin(phi)...
    +l_arm_z*cos(theta)*cos(phi);
z2 = z-(-l_arm_x)*sin(theta)+(l_arm_y)*cos(theta)*sin(phi)...
    +l_arm_z*cos(theta)*cos(phi);
z3 = z-(l_arm_x)*sin(theta)+(l_arm_y)*cos(theta)*sin(phi)...
    +l_arm_z*cos(theta)*cos(phi);
z4 = z-(-l_arm_x)*sin(theta)+(-l_arm_y)*cos(theta)*sin(phi)...
    +l_arm_z*cos(theta)*cos(phi);

% Traction ratio IGE/OGE
fge1 = 1/(1-(Rad/(4*z1)^2)-Rad^2*(z1/(((2*l_arm_x)^2+4*z1^2)^(3/2)))...
    -Rad^2/2*(z1/((2*(2*l_arm_x)^2+4*z1^2)^(3/2)))...
    -2*Rad^2*(z1/(((2*l)^2+4*z1^2)^(3/2)))*KB);
fge2 = 1/(1-(Rad/(4*z2)^2)-Rad^2*(z2/(((2*l_arm_x)^2+4*z2^2)^(3/2)))...
    -Rad^2/2*(z2/((2*(2*l_arm_x)^2+4*z2^2)^(3/2)))...
    -2*Rad^2*(z2/(((2*l)^2+4*z2^2)^(3/2)))*KB);
fge3 = 1/(1-(Rad/(4*z3)^2)-Rad^2*(z3/(((2*l_arm_x)^2+4*z3^2)^(3/2)))...
    -Rad^2/2*(z3/((2*(2*l_arm_x)^2+4*z3^2)^(3/2)))...
    -2*Rad^2*(z3/(((2*l)^2+4*z3^2)^(3/2)))*KB);
fge4 = 1/(1-(Rad/(4*z4)^2)-Rad^2*(z4/(((2*l_arm_x)^2+4*z4^2)^(3/2)))...
    -Rad^2/2*(z4/((2*(2*l_arm_x)^2+4*z4^2)^(3/2)))...
    -2*Rad^2*(z4/(((2*l)^2+4*z4^2)^(3/2)))*KB);

% Matrix multiplied elemt-wise with invMixer (Increase of Traction IGE/OGE)
fge = [fge1 fge2 fge3 fge4 ;
       fge1 fge2 fge3 fge4 ;
       fge1 fge2 fge3 fge4 ;
          1    1    1    1  ];
```

Código 4.2: Rotor altitude

invMixer, Ground Effect

En el código 4.3 se muestra el cálculo de las fuerzas de tracción y sus momentos en los tres ejes a partir de las velocidades de giro de cada rotor, de la matriz inversa de mezclado y de la matriz que contiene el efecto suelo de cada rotor y para cada uno de los términos de fuerzas y momentos.

```
function y = fcn(fge, u, invMixer)

% Inputs (Rotor speed, invMixer) -> invMixer.*fge -> ...
```

```
4  % Outputs (Traction and Moments)
5
6  % Inverse Mixer Matrix multiplied by fge to account for IGE
7  y = invMixer.*fge*u;
```

Código 4.3: Matriz de mezclado inversa y Efecto Suelo

Ground Reactions

En el código 4.4 se muestra el cálculo de las fuerzas y momentos de reacción en ejes cuerpo que serán introducidos en las ecuaciones (4.25) para su posterior integración. Mediante el modelo de una rigidez, amortiguamiento y fricción seca equivalente se consigue simular la presencia del suelo y su contacto con el quadrotor.

El procedimiento seguido consiste en un primer lugar en la determinación de la posición y velocidad en ejes globales de cada una las patas que es donde se ha asumido que se produce el contacto. Es importante considerar que si fuese necesario, se podrían añadir más puntos de contacto a los vectores *lxb*, *lyb* y *lzb* donde se introducen sus coordenadas en ejes locales, sin embargo, en este modelo no es necesario puesto que se asume que el quadrotor va a despegar desde el suelo y no va a maniobrar cerca de él. Posteriormente, se evalúan que patas se encuentran en contacto, se calcula para cada una de ellas las fuerzas y momentos de reacción en ejes globales y finalmente se convierten a ejes locales y se suman todas las contribuciones.

Es importante considerar que la integración numérica de las ecuaciones incluyendo el término de fricción de Coulomb es bastante costosa computacionalmente debido a la discontinuidad en el signo de la fuerza para valores positivos y negativos de la velocidad. El principal problema por el que a Simulink le resulta tan costosa esta operación es la definición del intervalo de simulación ya que se usa un criterio de satisfacción de tolerancias relativas y absolutas para decidir si es necesario aumentar o disminuir este intervalo. La tolerancia relativa se define sobre el valor del estado que se está integrando de tal forma que mayor es el estado, mayor es el error que puede haber y mayor es el intervalo de simulación, mientras que la tolerancia absoluta se define como el máximo error admisible antes de disminuir el intervalo temporal. De esta forma para definir el intervalo de simulación se verifica que el error de la simulación es menor que el mayor de la tolerancia relativa o absoluta.

Por tanto, a velocidades lo suficientemente bajas, el signo de la fuerza de fricción está continuamente cambiando, pero no su magnitud. Entonces, debido a la reducción considerable de la tolerancia admisible, el intervalo de simulación se ve drásticamente reducido, incrementando el tiempo de simulación total. Por esta razón, en los casos que sea necesaria la introducción del término de amortiguación de Coulomb es recomendable el incremento de la tolerancia admisible o trabajar con intervalos de simulación fijos. En este estudio han dado un mejor resultado el trabajar con la segunda opción aunque será necesaria un estudio de independencia del paso temporal para asegurar que es lo suficientemente bajo para garantizar que la solución no depende de él.

```
1  function [FGRb,MGRb]  = fcn(x, xDot, y, yDot, z, zDot, phi, phiDot,...
2      theta, thetaDot, psi, psiDot, l_arm_x, l_arm_y, l_feet, ka, ca, mua)
3
4  % Inputs (Global Frame [Position, Attitude, Angular and Lineal Velocity]...
5  % ,Dimensions, Contact Parameters) -> Output (Ground Reactions)
6
7  % Rotation Matrixes
8  % Global to Body
9  RBI = [cos(theta).*cos(psi),cos(theta).*sin(psi),(-1).*sin(theta);...
10        cos(psi).*sin(theta).*sin(phi)+(-1).*cos(phi).*sin(psi),...
11        cos(phi).*cos(psi)+sin(theta).*sin(phi).*sin(psi),...
```

```
        cos(theta).*sin(phi);...
        cos(phi).*cos(psi).*sin(theta)+sin(phi).*sin(psi),...
        (-1).*cos(psi).*sin(phi)+cos(phi).*sin(theta).*sin(psi),...
        cos(theta).*cos(phi)];
% Body to Global
RIB = [cos(theta).*cos(psi),...
        cos(psi).*sin(theta).*sin(phi)+(-1).*cos(phi).*sin(psi),...
        cos(phi).*cos(psi).*sin(theta)+sin(phi).*sin(psi);...
        cos(theta).*sin(psi),...
        cos(phi).*cos(psi)+sin(theta).*sin(phi).*sin(psi),...
        (-1).*cos(psi).*sin(phi)+cos(phi).*sin(theta).*sin(psi);...
        (-1).*sin(theta),cos(theta).*sin(phi),cos(theta).*cos(phi)];

% Feet Positions in body frame (Each column is a foot)
lxb = [l_arm_x,-l_arm_x,l_arm_x,-l_arm_x];
lyb = [-l_arm_y,l_arm_x,l_arm_x,-l_arm_x];
lzb = [-l_feet,-l_feet,-l_feet,-l_feet];

% Feet Positions in global frame (Each column is a foot)
xia = [x x x x; y y y y; z z z z]+(RIB*[lxb;lyb;lzb]);

% Feet Velocity in global frame (Each column is a foot)
xiaDot = [xDot+(-1).*lxb.*cos(psi).*thetaDot.*sin(theta)+(-1).*lxb.*...
    cos(theta).*psiDot.*sin(psi)+lzb.*(cos(theta).*cos(phi).*cos(psi).*...
    thetaDot+cos(psi).*psiDot.*sin(phi)+(-1).*cos(psi).*phiDot.*...
    sin(theta).*sin(phi)+cos(phi).*phiDot.*sin(psi)+(-1).*cos(phi).*...
    psiDot.*sin(theta).*sin(psi))+lyb.*((-1).*cos(phi).*cos(psi).*...
    psiDot+cos(phi).*cos(psi).*phiDot.*sin(theta)+cos(theta).*cos(psi).*...
    thetaDot.*sin(phi)+phiDot.*sin(phi).*sin(psi)+(-1).*psiDot.*...
    sin(theta).*sin(phi).*sin(psi));yDot+lxb.*cos(theta).*cos(psi).*...
    psiDot+(-1).*lxb.*thetaDot.*sin(theta).*sin(psi)+lyb.*((-1).* ...
    cos(psi).*phiDot.*sin(phi)+cos(psi).*psiDot.*sin(theta).*sin(phi)+...
    (-1).*cos(phi).*psiDot.*sin(psi)+cos(phi).*phiDot.*sin(theta).*...
    sin(psi)+cos(theta).*thetaDot.*sin(phi).*sin(psi))+lzb.*((-1).*...
    cos(phi).*cos(psi).*phiDot+cos(phi).*cos(psi).*psiDot.*sin(theta)+...
    cos(theta).*cos(phi).*thetaDot.*sin(psi)+psiDot.*sin(phi).*sin(psi)+...
    (-1).*phiDot.*sin(theta).*sin(phi).*sin(psi));zDot+(-1).*lxb.*...
    cos(theta).*thetaDot+lyb.*cos(theta).*cos(phi).*phiDot+(-1).*lzb.*...
    cos(phi).*thetaDot.*sin(theta)+(-1).*lzb.*cos(theta).*phiDot.*...
    sin(phi)+(-1).*lyb.*thetaDot.*sin(theta).*sin(phi)];

% For a given foot there is contact if z<0
xiaDot(3,xia(3,:)>0) = 0;
xia(3,xia(3,:)>0) = 0;

% Ground Reaction Forces in global frame (Each column is a foot)
FGR = [-abs(-ka*xia(3,:)-ca*xiaDot(3,:))*mua.*sign(xiaDot(1,:));...
       -abs(-ka*xia(3,:)-ca*xiaDot(3,:))*mua.*sign(xiaDot(2,:));...
       -ka*xia(3,:)-ca*xiaDot(3,:)];

% Ground Reaction Moments in global frame (Each column is a foot)
MGR = cross(RIB*[lxb;lyb;lzb],FGR);

% Ground Reaction in body frame (Columns are added to get total reaction)
FGRb = RBI*sum(FGR,2);
MGRb = RBI*sum(MGR,2);
```

Código 4.4: Reacciones del suelo

5. Diseño del sistema de control

A lo largo del presente capítulo se presentan las consideraciones tomadas en el diseño del controlador lineal que se utiliza para conseguir un correcto seguimiento de una trayectoria predefinida. El tipo de control utilizado se corresponde con un control óptimo lineal multivariable, más concretamente se utilizará la estrategia denominada LQR (*Lineal Quadratic Regulator*). También se hará uso de un observador de estados LQE (*Lineal Quadratic Estimator*) que junto con el LQR formarán un LQG (*Lineal Quadratic Gaussian*).

5.1 Linealización del modelo dinámico de un quadrotor

Para poder utilizar la estrategia de control descrita anteriormente es necesaria la expresión del modelo dinámico del sistema en espacio de estados. Este tipo de representación de un sistema matemático consiste en un modelo físico comprendido como un conjunto de entradas, salidas y estados que se encuentran relacionados por un conjunto de ecuaciones diferenciales de primer orden. En la ecuación (4.28) se representó el modelo dinámico no lineal del sistema del quadrotor con 6 gdl en espacio de estados. Sin embargo, para el desarrollo de una ley de control utilizando la estrategia LQR también es necesario que el sistema sea lineal, en las ecuaciones (5.1) se representa la estructura de sistema expresado en espacio de estados y formado por un conjunto de ecuaciones diferenciales de primer orden lineales.

$$\dot{\vec{x}}_{[m\times 1]} = A_{[m\times m]}\vec{x}_{[m\times 1]} + B_{[m\times n]}\vec{u}_{[n\times 1]} \tag{5.1a}$$

$$\vec{y}_{[p\times 1]} = C_{[p\times m]}\vec{x}_{[m\times 1]} + D_{[p\times n]}\vec{u}_{[n\times 1]} \tag{5.1b}$$

Donde $\vec{x}$ es el vector que contiene los estados del sistema, estos se definen como el conjunto más pequeño de variables que permiten representar el estado de un sistema en cada instante de tiempo. Además, cada una de las variables de estado de un sistema deben de ser linealmente independientes, es decir, ninguna de ellas puede ser expresada como combinación del resto. Por norma general, un sistema tiene tantas variables de estado como ecuaciones diferenciales la conforman y las dimensiones de este vector será por tanto de $m \times 1$, donde m es el número de estados internos del sistema. Por otro lado, $\dot{\vec{x}}$ es el vector que contiene las derivadas temporales de cada estado interno.

La variable $\vec{u}$ es el vector que contiene las acciones de control del sistema. Por norma general un sistema puede tener un número diferente de acciones de control independientemente del número de estados internos del sistema. El tamaño de este vector será por tanto $n \times 1$, donde n es el número de acciones de control.

La variable $\vec{y}$ es el vector que contiene las salidas del sistema. Este vector contiene aquellos estados que desean ser controlados en un control multivariable, el tamaño del vector será por tanto $p \times 1$, donde p dependerá de cuantas variables deseen ser controladas.

Las matrices que relacionan los estados y sus derivadas, las acciones de control y las salidas entre ellas son constantes en un modelo lineal y no dependen del tiempo. Por un lado, la matriz A se define como la matriz del sistema con dimensiones $m \times m$, B es la matriz de entradas con dimensiones $m \times n$, C es la matriz de salidas con dimensiones $p \times m$, y finalmente D es la matriz de *feedforward* con dimensiones $p \times n$. Generalmente D es cero para casi todos los sistemas.

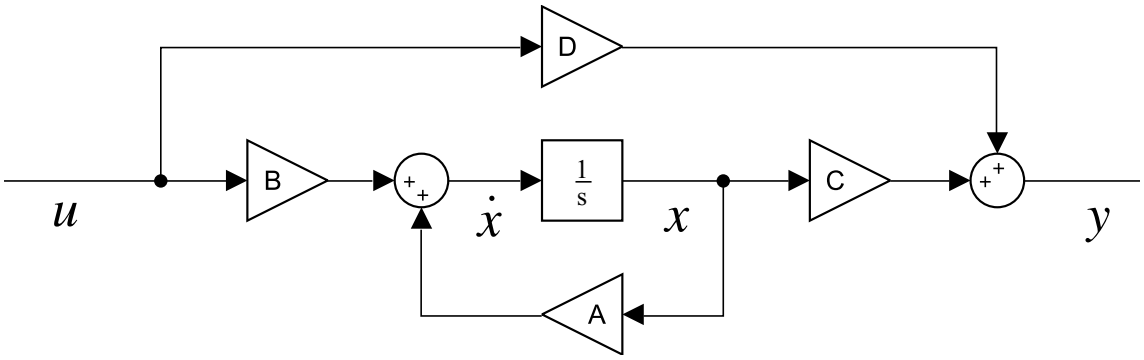


Figura 5.1: Representación en diagrama de bloques de un sistema lineal en espacio de estados

En la figura 5.1 se representa un sistema lineal en espacio de estados mediante la utilización de diagramas de bloques. Como se puede observar, el sistema tiene unas entradas que son las acciones de control y unas salidas que son los estados que se desean controlar. El proceso de integración comienza a partir de unas condiciones iniciales para cada uno de los estados del sistema. Cabe destacar que el bloque lineal representado en la figura 5.1 es equivalente al bloque que contenía el modelo no lineal y que se mostró en la sección 4.7 en la figura 4.8.

Es posible realizar una transformación de un sistema no lineal a uno que sí lo es mediante el desarrollo en serie de Taylor. Este proceso se realiza alrededor de un punto de equilibrio o funcionamiento del sistema y se truncará en el primer término de la serie tal forma que se supondrá que el sistema es lineal alrededor de este punto. El valor de cada una de las variables y acciones de control en el equilibrio se denominan $\vec{x}_0$ y $\vec{u}_0$ respectivamente.

Por tanto, este tipo de control, basado en un modelo de un sistema lineal alrededor de un punto de equilibrio, será válido para pequeñas variaciones de cada una de las variables de estado y de las acciones de control. De esta forma, se asume que el valor de cada variable de estado y acción de control se mide como pequeños incrementos sobre su valor en el equilibrio, $\vec{x} = \vec{x}_0 + \delta\vec{x}$, $\vec{u} = \vec{u}_0 + \delta\vec{u}$.

Un sistema no lineal expresado en espacio de estados tiene la forma de la ecuación (5.2), ya mostrada en la sección 4.6. 'f' es la función de estados con dimensiones $m \times 1$ que relaciona los estados y acciones de control del sistema con sus derivadas temporales, mientras que 'h' es la función de salida con dimensiones $p \times 1$.

$$\dot{\vec{x}} = f_{[m\times 1]}\left(t, \vec{x}(t), \vec{u}(t)\right) \tag{5.2a}$$

$$\vec{y} = h_{[p\times 1]}\left(t, \vec{x}(t), \vec{u}(t)\right) \tag{5.2b}$$

Para linealizar el sistema no lineal en un primer lugar es necesario calcular el punto de equilibrio. Se define punto de equilibrio aquel en el que los estados y salidas son constantes en el tiempo. Se asumirá que las salidas del sistema no lineal son una combinación lineal de las variables de estados y por tanto, si los estados son constantes en el tiempo, estas también lo serán. En la ecuación (5.3)

se muestra la condición necesaria que se ha de cumplir para encontrar el valor de los estados y acciones de control en los puntos de equilibrio.

$$0 = f\left(\vec{x}_0, \vec{u}_0\right) \tag{5.3}$$

Para el caso de un quadrotor, se asumirá que el punto de equilibrio es el vuelo en punto estacionario, también llamado *hover*. Se ha asumido que el control del quadrotor se produce cuando este ya está en vuelo y por tanto se van a ignorar las fuerzas debidas al efecto suelo y su reacción a la hora de establecer el punto de equilibrio, también se ignoran las perturbaciones debidas al viento y a fuerzas y momentos externos. Además, se ha considerado la resistencia aerodinámica despreciable a la hora de establecer el controlador.

Por tanto, ni las fuerzas ni momentos debidos a la presencia cercana del suelo, ni las perturbaciones, ni la resistencia aerodinámica son considerados por el controlador y se tratarán a estos términos como perturbaciones o dinámica no modelada. El valor de cada uno de los estados y de las acciones de control en el equilibrio se muestran en las ecuaciones (5.4).

$$\vec{x}_0 = \left[x_0, y_0, z_0, \phi_0, \theta_0, \psi_0, u_0, v_0, w_0, p_0, q_0, r_0\right]^T = \left[0,0,0,0,0,0,0,0,0,0,0,0\right]^T \tag{5.4a}$$

$$\vec{u}_0 = \left[T, M_{T_{x_0}}, M_{T_{y_0}}, M_{T_{z_0}}\right]^T = \left[mg, 0, 0, 0\right]^T \tag{5.4b}$$

En realidad, puesto que se ha asumido que la densidad es constante con la altura, ninguna de las ecuaciones depende de la posición y se podría seleccionar cualquiera en el espacio tridimensional como el punto de *hovering*. En este caso se asumen nulas para simplificar el problema. En cuanto a los ángulos de Euler, tanto el alabeo como el cabeceo deben ser nulos para poder estar en *hover*, sin embargo, la guiñada puede tener cualquier valor, en este documento se asumirá alineado el quadrotor con el norte.

En cuanto a las acciones de control en el equilibrio, la única de ellas que no es nula es la tracción que debe contrarrestar la fuerza gravitatoria.

Una vez que se ha definido el punto de funcionamiento, es posible linealizar las ecuaciones entorno a él. Para ello en la ecuación (5.5), se muestra el proceso de linealización mediante desarrollo en serie de Taylor para la ecuación no lineal 'i', es importante recordar que 'f' es en realidad un vector de funciones de tamaño igual al número de variables de estado, $f = \begin{bmatrix} f_1 & f_2 & \dots & f_m \end{bmatrix}^T$

$$\begin{aligned}
\overbrace{\dot{x}_i - \dot{x}_{i_0}}^{\delta \dot{x}_i} &= \left.\frac{\partial f_i}{\partial x_1}\right|_{\vec{x}_0, \vec{u}_0} \overbrace{(x_1 - x_{1_0})}^{\delta x_1} + \dots + \left.\frac{\partial f_i}{\partial x_m}\right|_{\vec{x}_0, \vec{u}_0} \overbrace{(x_m - x_{m_0})}^{\delta x_m} + \\
&+ \frac{1}{2!}\left.\frac{\partial^2 f_i}{\partial x_1^2}\right|_{\vec{x}_0, \vec{u}_0} \overbrace{\cancel{(x_1 - x_{1_0})^2}}^{\delta x_1^2 \approx 0} + \dots + \frac{1}{2!}\left.\frac{\partial^2 f_i}{\partial x_m^2}\right|_{\vec{x}_0, \vec{u}_0} \overbrace{\cancel{(x_m - x_{m_0})^2}}^{\delta x_m^2 \approx 0} + 0^3 \\
&+ \left.\frac{\partial f_i}{\partial u_1}\right|_{\vec{x}_0, \vec{u}_0} \overbrace{(u_1 - u_{1_0})}^{\delta u_1} + \dots + \left.\frac{\partial f_i}{\partial u_n}\right|_{\vec{x}_0, \vec{u}_0} \overbrace{(u_n - u_{n_0})}^{\delta u_n} + \\
&+ \frac{1}{2!}\left.\frac{\partial^2 f_i}{\partial u_1^2}\right|_{\vec{x}_0, \vec{u}_0} \overbrace{\cancel{(u_1 - u_{1_0})^2}}^{\delta u_1^2 \approx 0} + \dots + \frac{1}{2!}\left.\frac{\partial^2 f_i}{\partial u_n^2}\right|_{\vec{x}_0, \vec{u}_0} \overbrace{\cancel{(u_n - u_{n_0})^2}}^{\delta u_n^2 \approx 0} + 0^3
\end{aligned} \tag{5.5}$$

En la ecuación (5.5), como se ha indicado con anterioridad, es posible definir cada uno de los estados y acciones de control como incrementos sobre su valor en el equilibrio, empleando para

ello la letra griega 'δ'. La suposición de que las variaciones del sistema sobre su estado de equilibrio son pequeñas hacen que los términos con δ de órdenes mayores o iguales que 2 puedan ser despreciados ante los mismos términos de orden 1. Es por esto por lo que en la ecuación se tachan aquellos cuyo orden es 2 y se omiten los de orden mayor. Es importante observar que cada una de las derivadas de la función 'f_i' son evaluadas en el punto de equilibrio obteniendo un escalar, por tanto, como se puede observar, se ha convertido la función en lineal con respecto a los estados y acciones de control.

$$\overbrace{\begin{bmatrix}\delta\dot{x}_1\\ \delta\dot{x}_2\\ \vdots\\ \delta\dot{x}_m\end{bmatrix}}^{\delta\dot{\vec{x}}} = \overbrace{\begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m}\\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_m}\end{bmatrix}}^{A}\Bigg|_{\vec{x}_0,\vec{u}_0} \overbrace{\begin{bmatrix}\delta x_1\\ \delta x_2\\ \vdots\\ \delta x_m\end{bmatrix}}^{\delta\vec{x}} + \overbrace{\begin{bmatrix}\frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_n}\\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial f_m}{\partial u_1} & \frac{\partial f_m}{\partial u_2} & \cdots & \frac{\partial f_m}{\partial u_n}\end{bmatrix}}^{B}\Bigg|_{\vec{x}_0,\vec{u}_0} \overbrace{\begin{bmatrix}\delta u_1\\ \delta u_2\\ \vdots\\ \delta u_n\end{bmatrix}}^{\delta\vec{u}} \tag{5.6}$$

Si se expresa en forma matricial la ecuación (5.5) para cada una de las componentes del vector 'f' se obtiene la ecuación (5.6). Se obtendría, por tanto, un sistema equivalente al mostrado en (5.1) donde las matrices A y B son constantes y formadas por escalares.

A continuación se muestra el modelo de la dinámica de un quadrotor linealizado alrededor de la maniobra de 'hover', ecuación (5.7). Para conseguir este resultado se ha seguido la ecuación mostrada en (5.6), particularizada para este caso con sus estados y acciones de control. Para simplificar la nomenclatura no se incluirá la letra δ para indicar los incrementos con respecto al equilibrio, aunque es importante recordar esta definición de cada uno de los estados y acciones de control del sistema linealizado. El único estado o acción de control que no es nula en el equilibrio es la tracción, por tanto, todas las variables pueden ser manipuladas como absolutas excepto ella que será relativa y para calcular su valor absoluto habrá que añadirle el término mg.

En la ecuación (5.7) se ha separado en bloques cada una de las matrices del sistema, vectores de estados y acciones de control. De tal forma que las componentes dentro de un mismo sector pertenecen al mismo tipo de ecuaciones, cinemáticas o dinámicas, traslacionales o de rotación. El sistema se puede observar que se ha simplificado en gran medida, permitiendo realizar control lineal sobre él.

Analizando las ecuaciones es posible observar que la dinámica rotatoria no depende de la traslacional, aunque no al revés, por tanto se podría controlar la orientación del quadrotor sin tener en cuenta su posición o velocidad. Este fenómeno hace que se puedan desacoplar las dos dinámicas y controlar el sistema mediante dos leyes de control, la primera de ellas sería un control de posición o velocidad que calcularía cuáles serían las orientaciones deseadas como acciones de control, estas orientaciones serían por otro lado las entradas del control de los estados rotacionales. Aunque en un control multivariable no es necesario este tipo de distinción puesto que se puede controlar la referencia de los estados traslacionales mediante la realimentación del estado del sistema, este modo de proceder proporciona diversas ventajas y será el utilizado en el presente documento, tal y como se verá en la subsección 5.2.4.

$$
\overbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \hline \dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\\ \hline \dot{u}\\ \dot{v}\\ \dot{w}\\ \hline \dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix}}^{\dot{\vec{x}}}
=
\overbrace{\left[\begin{array}{c|ccc|c|c}
0_{[3\times3]} & \multicolumn{3}{c|}{0_{[3\times3]}} & I_{[3\times3]} & 0_{[3\times3]}\\ \hline
0_{[3\times3]} & \multicolumn{3}{c|}{0_{[3\times3]}} & 0_{[3\times3]} & I_{[3\times3]}\\ \hline
0_{[3\times3]} & 0 & g & 0 & 0_{[3\times3]} & 0_{[3\times3]}\\
 & -g & 0 & 0 & & \\
 & 0 & 0 & 0 & & \\ \hline
0_{[3\times3]} & \multicolumn{3}{c|}{0_{[3\times3]}} & 0_{[3\times3]} & 0_{[3\times3]}
\end{array}\right]}^{A}
\overbrace{\begin{bmatrix}x\\ y\\ z\\ \hline \phi\\ \theta\\ \psi\\ \hline u\\ v\\ w\\ \hline p\\ q\\ r\end{bmatrix}}^{\vec{x}}
+
$$

$$
+
\overbrace{\left[\begin{array}{cccc}
\multicolumn{4}{c}{0_{[3\times4]}}\\ \hline
\multicolumn{4}{c}{0_{[3\times4]}}\\ \hline
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
1/m & 0 & 0 & 0\\ \hline
0 & 1/I_{xx} & 0 & 0\\
0 & 0 & 1/I_{yy} & 0\\
0 & 0 & 0 & 1/I_{zz}
\end{array}\right]}^{B}
\overbrace{\begin{bmatrix}T\\ M_{T_x}\\ M_{T_y}\\ M_{T_z}\end{bmatrix}}^{\vec{u}}
\tag{5.7}
$$

Cabe destacar que en el presente documento se ha linealizado el modelo cinemático y dinámico del movimiento de un quadrotor con 6 gdl utilizando sus $m = 12$ estados y sus $n = 4$ acciones de control, por otro lado, las salidas dependen del tipo de control a realizar.

Se debe de tener en cuenta que existen algunos casos donde algunos estados no son necesarios para el tipo de control que se desea realizar. Un ejemplo es el caso donde se desea controlar la altura de un quadrotor ya que no serán necesarios los estados $x, y, \phi, \theta, \psi, u, v, p, q, r$ ni las acciones de control $M_{T_{x_0}}, M_{T_{y_0}}, M_{T_{z_0}}$. Si se desease realizar este control se deberían eliminar las filas y columnas correspondientes a estos estados y acciones de control. Por tanto, en función de los estados que se deseen controlar, el número de estados internos, de acciones de control y de salidas variarán.

5.2 Controlador LQR

Un sistema de control debe de cumplir principalmente tres objetivos primarios. El primero de ellos es conseguir una estabilización del sistema, el segundo de ellos es garantizar que los estados que desean ser controlados sean capaces de seguir una referencia y finalmente se debe asegurar el rechazo de perturbaciones y otros fenómenos que no han sido modelados o existe incertidumbre. Otras características con las que se puede diseñar un controlador es conseguir un determinado tiempo de establecimiento, de subida, limitar la sobreoscilación o conseguir un determinado error en estado permanente.

5.2.1. Estructura del controlador

Para que la dinámica de un sistema expresado en espacio de estados sea estable es necesario que todos y cada uno de los autovalores de la matriz A tengan parte real negativa. Sin embargo, en el caso de un quadrotor, sus 12 autovalores son nulos, dando lugar a una dinámica neutra que a efectos prácticos es inestable. Por tanto, el diseño del sistema de control debe de garantizar que partiendo de un estado en equilibrio, tras una determinada perturbación, se alcance de nuevo un punto de equilibrio, es decir, el equilibrio del sistema debe de ser asintóticamente estable.

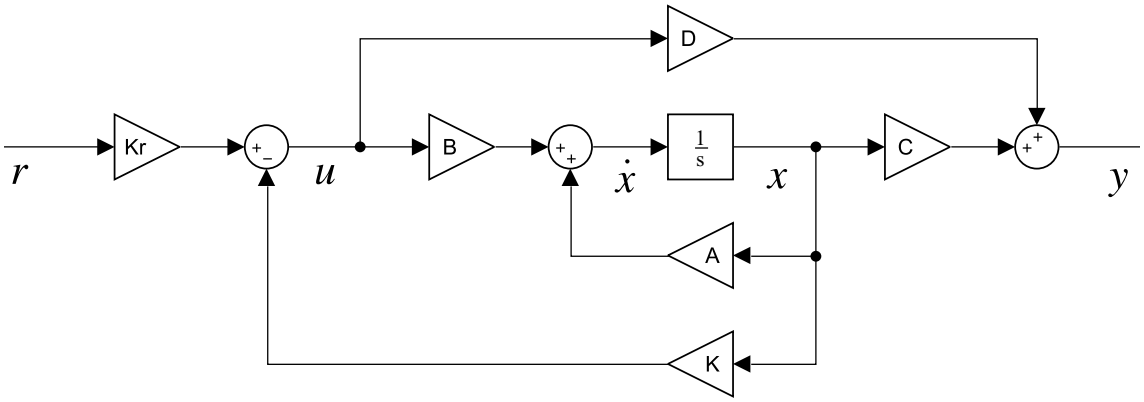


Figura 5.2: Estabilización del sistema mediante realimentación de estados

Asumiendo que todos los estados del sistema son medibles, es posible establecer una ley de control como combinación lineal de ellos y de las referencias, ecuación (5.8a). El vector columna que contiene las referencias, $\vec{r}$, y su ganancia, K_r, será definido posteriormente, por esta razón no se han especificado todavía sus dimensiones. Definiendo el resto de términos de la ecuación, K es una matriz con dimensiones $n \times m$. Si se sustituye esta ley de control en la ecuación (5.1), se obtiene la ecuación (5.8b) que representa la dinámica del sistema en bucle cerrado. En este caso, los autovalores del sistema en bucle cerrado estarían representados por $s = p_1, p_2, \ldots, p_m$, tal y como se muestra en la ecuación (5.8c).

$$\vec{u}(t) = K_r\vec{r}(t) - K_{[n\times m]}\vec{x}(t) \tag{5.8a}$$

$$\dot{\vec{x}}(t) = (A - BK)\,\vec{x}(t) + BK_r\vec{r}(t) \tag{5.8b}$$

$$|A - BK - sI| = 0 \tag{5.8c}$$

Mediante esta definición de la ley de control, escogiendo adecuadamente cada una de las componentes K_{ij} de la matriz de realimentación de estados K, es posible la modificación de los autovalores del sistema. Este procedimiento permite la elección del valor de cada uno de ellos para garantizar la estabilidad y una determinada respuesta dinámica ante perturbaciones y cambios de referencia. Mientras que todos los componentes de la matriz K deben ser reales, los polos del sistema pueden ser también imaginarios pero deben presentarse en pares conjugados.

En la figura 5.2 se muestra la estructura del controlador por realimentación de estado utilizada. Como se ha indicado anteriormente, el uso de una realimentación de estados permite una estabilización del sistema ante cambios de referencia y perturbaciones, sin embargo no garantiza que se siga una determinada referencia ni el rechazo de perturbaciones, es decir, existirá un determinado error en régimen permanente.

Para conseguir que un sistema de control por realimentación de estados cumpla con los objetivos descritos aparte de la estabilidad es necesario añadir una acción integral. La nueva estructura del sistema de control se muestra en la figura 5.3

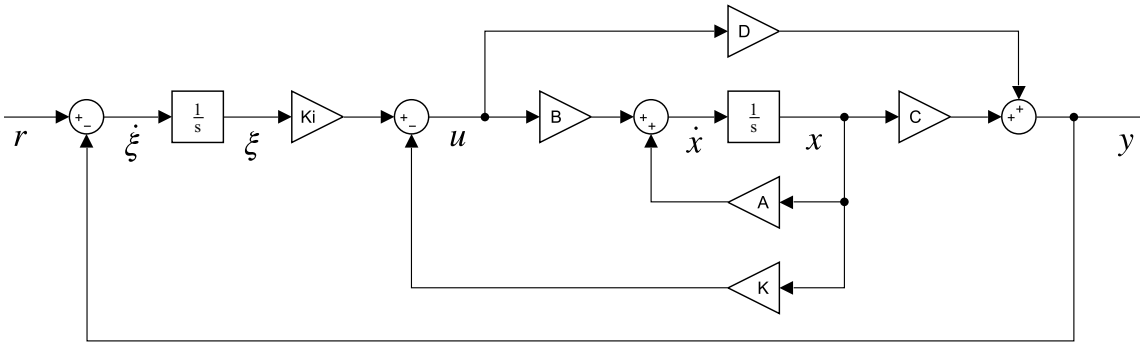


Figura 5.3: Acción integral del sistema de control

En un sistema expresado en espacio en estados, las salidas, $\vec{y}$, están definidas como una combinación lineal de los estados internos, $\vec{x}$, tal y como se mostraba en la ecuación (5.1). Las implicaciones que tiene en el control multivariable es que este se centra en garantizar que cada una de estas salidas siga una determinada referencia. Por ello, se define el vector que contiene las referencias a seguir en el tiempo de cada salida como $\vec{r}$, cuyas dimensiones serán, por tanto, las misma que las del vector de salidas, $\vec{y}$. Tal y como se muestra en la ecuación (5.9), la resta de $\vec{r}$ y $\vec{y}$ constituirán el vector de errores de las variables controladas, $\vec{\xi}$, es decir, la diferencia entre la referencia y el estado actual de cada salida. Por otro lado, se considera que la ganancia del vector de referencias, K_r es la matriz identidad.

$$\dot{\vec{\xi}}\,(t)_{[p\times 1]} = \overbrace{K_{r[p\times p]}}^{I}\vec{r}\,(t)_{[p\times 1]} - \vec{y}\,(t) = \vec{r}\,(t) - \vec{y}\,(t) \tag{5.9}$$

El vector de errores de las salidas es integrado y posteriormente multiplicado por la matriz K_i conocida como la ganancia del integrador y de dimensiones $n \times p$. La adición de este término integral es el que hace posible garantizar el seguimiento de las referencias y rechazo de perturbaciones con error de posición nulo. Mediante la modificación de la ganancia del integrador se podrá garantizar una respuesta más rápida y un menor tiempo de subida aunque la sobreoscilación o las magnitud de las acciones de control utilizadas aumentarán como contrapartida.

Para la elección de cada una de las componentes $K_{i_{ij}}$ de la ganancia del integrador K_i es necesario la definición del sistema ampliado mostrado en la ecuación (5.10a). Como se puede observar, es coherente con la estructura mostrada en la figura 5.3. Para ello se han redefinido cada una de las variables mediante el uso del guión sobre cada una de ellas, en el caso del vector de estados internos del sistema, '$\vec{x}$', indicaría el vector de estados internos del sistema ampliado.

$$\overbrace{\begin{pmatrix}\dot{\vec{x}}\\ \dot{\vec{\xi}}\end{pmatrix}}^{\dot{\overline{\vec{x}}}_{[(m+p)\times 1]}} = \overbrace{\begin{bmatrix}A_{[m\times m]} & 0_{[m\times p]}\\ -C_{[p\times m]} & 0_{[p\times p]}\end{bmatrix}}^{\overline{A}_{[(m+p)\times(m+p)]}}\overbrace{\begin{pmatrix}\vec{x}\\ \vec{\xi}\end{pmatrix}}^{\overline{\vec{x}}_{[(m+p)\times 1]}} + \overbrace{\begin{bmatrix}B_{[m\times n]}\\ 0_{[p\times n]}\end{bmatrix}}^{\overline{B}_{[(m+p)\times n]}}\vec{u} + \overbrace{\begin{bmatrix}0_{[m\times p]}\\ I_{[p\times p]}\end{bmatrix}}^{\overline{K}_{r[(m+p)\times p]}}\vec{r} \tag{5.10a}$$

$$\dot{\overline{\vec{x}}} = \overline{A}\,\overline{\vec{x}} + \overline{B}\,\vec{u} + \overline{K}_r\vec{r} \tag{5.10b}$$

En este nuevo sistema ampliado se debe redefinir la ley de control, que siguiendo la analogía del sistema simple debe de ser combinación lineal de los estados ampliados del sistema. En la

ecuación (5.11a) y (5.11b) se ha definido la matriz de realimentación de estados ampliada como $\overline{K}$.

$$\overline{K}_{[n\times(m+p)]} = \begin{bmatrix} K_{[n\times m]} & K_{i[n\times p]} \end{bmatrix} \tag{5.11a}$$

$$\vec{u} = -\overline{K}\vec{x} = -\begin{bmatrix} K & K_i \end{bmatrix} \begin{pmatrix} \vec{x} \\ \vec{\xi} \end{pmatrix} \tag{5.11b}$$

También, por analogía con el sistema simple, es posible definir la ecuación (5.12a) que representa la dinámica del sistema ampliado en bucle cerrado. Los autovalores de este sistema pueden ser calculados tal y como se muestran en la ecuación (5.12b), donde la variable $s = p_1, p_2, \ldots, p_{m+p}$ representa cada uno de los autovalores. Como se puede observar, mediante la definición del sistema ampliado es posible colocar cada uno de los autovalores del sistema y del integrador de forma idéntica a como sucedía en el caso simple. Presentan como aspecto común que ambos sistemas, simple y ampliado, permiten colocar los autovalores del sistema en bucle cerrado en el lugar deseado, pero como diferencia que el ampliado también permite ubicar los polos del integrador para conseguir un rechazo de perturbaciones y un seguimiento de referencia.

$$\dot{\vec{\tilde{x}}} = \left(\overline{A} - \overline{B}\,\overline{K}\right)\vec{\tilde{x}} + \overline{K}_r\vec{r} \tag{5.12a}$$

$$\left|\overline{A} - \overline{B}\,\overline{K} - sI\right| = 0 \tag{5.12b}$$

5.2.2. Controlabilidad del sistema

Un sistema se define como controlable en un instante de tiempo determinado, t_0, si es posible mediante unas acciones de control no restringidas transferir el sistema de un estado inicial $\vec{x}(t_0)$ a otro en un periodo finito de tiempo, [25].

Para que un sistema sea controlable es necesario que el rango de la matriz expresada en la ecuación (5.13) sea máximo, en caso contrario el sistema será incontrolable. La definición que se tomará en este documento de la matriz de controlabilidad se hará utilizando las matrices del sistema aumentado y permitirá determinar para cada tipo de control que estados son necesarios y cuáles no para conseguir que las salidas de un determinado sistema sean controlables. Por tanto, en cada uno de los sistemas que se muestran en el documento, control de orientación, posición y velocidad, se asegura que el rango de su matriz de controlabilidad es máximo.

$$CTR = \begin{bmatrix} \overline{B} & \overline{A}\,\overline{B} & \overline{A}^2\overline{B} & \ldots & \overline{A}^{m-1}\overline{B} \end{bmatrix} \tag{5.13}$$

5.2.3. Asignación de polos en bucle cerrado

El único término que queda por definir en la estructura de la ley de control mostrada en la figura 5.3 es la matriz de realimentación del estado del sistema ampliado, $\overline{K}$. Como se ha indicado en la subsección 5.2.1, mediante la modificación de la realimentación del sistema se podrá ajustar la ubicación de los polos del sistema y del integrador, de esta forma se podrá conseguir un sistema asintóticamente estable con seguimiento de referencias y rechazo de perturbaciones. Por esta razón, la correcta elección de $\overline{K}$ determinará la dinámica del sistema, se podrá ajustar el tiempo de establecimiento, sobreoscilación, amortiguamiento de la respuesta, y otros parámetros dinámicos.

Existen diferentes técnicas que permiten la obtención de las componentes de la matriz de realimentación. La más sencilla de ellas es la asignación de los polos manualmente por el diseñador,

para ello es necesario definir tantos polos como número de estados internos más variables de salida del sistema. Tal y como se muestra en la ecuación (5.14a), si se conocen todos los polos, $p_1, p_2, \ldots, p_{m+p}$, es posible el cálculo del polinomio característico del sistema ampliado, además mediante la igualación al del sistema ampliado en bucle cerrado, ecuación (5.14b). La única incógnita serían cada una de las componentes de la matriz de realimentación del sistema ampliado que podría ser despejada mediante igualación de términos.

$$pol\,(s) = (s-p_1)\,(s-p_2)\ldots(s-p_{m+p}) \tag{5.14a}$$

$$\left|\overline{A}-\overline{B}\,\overline{K}-sI\right| = (s-p_1)\,(s-p_2)\ldots(s-p_{m+p}) \tag{5.14b}$$

La principal ventaja de este método es que permite definir cada uno de los polos del sistema obteniendo la dinámica deseada. Sin embargo, este procedimiento no tiene en cuenta restricciones físicas de los estados internos del sistema ni posibles saturaciones de las acciones de control.

Si debido a temas de estabilidad o de diseño del sistema, un determinado estado no puede pasar cierto valor, mediante la asignación de los polos no se podrá diseñar la ley de control en base a este criterio. Por otro lado, las posibles limitaciones que pueden haber en los actuadores tampoco aparecen a la hora del diseño, ya que en ningún momento se ha tenido en cuenta el coste necesario para cumplir las determinadas especificaciones dinámicas. Debido a que en la realidad las acciones de control presentan límites superiores e inferiores, puede suceder que mediante la asignación de polos se llegue a la saturación, no siendo posible cumplir las especificaciones dinámicas de diseño.

El control óptimo es capaz de solventar estos dos problemas. Mediante el uso de una estrategia de diseño de leyes de control como el LQR es posible minimizar una función de coste cuadrática a la que se le han suministrado unos factores de ponderación de cada uno de los estados y acciones de control del sistema.

Generalmente esta función de coste se define como la suma cuadrática de las desviaciones de cada estado con su referencia, de esta forma, mediante la minimización de ella es posible encontrar la ley de control que permite obtener un funcionamiento óptimo, minimizando las desviaciones y limitando las acciones de control. En la ecuación (5.15) se muestra la estructura general la función de coste cuadrática.

$$J = \int_{t_0}^{t_1} \left(\vec{\overline{x}}^T Q\ \vec{\overline{x}} + \vec{u}^T R\ \vec{u}\right)\ dt \tag{5.15}$$

Donde Q y R son las matrices que ponderan los estados y las acciones de control respectivamente. Generalmente son matrices diagonales, semi-definida y definida positiva con dimensiones iguales al vector que ponderan cada una de ellas. El proceso de ponderación seguido permite limitar un estado o acción de control aumentando el parámetro de la matriz diagonal asociado.

Considerando la ley de control tal y como se muestra en la ecuación (5.11b) y combinándola con la función de coste se obtiene la ecuación (5.16).

$$J = \int_{t_0}^{t_1} \left[\vec{\overline{x}}^T \left(Q + \overline{K}^T R \overline{K}\right)\ \vec{\overline{x}}\right]\ dt \tag{5.16}$$

La matriz de realimentación de estado ampliada viene definida por la relación mostrada en las ecuaciones (5.17), donde P es la solución de la ecuación de Riccati mostrada en (5.17b).

$$\overline{K} = R^{-1}\overline{B}^T P \tag{5.17a}$$

$$\overline{A}^T P + P\overline{A} - P\overline{B}R^{-1}\overline{B}^T P + Q = 0 \tag{5.17b}$$

Para ajustar las ganancias de cada una de las matrices de ponderación existen diversas metodologías. Parte de ellas experimentales y centradas en ajuste y error y otras numéricas. La utilizada en el presente documento es la regla de Bryson, que permite obtener cada uno de las componentes de Q y R si se conocen los valores máximos de los estados y acciones de control tal y como se muestra en las ecuaciones (5.18), [26]. Por tanto, las dimensiones de Q son iguales al número de estados internos más el número de ellos que son controlados, las dimensiones de R son iguales al número de acciones de control. Q_{ii} y R_{jj} hacen referencia al elemento de la diagonal correspondiente al estado x_i y acción de control u_j respectivamente. Generalmente esta regla permite obtener unos valores iniciales de las matrices de ponderación, aunque posteriormente pueden ser modificadas para mejorar el desempeño dinámico del sistema. El criterio usado para el ajuste de las matrices Q y R para el caso del presente documento se muestra en la subsección 5.2.4.

$$Q_{ii} = \frac{1}{max\left(\bar{x}_i^2\right)} \tag{5.18a}$$

$$R_{jj} = \frac{1}{max\left(u_j^2\right)} \tag{5.18b}$$

Otras de las opciones que pueden ser utilizadas son optimizaciones numéricas como puede ser el uso de algoritmos genéticos. Partiendo de un conjunto inicial de matrices Q y R, para cada par de ellas se simula la respuesta del sistema y mediante la evolución, mutación y reproducción de esta población inicial se alcanza un óptimo local para un determinado par de matrices. Puesto que se simula la evolución genética, al final del proceso la mayoría de pares de matrices serán similares, próximas al óptimo local mencionado anteriormente. Para evaluar la actuación de cada individuo dentro de una población se hace uso de una función de coste que evalúa si la respuesta del sistema es estable, el tiempo de establecimiento y la máxima acción de control y respuesta. En la referencia [27] se hace uso de este tipo de algoritmos para ajustar las matrices Q y R del problema del péndulo invertido.

5.2.4. Implementación del controlador LQR

La arquitectura con la que se ha implementado el controlador LQR es mediante una doble realimentación. El sistema está compuesto por un bucle interno que se encarga de controlar la altitud y la orientación del quadrotor cuyas referencias están suministradas por el bucle externo que se encarga de controlar la posición en los ejes x_G y y_G o la velocidad en los ejes x_B y y_B, en función de las necesidades del control.

A su vez, esta arquitectura del sistema de control permite una reutilización del control de orientación tanto en el control de posición como de velocidad o el control manual por un usuario. Además, se facilita en gran medida el ajuste de las matrices Q y R para ambas realimentaciones y permite saturar las referencias de los ángulos de cabeceo y guiñada suministradas al bucle interno por el externo. De esta forma se limitan las actuaciones del quadrotor para evitar que se vuelva inestable a grandes ángulos.

Entonces es necesario el diseño de dos leyes de control, para el bucle interno y externo. Para ello se representa en espacio de estados la dinámica de ambos bucles en la ecuación (5.19). Es importante notar que a partir de ahora se utilizará el subíndice ‘1’ para referirse al bucle interno y ‘2’ al externo.

$$\overbrace{\begin{bmatrix}\dot{z}\\ \dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\\ \dot{w}\\ \dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix}}^{\dot{\vec{x}}_1} = \overbrace{\left[\begin{array}{c|ccc|c|ccc} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & & & & 0 & & & \\ 0 & & 0_{[3\times3]} & & 0 & & I_{[3\times3]} & \\ 0 & & & & 0 & & & \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & & & & 0 & & & \\ 0 & & 0_{[3\times3]} & & 0 & & 0_{[3\times3]} & \\ 0 & & & & 0 & & & \end{array}\right]}^{A_1} \overbrace{\begin{bmatrix}z\\ \phi\\ \theta\\ \psi\\ w\\ p\\ q\\ r\end{bmatrix}}^{\vec{x}_1} + \overbrace{\left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ \hline & & & \\ & & 0_{[3\times4]} & \\ & & & \\ \hline 1/m & 0 & 0 & 0 \\ \hline 0 & 1/I_{xx} & 0 & 0 \\ 0 & 0 & 1/I_{yy} & 0 \\ 0 & 0 & 0 & 1/I_{zz} \end{array}\right]}^{B_1} \overbrace{\begin{bmatrix}T\\ M_{T_x}\\ M_{T_y}\\ M_{T_z}\end{bmatrix}}^{\vec{u}_1} \tag{5.19a}$$

$$\overbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{u}\\ \dot{v}\end{bmatrix}}^{\dot{\vec{x}}_2} = \overbrace{\left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]}^{A_2} \overbrace{\begin{bmatrix}x\\ y\\ u\\ v\end{bmatrix}}^{\vec{x}_2} + \overbrace{\left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ \hline 0 & g \\ -g & 0 \end{array}\right]}^{B_2} \overbrace{\begin{bmatrix}\phi\\ \theta\end{bmatrix}}^{\vec{u}_2} \tag{5.19b}$$

También se debe destacar que la hipótesis de que el quadrotor se encuentra cerca del 'hover' es bastante cercana a la realidad y permite obtener buenos resultados en el control de posición y velocidad, ya que tanto las velocidades y ángulos de alabeo y cabeceo a las que se mueve un quadrotor no son elevadas. Para el caso de estos dos ángulos utilizados para describir la orientación del quadrotor, no se esperan que sean mayores de alrededor 30° en valor absoluto, rango donde la suposición de ángulos pequeños, aunque pierde precisión, continúa siendo válida. Los errores de modelado debidos a esta hipótesis pueden ser considerados como perturbaciones y deben ser contrarrestados por el diseño del sistema de control.

Sin embargo, la hipótesis de ángulos de guiñada pequeños no es realmente válida, puesto que en un vuelo de un quadrotor real uno desea que sea capaz de estar orientado en cualquier dirección, es decir, la guiñada puede comprender valores de entre -180° y 180° que están lejos de poder ser considerados pequeños. La razón por la que se consideró está hipótesis y se linealizó en torno al punto de guiñada nula es para evitar los fuertes acoplamientos que aparecen en la dinámica rotacional y traslacional del quadrotor, desarrollando movimientos indeseados, [28]. Debido a que el sistema dinámico del quadrotor está subactuado, tal y como se indicó en la sección 4.6, si se considera que solamente puede moverse con ángulos de guiñada nulos, se limitan las capacidades dinámicas en gran medida. Por otro lado, es deseable poder modificar la guiñada en aquellas misiones donde se quiere mantener contacto visual con un objetivo mediante una cámara, como por ejemplo en tareas de vigilancia.

En la gran mayoría de estudios anteriores donde se diseñan reguladores del tipo presentado en este documento se hace la hipótesis de pequeños valores de guiñada y se realiza el control con el quadrotor orientado al norte sin cambios de referencia, [29], [30]. Sin embargo, en el presente documento se hará uso de una estrategia de control mediante 'Gain Scheduling' que permitirá, de forma iterativa, ajustar la ganancia de la realimentación en función de los estados del sistema, permitiendo extender la ley de control a cualquier ángulo de guiñada.

Por esta razón, se propone un controlador LQR con una matriz de realimentación que varíe en función del tiempo y de cuál sea el ángulo de guiñada en cada instante. Por ello, las ecuaciones (5.19) se reformularán como linealizadas en torno a una guiñada genérica, ψ. En concreto, la dinámica del el bucle interno mostrada en al ecuación (5.19a) no se ve modificada, aunque si la dinámica del bucle externo de la ecuación (5.19b), por ello, en la ecuación (5.20) se incluye la posible variación con la guiñada. De esta forma, la matriz de realimentación del sistema externo aumentado, $\overline{K}_2$, se podrá obtener como dependiente de la guiñada actual del quadrotor que será a

su vez dependiente del tiempo. Por tanto, las matrices $\overline{K}_2(t)$ y $\overline{A}_2(t)$ son ahora dependientes del tiempo.

$$\overbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{u}\\ \dot{v}\end{bmatrix}}^{\dot{\vec{x}}_2} = \overbrace{\left[\begin{array}{cc|cc} 0 & 0 & \cos(\psi) & -\sin(\psi)\\ 0 & 0 & \sin(\psi) & \cos(\psi)\\ \hline 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{array}\right]}^{A_2} \overbrace{\begin{bmatrix}x\\ y\\ u\\ v\end{bmatrix}}^{\vec{x}_2} + \overbrace{\left[\begin{array}{cc} 0 & 0\\ 0 & 0\\ \hline 0 & g\\ -g & 0\end{array}\right]}^{B_2} \overbrace{\begin{bmatrix}\phi\\ \theta\end{bmatrix}}^{\vec{u}_2} \tag{5.20}$$

Antes de pasar al diseño de cada uno de los reguladores LQR, interno y externo, en la figura 5.4 se muestra el esquema de control utilizado y que será genérico tanto para el control de velocidad como de posición.

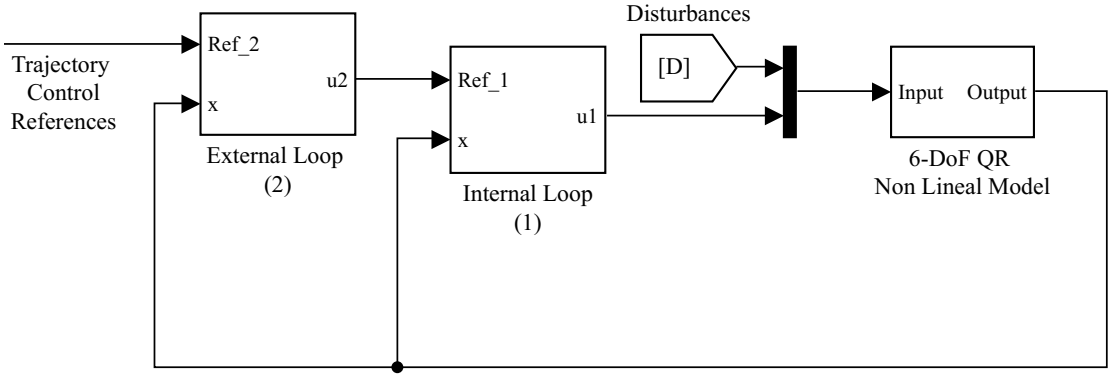


Figura 5.4: Estructura del controlador: Bucle interno y externo

Como se ha indicado anteriormente, la estructura del controlador está dividida en dos bucles, el interno y externo. El controlador externo recibe las referencias procedentes del control de trayectoria, estás pueden ser o posiciones en los ejes x_G y y_G o velocidades en los ejes x_B y y_B en función de las variables que deseen ser controladas. Aunque no se utilicen directamente en este bucle, en el externo también se reciben como referencias la altitud de vuelo y el ángulo de rumbo deseado que serán pasadas directamente al bucle interno como entradas y sin ningún procesamiento. Mediante la realimentación de estados con acción integral descrita en la figura 5.3 se generan las acciones de control del bucle interno que son la altitud de referencia y los tres ángulos de Euler que describen su orientación. Posteriormente, en el control de orientación y altura que se produce en el bucle interno, de forma similar a como ocurría en el bucle externo, se generan las acciones de control del sistema, $\vec{u}$, para seguir las referencias generadas por el externo. Finalmente, al vector de acciones de control se añaden las perturbaciones, tal y como se indicó en la sección 4.7.

En la figura 5.4, el bloque '6-Dof QR Non Lineal Model' es el bloque utilizado para describir la cinemática y dinámica del movimiento del quadrotor y cuyo funcionamiento ya había sido descrito en la sección 4.7. En las figuras 5.5 y 5.6 se muestra, finalmente, la implementación de cada uno de los dos bucles. Se puede observar en estos diagramas aquellas referencias y estados que son utilizados en cada bucle.

Como se puede observar, ambas estructuras son idénticas a la mostrada en la figura 5.3. La única diferencia aparece en el bucle externo ya que, las matrices de realimentación del sistema, K_2 y K_{i2}, se recalculan en cada iteración del sistema de control en función de la guiñada en ese instante del quadrotor. De esta forma se consigue una ley de control que es válida para todo el rango de ángulos posible. Además, también es importante destacar que una vez que se han calculado las referencias que debe de seguir el quadrotor en ϕ y θ, estas se saturan para evitar grandes valores que puedan hacer entrar al quadrotor en una zona de su dinámica donde pueda ser inestable la ley

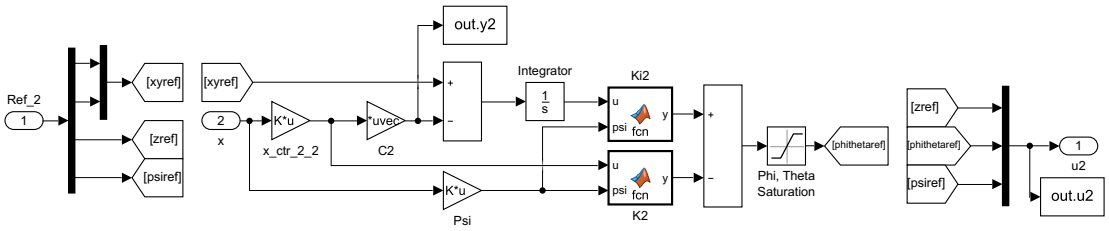


Figura 5.5: Estructura del controlador: Bucle externo

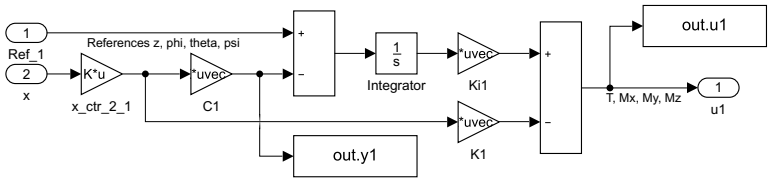


Figura 5.6: Estructura del controlador: Bucle interno

de control aplicada. De esta forma se asegura que el control de orientación es válido en todo el rango de aplicación que es entre $-30°$ y $30°$ para el cabeceo y alabeo.

También es destacable que la operación matricial, x_{ctr_2}, utilizada para seleccionar los estados necesarios en el control, se ha sacado fuera del modelo dinámico del sistema mostrado en la figura 4.8. De esta forma cada bucle, interno y externo, tienen implementados esta misma operación matricial para poder separar aquellos estados que son necesarios del conjunto de 12 estados internos totales del sistema.

Diseño del bucle interno del regulador

Como se ha comentado anteriormente, el bucle interno es el correspondiente al control de la altitud de vuelo y orientación del quadrotor. Por tanto, los estados del sistema serán $m = 8$, mostrados en la ecuación (5.21a). En cuanto a los estados del sistema que serán controlados, estos serán los $p = 4$ mostrados en la ecuación (5.21b), es decir, la altura y los ángulos de Euler. Finalmente, las acciones de control del bucle interno son las $n = 4$ mostradas en la ecuación (5.21c).

$$\vec{x}_1 = \begin{bmatrix} z & \phi & \theta & \psi & w & p & q & r \end{bmatrix}^T \tag{5.21a}$$

$$\vec{y}_1 = \begin{bmatrix} z & \phi & \theta & \psi \end{bmatrix}^T \tag{5.21b}$$

$$\vec{u}_1 = \begin{bmatrix} T & M_{T_x} & M_{T_y} & M_{T_z} \end{bmatrix}^T \tag{5.21c}$$

Esta definición de estados, referencias y acciones de control también puede ser observada en la ecuación (5.19a) donde se mostraba la dinámica del sistema del bucle interno.

$$C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.22a}$$

$$D_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.22b}$$

$$
\overbrace{\begin{pmatrix} \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\xi}_z \\ \dot{\xi}_\phi \\ \dot{\xi}_\theta \\ \dot{\xi}_\psi \end{pmatrix}}^{\dot{\vec{x}}_1}
=
\overbrace{\left[\begin{array}{cccccccc|cccc}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]}^{\overline{A}_1}
\overbrace{\begin{pmatrix} z \\ \phi \\ \theta \\ \psi \\ w \\ p \\ q \\ r \\ \xi_z \\ \xi_\phi \\ \xi_\theta \\ \xi_\psi \end{pmatrix}}^{\vec{x}_1}
+
$$

$$
+
\overbrace{\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1/m & 0 & 0 & 0 \\
0 & 1/I_{xx} & 0 & 0 \\
0 & 0 & 1/I_{yy} & 0 \\
0 & 0 & 0 & 1/I_{zz} \\
\hline
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}}^{\overline{B}_1}
\overbrace{\begin{pmatrix} T \\ M_{x_T} \\ M_{y_T} \\ M_{z_T} \end{pmatrix}}^{\vec{u}_1}
+
\overbrace{\left[\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{array}\right]}^{\overline{K}_{r1}}
\overbrace{\begin{pmatrix} r_z \\ r_\phi \\ r_\theta \\ r_\psi \end{pmatrix}}^{\vec{r}_1}
\tag{5.23}
$$

Las matrices A_1 y B_1 del bucle interno pueden encontrarse en la ecuación (5.19a). En cuanto a las matrices C_1 y D_1 del sistema dinámico del bucle interno se muestran en la ecuación (5.22). Los estados que son controlados en el bucle interno, tal y como puede observarse mediante la matriz C_1 son z, ϕ, θ, ψ.

Una vez definido el sistema completamente en espacio de estados, se define el sistema ampliado (5.23), de tal y como se mostró en la ecuación (5.10a).

Para la obtención de la matriz de realimentación del sistema ampliado, $\overline{K}$, es necesario definir las matrices Q y R. Para ello se hace uso de la regla de Bryson para obtener unos valores iniciales y después son modificados hasta que se obtiene la respuesta dinámica deseada. Cabe destacar que se ha definido el tiempo de subida como el necesario para alcanzar el 0.90 desde el 0.1 del valor final y antes de la sobreoscilación.

1. Control de altura: Para una referencia de 5 metros se desea que:

   a) Sobreoscilación $\leq 10\,\%$

   b) Tiempo de subida ≤ 2 s

En cuanto al control de orientación, se desea que su dinámica sea lo suficientemente rápida cómo para que pueda ser despreciada con la dinámica de traslación. Por ello, se utilizarán como referencias los máximos ángulos de cada uno de los ángulos de Euler admisibles:

2. Control de guiñada: Para una referencia de $\pm 180^\circ$
   a) Sobreoscilación $\leq 8\,\%$
   b) Tiempo de subida ≤ 2.5 s
3. Control de alabeo: Para una referencia de $\pm 30^\circ$
   a) Sobreoscilación $\leq 5\,\%$
   b) Tiempo de subida ≤ 0.5 s
4. Control de cabeceo: Para una referencia de $\pm 30^\circ$
   a) Sobreoscilación $\leq 5\,\%$
   b) Tiempo de subida ≤ 0.5 s

A su vez, en el diseño de las leyes de control mediante el ajuste de las matrices Q y R se busca que las velocidades de giro mandadas a cada rotor no saturen y además se deje un margen de 0.1 veces la velocidad de giro máxima. En el capítulo 7 se muestran las dimensiones y parámetros físicos, así como las máximas velocidades de giro de los rotores del quadrotor utilizado en las diferentes simulaciones del presente documento. De esta forma, en un vuelo real se deja un margen de seguridad que considere los fenómenos no modelados o se evite la saturación en maniobras que incluyan cambios de altura de vuelo y de orientación simultáneos. Puesto que las referencias a las que se ha diseñado se han considerado como las máximas que serán suministradas, las acciones de control serán también las máximas, por ello es importante asegurar que en todo momento las referencias comandadas no superan este límite para asegurar la estabilidad del sistema de control.

En la simulación utilizada para controlar la altura y orientación del quadrotor se ha omitido el bucle externo de la estructura del controlador mostrada en la figura 5.4. De esta forma, las referencias del bucle interno son suministradas manualmente y no por el bucle externo.

En las ecuaciones (5.24) se incluyen las matrices Q y R que permiten satisfacer las especificaciones dinámicas deseadas, 'diag()' es la matriz diagonal formada por el vector que contiene entre paréntesis. Es importante destacar que las limitaciones de las acciones de control se han impuesto mediante la modificación de las componentes correspondientes a los integradores, por otro lado, no se han limitado las alturas máximas, el ángulo de guiñada, la velocidad de ascenso ni las velocidades de giro en los tres ejes.

$$\begin{aligned} Q &= diag\begin{pmatrix} Q_z & Q_\phi & Q_\theta & Q_\psi & Q_w & Q_p & Q_q & Q_r & Q_{\xi_z} & Q_{\xi_\phi} & Q_{\xi_\theta} & Q_{\xi_\psi} \end{pmatrix} = \\ &= diag\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0.6 & 60 & 60 & 1\cdot 10^{-5} \end{pmatrix} \end{aligned} \tag{5.24a}$$

$$\begin{aligned} R &= diag\begin{pmatrix} R_T & R_{M_{T_x}} & R_{M_{T_y}} & R_{M_{T_z}} \end{pmatrix} = \\ &= diag\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \cdot 1 \cdot 10^{-2} \end{aligned} \tag{5.24b}$$

La matriz de realimentación del sistema aumentado que se obtiene con la definición de las matrices Q y R de la ecuación (5.24) se muestra en la ecuación (5.25). Esta matriz se encuentra compuesta por la realimentación del sistema que permite modificar sus autovalores para hacerlo estable y las ganancias del integrador que permiten conseguir un error nulo en el seguimiento de las referencias.

$$\overline{K} = \begin{bmatrix} K \mid K_i \end{bmatrix} =$$

$$\left[\begin{array}{cccccccc|cccc} 7.83 & 0 & 0 & 0 & 3.96 & 0 & 0 & 0 & -7.75 & 0 & 0 & 0 \\ 0 & 12.82 & 0 & 0 & 0 & 0.42 & 0 & 0 & 0 & -77.46 & 0 & 0 \\ 0 & 0 & 13.08 & 0 & 0 & 0 & 0.46 & 0 & 0 & 0 & -77.46 & 0 \\ 0 & 0 & 0 & 0.05 & 0 & 0 & 0 & 0.04 & 0 & 0 & 0 & -0.03 \end{array}\right] \tag{5.25}$$

En las figuras 5.7[1], 5.8[2] y 5.9[3] se muestran las respuestas del sistema ante las referencias indicadas anteriormente. Junto a las respuestas del sistema de añaden las velocidades de giro de cada rotor donde se puede asegurar que la saturación no se produce, aparte de la limitación física de saturación también se ha incluido la limitación impuesta de 0.9 veces la velocidad de giro máxima. Además, en las tablas 5.1, 5.2, 5.3 y 5.4 se adjuntan los parámetros dinámicos[4] de cada respuesta del bucle interno mostrados en cada figura, también se añade el valor de cada uno de los autovalores del sistema en bucle cerrado para cada uno de los estados internos e integradores.

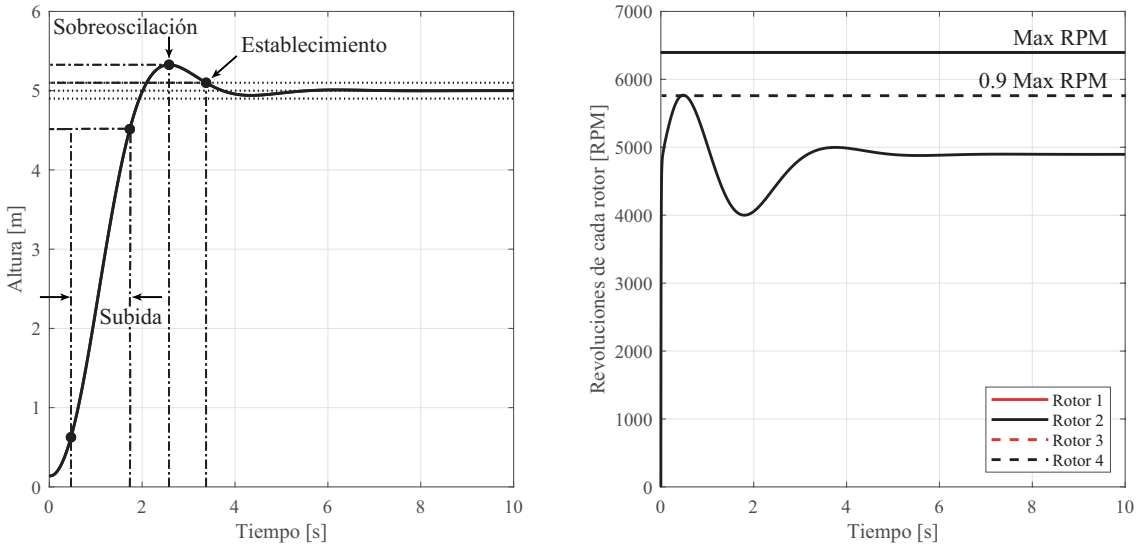


Figura 5.7: Respuesta del quadrotor ante un cambio de referencia en la altura.

Características dinámicas	
Tiempo de subida [s]	1.29
Tiempo de pico [s]	2.58
Tiempo de establecimiento [s]	3.38
Sobreoscilación [%]	6.29

Autovalores en bucle cerrado	
Estados internos (z, w)	-0.99±1.71i
Estado del integrador	-1.98±0i

Tabla 5.1: Características dinámicas y autovalores en bucle cerrado del control de altura

[1]El quadrotor comienza a una altura distinta de 0 puesto que en el instante inicial se encuentra con las patas en contacto con el suelo y la medición se produce desde el cdg del vehículo.

[2]Notar que cada referencia se ha impuesto una detrás de otra esperando que se establezca la previa antes de cambiar, por esta razón el eje 'X' empieza en el segundo 10.

[3]Debido a la simetría radial del quadrotor, la respuesta ante un cambio en la referencia de alabeo y cabeceo son casi idénticos, razón por la cual sólo se muestra una de ellas.

[4]Se ha definido el tiempo de establecimiento como el necesario para que la amplitud de oscilación de la respuesta sea menor que el 2 % de la referencia.

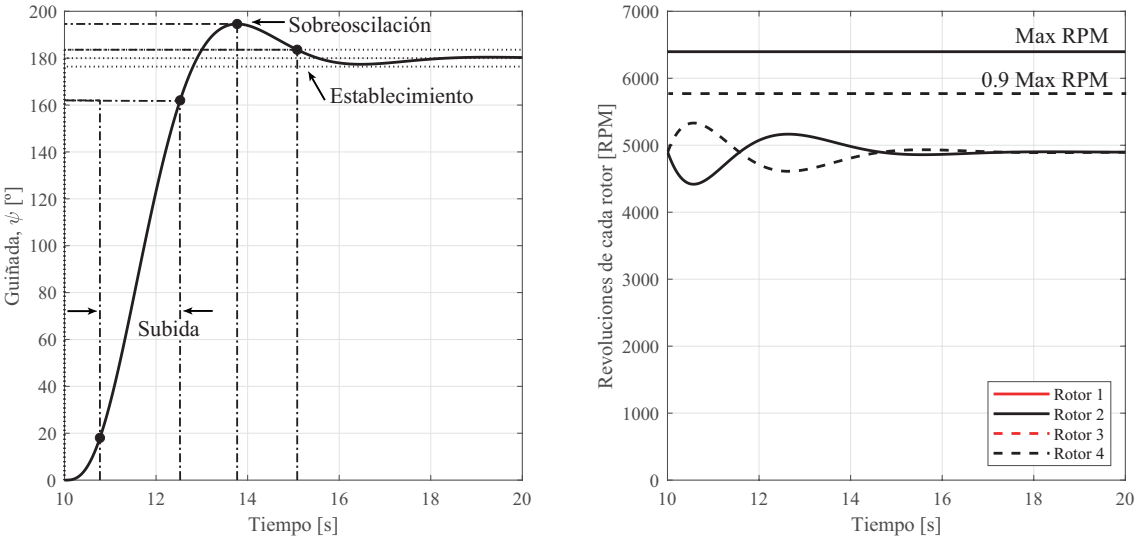


Figura 5.8: Respuesta del quadrotor ante un cambio de referencia en la guiñada.

Características dinámicas	
Tiempo de subida [s]	1.76
Tiempo de pico [s]	3.77
Tiempo de establecimiento [s]	5.08
Sobreoscilación [%]	7.35

Autovalores en bucle cerrado	
Estados internos (ψ, r)	-0.65±1.131i
Estado del integrador	-1.30±0i

Tabla 5.2: Características dinámicas y autovalores en bucle cerrado del control de guiñada

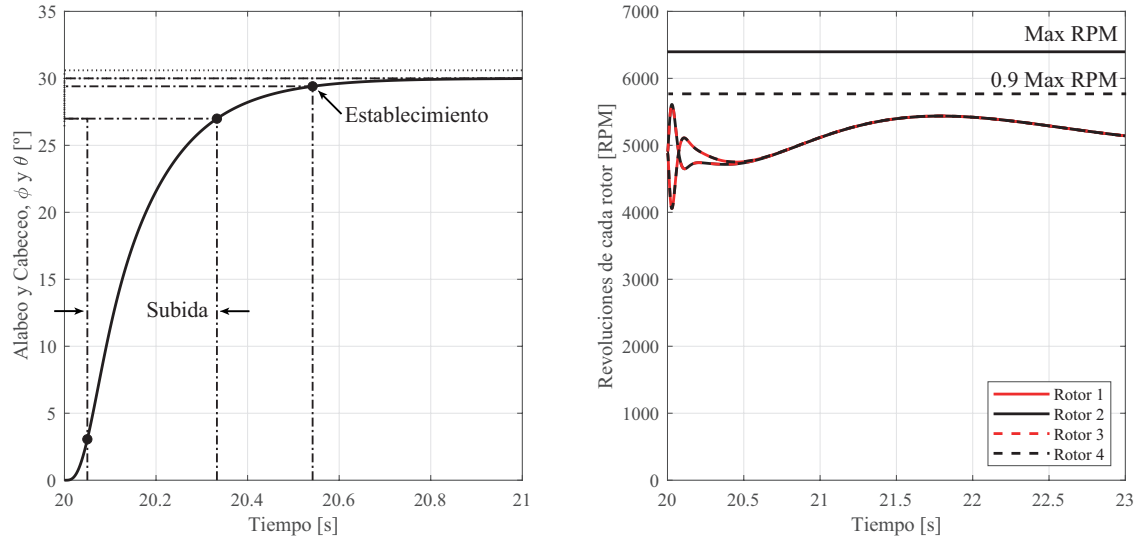


Figura 5.9: Respuesta del quadrotor ante un cambio de referencia en el alabeo o el cabeceo.

Características dinámicas	
Tiempo de subida [s]	0.28
Tiempo de establecimiento [s]	0.54
Sobreoscilación [%]	0

Autovalores en bucle cerrado	
Estados internos (ϕ,p)	-27.01±27.561i
Estado del integrador	-7.74±0i

Tabla 5.3: Características dinámicas y autovalores en bucle cerrado del control de alabeo

Características dinámicas	
Tiempo de subida [s]	0.29
Tiempo de establecimiento [s]	0.55
Sobreoscilación [%]	0

Autovalores en bucle cerrado	
Estados internos (θ, q)	-24.65±25.251i
Estado del integrador	-7.74±0i

Tabla 5.4: Características dinámicas y autovalores en bucle cerrado del control de cabeceo

Si se analiza la evolución del seguimiento de todas las referencias en su conjunto es posible observar que el seguimiento de la altura es considerablemente más lento que el de las tres orientaciones, tal y como se introdujo anteriormente.

Por otro lado, observando con detenimiento la figura 5.9 es posible ver que las revoluciones de los cuatro rotores experimentan un aumento una vez que el ángulo de alabeo o cabeceo se ha estabilizado. Cuando ϕ o θ son diferentes de cero, el vector tracción no está alineado con el eje z_G y se produce por tanto una disminución de la componente de tracción que hace al quadrotor mantener su altura, por esta razón el controlador hace aumentar la velocidad de giro de los rotores.

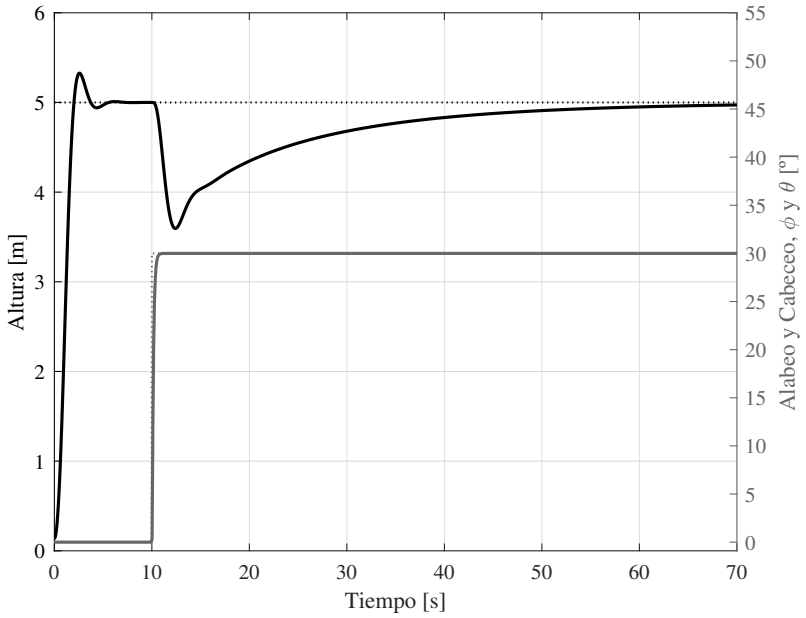


Figura 5.10: Acoplamiento de la altura con los estados de alabeo y cabeceo.

De esta forma, el estado de la altura se encuentra acoplado con la orientación del quadrotor en los ejes x_B y y_B. En la figura 5.10 se puede observar la disminución de altura debido al aumento de cabeceo o alabeo y su posterior corrección. Aunque pueda parecer que es elevada la variación de altura, en control de trayectoria con bucle externo no será tan elevado ya que los grandes ángulos solamente serán utilizados para acelerar y no de forma permanente, además, en pocas ocasiones

serán tan grandes. El rechazo de la perturbación debida al cambio de ángulo de cabeceo o alabeo es considerablemente lenta debido a que las acciones de control están limitadas para no sobrepasar la saturación.

Finalmente, en la figura 5.11 se representa la evolución de velocidades de giro de los rotores del quadrotor ante un cambio simultáneo en todas las referencias. De esta forma se puede ver cómo, en la situación más restrictiva, no se supera la máxima velocidad de giro de los rotores y no se produce la saturación.

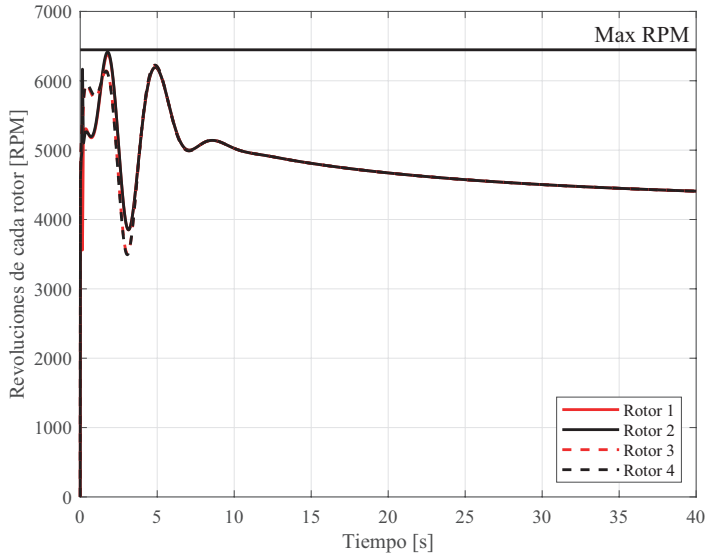


Figura 5.11: Acciones de control ante un cambio de referencia simultáneo en todas las variables controladas.

Antes de pasar a detallar el diseño del controlador del bucle externo es necesario establecer algunas pautas sobre la referencia de la guiñada que presenta alguna complejidad comparada con el alabeo y el cabeceo. Debido a que el rango de la guiñada está comprendida entre -180° y 180° se debe llevar cuidado a la hora de imponer la referencia ya que existe una discontinuidad en los valores de la guiñada. Para solucionar este problema se va a asumir que cuando el quadrotor pasa por la discontinuidad su valor de guiñada continúa aumentando en valor absoluto en vez de cambiar de signo para volver al origen, es decir, cuando el quadrotor pase por la discontinuidad, el valor de la guiñada continuará aumentando o disminuyendo de la forma en que lo hacía. Por tanto se va a asumir que la guiñada del quadrotor puede tener cualquier valor comprendido entre $-\infty$ y ∞.

Entonces, si se impone una referencia sin considerar el valor actual de guiñada del quadrotor puede pasar que el vehículo comience a girar sobre sí mismo sin control. A modo de ejemplo y para ilustrar el problema, imagínese el caso en el que el quadrotor ha dado dos vueltas sobre sí mismo en una trayectoria en espiral ascendente, entonces la guiñada actual del quadrotor es de 720°, si se desea que el quadrotor cambie de guiñada para seguir el rumbo de $+10^\circ$ y se impone directamente esa referencia, entonces el quadrotor volverá a girar las dos vueltas que había hecho anteriormente hasta igualar la referencia. Es posible ver como esto no es eficiente ya que está tomando un camino, que aunque pueda tender a la referencia, es demasiado largo. La solución a este problema es asegurar que el valor de la referencia esté comprendida entre $-180^\circ + \psi$ y $180^\circ + \psi$, siendo ψ el valor actual de la guiñada del quadrotor.

Diseño del bucle externo del regulador

En el diseño de un control de un quadrotor es obligatorio el uso del bucle interno debido a que el sistema dinámico es inestable de por sí solo, de tal forma que sería muy difícil para un usuario mediante el uso de un *joystick* controlar las acciones de control del quadrotor. Sin embargo, con la correcta implementación de una ley de control de los ángulos de Euler es posible suministrar estas referencias que serán seguidas, facilitándose en gran medida la tarea de controlar la aeronave.

En el presente documento se sustituye el usuario, que era el encargado de comandar las referencias de los ángulos de Euler, por el bucle externo que será capaz de generar estas referencias en función de dónde y cómo se encuentre el quadrotor para conseguir el seguimiento de estas referencias. Se pueden proponer dos definiciones de controladores de bucle externo, el primero de ellos es el de posición, mientras que el segundo es el de velocidad.

Antes era el usuario el que, observando al quadrotor, decidía cuál era la mejor orientación que debía de aportarle al control de bucle interno para conseguir una determinada posición o velocidad, sin embargo, esta tarea la hará el bucle externo y es el usuario el que se encarga de decidir cuál es el punto al que desea que el vehículo se mueva. Como se puede observar, la utilización del controlador del bucle externo reduce en gran medida la carga de trabajo del usuario.

En el presente documento se muestran las dos leyes de control correspondientes a las dos posibles elecciones del bucle externo:

- **Control de posición**

Para el control de posición serán necesarios los $m = 4$ estados que no eran considerados en el bucle interno y que se muestran en la ecuación (5.26a). En cuanto a los estados del bucle externo que serán controlados y considerados como las salidas serán los $n = 2$ mostrados en la ecuación (5.26b) que se corresponden con la posición del quadrotor en el plano horizontal. Finalmente, las acciones de control serán los ángulos de cabeceo y alabeo necesarios para llegar a la determinada referencia, (5.26c).

$$\vec{x}_2 = \begin{bmatrix} x & y & u & v \end{bmatrix}^T \tag{5.26a}$$

$$\vec{y}_2 = \begin{bmatrix} x & y \end{bmatrix}^T \tag{5.26b}$$

$$\vec{u}_2 = \begin{bmatrix} \phi & \theta \end{bmatrix}^T \tag{5.26c}$$

La previa definición de los estados, salidas y acciones de control también ha sido expuesta en la ecuación (5.20) donde se representó la dinámica del sistema en espacio de estados. En ella se pueden encontrar las definiciones de las matrices A_2 y B_2, sin embargo, C_2 y D_2 se pueden encontrar en la ecuación (5.27). Mediante ellas es posible determinar las salidas del sistema.

$$C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.27a}$$

$$D_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.27b}$$

A continuación, una vez que se ha definido completamente el sistema del bucle externo a controlar, se debe de establecer el sistema ampliado que permita un error de posición nulo, ecuación (5.28). A diferencia de como ocurría en el sistema ampliado del bucle interno, (5.23), para el caso del bucle externo es necesario considerar los grandes ángulos de rumbo que pueden existir y por ello

no se ha asumido la hipótesis de pequeñas variaciones obteniendo una matriz $\overline{A}_2$ dependiente del tiempo.

$$\overbrace{\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{u} \\ \dot{v} \\ \dot{\xi}_x \\ \dot{\xi}_y \end{pmatrix}}^{\dot{\vec{x}}_2} = \overbrace{\left[\begin{array}{cccc|cc} 0 & 0 & \cos(\psi) & -\sin(\psi) & 0 & 0 \\ 0 & 0 & \sin(\psi) & \cos(\psi) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{array}\right]}^{\overline{A}_2} \overbrace{\begin{pmatrix} x \\ y \\ u \\ v \\ \xi_x \\ \xi_y \end{pmatrix}}^{\vec{x}_2} + \overbrace{\left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 0 & g \\ -g & 0 \\ \hline 0 & 0 \\ 0 & 0 \end{array}\right]}^{\overline{B}_2} \overbrace{\begin{pmatrix} \phi \\ \theta \end{pmatrix}}^{\vec{u}_2} + \overbrace{\left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \hline 1 & 0 \\ 0 & 1 \end{array}\right]}^{\overline{K}_{r2}} \overbrace{\begin{pmatrix} r_x \\ r_y \end{pmatrix}}^{\vec{r}_2} \tag{5.28}$$

Para la obtención de la matriz de realimentación del sistema aumentado es necesaria la definición de las matrices Q y R. Para ello, como se ha hecho con anterioridad se aplicará la regla de Bryson y se modificarán hasta obtener la respuesta dinámica deseada.

1. Control de posición en x_G: Para una referencia de 5 metros se desea que:
   a) Sobreoscilación $\leq 10\,\%$
   b) Tiempo de subida ≤ 2 s
2. Control de posición en y_G: Para una referencia de 5 metros se desea que:
   a) Sobreoscilación $\leq 10\,\%$
   b) Tiempo de subida ≤ 2 s

Las limitaciones físicas que se encuentran en los estados y acciones de control son expuestos a continuación. La importancia reside en que, para asegurar la estabilidad del sistema no se deben superar. En cuanto a las posiciones no debe de existir ninguna limitación ya que se desea que el quadrotor sea capaz de moverse en cualquiera de ellas, de igual forma pasa para las velocidades, sin embargo, las orientaciones mandadas como referencias al bucle interno no deben de ser grandes, aunque a la salida del bucle externo se saturen a $\pm 30^\circ$ también se ha diseñado el sistema para que a referencias menores de 5 m no se superen estos ángulos. El proceso de saturación está por tanto como una medida de seguridad para asegurar que en el caso que la generación de la trayectoria falle, no se llega a la inestabilidad. También se ha asegurado que la máxima acción de control sea al menos un 15 % menor que la máxima admisible.

Para la simulación del sistema se asumirá la ley de control en su conjunto, es decir, las referencias de los ángulos de Euler calculados por el bucle externo serán suministrados al bucle interno.

En las ecuaciones (5.29) se incluyen las matrices Q y R que permiten satisfacer las especificaciones dinámicas deseadas. Es importante destacar que las limitaciones de las acciones de control se han impuesto mediante la modificación de las componentes correspondientes a los integradores, por otro lado, no se han limitado las posiciones ni velocidades y por esta razón la componente asociada a estos estados es nula.

$$\begin{aligned} Q &= diag\begin{pmatrix} Q_x & Q_y & Q_u & Q_v & Q_{\xi_x} & Q_{\xi_y} \end{pmatrix} = \\ &= diag\begin{pmatrix} 0 & 0 & 0 & 0 & 0.07 & 0.07 \end{pmatrix} \end{aligned} \tag{5.29a}$$

$$\begin{aligned} R &= diag\begin{pmatrix} R_\phi & R_\theta \end{pmatrix} = \\ &= diag\begin{pmatrix} 1 & 1 \end{pmatrix} \cdot 1 \cdot 10^{-2} \end{aligned} \tag{5.29b}$$

La matriz de realimentación del sistema aumentado que se obtiene con la definición de las matrices Q y R de la ecuación (5.29) se muestra en la ecuación (5.30). Esta matriz se encuentra compuesta por la realimentación del sistema que permite modificar sus autovalores para hacerlo estable y las ganancias del integrador que permiten conseguir un error nulo en el seguimiento de las referencias. Como se puede observar, tanto la realimentación del sistema como la ganancia del integrador dependen de la guiñada actual del sistema que a su vez depende del tiempo, de esta forma se ha conseguido extender la ley de control a grandes valores de la guiñada y no sólo cuando se encuentra orientado con el norte. El regulador diseñado, que recibe el nombre de regulador LQR con Planificación de Ganancia o 'Gain Scheduling', se constituye como una sucesión de reguladores lineales entre los cuáles se va conmutando en función de cuál sea el punto de operación del sistema en cada una de las iteraciones en las que se evalúe.

$$\overline{K} = \begin{bmatrix} K \mid K_i \end{bmatrix} = \left[\begin{array}{cccc|cc} 0.39\sin(\psi) & -0.39\cos(\psi) & 0 & -0.28 & 0.26\sin(\psi) & -0.26\cos(\psi) \\ 0.39\cos(\psi) & 0.39\sin(\psi) & 0.28 & 0 & 0.26\cos(\psi) & 0.26\sin(\psi) \end{array}\right] \tag{5.30}$$

En la figura 5.12 se muestran las respuestas del sistema ante las referencias indicadas anteriormente. Debido a la simetría radial del quadrotor, al respuesta ante un cambio en la referencia en x e y son idénticas y por ello solamente se incluye la figura de una de ellas. Junto a las respuestas del sistema de añaden los ángulos de Euler de alabeo y cabeceo mandados como referencia al bucle interno y donde se puede asegurar que no se produce un valor máximo del admisible. Además, en la tabla 5.5 se adjuntan los parámetros dinámicos de cada respuesta del bucle externo mostrados en la figura, también se añade el valor de cada uno de los autovalores del sistema en bucle cerrado para cada uno de los estados internos e integradores.

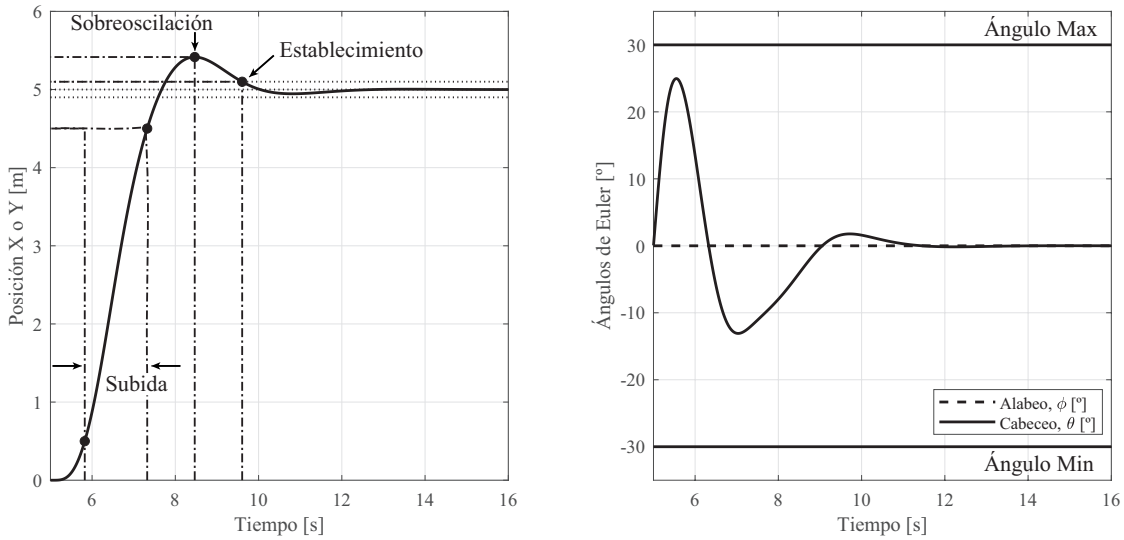


Figura 5.12: Respuesta del quadrotor ante un cambio de referencia en la posición en X o Y.

Características dinámicas	
Tiempo de subida [s]	1.5
Tiempo de pico [s]	3.47
Tiempo de establecimiento [s]	4.61
Sobreoscilación [%]	7.68

Autovalores en bucle cerrado	
Estados internos (x o y, u o v)	-0.69±1.19i
Estado del integrador	-1.38±0i

Tabla 5.5: Características dinámicas y autovalores en bucle cerrado del control de posición X o Y

Debido a que el quadrotor es un sistema subactuado, el seguimiento de las referencias de posición es considerablemente más lento que las de orientación tal y como se introdujo en el diseño del controlador del bucle interno. Esto es debido a que para conseguir un movimiento en avance es necesario en un primer lugar orientar el vector tracción en esa dirección y por esta razón se debe cambiar el ángulo de alabeo o cabeceo en función del tipo de movimiento que se desee realizar. Además cabe destacar que se ha representado las acciones de control para una posición en x positiva, en el caso de que la posición fuese en y la representación del cabeceo y alabeo estaría invertida.

- **Control de velocidad**

Para controlar la velocidad del quadrotor serán necesarios únicamente $m = 2$ estados de los que no eran considerados en el bucle interno, es decir, a diferencia del control de posición, no serán utilizados los estados x e y, ecuación (5.31a). En cuanto a las salidas del sistema o estados controlados serán estos dos únicos necesarios en el control, ecuación (5.31b). Finalmente, las acciones de control utilizadas serán las mismas que en el control de posición, $p = 2$, ecuación (5.31c).

$$\vec{x}_3 = \begin{bmatrix} u & v \end{bmatrix}^T \tag{5.31a}$$

$$\vec{y}_3 = \begin{bmatrix} u & v \end{bmatrix}^T \tag{5.31b}$$

$$\vec{u}_3 = \begin{bmatrix} \phi & \theta \end{bmatrix}^T \tag{5.31c}$$

Esta definición de los estados, salidas y acciones de control puede ser encontrada en la ecuación (5.20) donde se representaba la dinámica del bucle externo en espacio de estados. Sin embargo, se han eliminado los estados correspondientes a las posiciones ya que no son necesarios. Por ello, para el control de velocidad se redefinen las matrices A_2, B_2, C_2 y D_2, ecuación (5.32)

$$A_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.32a}$$

$$B_2 = \begin{bmatrix} 0 & g \\ -g & 0 \end{bmatrix} \tag{5.32b}$$

$$C_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.32c}$$

$$D_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.32d}$$

A continuación, una vez que se ha definido completamente el sistema del bucle externo a controlar, se debe de establecer el sistema ampliado que permita un error de posición nulo, ecuación (5.33). A diferencia de como ocurría en el sistema ampliado del bucle externo de control de posición,

(5.28), no es necesario considerar que los ángulos de rumbo pueden ser grandes ya que se está controlando la velocidad en el sistema de referencia cuerpo.

$$\overbrace{\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{\xi}_u \\ \dot{\xi}_v \end{pmatrix}}^{\dot{\vec{x}}_2} = \overbrace{\left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{array}\right]}^{\overline{A}_2} \overbrace{\begin{pmatrix} u \\ v \\ \xi_u \\ \xi_v \end{pmatrix}}^{\vec{x}_2} + \\ + \overbrace{\left[\begin{array}{cc} 0 & g \\ -g & 0 \\ \hline 0 & 0 \\ 0 & 0 \end{array}\right]}^{\overline{B}_2} \overbrace{\begin{pmatrix} \phi \\ \theta \end{pmatrix}}^{\vec{u}_2} + \overbrace{\left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ \hline 1 & 0 \\ 0 & 1 \end{array}\right]}^{\overline{K}_{r2}} \overbrace{\begin{pmatrix} r_u \\ r_v \end{pmatrix}}^{\vec{r}_2} \tag{5.33}$$

Para la obtención de la matriz de realimentación del sistema aumentado es necesaria la definición de las matrices Q y R. Para ello, como se ha hecho con anterioridad se aplicará la regla de Bryson y se modificarán hasta obtener la respuesta dinámica deseada.

1. Control de velocidad en x_B: Para una referencia de 10 m/s se desea que:
   a) Sobreoscilación $\leq 10\,\%$
   b) Tiempo de subida ≤ 2 s
2. Control de posición en y_B: Para una referencia de 10 m/s se desea que:
   a) Sobreoscilación $\leq 10\,\%$
   b) Tiempo de subida ≤ 2 s

En las ecuaciones (5.34) se adjuntan las matrices Q y R que permiten satisfacer las especificaciones dinámicas. Es importante destacar que las limitaciones de las acciones de control se han impuesto mediante la modificación de las componentes correspondientes a los integradores, para ello se ha considerado que el máximo ángulo de alabeo o cabeceo se al menos 0.15 veces menor que la máxima admisible. Las limitaciones de los estados de las velocidades se han considerado nulas tal y como se puede observar. Para simular la respuesta del sistema ante la ley de control de velocidad se han considerado los dos bloques, interno y externo, de igual forma que se hizo para el control de posición.

$$\begin{aligned} Q &= diag\begin{pmatrix} Q_u & Q_v & Q_{\xi_u} & Q_{\xi_v} \end{pmatrix} = \\ &= diag\begin{pmatrix} 0 & 0 & 0.004 & 0.004 \end{pmatrix} \end{aligned} \tag{5.34a}$$

$$\begin{aligned} R &= diag\begin{pmatrix} R_\phi & R_\theta \end{pmatrix} = \\ &= diag\begin{pmatrix} 1 & 1 \end{pmatrix} \end{aligned} \tag{5.34b}$$

La matriz de realimentación del sistema aumentado que se obtiene con la definición de las matrices Q y R de la ecuación (5.34) se muestra en la ecuación (5.35). A diferencia del caso de control de velocidad, ests matriz de realimentación del sistema aumentado es constante en el tiempo y con los estados internos del sistema.

$$\overline{K} = \left[\begin{array}{c|c} K & K_i \end{array}\right] = \left[\begin{array}{cc|cc} 0 & -0.11 & 0 & 0.06 \\ 0.11 & 0 & -0.06 & 0 \end{array}\right] \tag{5.35}$$

En la figura 5.13 se muestran las respuestas del sistema ante las referencias indicadas anteriormente. Debido a la simetría radial del quadrotor, al respuesta ante un cambio en la referencia en u y v son idénticas y por ello solamente se incluye la figura de una de ellas. Junto a las respuestas del sistema de añaden los ángulos de Euler de alabeo y cabeceo mandados como referencia al bucle interno y donde se puede asegurar que no se produce un valor máximo del admisible. Además, en la tabla 5.6 se adjuntan los parámetros dinámicos de cada respuesta del bucle externo mostrados en la figura, también se añade el valor de cada uno de los autovalores del sistema en bucle cerrado para cada uno de los estados internos e integradores.

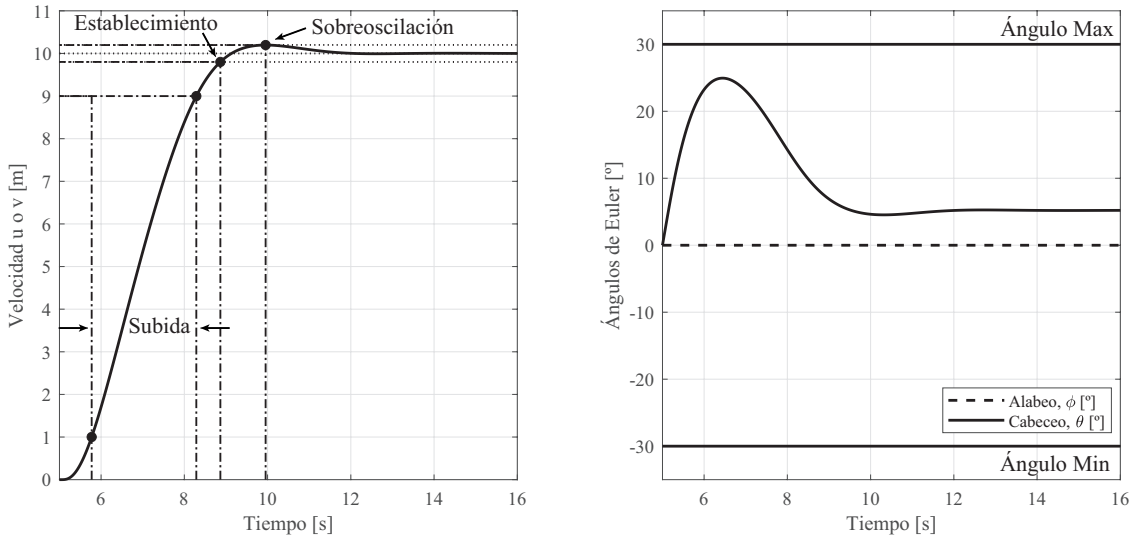


Figura 5.13: Respuesta del quadrotor ante un cambio de referencia en la velocidad en u o v.

Características dinámicas	
Tiempo de subida [s]	2.51
Tiempo de pico [s]	4.95
Tiempo de establecimiento [s]	3.87
Sobreoscilación [%]	1.93

Autovalores en bucle cerrado	
Estados internos (x o y, u o v)	-0.56±0.56i
Estado del integrador	-0.56±0.56i

Tabla 5.6: Características dinámicas y autovalores en bucle cerrado del control de velocidad u o v

En la figura 5.13 se puede observar como el ángulo de cabeceo necesario para mantener una velocidad constante es no nulo debido a la presencia de la resistencia aerodinámica. Además cabe destacar que se ha representado las acciones de control para una velocidad en u positiva, en el caso de que la velocidad fuese en v la representación del cabeceo y alabeo estaría invertida.

5.3 Filtro de Kalman extendido

Para la utilización de control mediante realimentación de estados se asumió que todos los estados internos del sistema eran accesibles y se podía obtener una medición de ellos exacta y sin ruido. Sin embargo, en la práctica es difícil tener sensores que sean capaces de medir todos y cada uno de ellos, por ello, es necesario estimar aquellos estados que no se pueden medir a partir de los que sí son accesibles, este proceso se llama observación y se consigue mediante la utilización de un observador. Además, en función de si se observan todos los estados del sistema o únicamente

aquellos que no están disponibles, el observador se podrá clasificar como un observador de orden completo o de orden reducido.

Para la estimación de los estados del sistema, un observador utilizará tanto las salidas o estados medibles del sistema y las acciones de control suministradas. Para determinar si es posible realizar el proceso de observación se utilizará la matriz de observabilidad descrita en la subsección 5.3.1

El observador que será utilizado en el presente documento se corresponde con el filtro de Kalman, más concretamente el filtro de Kalman extendido. Cuando se habla de teoría de control, el filtro de Kalman, también conocido como LQE, es un algoritmo que utiliza una serie de mediciones procedentes de los sensores del sistema y que contienen perturbaciones y ruido blanco de tal forma que se obtiene una estimación más precisa, incluyendo aquellos estados que no pueden ser medidos. Lo errores de medición y de proceso que aparecen en el sistema son asumidos como Gausianos, es decir, con media nula y covarianza conocida. El funcionamiento del algoritmo está dividido en dos fases, en la primer de ellas, llamada predicción, se produce una primera estimación de los estados internos del sistema, junto con sus incertidumbres. Posteriormente, en el paso de corrección, la primer estimación es actualizada mediante una media ponderada entre la estimación y la medición, dándole un mayor peso a aquellos valores que tienen una mayor certidumbre.

La implementación de este algoritmo es discreta y recursiva, pudiendo ser utilizado en tiempo real utilizando únicamente las acciones de control suministradas y las matrices de estado e incertidumbre que han sido calculadas en la iteración anterior. De esta forma no es necesaria más información de previas iteraciones. Es importante considerar que la dinámica del sistema sobre la que se produzca el proceso de observación mediante el filtro de Kalman debe de ser considerada como lineal, o en caso contrario como lineal en torno a un punto de equilibrio cuyas variaciones deben ser pequeñas. Existen diferentes alternativas que permiten utilizar el filtro de Kalman en sistemas no lineales como son el ya mencionado filtro de Kalman extendido o el filtro de Kalman *unscented*.

$$\dot{\vec{x}}(t) = f(t, \vec{x}(t), \vec{u}(t)) + \frac{d}{dt} w_k(t) \tag{5.36a}$$

$$\vec{y}(t) = h(t, \vec{x}(t), \vec{u}(t)) + v_k(t) \tag{5.36b}$$

El filtro de Kalman extendido (EKF), en cada una de las iteraciones, linealiza las ecuaciones dinámicas del sistema en torno a la estimación de los estados de la iteración previa, de esta forma se consigue extender el algoritmo a sistemas no lineales que pueden llegar a operar en distintos puntos de equilibrio, alejados unos de otros y donde la hipótesis de pequeñas perturbaciones dejaría de ser válida.

En las ecuaciones (5.36) se muestra el sistema dinámico no lineal del quadrotor donde todas las variables han sido introducidas excepto, w_k y v_k que son el ruido de proceso y medida con media cero y covarianza Q y R respectivamente, es decir, en el filtro de Kalman se asume que las perturbaciones son ruido blanco. Esta suposición, junto con que la dinámica del sistema sea lineal, es necesaria para que el estimador sea óptimo, [31].

$$w_{[m\times 1]}(t) = N(0, Q) \tag{5.37a}$$

$$v_{[p\times 1]}(t) = N(0, R) \tag{5.37b}$$

$$Q_{ij} = cov(x_i, x_j) \tag{5.37c}$$

$$R_{ij} = cov(u_i, u_j) \tag{5.37d}$$

Las dimensiones de w_k son iguales que la del vector de estados internos del sistema, $m \times 1$, mientras que las dimensiones de v_k es igual al la del vector de salidas del sistema, $p \times 1$. Además, en el

desarrollo se permite que cada una de las componentes de cada uno de los vectores de ruido tengan diferente covarianza, entonces la perturbación de cada uno de los estados podrá tener diferente amplitud. Por tanto, se define Q como la matriz de dimensiones $m \times m$, donde cada componente se corresponde con la covarianza de la perturbación de cada uno de los estados internos del sistema. La definición de R es análoga a Q pero para las salidas del sistema. Estas definiciones se muestran en la ecuación (5.36), donde 'i' y 'j' hacen referencia a las coordenadas de cada componente dentro de la matriz. Por otro lado, el filtro de Kalman asume que el ruido de medida y proceso no están correlacionados, [32].

Por otro lado, en las ecuaciones (5.38) se muestra el proceso de linealización del sistema dinámico del quadrotor entorno a la estimación de la iteración anterior, para ello se hace uso del Jacobiano. Se ha asumido que C es constante y que se calcula como una combinación lineal de los estados, por otro lado, D es nula. Cabe indicar que el subíndice 'k' hace referencia a la iteración actual, mientras que 'k-1|k-1' junto con el signo '^' hace referencia a la estimación de los estados internos de la iteración anterior. La evaluación de las acciones de control se hace en la iteración de evaluación del Jacobiano, tal y como se puede observar.

$$A_k = \left[\begin{matrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_m} \end{matrix}\right]\Bigg|_{\hat{\vec{x}}_{k-1|k-1},\vec{u}_k} \qquad B_k = \left[\begin{matrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_n} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial u_1} & \frac{\partial f_m}{\partial u_2} & \cdots & \frac{\partial f_m}{\partial u_n} \end{matrix}\right]\Bigg|_{\hat{\vec{x}}_{k-1|k-1},\vec{u}_k} \tag{5.38a}$$

$$C_k = C = \left[\begin{matrix} \frac{\partial h_1}{\partial h_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_m} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_p}{\partial x_1} & \frac{\partial h_p}{\partial x_2} & \cdots & \frac{\partial h_p}{\partial x_m} \end{matrix}\right]\Bigg|_{\hat{\vec{x}}_{k-1|k-1},\vec{u}_k} \qquad D_k = 0_{[p\times n]} = \left[\begin{matrix} \frac{\partial h_1}{\partial u_1} & \frac{\partial h_1}{\partial u_2} & \cdots & \frac{\partial h_1}{\partial u_n} \\ \frac{\partial h_2}{\partial u_1} & \frac{\partial h_2}{\partial u_2} & \cdots & \frac{\partial h_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_p}{\partial u_1} & \frac{\partial h_p}{\partial u_2} & \cdots & \frac{\partial h_p}{\partial u_n} \end{matrix}\right]\Bigg|_{\hat{\vec{x}}_{k-1|k-1},\vec{u}_k} \tag{5.38b}$$

A continuación se describen cada uno de los pasos que deben ser considerados en el filtro de Kalman extendido. Es importante considerar que la implementación que se ha hecho en el presente documento es híbrida, es decir, parte del algoritmo estará expresada en el dominio continuo (predicción) y la otra parte en el dominio discreto (corrección). La principal razón por la que se ha hecho esta distinción es debido a que el sistema físico está expresado en el dominio continuo mientras que las mediciones que se realizan de las salidas se hacen en un dominio discreto, de esta forma se puede aplicar el algoritmo del filtro de Kalman extendido sin la necesidad de convertir el sistema dinámico del quadrotor al dominio discreto.

1. Predicción

   a) Predicción de estimación de estados:

$$\dot{\hat{\vec{x}}}(t) = f\left(\hat{\vec{x}}(t), u(t)\right)$$

$$\hat{\vec{x}}_{k|k-1} = \hat{\vec{x}}(t_k)$$

   Con la estimación obtenida en la iteración anterior con el filtro de Kalman, $\hat{\vec{x}}_{k-1|k-1}$, y las acciones de control actuales que se suministran al sistema, u_k, es posible predecir cuáles serán los estados en la iteración siguiente. Para ello se utiliza el modelo no lineal de la dinámica del quadrotor donde no se consideran ni las perturbaciones, ni el

ruido de proceso o medida, ni las fuerzas aerodinámicas o reacciones procedentes del suelo que van a ser consideradas como perturbaciones. Por tanto, en una determinada iteración k, se debería sustituir $\hat{\vec{x}}(t)$ por $\hat{\vec{x}}_{k-1|k-1}$ y $u(t)$ por u_k.

El proceso seguido se corresponde con el cálculo de la derivada temporal de los estados internos del sistema y su posterior integración durante el intervalo con el que se evalúa el algoritmo de Kalman. La variable $\hat{\vec{x}}_{k|k-1}$ hace referencia a la predicción de la estimación del filtro de Kalman en la iteración k.

b) Predicción de la estimación de la covarianza:

$$\dot{P}(t) = A(t)P(t) + P(t)A(t)^T + Q(t)$$

$$P_{k|k-1} = P(t_k)$$

Para la estimación de la predicción de la estimación de la covarianza, $P_{k|k-1}$, en una determinada iteración del algoritmo k, es necesaria la estimación de la iteración anterior, $P_{k-1|k-1}$. Además son necesarias las matrices del sistema, A_k, linealizadas en la estimación de la iteración anterior del estado del sistema, tal y como se muestra en la ecuación (5.38). También se debe hacer uso de la covarianza del ruido de proceso en la iteración de evaluación, Q_k. Por tanto, $A(t)$ es A_k, $P(t)$ es $P_{k|k-1}$ y $Q(t)$ es Q_k.

El proceso es similar a la predicción de la estimación del estado, ya que se debe evaluar en primer lugar la derivada temporal para luego integrar durante el intervalo con el que se evalúa el algoritmo de Kalman. Las dimensiones de la matriz de estimación de covarianza es $m \times m$.

2. Corrección

a) Ganancia óptima de Kalman:

$$K_k = P_{k|k-1}C^T \left(CP_{k|k-1}C^T + R_k\right)^{-1}$$

La ganancia óptima del filtro de Kalman se utiliza para determinar a que variable dar más peso en la corrección de la estimación, si a la predicción de la estimación de estados o a la medición, en función de la incertidumbre de ellas. Cuanto mayor es K_k, más fiable es la medición y viceversa, siendo su valor comprendido entre 0 y 1. Si se observa la expresión de la ganancia es posible concluir que cuando la covarianza del ruido de medida R_k es elevada, K_k tendrá un valor pequeño puesto que las medidas no serán tan confiables dándole un mayor peso a las estimaciones mediante el sistema dinámico. Las dimensiones de la ganancia de Kalman son $m \times p$.

b) Corrección de la estimación de estados:

$$\hat{\vec{x}}_{k|k} = \hat{\vec{x}}_{k|k-1} + K_k\left(y_k - C\hat{\vec{x}}_{k|k-1}\right)$$

En función de la ganancia óptima de Kalman que le da más peso a la estimación o a la medición en función de la incertidumbre que tengan se corrigen las estimaciones de los estados. Esta estimación será la utilizada en la siguiente iteración para predecir el estado del sistema.

c) Corrección de la estimación de la covarianza:

$$P_{k|k} = (I - K_k C)P_{k|k-1}$$

De igual forma que la corrección de los estados, en función de la ganacia óptima de Kalman se corrige la estimación de la covarianza. Esta estimación será la utilizada en la siguiente iteración en la parte de predicción.

A la hora de implementar el filtro de Kalman extendido es importante proporcionar una buena estimación inicial de los estados del sistema, x_0, junto con la estimación de la covarianza inicial, P_0, ya que en caso contrario el algoritmo puede llegar a divergir. Por otro lado, en el caso que la dinámica del sistema esté modelada incorrectamente, el algoritmo también puede llegar a ser inestable. Si se consideran estos inconvenientes, la implementación del filtro de Kalman puede llegar a proporcionar estimaciones lo suficientemente buenas con las perturbaciones y ruidos filtrados.

La matriz inicial de covarianza, P_0, debe ser estimada en función del error de inicialización de los estados internos del sistema, es decir, si estos estados iniciales están alejados de los reales, se debe imponer que tenga valores elevados para considerar este error. A través de las iteraciones del algoritmo se irá disminuyendo poco a poco este error y la matriz de covarianza. Si la inicialización de los estados es próxima a los reales y hay certidumbre de ello, se podrá asumir un valor de P_0 pequeño.

Por otro lado, en sistemas con un alto grado de no linealidad, el filtro de Kalman extendido puede llegar a no dar unos buenos resultados. Para este tipo de sistemas se recomienda el uso del filtro de Kalman *unscented*.

5.3.1. Observabilidad del sistema

Un sistema se define observable en el instante de tiempo t_0 si es posible determinar los estados de un sistema, $x(t_0)$, a partir de la observación de la salida del sistema, $y(t_0)$, en un tiempo finito de tiempo, [25]. Un sistema deja de ser observable cuando los valores de alguno de los estados internos no pueden ser determinados mediante las salidas del sistema que son medidos mediante los sensores.

Para que un sistema sea observable es necesario que el rango de la matriz expresada en la ecuación (5.39) sea máximo, en caso contrario el sistema será incontrolable. La definición que se tomará en este documento de la matriz de observabilidad se hará utilizando las matrices del sistema simple y permitirá determinar que estados internos necesitan ser medidos directamente con los sensores y cuáles pueden ser estimados a partir de ellos.

$$OBS = \begin{bmatrix} C & CA & CA^2 & \dots & CA^{m-1} \end{bmatrix}^T \tag{5.39}$$

5.3.2. Implementación del observador de estados

La implementación del filtro de Kalman extendido se hará, de igual forma que el modelo dinámico del sistema y el controlador, mediante diagama de bloques de Simulink. La estructura que se seguirá será la creación de un bloque cuyas entradas sean las acciones de control del sistema y las mediciones de los sensores y la salida será el vector de estados estimado del quadrotor. De esta forma, la salida del observador de estados será la entrada a los dos bucles de control del sistema que utilizarán las estimaciones proporcionadas para evaluar la ley de control y calcular las acciones de control necesarias para conseguir un determinado seguimiento de referencias y rechazo de perturbaciones. A su vez, al sistema se le añadirán los diferentes ruidos de proceso y medición que serán filtrados por el observador de estados en la estimación de estados.

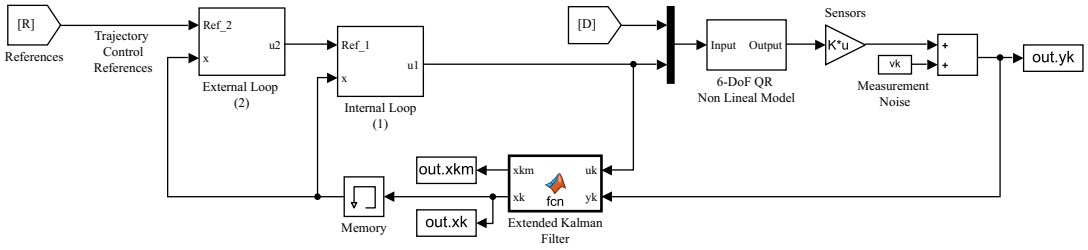


Figura 5.14: Implementación del Filtro de Kalman Extendido (EKF) en la estructura del controlador.

En la figura 5.14 se muestra la implementación del filtro dentro del entorno de control junto con la adición de los dos ruidos, de proceso y medida. Cabe destacar que el ruido de proceso se añadió dentro del bloque no lineal que simula la dinámica del sistema, tal y como se mostró en la figura 4.9. A la salida del bloque que se encarga de simular la dinámica del quadrotor, '6-DoF QR Non Lineal Model', donde se obtendría el vector de estados del sistema, se añade el bloque de sensores. La aplicación de los sensores se ha hecho de tal forma que se indica que estados son accesibles y se pueden medir directamente. En la ecuación (5.40) se muestra la operación matricial realizada, donde $\vec{y}$ son las p salidas del sistema, es decir, los estados que pueden ser medidos por los sensores y $\vec{x}$ es el vector de estado que contiene los m estados internos del sistema, por otro lado, se hace uso de la matriz de salidas C_k para pasar de los estados a las salidas de los sensores.

$$\vec{y} = C_k \vec{x} \tag{5.40}$$

Los sensores que usualmente un quadrotor tiene abordo son una IMU o *Inertial Measurement Unit*, magnetómetro, sensor de ultrasonidos y GPS. Mediante el uso del acelerómetro y del giroscopio de la IMU del quadrotor se pueden estimar dos de los ángulos de Euler que serían el alabeo y el cabeceo, el tercero de ellos, la guiñada, puede ser medida mediante el uso del magnetómetro. Por otro lado, mediante el uso del GPS se puede medir la posición del quadrotor en el plano horizontal, es decir, 'X' e 'Y', la altura puede ser medida mediante el uso del sensor de ultrasonidos abordo de vehículo. En la referencia [32] se muestra la medición de la orientación del sistema a partir de las mediciones del acelerómetro y giroscopio. En el presente documento se va a asumir que las estimaciones de la posición y orientación ya se han producido y por tanto se pueden medir directamente del sistema con sus correspondientes ruidos de medición.

Por ello, en la ecuación (5.41) se muestra la expresión de la matriz C_k que permite seleccionar los estados que sí pueden ser medidos. Si se comprueba el rango de la matriz de observabilidad del sistema dinámico del quadorotor es posible observar que es máximo y por tanto es observable. Puesto que esta matriz depende de los sensores que monte el vehículo y estos suelen ser genéricos para la mayoría de ellos, y además no depende del tipo de control a realizar (posición, velocidad, orientación), se podrá utilizar la misma implementación del filtro de Kalman en cada uno de los controles que se desee realizar.

$$C_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.41}$$

Aunque es evidente que para el control de orientación no son necesarias las mediciones del GPS y un quadrotor que solamente fuese controlado por esta ley podría volar sin problema sin este sensor, se va a asumir que se dispone de él para estimar todos los estados del sistema y luego en el bucle de control se separarán aquellos estados que sí sean necesarios.

Una vez que se han seleccionado aquellos estados que son medidos por los sensores, se le añade el ruido de medida que, como se ha indicado anteriormente, tiene media nula y covarianza conocida, es decir, se está asumiendo que las medidas no tienen ningún *bias* o que es lo suficientemente pequeño como para poder ser despreciado con el ruido Gaussiano.

El algoritmo de Kalman, como cualquier algoritmo discreto, es ejecutado cada cierto intervalo de tiempo, por ello se ha definido el periodo de muestreo como T. Cuanto menor sea el periodo, mayor será la frecuencia con la que se estimen los estados del sistema y más precisas serán las estimaciones. Sin embargo, no se puede reducir todo lo que se desee ya que aumenta el coste computacional y además puede llegar el momento en que el tiempo de ejecución sea mayor que el de muestreo. Se debe buscar un balance entre coste computacional y precisión de la estimación.

Puesto que se ha implementado el filtro de Kalman extendido mediante un bloque 'Matlab Function' de Simulink, es necesario indicar que el método de actualización es discreto y además indicarle cuál será el período utilizado, T. De esta forma solamente se actualizarán las salidas estimaciones cada T segundos.

Es importante destacar que las salidas del observador de estados se corresponden con los estados estimados en la predicción, *xkm*, y en la corrección, *xk*, haciéndose uso de los segundos para completar la realimentación de estados a los dos bucles de control. Finalmente, antes de realimentar los estados estimados, es necesario pasarlos por el bloque de Simulink 'Memory'. Este bloque es necesario para inicializar la salida al estado estimado de inicialización.

Las covarianzas del ruido de medida y proceso seleccionados se muestran en la ecuación (5.42)

$$Q = diag\,(10,10,10,0.1,0.1,0.1,100,100,100,1,1,1)^T \cdot 10^{-6} \tag{5.42a}$$

$$R = diag\,(3,3,3,0.1,0.1,0.1)^T \cdot 10^{-2} \tag{5.42b}$$

$$\tag{5.42c}$$

Finalmente, en las figuras 5.15 y 5.16 se representan las mediciones de los estados accesibles con el ruido de medida y proceso asociado junto con las estimaciones del filtro de Kalman extendido. Se está representando un cambio en la referencia de la posición de 5 m en los tres ejes de forma sucesiva.

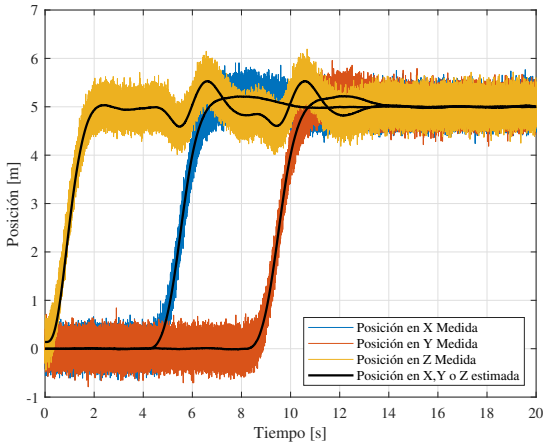


Figura 5.15: Posición medida y estimada por el filtro de Kalman ante cambios de referencia sucesivos en los tres ejes

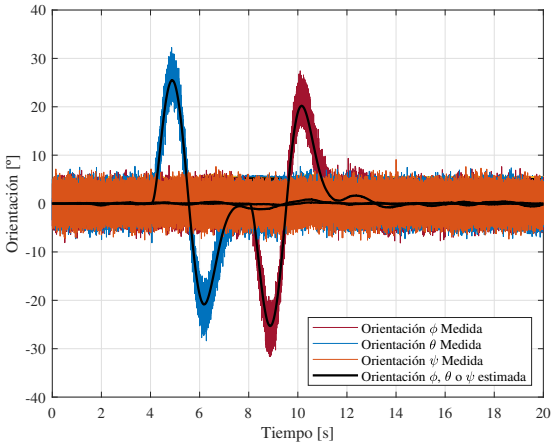


Figura 5.16: Orientación medida y estimada por el filtro de Kalman ante cambios de referencia sucesivos en los tres ejes

El observador de estados de Kalman ampliado es capaz de filtrar las mediciones ruidosas de los sensores y conseguir una estimación de los estados medidos que permiten ser usados en el control

del quadrotor mediante la estrategia de control LQR. Por otro lado, en las figuras 5.17 y 5.18 se puede observar como el filtro de Kalman es además capaz de estimar aquellos estados que no eran accesibles a partir de las medicione ruidosas de los que si que lo eran.

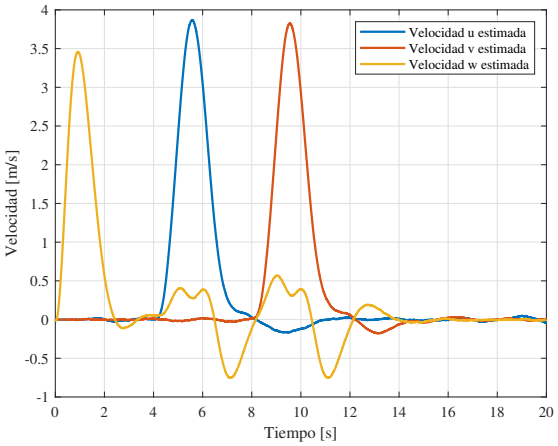


Figura 5.17: Velocidad lineal estimada por el filtro de Kalman a partir de las mediciones de los estados accesibles

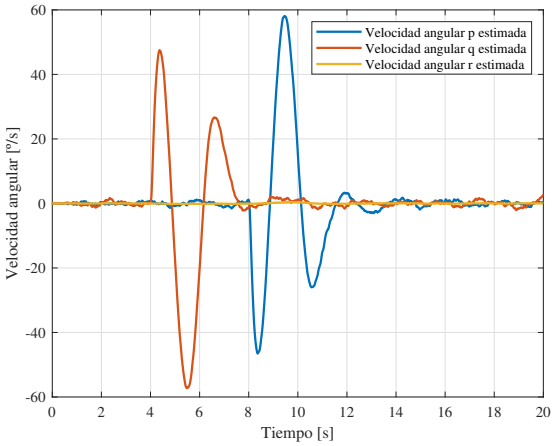


Figura 5.18: Velocidad angular estimada por el filtro de Kalman a partir de las mediciones de los estados accesibles

Código de bloque de Simulink 'Matlab Function'

A continuación se muestra el algoritmo implementado en el bloque de Simulink que permite estimar los estados del sistema a partir de mediciones ruidosas y de sus acciones de control.

En un primer lugar se definen las variables que van a ser utilizadas de previas iteraciones que son las estimaciones de estados y covarianza. En caso de que sea la primera iteración se inicializarán con los valores deseados y guardados en la 'Workspace' de Matlab.

Posteriormente, a partir de las acciones de control y estimación corregida de estados de la anterior iteración se predice la estimación de los estados. Para ello se hace uso del método numérico de integración de Euler que permite obtener buenos resultados con un intervalo de integración realtivamente bajo como es el asumido y con un coste computacional extremadamente bajo. Posteriormente, con las estimaciones corregidas de la previa iteración se estima la matriz del sistema A_k y junto con la corrección de la covarianza anterior se predice la estimación de la actual. Para ello se hace uso de un proceso de integración mediante Euler.

Finalmente, se calcula la ganacia óptima de Kalman que permite corregir las estimaciones de los estados y covarianzas. Esta parte de corrección se realiza en el dominio discreto a diferencia de la predcción que se hizo en dominio continuo.

```
1  function [xkm,xk]= fcn(uk,yk,...
2      Izz,Iyy,Ixx,m,g,invMixer,Mixer,max_omega,u0,x0,Pk0,Ck,Qk,Rk,T)
3
4  % Persistent variables over iterations
5  % To save state and covariance estimates from previous iteration
6  persistent xkint Pk
7
8  % State Estimate Initialization
9  if isempty(xkint)
10   xkint = x0;
11 end
12
13 % Covariance Estimate Initialization
```

```
14 if isempty(Pk)
15  Pk = Pk0;
16 end
17
18 %% PREDICTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19
20
21 % 1) STATE ESTIMATION PREDICTION
22
23 % Actual Internal States
24 % x    = xkint(1);  y     = xkint(2);  z    = xkint(3);
25   phi  = xkint(4);  theta = xkint(5);  psi  = xkint(6);
26   u    = xkint(7);  v     = xkint(8);  w    = xkint(9);
27   p    = xkint(10); q     = xkint(11); r    = xkint(12);
28
29 % Actual Control Actions
30 uk = Mixer*(uk+u0);
31 uk(uk>max_omega^2) = max_omega^2; uk(uk<1e-3) = 0;
32 uk = sqrt(uk);
33 uk = (invMixer*uk.^2);
34
35 Tr = uk(1); MxT = uk(2); MyT = uk(3); MzT = uk(4);
36
37 % Internal States Time Derivatives
38 xDot = (u.*cos(theta).*cos(psi)+v.*(cos(psi).*sin(theta).*...
39     sin(phi)+(-1).*cos(phi).*sin(psi))+w.*(cos(phi).*cos(psi).*...
40     sin(theta)+sin(phi).*sin(psi)));
41 yDot = (u.*cos(theta).*sin(psi)+w.*((-1).*cos(psi).*sin(phi)+...
42     cos(phi).*sin(theta).*sin(psi))+v.*(cos(phi).*cos(psi)+sin(theta).*...
43     sin(phi).*sin(psi)));
44 zDot = (w.*cos(theta).*cos(phi)+(-1).*u.*sin(theta)+v.*cos(theta).*...
45     sin(phi));
46
47 phiDot   = (p+r.*cos(phi).*tan(theta)+q.*sin(phi).*tan(theta));
48 thetaDot = (q.*cos(phi)+(-1).*r.*sin(phi));
49 psiDot   = (r.*cos(phi).*sec(theta)+q.*sec(theta).*sin(phi));
50
51 uDot = (g.*sin(theta)+r.*v+(-1).*q.*w);
52 vDot = (-g.*cos(theta).*sin(phi)+(-1).*r.*u+p.*w);
53 wDot = (-g.*cos(theta).*cos(phi)+Tr.*m.^(-1)+q.*u+(-1).*p.*v);
54
55 pDot = (Ixx.^(-1).*(Iyy.*q.*r+(-1).*Izz.*q.*r+MxT));
56 qDot = (Iyy.^(-1).*((-1).*Ixx.*p.*r+Izz.*p.*r+MyT));
57 rDot = (Izz.^(-1).*(Ixx.*p.*q+(-1).*Iyy.*p.*q+MzT));
58
59 xkmDot = [xDot;yDot;zDot;phiDot;thetaDot;psiDot;uDot;vDot;wDot;...
60     pDot;qDot;rDot];
61
62 % Integration process
63 xkm = xkint+xkmDot*T;
64
65
66 % 2) COVARIANCE ESTIMATE PREDICTION
67 % System Matrix at the given estimate
68 Ak = ...
69 [0,0,0,...
70     w.*((-1).*cos(psi).*sin(theta).*sin(phi)+cos(phi).*sin(psi))+...
71     v.*(cos(phi).*cos(psi).*sin(theta)+sin(phi).*sin(psi)),...
72     w.*cos(theta).*cos(phi).*cos(psi)+(-1).*u.*cos(psi).*sin(theta)+...
73     v.*cos(theta).*cos(psi).*sin(phi),...
74     (-1).*u.*cos(theta).*sin(psi)+...
75     w.*(cos(psi).*sin(phi)+(-1).*cos(phi).*sin(theta).*sin(psi))+...
76     v.*((-1).*cos(phi).*cos(psi)+(-1).*sin(theta).*sin(phi).*sin(psi)),...
```

```
      cos(theta).*cos(psi),...
      cos(psi).*sin(theta).*sin(phi)+(-1).*cos(phi).*sin(psi),...
      cos(phi).*cos(psi).*sin(theta)+sin(phi).*sin(psi),0,0,0;...
   0,0,0,...
      v.*((-1).*cos(psi).*sin(phi)+cos(phi).*sin(theta).*sin(psi))+...
      w.*((-1).*cos(phi).*cos(psi)+(-1).*sin(theta).*sin(phi).*sin(psi)),...
      w.*cos(theta).*cos(phi).*sin(psi)+(-1).*u.*sin(theta).*sin(psi)+...
      v.*cos(theta).*sin(phi).*sin(psi),...
      u.*cos(theta).*cos(psi)+...
      v.*(cos(psi).*sin(theta).*sin(phi)+(-1).*cos(phi).*sin(psi))+...
      w.*(cos(phi).*cos(psi).*sin(theta)+sin(phi).*sin(psi)),...
      cos(theta).*sin(psi),...
      cos(phi).*cos(psi)+sin(theta).*sin(phi).*sin(psi),...
      (-1).*cos(psi).*sin(phi)+cos(phi).*sin(theta).*sin(psi),0,0,0;...
    0,0,0,...
      v.*cos(theta).*cos(phi)+(-1).*w.*cos(theta).*sin(phi),...
      (-1).*u.*cos(theta)+(-1).*w.*cos(phi).*sin(theta)+(-1).* ...
      v.*sin(theta).*sin(phi),...
      0,(-1).*sin(theta),cos(theta).*sin(phi),cos(theta).*cos(phi),0,0,0;...
    0,0,0,...
      q.*cos(phi).*tan(theta)+(-1).*r.*sin(phi).*tan(theta),...
      r.*cos(phi).*sec(theta).^2+q.*sec(theta).^2.*sin(phi),...
      0,0,0,0,1,sin(phi).*tan(theta),cos(phi).*tan(theta);...
    0,0,0,...
      (-1).*r.*cos(phi)+(-1).*q.*sin(phi),0,0,0,0,0,0,...
      cos(phi),(-1).*sin(phi);...
    0,0,0,...
      q.*cos(phi).*sec(theta)+(-1).*r.*sec(theta).*sin(phi),...
      r.*cos(phi).*sec(theta).*tan(theta)+...
      q.*sec(theta).*sin(phi).*tan(theta),0,0,0,0,0,sec(theta).*sin(phi),...
      cos(phi).*sec(theta);...
    0,0,0,...
      0,g.*cos(theta),0,0,r,(-1).*q,0,(-1).*w,v;...
    0,0,0,...
      (-1).*g.*cos(theta).*cos(phi),g.*sin(theta).*sin(phi),0,...
      (-1).*r,0,p,w,0,(-1).*u;...
    0,0,0,...
      g.*cos(theta).*sin(phi),g.*cos(phi).*sin(theta),0,...
      q,(-1).*p,0,(-1).*v,u,0;...
    0,0,0,...
      0,0,0,0,0,0,0,...
      Ixx.^(-1).*(Iyy.*r+(-1).*Izz.*r),Ixx.^(-1).*(Iyy.*q+(-1).*Izz.*q);...
    0,0,0,...
      0,0,0,0,0, ...
      0,Iyy.^(-1).*((-1).*Ixx.*r+Izz.*r),0,Iyy.^(-1).*((-1).*Ixx.*p+ ...
      Izz.*p);...
    0,0,0,...
      0,0,0,0,0,0,Izz.^(-1).*(Ixx.*q+(-1) ...
      .*Iyy.*q),Izz.^(-1).*(Ixx.*p+(-1).*Iyy.*p),0];

% Error Covariance Prediction
PkmDot = Ak*Pk+Pk*Ak'+Qk;
Pkm = Pk+PkmDot*T;


%% CORRECTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% 1) OPTIMAL KALMAN GAIN
Kk = Pkm*Ck'/(Ck*Pkm*Ck'+Rk);


% 2) STATE ESTIMATION CORRECTION
```

```
140  xkint = xkm+Kk*(yk-Ck*xkm);
141  xk    = xkint;
142
143
144  % 3) COVARIANCE ESTIMATION CORRECTION
145  Pk = (eye(size(Ak,1))-Kk*Ck)*Pkm;
```

Código 5.1: Filtro de Kalman Extendido

6. Seguimiento de trayectorias

Una vez que se han diseñado las leyes de control que van a ser utilizadas para conseguir que el quadrotor sea capaz de seguir una determinada referencia, es necesario establecer el control de trayectoria que será el encargado de, en función de dónde se encuentre el quadrotor en el espacio, generar las referencias adecuadas de posición o velocidad que permitan el seguimiento de una trayectoria preestablecida.

La estructura del controlador del quadrotor quedó establecida en el capítulo 5. Éste estaba compuesto por un bucle interno que se encargaba de controlar la orientación y altura de referencia que habían sido proporcionadas por el bucle externo, para ello las acciones de control disponibles eran la fuerza de tracción y los momentos de reacción. El bucle externo generaba las orientaciones y alturas de referencia a partir de unas referencias de velocidad o posición, en función de la necesidad.

El controlador de trayectoria es un bloque que se encarga de generar las posiciones o velocidades de referencia que son suministradas al bucle externo para conseguir que el quadrotor termine ajustándose a una determinada trayectoria previamente predefinida o que se está continuamente calculando, este sería el caso donde se tiene las mediciones de un sensor lidar para evadir obstáculos, sin embargo, en este documento no se va a considerar este caso, y se va a asumir que la trayectoria ya está definida de antemano.

En la figura 6.1 se muestra el control en cascada implementado para conseguir un control de trayectoria exitoso. Como se puede observar, se produce la realimentación de los estados medibles por parte de los sensores y mediante el observador de estados se consigue una estimación de todos los estados internos del sistema. Estos estados internos estimados serán los utilizados en cada bloque del sistema en cascada para generar las referencias al siguiente.

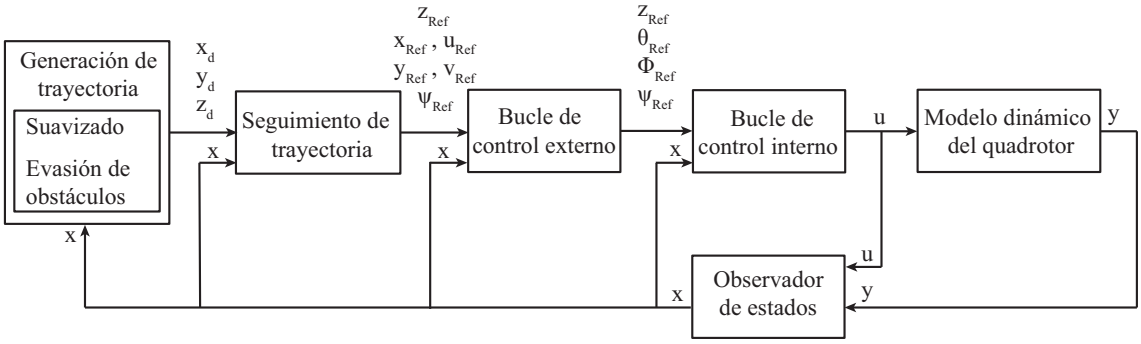


Figura 6.1: Esquema de control completo para el control de trayectorias de un vehículo del tipo quadrotor

El primero de los bloques en cascada se corresponde con el generador de los puntos que corresponden la trayectoria. Este proceso puede realizarse de diferentes formas, en [33] se utiliza una trayectoria generada a partir de arcos y lineas tangente a ellos de tal forma que se puede garantizar

que el error tiende asintóticamente a cero. Por otro lado existe la posibilidad de unir una sucesión de *waypoints* mediante el uso de líneas rectas que posteriormente pueden ser suavizadas o mediante el uso de splines. También existen algoritmos que permiten buscar el camino más adecuado entre dos puntos considerando el entorno de vuelo y evitando los obstáculos fijos que hayan en el momento de la generación de una trayectoria como puede ser el algoritmo de Dijkstra [34]. La generación de la trayectoria también puede ser un proceso iterativo que se recalcule en cada iteración del vuelo para evitar obstáculos que puedan aparecer y se detecten mediante los sensores de abordo como un lidar. En este documento se asumirá que la trayectoria generada y a seguir se mantiene fija a lo largo de las iteraciones, es decir, no se asume que el quadrotor cuente con ningún sensor que permita detectar los obtáculos.

Una vez que se ha generado la sucesión de puntos de la trayectoria, $pd = \begin{bmatrix} x_d & y_d & z_d \end{bmatrix}^T$, esta referencia se pasa al bloque de seguimiento de la trayectoria. Mediante la realimentación de los estados estimados del sistema se permite generar las altitudes y velocidades o posiciones de referencia que debe mandarse al bucle de control externo. El resto del diagrama ya ha sido introducido en el capítulo 5. Cabe destacar que la implementación en bloques del sistema de control completo permite sustituir cualquiera de ellos en caso que se desee implementar una aproximación diferente.

Existen algunos algoritmos de control de trayectoria como pueden ser el de *backstepping* o el de *feedback linearization* mostrados en [12] que no siguen necesariamente la misma estructura. En concreto no necesitan de los bloques de control interno y externo, ya que simplemente las acciones de control del sistema del quadrotor son generadas automáticamente por el bloque del control de trayectoria. Los algoritmos que si que se ajustan a ese modelo son el NLGL o el *carrot chase* entre otros.

6.1 Seguimiento de trayectoria: Algoritmo NLGL (*Non Lineal Guidance Law*)

El algoritmo NLGL se corresponde con un método geométrico que está basado en la idea del *virtual target point* o VTP. El VTP es el punto de la curva deseada que el quadrotor está tratando de seguir en todo momento, su actualización se produce en cada iteración del bucle de control de la trayectoria y su posición se calcula como el punto de la curva que está situado a una distancia L del quadrotor. Una vez conocido el VTP se calcula el ángulo de guiñada ψ necesario para seguir la dirección del VTP. Para añadir la operación en el espacio tridimensional se debe obtener la diferencia de alturas que hay entre el quadrotor y el punto de la trayectoria más cercano al quadrotor para marcar la nueva referencia.

La evaluación del algoritmo se hace conociendo la posición del quadrotor en ese instante, entonces se genera un arco de radio L que cortará la trayectoria deseada en dos puntos, en función de la dirección de movimiento que se desee en el quadrotor se escogerá uno u otro como el VTP a seguir. Considerando la posición del quadrotor y la velocidad deseada en ejes cuerpo que se quiere que siga, se calcula la guiñada, por otro lado, la altitud de referencia se calcula como la del punto de la trayectoria más cercano al quadrotor. En la figura 6.2 se puede ver el procedimiento seguido en la actualización de la referencia en la guiñada en función de la posición del VTP.

Cabe destacar que el algoritmo que se utiliza en el presente documento está basado en el mostrado en la referencia [16] y en [12] aunque con ciertas modificaciones para permitir el movimiento siguiendo una trayectoria curva en tres dimensiones ya que el mostrado en la referencia es únicamente válido para seguimiento de líneas rectas o circulares con altitud constante. Además, en las referencias se asume que la velocidad lateral es nula.

En el algoritmo 1 se muestra la estructura del código del control de trayectoria empleando NLGL.

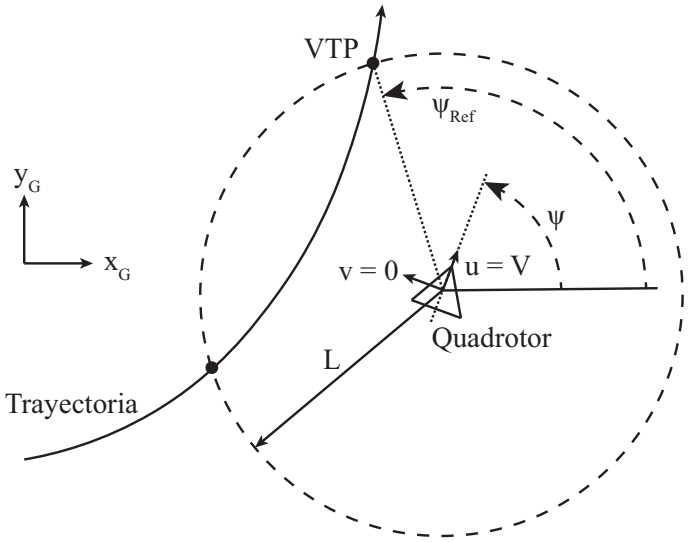


Figura 6.2: Algoritmo para el seguimiento de trayectorias NLGL

En un primer lugar se realimentan los estados estimados mediante el observador para evaluar la posición y guiñada actual del quadrotor. Como otros argumentos de entrada se encuentra el vector, p_d, que contiene los puntos que forman la trayectoria en función de la coordenada curvilínea s. El vector V contiene la velocidad del quadrotor en función de la coordenada curvilínea mencionada anteriormente y el parámetro L es la distancia del quadrotor al VTP.

Algoritmo 1: *Non Lineal Guidance Law* - NLGL[12]

Data: $\xi = (x,y,z)^T$, ψ
$p_d(s) = (x_d(s), y_d(s), z_d(s))^T$, $V(s)$, L

Cálculo del punto más cercano $(d_{min}, s_{d_{min}})$
$[d_{min}, s_{d_{min}}] = min(|\xi - p_d(s)|)$;

Cálculo del VTP
if $d_{min} > L$ **then**
  $s_{VTP} = s_{d_{min}}$;
else
  $[0, s_{VTP}] = min(||\xi - p_d(s)| - L|)$;
end

Cálculo de las referencias a seguir
$\psi_{Ref} = atan2(y_d(s_{VTP}) - y, x_d(s_{VTP}) - x)$;
$z_{Ref} = z_d(s_{d_{min}})$;
$v_{Ref} = 0$;
$u_{Ref} = V(s_{d_{min}})$;

return $\psi_{Ref}, z_{Ref}, v_{Ref}, u_{Ref}$

El primero de los pasos del algoritmo consiste en el cálculo del punto de la trayectoria más cercano al quadrotor. Los resultados que se obtendrán será la distancia mínima y la coordenada curvilínea a la que ocurre este suceso. En el cálculo de la posición del VTP se comprobará que la distancia mínima sea menor que L, ya que si esto no ocurre se ubicará el VTP en el punto de distancia mínima aunque sea mayor que L. Posteriormente se calculará la coordenada curvilínea donde se

encontraría el VTP. Debido a la naturaleza del VTP con respecto al quadrotor, aparecerán dos posibles candidatos como el objetivo. Se escogerá el que esté en sentido del movimiento deseado.

Una vez que se ha calculado la posición donde se encontraría el VTP, se calculará la guiñada necesaria que se le debe suministrar al sistema de control como referencia. Para ello se asumirá que la velocidad lateral del quadrotor es nula en toda la trayectoria, esta hipótesis tiene sentido especialmente si se considera que en un vuelo real de quadrotor se desea que la cámara este apuntando en dirección del movimiento.

Finalmente se calcularán las referencias que serán suministradas al bucle de control externo. Es importante notar que el tipo de control realizado es de velocidad porque lo que es posible que el punto final de la trayectoria no sea exacto, para ello se deberá añadir una conmutación con el control de posición que asegure que se alcanza el punto final deseado.

Cabe destacar que este algoritmo presenta de ciertas complejidades cuando las trayectorias tienen cambios de referencia en la altura continuos, la trayectoria es del tipo ascendente helicoidal con tramos unos por encima de otros o cuando existen numerosas curvas de forma que los tramos pasan muy cerca unos de otros. Por ello, en el presente documento se propone un control de trayectoria basado en el control de posición como alternativa.

6.2 Seguimiento de trayectoria: Control mediante cambios en la referencia de posición

El algoritmo que va a ser utilizado para conseguir el seguimiento de trayectoria se construye a partir del controlador de posición. El algoritmo NLGL está diseñado para cualquier tipo de UAV, es por ello que no tiene en cuenta la capacidad de un quadrotor de poder moverse en cualquier dirección independientemente de cuál sea su ángulo de guiñada y tampoco considera su capacidad de mantenerse en vuelo en punto fijo, por lo que el punto final de la trayectoria no será exactamente el deseado. Otra desventaja que tiene el algoritmo NLGL es que no permite el seguimiento de trayectorias del tipo del ascenso helicoidal donde la trayectoria está solapada a diferentes alturas ni trayectorias con curvas muy cerradas con varios tramos unos muy cerca de los otros. Por ello, en esta sección se diseña un controlador de trayectoria particular para los vehículos del tipo multirotor aprovechando la capacidad que tienen de moverse en cualquier dirección independientemente de su orientación.

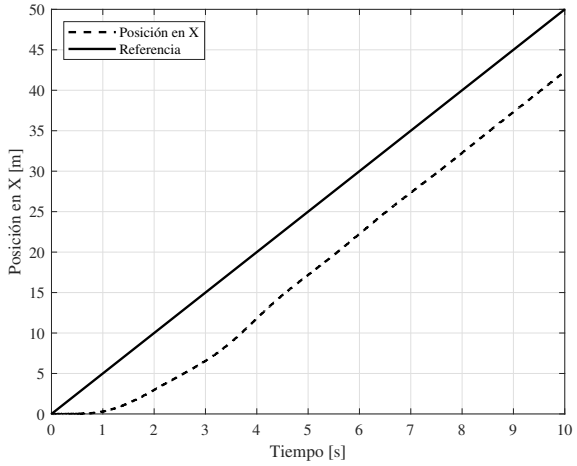


Figura 6.3: Control de trayectoria mediante el control de la posición

La idea principal del algoritmo consiste en la selección de referencias de posición que puedan variar con el tiempo. Poniendo el caso en que se desea controlar la trayectoria en el eje X, si se

desea que el quadrotor se mueva con una determinada velocidad en ese eje, es posible controlar la trayectoria mediante una referencia que se mueva a la velocidad deseada, de esta forma en el instante inicial el quadrotor tendrá la referencia justo en su posición, pero a medida que pasa el tiempo la referencia se moverá alrededor de la trayectoria a la velocidad deseada y puesto que se aleja del quadrotor, éste tenderá a seguirla, tras pasar un cierto transitorio el quadrotor comenzará a moverse a la misma velocidad que el punto de referencia y a una cierta distancia de él. Como se puede observar, el concepto es considerablemente similar al del VTP, ya que el quadrotor estaría siguiendo un objetivo.

La principal desventaja que tiene este algoritmo es que en el caso que el quadrotor comience en un punto alejado de la trayectoria y con velocidad no nula, se debería generar un conjunto de referencias que tiendan asintóticamente a la trayectoria deseada, es decir, la trayectoria debe de empezar desde la posición del quadrotor. Como punto a favor tiene que se podría seleccionar la velocidad a la que se debe mover el quadrotor, variando la velocidad a la que se mueve la referencia y además variar la guiñada independientemente de la velocidad que tenga en cada uno de los puntos de la trayectoria.

Para ilustrar el funcionamiento de este tipo de control se añade la figura 6.3. En esta figura se está representando el movimiento de un quadrotor a lo largo del eje X con una velocidad determinada. Como se puede observar, a medida que la referencia se aleja del quadrotor, este comienza a aumentar su velocidad hasta que se iguala con la de la referencia y comienza a seguirla a una cierta distancia.

7. Parámetros físicos del quadrotor simulado

En el presente capítulo se muestran los parámetros físicos del quadrotor escogido para la simulación. Concretamente se trata del quadrotor Parrot AR Drone 2.0, mostrado en la figura 7.1[1]. Cabe destacar que la obtención de los parámetros físicos se ha hecho del *software* utilizado en la validación del sistema de control, AirSim, de esta forma se asegura que en ambos entornos se simula el mismo vehículo.

Figura 7.1: Quadrotor simulado: Parrot AR Drone 2.0

Parámetro másicos	Variable	Valor	Unidades
Configuración	-	X	-
Número de rotores	n_{motor}	4	-
Longitud del brazo	l_{arm}	0.23	m
	l_{box_x}	0.18	m
Longitud del cuerpo	l_{box_y}	0.11	m
	l_{box_z}	0.04	m
Longitud tren aterrizaje	l_{feet}	0.14	m
Diámetro del rotor	D	0.23	m
Masa	m	1	kg
Masa motor	m_{motor}	0.055	kg
	I_{xx}	6.7e-3	$kg\,m^2$
Momentos de inercia	I_{yy}	8e-3	$kg\,m^2$
	I_{zz}	0.0143	$kg\,m^2$
Inercia del rotor	I_R	2.03e-5	kg m^2
Gravedad	g	9.8	$m\,s^{-2}$
Densidad del aire	ρ	1.225	$kg\,m^{-3}$

Tabla 7.1: Parámetros másicos del quadrotor

[1] `https://www.parrot.com/es/drones/parrot-ardrone-20-elite-edition`

Además, se añaden los parámetros que se han considerado en la simulación que no son pertenecientes al quadrotor, como son los referidos al contacto con el suelo o el intervalo de simulación utilizado para ejecutar el algoritmo de Kalman extendido o muestrear los parámetros de interés para luego representarlos.

Parámetro del motor	Variable	Valor	Unidades
Máx RPM del motor	max_{RPM}	6396.67	rpm
Constante de tiempo del filtro paso bajo del motor	T_m	0.005	s^{-1}

Tabla 7.2: Parámetros del motor del quadrotor

Parámetro aerodinámicos	Variable	Valor	Unidades
Máxima tracción	T_{max}	4.18	N
Máximo par	Q_{max}	0.056	N m
Coeficiente de tracción	k	9.31e-6	$rad\,s^{-2}$
Coeficiente de resistencia	b	1.28e-7	$rad\,s^{-2}$
Coeficiente de resistencia aerodinámica Lumped	$\overline{c}$	0.037	$s\,m^{-1}$
Constante de efecto fuente en efecto suelo	K_B	2	-

Tabla 7.3: Parámetros aerodinámicos del quadrotor

Parámetro del simulador	Variable	Valor	Unidades
Amortiguamiento relativo del contacto	ξ_a	0.95	-
Amortiguamiento de contacto	c_a	47.02	$N\,s\,m^{-1}$
Rigidez de contacto	k_a	2450	$N\,m$
Coeficiente de fricción de Coulomb	μ	0.5	-
Periodo de muestreo y actualización del filtro de Kalman	T	0.0003	s

Tabla 7.4: Parámetros de la simulación

8. Simulación y validación del modelo dinámico y sistema de control

En este capítulo se muestra en un primer lugar el software externo que se ha empleado para validar el modelo dinámico y el sistema de control. Este programa servirá también para la validación del sistema de control en el dominio discreto. Por otro lado, también se incluyen las simulaciones empleadas, en las que se han generado diversas trayectorias que debe seguir el quadrotor que partirá con diferentes condiciones y no siempre desde un punto dentro de la trayectoria. Para la correcta validación del sistema de control se debe asegurar que el quadrotor converge de forma asintótica a la trayectoria impuesta. Se podrán comparar las respuestas en las diferentes plataformas de simulación para comprobar la validez de ambas opciones.

8.1 Simulador empleado en la validación

A lo largo de todo el presente documento se ha hecho uso del modelo dinámico del quadrotor modelado en un bloque de Simulink para poder simular la respuesta del sistema ante diferentes leyes de control y poder ajustar correctamente las ganancias del sistema hasta obtener la respuesta deseada, en la sección 4.7 se mostró la implementación de este bloque. De esta forma puede comprenderse la ventaja de utilizar los simuladores ante cualquier técnica experimental y es que se pueden reducir los costes en gran medida en la etapa de prediseño hasta tener un primer controlador que tenga las características deseadas. Sin embargo, posteriormente se debe validar el diseño obtenido mediante técnicas experimentales u otros simuladores que presenten un sistema dinámico más complejo e incluya los efectos no modelados de forma que se ajuste a la realidad, ya que es posible que un regulador que en simulación funciones bien y se ajuste a los requerimientos en la etapa de validación falle debido a que el sistema dinámico empleado no se ajustaba a la realidad de forma precisa.

En el presente documento se empleará la simulación para validar el regulador obtenido. En concreto se utilizará el software AirSim, éste es un simulador de Microsoft para drones, coches y otros que corre en el motor gráfico de Unreal Engine. Se caracteriza por ser un *software* de código abierto, es decir, cualquier persona puede ver y editar el código por el que está compuesto, permite la ejecución en varios sitemas operativos, entre ellos Windows y Linux y además soporta el llamado *hardware* y *software in the loop*. Por tanto, esta herramienta puede ser utilizada para simular el comportamiento dinámico del sistema de igual forma que se hizo con el modelo dinámico.

Este software esta pensado para el desarrollo de algoritmos para el control de los vehículos basados en inteligencia artificial, aprendizaje profundo, visión artificial y aprendizaje reforzado. Para ello, el software cuenta con APIs que permiten la conexión con Python o C++ donde se puede recibir la información procedente de los sensores del vehículo y se permite cambiar las referencias del sistema de control e incluso ejecutar la propia ley de control. Aunque el software esté centrado

en el control por inteligencia artificial, con el uso de las APIs adecuadas es posible desarrollar el control en bajo nivel. Se considera control de bajo nivel el que se ha desarrollado en el presente documento, donde se reciben unas mediciones de los sensores y se calculan las acciones de control que se deben suministrar al sistema para el seguimiento de una referencia. Cabe destacar que el *software* empleado en el presente documento en la comunicación con AirSim es Python, por ello, todo el desarrollo mostrado es únicamente válido si se emplea el mismo lenguaje de programación. En caso que se desee utilizar C++ se recomienda consultar la página de Github del simulador[1].

Se puede encontrar la información necesaria acerca de la instalación del simulador en el apéndice A.

8.1.1. APIs de AirSim

Las APIs tienen como principal función permitir la comunicación e interacción con AirSim por el ususario empleando para ello un lenguaje de programación como es Python. Algunas de las funciones que pueden realizarse son la recepción de las mediciones de los sensores, el estado del quadrotor, imágenes de la cámara a bordo e incluso controlar las velocidades de giro de los rotores. Cabe destacar que AirSim ya tiene implementadas las leyes de control que permiten controlar el vehículo suministrando las referencias, sin embargo, en este documento se hará uso de exclusivo del control de los motores y no se empleará la controlador a ya implementada. De esta forma se podrá implementar la misma ley de control que la desarrollada en el documento mediante LQG para su validación.

Para el uso de las APIs mediante Python se recomienda tener instalada la versión 3.5 o superior. En un primer lugar es necesaria la instalación del paquete de AirSim de Python, esto se realizará mediante el comando en la consola de comandos 'pip install airsim' que automáticamente buscará la librería en el repositorio y la instalará para su uso.

En la carpeta de instlación, concretamente en el directorio 'PythonClient\multirotor' se pueden encontrar algunas ejemplos de como utilizar las APIs. En el código 8.1 se muestra un ejemplo de la estructura a seguir para recibir información y enviar acciones de control al simulador.

```
1  import setup_path
2  import airsim
3
4  import numpy as np
5
6  # connect to the AirSim simulator
7  client = airsim.MultirotorClient()
8  client.confirmConnection()
9  client.enableApiControl(True)
10 client.armDisarm(True)
11
12 # Get state of the vehicle
13 state = client.getMultirotorState()
14
15 # Get sensors data
16 imu_data          = client.getImuData()
17 barometer_data    = client.getBarometerData()
18 magnetometer_data = client.getMagnetometerData()
19 gps_data          = client.getGpsData()
20
21 # Wait for tkae off
22 airsim.wait_key('Press any key to takeoff')
23 client.takeoffAsync().join()
```

[1] https://microsoft.github.io/AirSim/apis/

```
24
25 # Wait to move the vehicle
26 airsim.wait_key('Press any key to move vehicle to (3, 3, -3) at 5 m/s')
27 client.moveToPositionAsync(3, 3, -3, 5).join()
28
29 # Wait to stop control
30 airsim.wait_key('Press any key to reset to original state')
31
32 # Disarm and reset to original state
33 client.armDisarm(False)
34 client.reset()
35
36 # that's enough fun for now. let's quit cleanly
37 client.enableApiControl(False)
```

Código 8.1: Ejemplo de comunicación con AirSim: Hello Drone

En un primer lugar es necesario importar el archivo 'setup_path.py' que se puede encontrar en la carpeta mencionada anteriormente,'PythoClient\multirotor'. Este archivo permite detectar si el paquete de AirSim de Python está disponible en la carpeta raíz para utilizarlo en lugar del instlado mediante el comando 'pip install', de esta forma se utiliza siempre la versión actualizada. Posteriormente se debe incluir la librería de AirSim que permite utilizar las APIs mediante la línea de código 'import airsim'. El paquete de 'numpy' permite la utilización de muchas de las operaciones matemáticas en el lenguaje de Python y se recomienda su uso debido a la rapidez con la que permite realizar los cálculos.

Posteriormente se debe conectar Python con al quadrotor mediante la creación de la clase 'client' con la API 'airsim.MultirotorClient()'. Sobre esta clase, que haría referencia al quadrotor del simulador, es sobre quien se aplican los diferentes métodos que permiten obtener los estados o controlar el quadrotor. A continuación se detallan algunos de los métodos más utilizados. Notad en el código 8.1 que, en el caso de querer leer los datos de la IMU, se debe llamar al método mediante 'client.getImuData'. Para más información se recomienda acceder a la página de Github del simulador[2] o la lectura del código del simulador donde se pueden encontrar todas las APIs disponibles. Aunque no se muestran en el presente documento, existe la posibilidad de cambiar la meteorología o la parte del día en la que se simula. La utilidad de estas APIs vienen del lado del control con visión artificial que no será utilizado en este documento.

- 'confirmConnection()': Comprueba que existe conexión con el simulador cada 1 s y se muestra en la consola para que el usuario lo pueda comprobar.
- 'enableApiControl(True/False)': Por defecto el control mediante las APIs está desactivado, si se desea controlar el vehículo mediante ellas es necesario activarlo. Para comprobar si el control mediante APIs está activado se puede llamar a 'isApiControlEnabled()' que devolverá True/False.
- 'armDisarm(True/False)': Se encarga de armar y desarmar el quadrotor ya que por defecto el giro de los rotores no está activado.
- 'reset()': Restablece el estado del vehículo a la posición original. Cabe destacar que tras llamarlo es necesario volver a ejecutar 'enableApiControl()' y 'armDisarm()'.
- 'getMultirotorState()': Devuelve una variable que contiene el estado del quadrotor. Concretamente se muestra la posición, velocidad y aceleración lineal y orientación, velocidad y aceleración angular. Cabe destacar que estas mediciones son absolutas y no presentan ruido o sesgos.

[2] https://microsoft.github.io/AirSim/apis/

- 'getImuData()','getBarometerData()','getMagnetometerData()','getGpsData()': Mediante ellos se puede obtener las mediciones del correspondiente sensor que sí incluyen los correspondientes ruidos.
- 'moveByMotorPWMsAsync($rotor_1$, $rotor_2$, $rotor_3$, $rotor_4$, duración)': Mediante esta API es posible controlar la velocidad de giro de cada uno de los rotores. La entrada se corresponde con el valor pwm de giro, es decir, la velocidad deseada dividida entre la velocidad máxima. Este tipo de APIs de control son ejecutadas durante un cierto tiempo y son canceladas cuando se llama a otra API de control. Si se desea que se espere a que termine la acción de control para continuar con el código se debe añadir el método '.join()' a la llamada de la API de control. Puesto que esta es la única API utilizada para controlar el quadrotor no se incluirán el resto de opciones, si se desea consultar el resto de APIs destinadas al control del quadrotor se recomienda acceder a la documentación del simulador. En la documentación se pueden encontrar APIs que permiten controlar la posición, velocidad, orientación simplemente seleccionando la referencia, este tipo de control en AirSim se le llama de alto nivel mientras que el seleccionar las velocidades de giro de los rotores se conoce como bajo nivel como se introdujo anteriormente.

Aunque no será utilizado en el presente documento, mediante AirSim también es posible la simulación con múltiples vehículos simultáneos. Las aplicaciones que tiene es la simulación de control de formaciones o la simulación de tareas de mapeado del suelo. Para más información se recomienda la lectura de la documetnación del simulador[3].

Finalmente, cabe destacar que la velocidad de comunicación mediante las APIs con AirSim está restringida a 50 Hz, sin embargo para el control de bajo nivel realizado serán necesarias frecuencias de alrededor 500 Hz. Para solucionar el problema es posible la disminución de la velocidad del reloj de simulación a 0.1, de tal forma que serán necesarios 10 s de tiempo real para simular 1 s. Esta disminución de la velocidad del reloj se puede hacer en el documento en formato '.json' que se crea en la carpeta de documentos del ordenador la primera vez que se ejecuta el simulador, para ello se debe añadir la línea ' "ClockSpeed": 0.1' a este archivo. El resto de configuraciones que pueden ser modificadas con este documento se pueden encontrar en la página del simulador[4].

8.2 Validación del sistema dinámico y del sistema de control

En un primer lugar se presenta la validación del modelo dinámico y del sistema de control. Cabe destacar que el modelo dinámico se validará mediante simulaciones donde se evaluará si el control es capaz de cumplir con sus objetivos, en el caso de que la respuesta del sistema de Simulink y AirSim sean similares se podrá concluir que el modelo dinámico se ajusta al del simulador.

De igual forma, si la respuesta del sistema de control en Simulink y AirSim son similares se podrá validar el sistema de control. Para ello se utilizarán las mismas simulaciones en Simulink y AirSim con las mismas referencias, acciones de control y ruidos de medida y proceso.

Además, el proceso de validación se harán para los dos controladores diseñados, el de velocidad y posición. De esta forma, se podrá validar al mismo tiempo que el modelo dinámico asumido en Simulink es lo suficientemente realista comparado con el del simulador y que los diferentes controladores cumplen con sus criterios de diseño en el dominio continuo (Simulink) y en el dominio discreto (AirSim).

Cabe destacar que el controlador utilizado en la simulación mediante AirSim y Python debe de estar definido en el dominio discreto ya que no permite una comunicación continua en el tiempo.

[3] `https://microsoft.github.io/AirSim/multi_vehicle/`

[4] `https://microsoft.github.io/AirSim/settings/`

8.2.1. Controlador de posición

La primera de las simulaciones utilizadas para la doble validación, del sistema dinámico y del sistema de control, consiste en un control de posición. Para ello se asumirán las siguientes referencias mostradas en la figura 8.1 y 8.2. Es decir partiendo del quadrotor apoyado en el suelo, en el instante inicial se asumirá una referencia del tipo escalón de magnitud 7 m en la coordenada Z, posteriormente en el instante de $t = 4s$ se asumirá un cambio en la referencia del tipo escalón y de magnitud igual a 5 m en la posición en Y, finalmente en el instante de $t = 8s$ se utilizará un cambio en la referencia del tipo escalón de magnitud 3 m en la posición X. En cuanto a la referencia de la guiñada se asumira un escalón en el instante $t = 3s$ de magnitud igual a 180°

En cuanto a las covarianzas de los ruidos de medida y proceso se asumirán los mostrados en las ecuaciones (5.42).

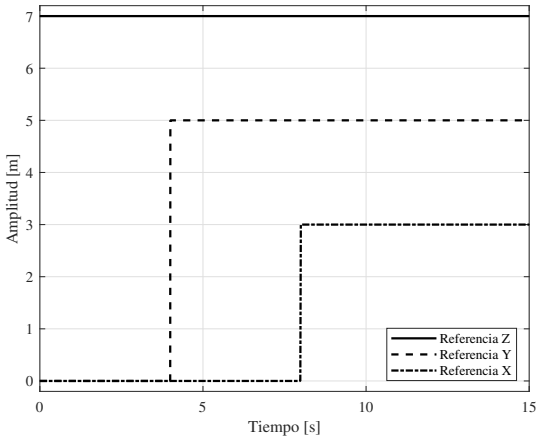


Figura 8.1: Referencias de la posición del quadrotor empleadas para validar el control de posición

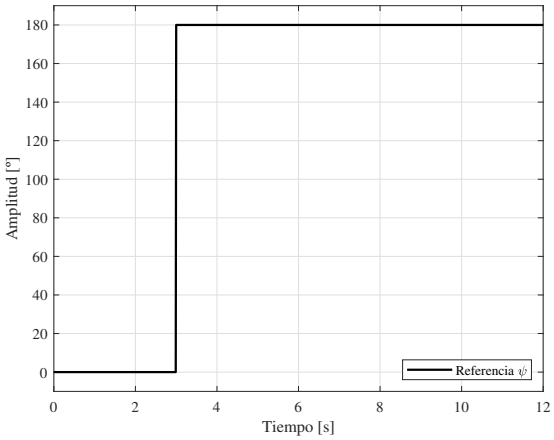


Figura 8.2: Referencias de la guiñada del quadrotor empleadas para validar el control de posición

Cabe destacar que las mediciones de la guiñada en el simulador de AirSim se devuelven mapeadas entre -180° y 180° apareciendo el problema de la discontinuidad descrito en la subsección 5.2.4. Para solucionar el problema se implementará una función en Python que detecte el paso por la discontinuidad y corregirá los valores de la guiñada proporcionados por el programa añadiéndole o restándole un múltiplo de 360° en función del número de vueltas que haya dado el quadrotor y en función del sentido en el que se pase por la discontinuidad. En cuánto a la referencia de la guiñada suministrada se asegurará que se encuentra dentro del rango $-180^\circ + \psi$ y $180^\circ + \psi$, siendo ψ el valor de la guiñada actual del quadrotor, de esta forma se asegurará que el quadrotor seguirá la referencia de la guiñada por el camino más corto.

Además, es importante considerar que las mediciones de la orientación del quadrotor se miden mediante el uso de quaterniones, por ello se deberá transformar estas mediciones a ángulos de Euler que permitan emplear el control diseñado. Por otro lado, las mediciones de velocidad del quadrotor se miden en ejes globales, siendo necesarias las medidas en ejes locales en el control diseñado en espacio de estados, por esta razón los valores proporcionados por el programa en cuanto a la velocidad se deberán rotar al sistema de referencia local.

En el apéndice B se muestran todos los códigos de los programas utilizados en las diferentes simulaciones. Concretamente en la sección B.1 del apéndice se muestran los códigos empleados en la validación del sistema dinámico y de control.

La duración de la simulación será de 15 s con un periodo de actualización del filtro de Kalman de $3e-4$. El periodo de actualización del algoritmo discreto en Python se asumirá igual que el periodo de actualización de Kalman.

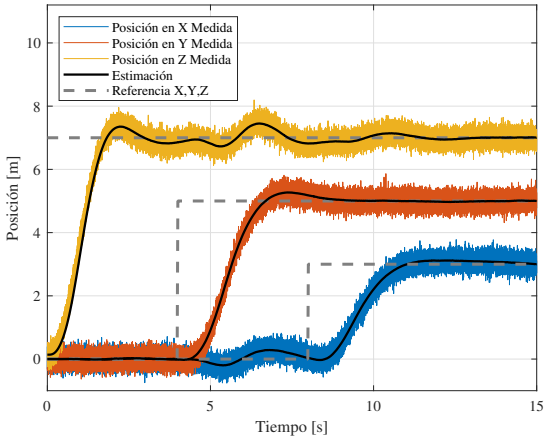


Figura 8.3: Variación de la posición del quadrotor por cambio de referencia de posición: Simulink

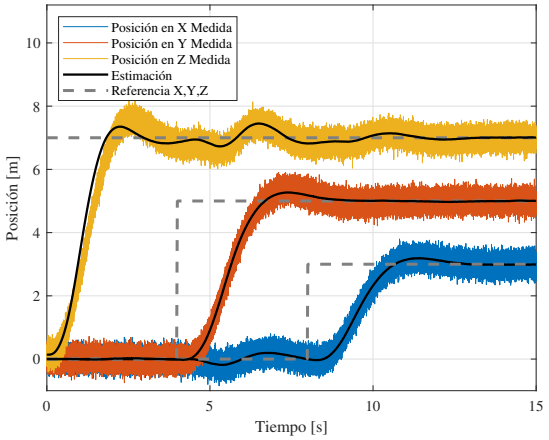


Figura 8.4: Variación de la posición del quadrotor por cambio de referencia de posición: AirSim

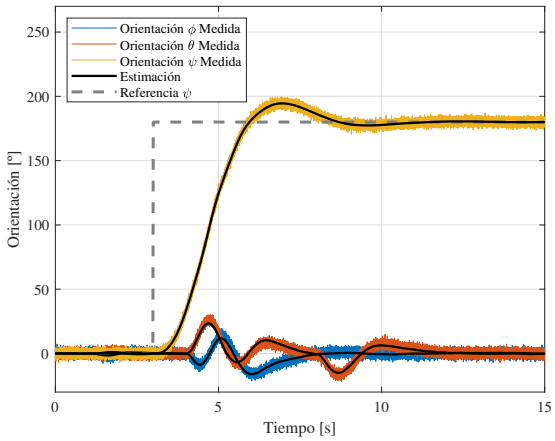


Figura 8.5: Variación de la orientación del quadrotor por cambio de referencia de posición: Simulink

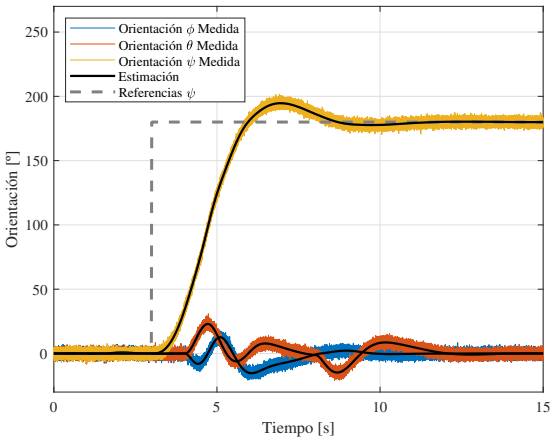


Figura 8.6: Variación de la orientación del quadrotor por cambio de referencia de posición: AirSim

En las figuras 8.3 y 8.4 se representa la estimación de la posición del quadrotor a partir de las mediciones de los sensores con un cierto ruido de medición para el modelo desarrollado en Simulink y el simulador de AirSim.

Por otro lado, en las figuras En las figuras 8.5 y 8.6 se representa la estimación de la orientación del quadrotor a partir de las mediciones de los sensores con un cierto ruido de medición para el modelo desarrollado en Simulink y el simulador de AirSim.

Cabe destacar que se ha representado la respuesta de cada software por separado para reducir la carga visual de las figuras. A continuación, se representan las estimaciones de todos los estados, incluso los no accesibles directamente por los sensores, donde se podrá comparar en la misma figura los mismos estados estimados mediante AirSim y Simulink. De esta forma se podrá comparar la respuesta de forma más precisa.

En la figura 8.7 se muestra la comparación de la respuesta del sistema ante un cambio de referencia en la posición del modelo de Simulink y el simulador de AirSim. En la figura 8.8 se muestra la comparación de la evolución de la orientación de Simulink y AirSim. Finamente las figuras 8.9 y 8.10 muestran la comparación de las velocidades lineales y angulares.

Como se puede observar en las comparaciones, la respuesta de Simulink y AirSim es idéntica en la evolución de los estados de la posición orientación y sus velocidades, siendo sus diferencias despreciables. Estas se pueden asumir debido a los fenómenos no modelados que pueden existir

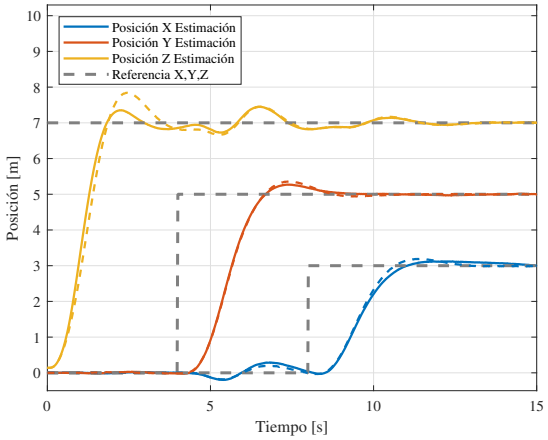


Figura 8.7: Comparación de la posición del quadrotor por cambio de referencia de posición: Simulink (línea continua) y AirSim (discontinua)

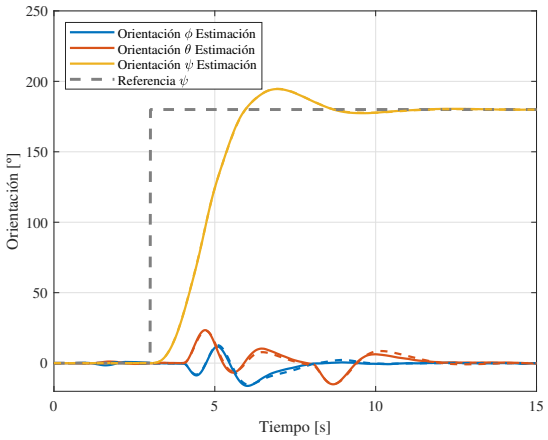


Figura 8.8: Comparación de la orientación del quadrotor por cambio de referencia de posición: Simulink (línea continua) y AirSim (discontinua)

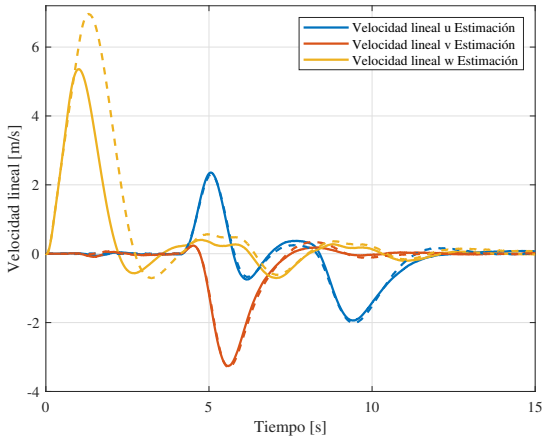


Figura 8.9: Comparación de la velocidad lineal del quadrotor por cambio de referencia de posición: Simulink (línea continua) y AirSim (discontinua)

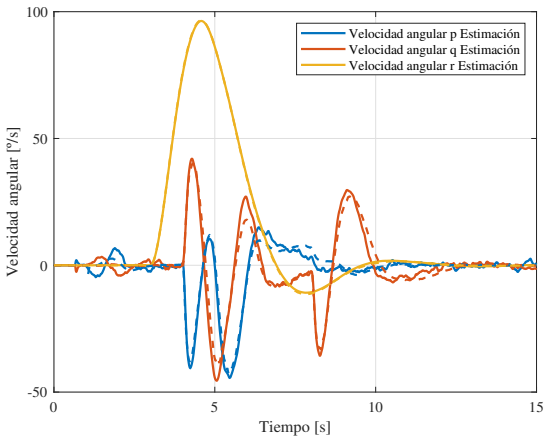


Figura 8.10: Comparación de la velocidad angular del quadrotor por cambio de referencia de posición: Simulink (línea continua) y AirSim (discontinua)

en los dos modelos y al hecho de que en Simulink el control realizado es continuo y en AirSim es discreto.

Las principales diferencias aparecen en el transitorio de la respuesta, teniendo diferencias despreciables en el estacionario alcanzado. Especialmente, aparecen ciertas discrepancias en la posición en Z y su velocidad lineal w que se pueden ser explicadas debido a que en el modelo de Simulink se despreció la resistencia en Z del quadrotor. Como se puede observar en la figura 8.9, las diferencias en cuanto a la velocidad lineal en Z aparecen únicamente a altos valores que es cuando la resistencia empieza a tener relevancia.

A pesar de haber realizado ciertas simplificaciones en el modelo dinámico que se ha utilizado para la creación del bloque no lineal que representa el movimiento del quadrotor en Simulink, se puede observar que la respuesta se ajusta con pocas discrepancias a la de AirSim. De esta forma se puede validar el modelo dinámico simplificado.

En cuanto al controlador empleado para seguir unas determinadas referencias de posición se puede validar su funcionamiento ya que, como se ha podido observar, este ha sido capaz de cumplir las referencias con ningún error de posición y con la misma respuesta dinámica del sistema tanto con el simulador dinámico en Simulink y el simulador de AirSim.

8.2.2. Controlador de velocidad

Para validar el modelo dinámico y el sistema de control también se utilizará el control de velocidad desarrollado en el capítulo 5. Las referencias que se utilizarán será un cambio de altura de magnitud 5 m en el instante inicial y en cambio de velocidad en el instante $t = 2s$ en u y v con valor de 5 m/s. En cuanto a la guiñada se utilizará un cambio del tipo escalón en el segundo 2 de valor $-30°$.

El valor de la covarianza de cada uno de los ruidos será el mostrado en la ecuación (5.42). La duración de la simulación será de 10 s y el periodo de muestreo de 0.0003 s.

En las figuras 8.11 y 8.12 se muestra la evolución de la posición del quadrotor ante las referencias anteriores. Se añaden las mediciones de los estados que son accesibles mediante los sensores con su ruido de medida y la estimación del filtro de Kalman.

Por otro lado, en las figuras 8.13 y 8.14 se muestra la evolución de la orientación del quadrotor ante las referencias anteriores. Se añaden las mediciones de los estados que son accesibles mediante los sensores con su ruido de medida y la estimación del filtro de Kalman.

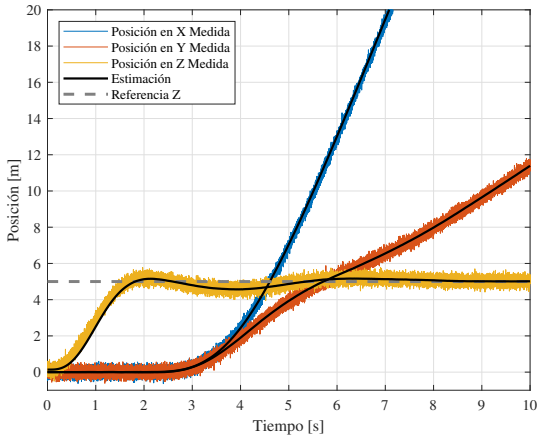


Figura 8.11: Variación de la posición del quadrotor por cambio de referencia de velocidad: Simulink

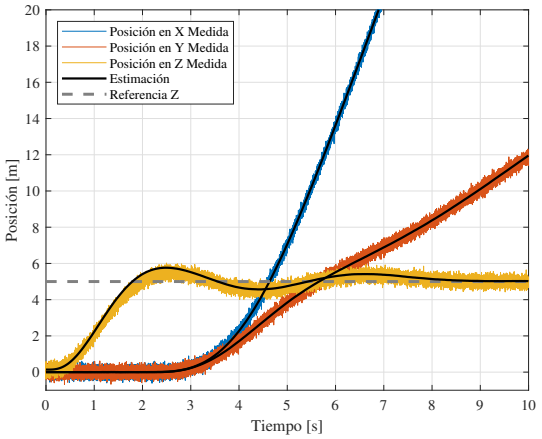


Figura 8.12: Variación de la posición del quadrotor por cambio de referencia de velocidad: AirSim

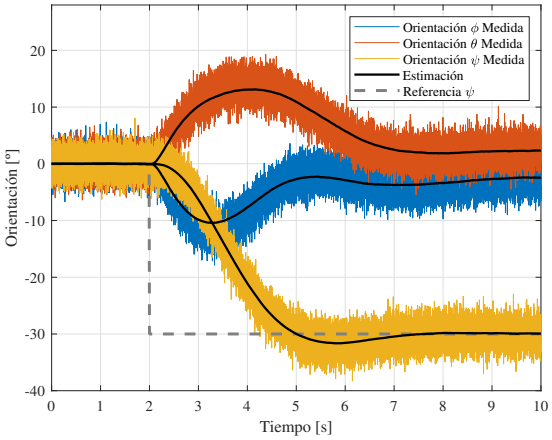


Figura 8.13: Variación de la orientación del quadrotor por cambio de referencia de velocidad: Simulink

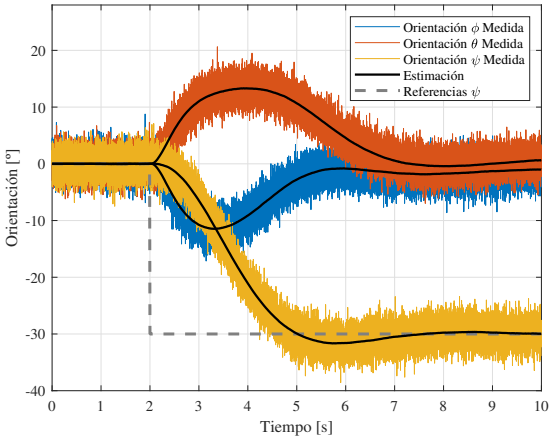


Figura 8.14: Variación de la orientación del quadrotor por cambio de referencia de velocidad: AirSim

Al igual que se hizo con el control de posición se añaden los estados estimados del quadrotor ante los mismos cambios de referencia en AirSim y Simulink para permitir su comparación.

En las figuras 8.15 y 8.16 se muestra la comparación entre las estimaciones de posición y orientación del quadrotor ante un cambio en las referencias de velocidad. Como se puede observar las

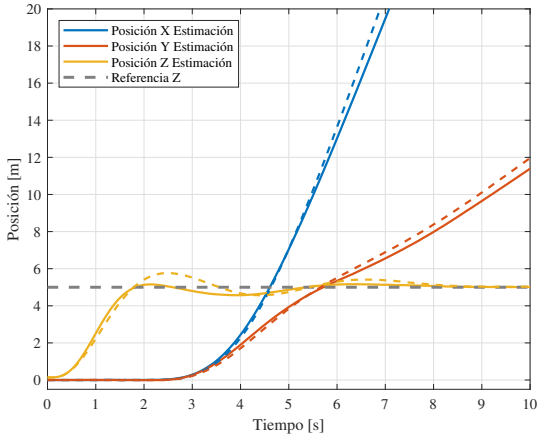


Figura 8.15: Comparación de la posición del quadrotor por cambio de referencia de velocidad: Simulink (línea continua) y AirSim (discontinua)

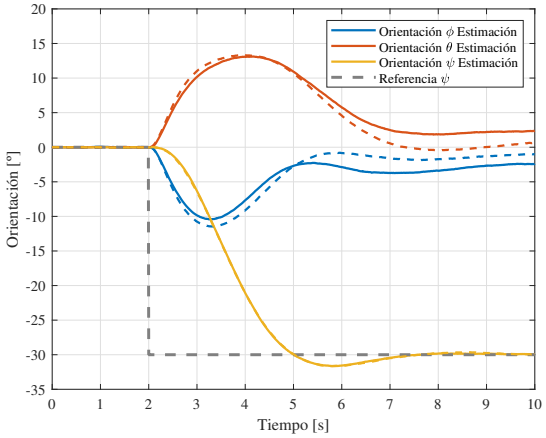


Figura 8.16: Comparación de la orientación del quadrotor por cambio de referencia de velocidad: Simulink (línea continua) y AirSim (discontinua)

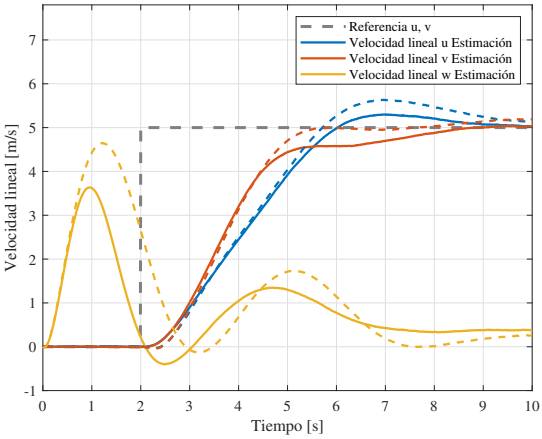


Figura 8.17: Comparación de la velocidad lineal del quadrotor por cambio de referencia de velocidad: Simulink (línea continua) y AirSim (discontinua)

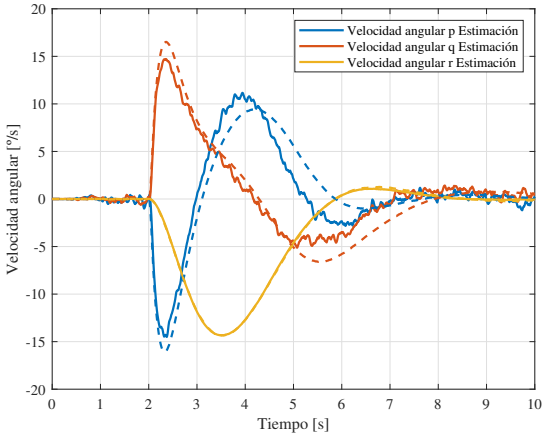


Figura 8.18: Comparación de la velocidad angular del quadrotor por cambio de referencia de velocidad: Simulink (línea continua) y AirSim (discontinua)

tendencias son similares a las mostradas en la comparación del control de posición. Existen ligeras diferencias, sobre todo en la respuesta de la posición en Z que pueden ser explicadas debido a los fenómenos no modelados como pueden ser la resistencia aerodinámica en este eje. Al igual que antes, se puede concluir que aunque no se considere este fenómeno, la dinámica del sistema continúa siendo lo suficientemente aproximada como para ser despreciado. En cuanto a las orientaciones del sistema, existen ligeras diferencias entre los ángulos de Euler de cabeceo y alabeo que son debidas a las diferentes definiciones de la resistencia aerodinámica en Simulink y AirSim. Como se puede observar, la resistencia aerodinámica en el modelo de AirSim es menor que en Simulink ya que el ángulo de Euler necesario para contrarrestar la resistencia es menor.

En cuanto a los estados internos correspondientes a la velocidad lineal y angular, cuyas comparaciones se muestran en las figuras 8.17 y 8.18, se puede observar que las diferencias mostradas son reducidas explicable debido a las diferencias en el modelado entre AirSim y Simulink y debido al tipo de control continuo en Simulink y discreto de AirSim.

A pesar de las diferencias, que pueden ser consideradas como despreciables, es posible observar cómo las tendencias observadas en la evolución de los estados internos del quadrotor son las mismas. Estas figuras pueden ser utilizadas para validar el sistema dinámico del quadrotor y validar que pueda ser utilizado para el diseño de las leyes de control, ya que, como se puede observar, el sistema de control es capaz de seguir las referencias impuestas incluso con las correspondientes

diferencias en el modelado. Por ello, también se puede validar el sistema de control y asegurar por medio de la simulación que cumple con las características dinámicas y que podría ser utilizado en el sistema de control en un quadrotor real.

8.3 Seguimiento de trayectoria y simulaciones

En la siguiente sección se presentan los algoritmos introducidos en el capítulo 6 para conseguir un determinado control de trayectoria del quadrotor. En el primero de los algoritmos se presentaba uno genérico para cualquier tipo de UAV, llamado *Non Lineal Guidance Law* o NLGL y que fue introducido en la sección 6.1. Este algoritmo no permite el seguimiento de una trayectoria a diferentes alturas pero fue modificado para permitirlo, aunque el vuelo en trayectorias del tipo ascendente helicoidal o con tramos que pasan uno muy cerca de otros presenta dificultades para conseguir el seguimiento de la trayectoria, el rango de aplicación de este algoritmo está limitado a trayectorias rectas o con pocas curvas.

El segundo de ellos consiste en el seguimiento de una trayectoria mediante el uso de un control de posición cuya referencia cambia con el tiempo para conseguir que el quadrotor siga la determinada trayectoria. Este control de trayectoria se introdujo en la sección 6.2.

Debido al mayor rango de aplicabilidad del segundo algoritmo, su comportamiento será simulado mediante Simulink y AirSim para comprobar su validez en ambos entornos. Además, puesto que un algoritmo de seguimiento de trayectoria debe de ser capaz de corregir las posibles perturbaciones que puedan haber debido al viento u otros agentes externos, se realizará una posterior simulación en Simulink donde se asumirán rachas de viento variables que tiendan a sacar al quadrotor fuera de su trayectoria a seguir. Cabe destacar que en AirSim se puede introducir la presencia del viento mediante la modificación del documento de 'settings.json' presente en la carpeta de documentos del sistema. Sin embargo, no permite introducir rachas de viento variable y únicamente se simulará este fenómeno en Simulink.

8.3.1. Control de trayectoria por control de posición

Generación de la trayectoria a partir de *waypoints*

En un primer lugar se introducirá el método que ha sido utilizado en la generación de la trayectoria. La obtención de esta se hará a partir de un conjunto de *waypoints* por donde se desea que pase el quadrotor. En la figura 8.19 se muestra un ejemplo de trayectoria compuesto por estos puntos de interés, cabe destacar que se asumirá que los tramos que unen dos *waypoints* sucesivos serán asumidos rectos o curvos en función de lo deseado. Además, en función de si son curvos o rectos, la generación de la trayectoria se hará de una forma u otra.

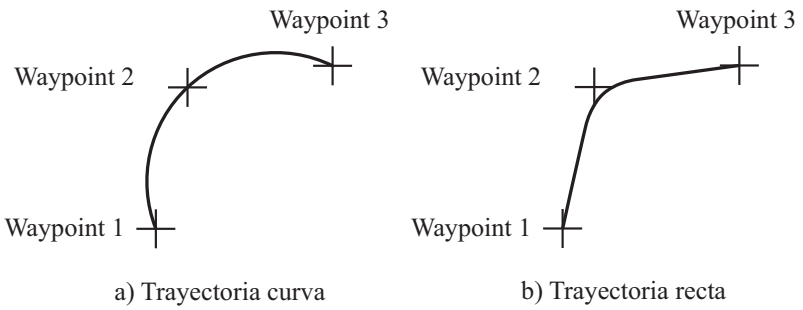


Figura 8.19: Generación de trayectoria a partir de una serie de waypoints

En el caso que se asuma que los tramos que unen dos *waypoints* sucesivos son curvos, se utilizará una interpolación mediante el uso de *splines* para generar los puntos intermedios por donde deberá pasar el quadrotor. En caso contrario se utilizará una interpolación lineal entre el rango intermedio de los sucesivos *waypoints* para generar la trayectoria intermedia, sin embargo, en las zonas cercanas a cada *waypoint* que formarían una esquina no se interpolará ningún punto y además se eliminará el *waypoint*, de esta forma se suavizarán las curvas y se evitarán cambios bruscos de dirección. Cabe destacar que para cada *waypoint* se especificará una velocidad, una altura y un ángulo de guiñada con la que el quadrotor debe pasar, la interpolación de estos parámetros puede ser realizada mediante una interpolación lineal o curva, al igual que la trayectoria en el plano horizontal. En cuanto a la guiñada también existe la posibilidad que sea paralela a la velocidad.

A continuación se muestra la implementación de la función utilizada para generar la trayectoria a seguir por el quadrotor, código 8.2.

```
function [waypoints] = ...
    trayectory_generation(waypoints,type_int,type_yaw,r,t)

%% WAYPOINTS COORDINATES, YAW AND SPEED
x   = waypoints(:,1);
y   = waypoints(:,2);
z   = waypoints(:,3);
yaw = waypoints(:,4);
v   = waypoints(:,5);

% Curvilinear coordinate distance in horizontal plane
s = ...
    cumsum(sqrt(sum((waypoints(2:end,1:2)-waypoints(1:end-1,1:2)).^2,2)));

% If interpolation function is lineal: Intermediate lineal points are
% calculated and saved in vector sq: Curvilinear distances
% If 1: lineal interpolation, If other: curved interpolation
if type_int == 1
    npoints = 100;
    sq      = zeros(length(s),npoints);
    lows    = [0;s+r];
    upps    = [s-r;0];
    for i=1:length(s)
        sq(i,:) = linspace(lows(i),upps(i),npoints);
    end
    sq = reshape(sq',1,[])';
else
    sq = [0;s];
end
s  = [0;s];

% Coordinates of intermediate points
xq   = interp1(s,x,sq,'linear');
yq   = interp1(s,y,sq,'linear');
zq   = interp1(s,z,sq,'linear');
yawq = interp1(s,yaw,sq,'linear');
vq   = interp1(s,v,sq,'linear');

% Coordinates of points in the trajectory
sq2   = 0:0.01:s(end);
xq2   = interp1(sq,xq,sq2,'spline');
yq2   = interp1(sq,yq,sq2,'spline');
zq2   = interp1(sq,zq,sq2,'spline');
yawq2 = interp1(sq,yawq,sq2,'spline');
vq2   = interp1(sq,vq,sq2,'spline');

```

```
% Check type of yaw: If 1 yaw parallel to velocity
if type_yaw==1
    yawq2 = atan2(yq2(2:end)-yq2(1:end-1),xq2(2:end)-xq2(1:end-1));
    yawq2 = [yawq2,yawq2(end)];
end

% References as function of time considering speed
s_check = 0;
waypoints = zeros(length(t),5);
for i = 1:length(t)-1
    [¬,ind]           = min(abs(sq2-s_check));
    waypoints(i+1,:) = [xq2(ind),yq2(ind),zq2(ind),yawq2(ind),vq2(ind)];
    s_check           = s_check+vq2(ind)*(t(i+1)-t(i));
end
waypoints(1,:) = waypoints(2,:);

end
```

Código 8.2: Generación de la trayectoria a partir de *waypoints*

La información que debe ser suministrada a esta función es una matriz cuyas filas son cada uno de los *waypoints*. Las tres primeras columnas se corresponden con la posición en ejes globales del waypoint, la cuarta columna es el ángulo de guiñada con el cual el quadrotor debe pasar por ese *waypoint* y la última columna se corresponde con la velocidad de paso por el *waypoint*. A parte se debe suministrar que tipo de interpolación se desea realizar, *spline* o lineal, según se muestra en la figura 8.19. Si el parámetro *type_int* es 1 la interpolación será lineal, en caso contrario será curva. Por otro lado, si el parámetro *type_yaw* es 1 la guiñada será paralela a la velocidad del quadrotor a lo largo de toda la trayectoria. Los últimos dos parámetros se corresponden con el radio que se utilizará para suavizar las esquinas en caso que el tipo de interpolación sea lineal, r, y el vector de tiempos donde se debe evaluar la referencia que será suministrada al quadrotor para seguir la referencia.

El primer paso que se sigue en la función es el cálculo de la coordenada curvilínea de cada uno de los *waypoints*, es decir, a que distancia se encuentran del origen asumiendo que los tramos son rectos. Si el tipo de interpolación es lineal, se calculan los puntos intermedios de cada tramo que será lineal, en estos puntos se excluyen los puntos de intersección de cada tramo. Posteriormente se produce la interpolación en cada uno de los tramos para cada una de las variables. Si se ha impuesto que la guiñada esté alineada con la velocidad se calcula este parámetro en cada punto interpolado.

La salida de la función se corresponde con una matriz que contiene el valor de las referencias de posición X, Y y Z y orientación ψ en función del tiempo impuesto como entrada que permiten satisfacer que la trayectoria se va a recorrer a la velocidad impuesta.

Número de Waypoint	X [m]	Y [m]	Z [m]	ψ [º]	V [m/s]
1	0	0	1	0	5
2	81.5	0	5	0	7
3	81.5	130.1	10	0	10
4	-128.1	130.1	15	0	10
5	-128.1	0	20	0	7
6	0	0	25	0	2

Tabla 8.1: Waypoints utilizados en la generación de la trayectoria

Para comprobar la validez del control de trayectoria mediante el control de posición se va a generar una trayectoria que pase por una serie de *waypoints* determinados. Cabe destacar que el *waypoint*

inicial en el momento de la generación de la trayectoria debe de ubicarse en la posición inicial del quadrotor, asegurando que este se encuentra con velocidad nula para conseguir que el seguimiento de la referencia sea correcto, en caso contrario puede suceder que las referencias de posición se alejen demasiado rápido del quadrotor causando su inestabilidad. También se debe garantizar que la velocidad de vuelo de paso en cada uno de los *waypoints* está dentro del rango de velocidades de vuelo admisibles del quadrotor.

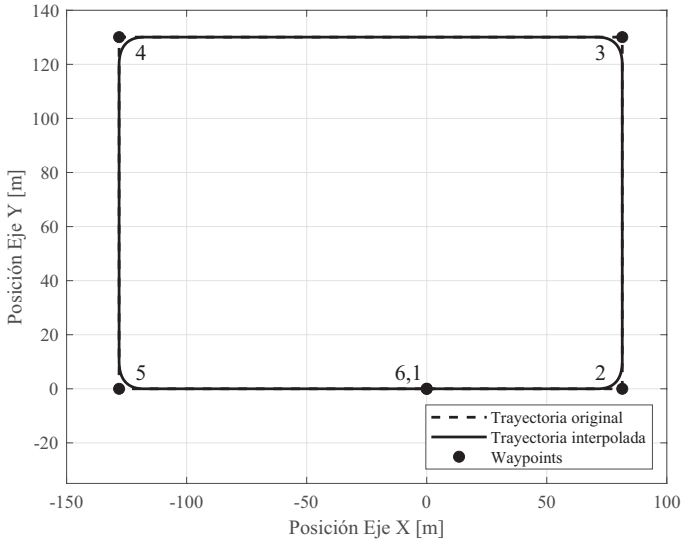


Figura 8.20: Interpolación de la trayectoria a seguir por el quadrotor mediante el uso de *splines*

La trayectoria que será generada se corresponde con un vuelo rectangular simulando el seguimiento de las calles de una manzana en una ciudad. En la tabla 8.1 se muestran las coordenadas de cada *waypoint* junto con sus velocidades y guiñadas de paso. Cabe destacar que, como se ha impuesto anteriormente, el *waypoint* inicial se corresponde con el la posición inicial del quadrotor aunque con una altura ligeramente superior para asegurar que despegue del suelo en el momento que se empiece a seguir la referencia. En la trayectoria seguida han asumido cambios de altura de vuelo y velocidad para validar que el algoritmo utilizado sea capaz de garantizar el seguimiento de la altura. En cuanto al valor de la guiñada, se ha asumido que será en todo momento alineada con el vector velocidad de vuelo aunque en la tabla se han puesto ceros para rellenarla.

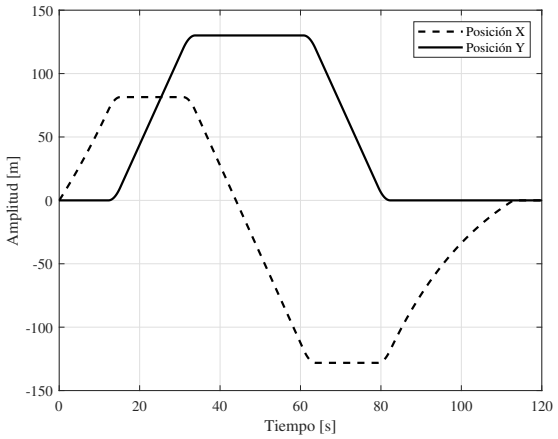


Figura 8.21: Referencias de la posición del quadrotor en ejes X e Y en función de la distancia curvilínea

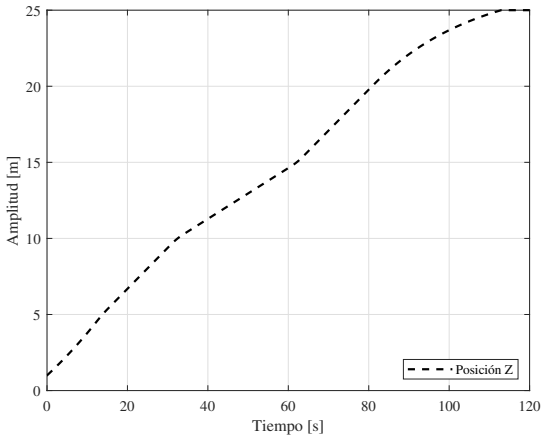


Figura 8.22: Referencias de la posición del quadrotor en eje Z en función de la distancia curvilínea

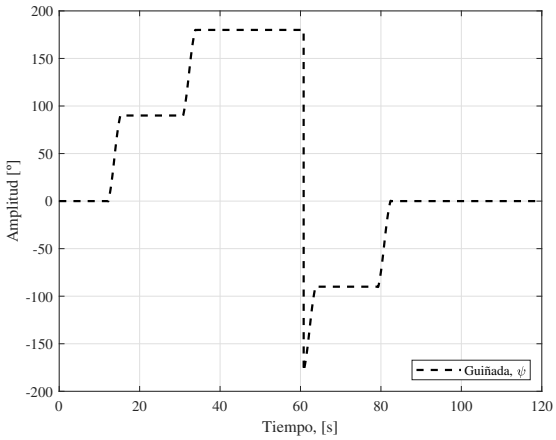


Figura 8.23: Referencias de la guiñada del quadrotor en función de la distancia curvilínea

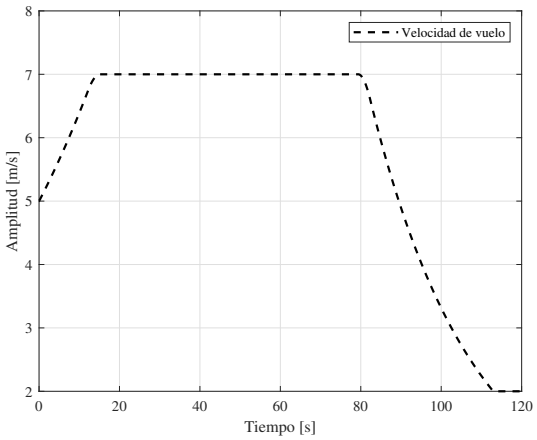


Figura 8.24: Referencias de la velocidad del quadrotor en función de la distancia curvilínea

En la figura 8.20 se representa gráficamente la disposición de estos *waypoints* en el espacio en dos dimensiones. Además se incluye sobre esta figura la trayectoria interpolada que seguirá el quadrotor utilizando la interpolación lineal con el uso de *splines* en las esquinas donde se intersectan los dos tramos rectos.

Por otro lado, en las figuras 8.21, 8.22, 8.23 y 8.24 se representa la evolución de cada una de las referencias que son suministradas al sistema con el tiempo. Cabe destacar que la velocidad no es en sí una entrada del sistema, sino que se cumplirá esta referencia mediante el cumplimiento de las referencias de posición.

Validación del control de trayectoria mediante el control de posición

Para validar el funcionamiento del control de la trayectoria se simularán las respuestas del sistema mediante el simulador de Simulink diseñado en el presente documento y el simulador de AirSim. Cabe destacar que las referencias de X,Y,Z y ψ serán impuestas de la forma que se mostró en las figuras 8.21, 8.22 y 8.23.

En las figuras 8.25 y 8.26 se representan las trayectorias de referencias que tiene que seguir el quadrotor y la respuesta del sistema para conseguir el seguimiento de la referencia siendo la figura de la izquierda la respuesta de Simulink y la figura de la derecha la respuesta de AirSim.

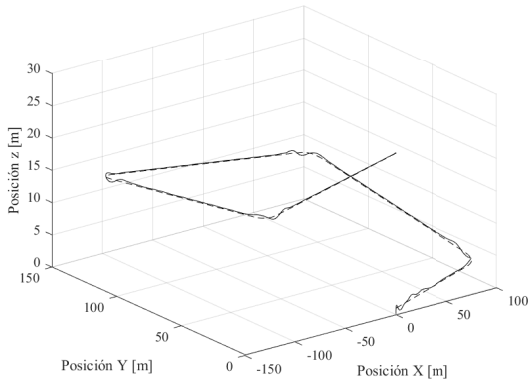


Figura 8.25: Trayectoria seguida por el quadrotor (línea continua) ante una trayectoria de referencia (línea discontinua): Simulink

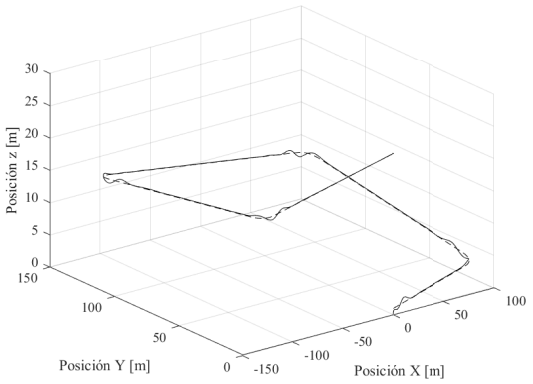


Figura 8.26: Trayectoria seguida por el quadrotor (línea continua) ante una trayectoria de referencia (línea discontinua): AirSim

Como se puede observar, el sistema de control es capaz de seguir la trayectoria de referencia en Simulink y AirSim de forma bastante precisa. En los cambios de tramos se pueden observar disminuciones de altura de vuelo debido al aumento de los ángulos de Euler para cambiar la dirección de la velocidad, sin embargo, tras realizar el giro, el quadrotor se vuelve a estabilizar en el seguimiento de la trayectoria.

Para comparar la respuesta de cada uno de los simuladores, Simulink y AirSim, en las figuras 8.27, 8.28, 8.29 y 8.30 se muestran los valores de las variables controladas a lo largo de la coordenada curvilínea de la trayectoria proyectada en el pano horizontal.

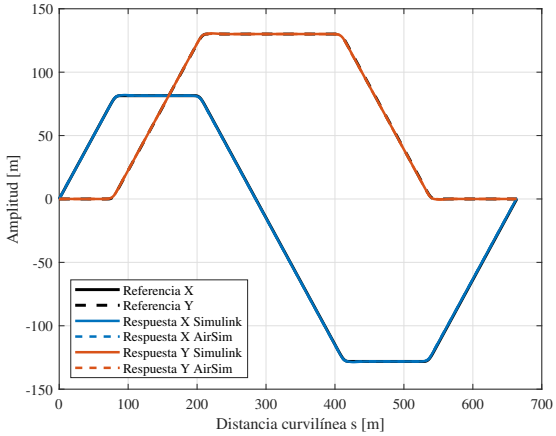


Figura 8.27: Comparación entre AirSim y Simulink de la posición de X e Y en el seguimiento de la trayectoria

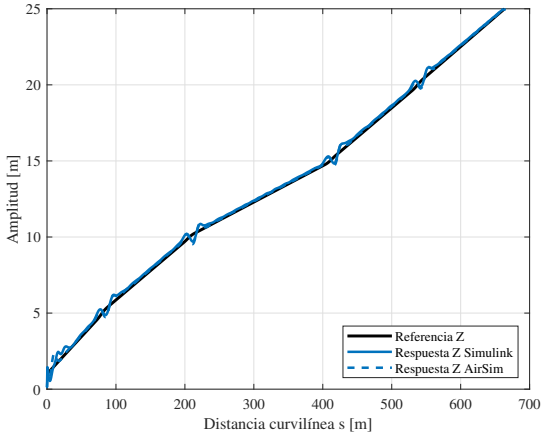


Figura 8.28: Comparación entre AirSim y Simulink de la posición de Z en el seguimiento de la trayectoria

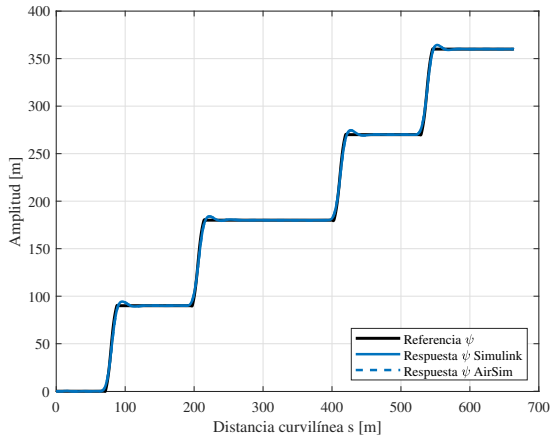


Figura 8.29: Comparación entre AirSim y Simulink de la posición de ψ en el seguimiento de la trayectoria

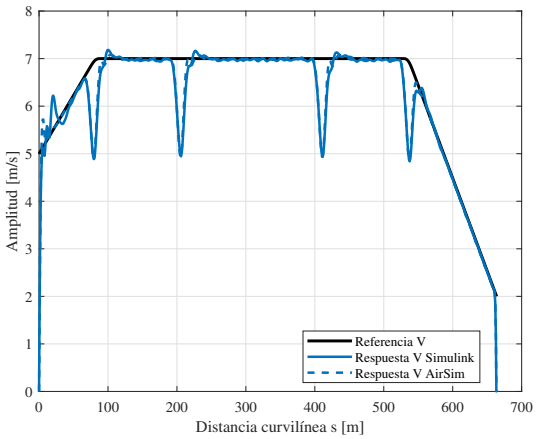


Figura 8.30: Comparación entre AirSim y Simulink de la velocidad en el seguimiento de la trayectoria

Como se puede observar, en el seguimiento de las posiciones en el plano horizontal no existen apenas diferencias entre las referencias y las posiciones del quadrotor calculadas en Simulink y AirSim lo que da una validación más del modelo dinámico desarrollado en el presente documento. En cuanto al control de trayectoria se puede concluir que el seguimiento de la referencia en el plano horizontal es válida.

En cuanto al seguimiento de la altura se puede observar como el quadrotor es capaz de seguir en todo momento la referencia de altura impuesta, tanto en el simulador desarrollado en Simulink y en AirSim. Sin embargo, se pueden observar ciertas perturbaciones en los cambios de dirección que son debido al aumento de los ángulos de Euler para cambiar la dirección de la velocidad

del quadrotor. Tras un cierto transitorio, el quadrotor es capaz de volver a estabilizar su altura y continuar el seguimiento de la referencia.

Por otro lado, el seguimiento del ángulo de guiñada impuesto es capaz de cumplir con las especificaciones deseadas. Como se puede observar existe una ligera sobreoscilación en los cambios de referencia que es capaz de estabilizar el sistema de control.

Finalmente el seguimiento de la velocidad se ajusta a la referencia a lo largo de la trayectoria, excepto en los cambios de tramos debido a que el quadrotor debe frenar en una dirección del tramo de salida y acelerar en la dirección del tramo de salida. Es decir, la reducción de la velocidad se debe al cambio de dirección del vector velocidad. Sin embargo, tras un ligero transitorio el quadrotor es capaz de continuar con el seguimiento de la referencia.

Por tanto, observando las comparaciones entre los diferentes simuladores en el seguimiento de la trayectoria es posible validar el sistema de control empleado para conseguir el seguimiento de las referencias. Además, como se ha podido comprobar, la respuesta del sistema de control en ambos simuladores es idéntica por lo que se consigue validar de nuevo el modelo dinámico.

Robustez del control de trayectoria mediante el control de posición ante rachas de viento del tipo armónico

En esta sección se va a evaluar el comportamiento del sistema de control del quadrotor ante fuertes rachas de viento que van a ser consideradas armónicas. Este tipo de perturbaciones suelen requerir de bastante coste de las leyes de control ya que su amplitud y signo varían con el tiempo. Por ello es necesario que las acciones de control calculadas por el sistema sean capaces de estabilizar y garantizar el seguimiento de las referencias incluso en estas condiciones. Mediante la simulación de dos casos diferentes se va a poder estudiar el comportamiento del control de trayectoria ante estas perturbaciones.

Se van a realizar dos simulaciones, la primera de ellas con velocidad de la corriente de aire igual a 5 m/s y la segunda de ellas con velocidad de corriente de aire igual a 10 m/s. En cuanto a la frecuencia de las leyes armónicas de la perturbación se va a asumir constante en los dos casos e igual a 1 Hz.

Por otro lado, cabe destacar que esta validación se va a realizar únicamente en Simulink, puesto que en AirSim no se encuentra disponible la posibilidad de introducir rachas de viento variables.

En las figuras 8.31, 8.32, 8.33, 8.34 y 8.35 se representan los casos donde se asumen rachas de viento de amplitud igual a 5 m con frecuencia de 1 Hz. En un primer lugar se muestra el seguimiento tridimensional de la trayectoria y posteriormente la evolución de las variables controladas a lo largo del tiempo.

En las figuras 8.36, 8.37, 8.38, 8.39 y 8.40 se representan los casos donde se asumen rachas de viento de amplitud igual a 10 m con frecuencia de 1 Hz. En un primer lugar se muestra el seguimiento tridimensional de la trayectoria y posteriormente la evolución de las variables controladas a lo largo del tiempo.

Como se puede observar, la ley de control es capaz de estabilizar el sistema ante las perturbaciones experimentadas y además permite garantizar el seguimiento de las referencias. A medida que aumenta la amplitud de las rachas de viento no se experimentan fuertes variaciones en el seguimiento de la trayectoria sino que el quadrotor es capaz de seguir en todo momento las referencias de igual forma que en el caso sin ellas. En cuanto al seguimiento de la velocidad se aprecian oscilaciones en el valor estacionario que tienden a aumentar la amplitud cuando aumenta la de las rachas de viento debido a que el quadrotor debe modificar sus ángulos de cabeceo y alabeo para contrarrestar los efectos de las rachas de viento. De esta forma es posible ver como el sistema de control es capaz de cumplir con el tercero de sus objetivos que es el rechazo de perturbaciones.

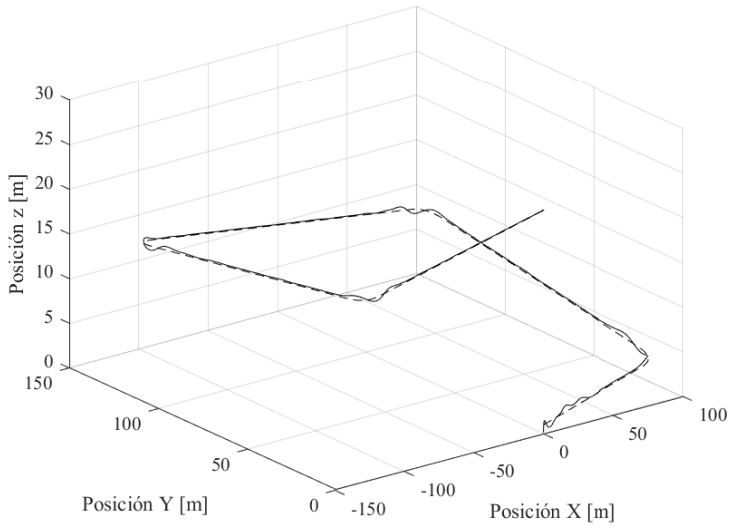


Figura 8.31: Seguimiento de la trayectoria tridimensional bajo perturbaciones de rachas de vientos armónicas de amplitud 5 m/s y frecuencia 1 Hz

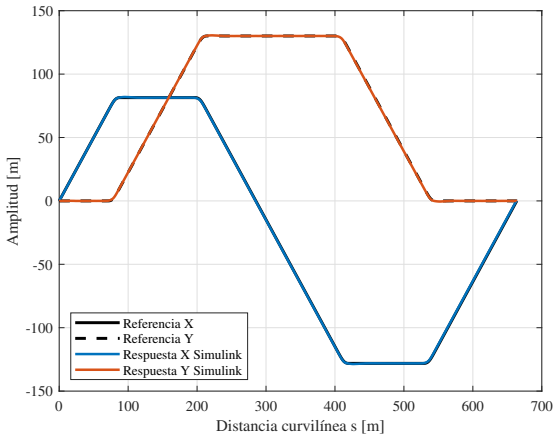


Figura 8.32: Seguimiento de la referencia de posición en X e Y bajo perturbaciones de rachas de vientos armónicas de amplitud 5 m/s y frecuencia 1 Hz

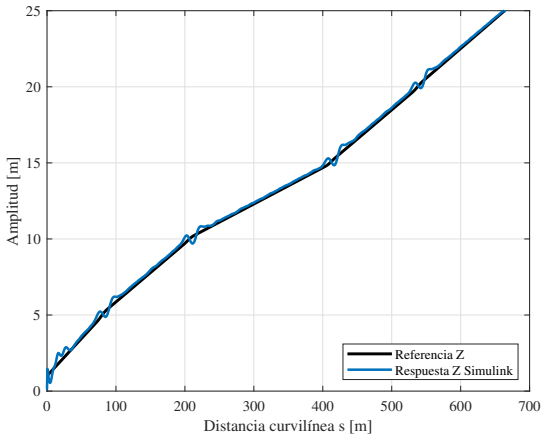


Figura 8.33: Seguimiento de la referencia de posición en Z bajo perturbaciones de rachas de vientos armónicas de amplitud 5 m/s y frecuencia 1 Hz

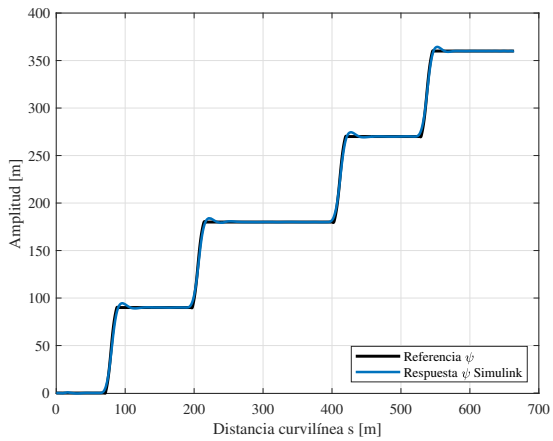


Figura 8.34: Seguimiento de la referencia de posición en ψ bajo perturbaciones de rachas de vientos armónicas de amplitud 5 m/s y frecuencia 1 Hz

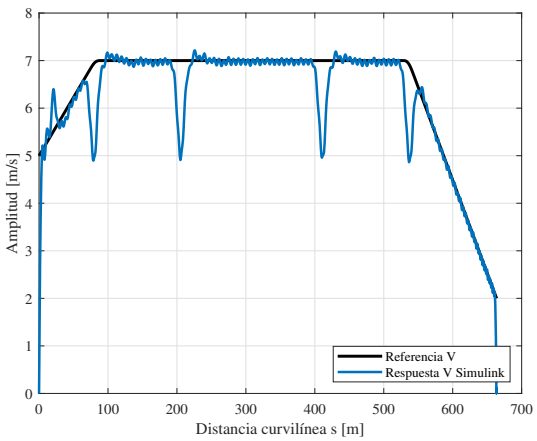


Figura 8.35: Seguimiento de la referencia de posición en V bajo perturbaciones de rachas de vientos armónicas de amplitud 5 m/s y frecuencia 1 Hz

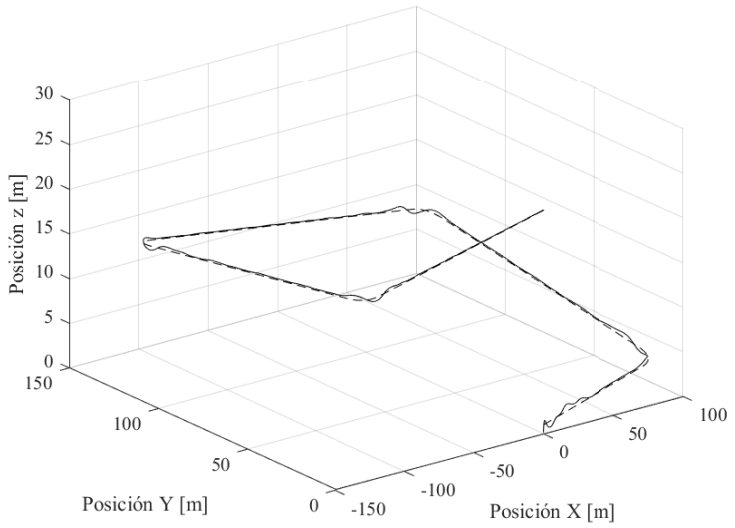


Figura 8.36: Seguimiento de la trayectoria tridimensional bajo perturbaciones de rachas de vientos armónicas de amplitud 10 m/s y frecuencia 1 Hz

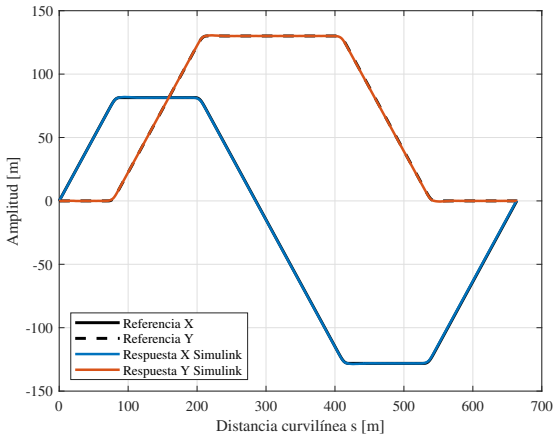


Figura 8.37: Seguimiento de la referencia de posición en X e Y bajo perturbaciones de rachas de vientos armónicas de amplitud 10 m/s y frecuencia 1 Hz

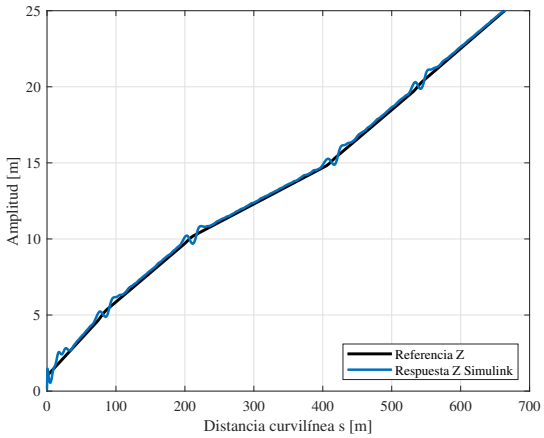


Figura 8.38: Seguimiento de la referencia de posición en Z bajo perturbaciones de rachas de vientos armónicas de amplitud 10 m/s y frecuencia 1 Hz

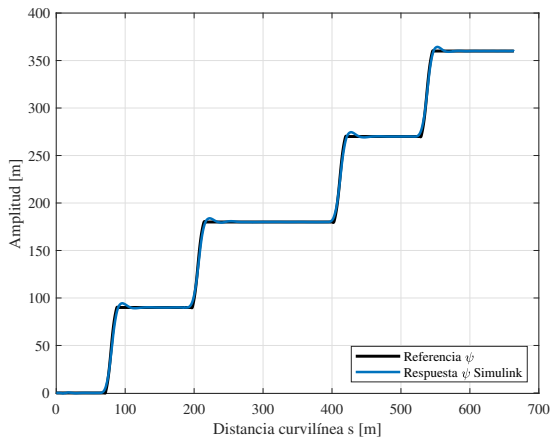


Figura 8.39: Seguimiento de la referencia de posición en ψ bajo perturbaciones de rachas de vientos armónicas de amplitud 10 m/s y frecuencia 1 Hz

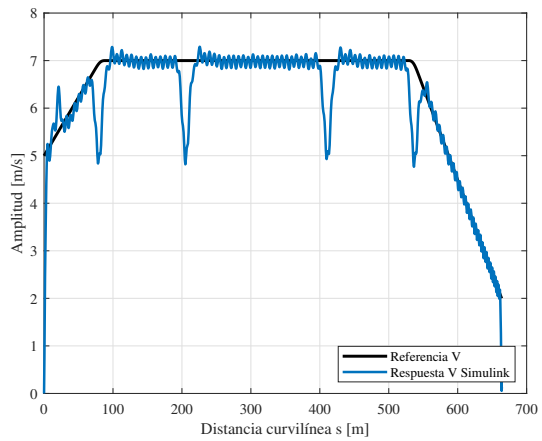


Figura 8.40: Seguimiento de la referencia de posición en V bajo perturbaciones de rachas de vientos armónicas de amplitud 10 m/s y frecuencia 1 Hz

9. Conclusiones y futuros trabajos

A lo largo del presente trabajo se han ido cumpliendo los diferentes pasos necesarios para diseñar el sistema de seguimiento de trayectoria de un quadrotor. En un primer lugar se llevó a cabo el modelado del sistema dinámico del movimiento de un quadrotor con 6 gdl. Los requisitos que se impusieron a este modelo era que fuese lo más realistas posible y que se incluyesen los fenómenos más representativos que afectan a la dinámica del quadrotor. Por ello se asumen los términos relacionados con el contacto con el suelo y el aumento de la tracción sobre cada uno de los rotores debido a su actuación a bajas alturas. Mediante el modelado de un sistema lo más cercano a la realidad se podrá facilitar la tarea del ajuste de las ganancias del regulador de forma más precisa, obteniendo un control que pueda ser extrapolado a una aplicación real y donde serán necesarios pocos cambios. De esta forma, el modelo dinámico utilizado en simulación constituye una opción válida para obtener las primeras iteraciones de las leyes de control rápidamente y prácticamente sin coste o riesgo.

El diseño de las leyes de control del quadrotor se ha considerado lineal, en concreto se ha utilizado la ley de control LQR. Las hipótesis consideradas para el desarrollo del sistema de control se corresponden con pequeños desplazamientos alrededor del equilibrio del quadrotor que es el vuelo en punto fijo alineado con el norte. Para la mayoría de los estados esta suposición es lo suficientemente aproximada a la realidad como para permitir que la actuación de las leyes de control sea la correcta, sin embargo la guiñada no la cumple. Debido a que puede tomar cualquier valor comprendido entre -180° y 180°, se ha utilizado el control LQR con una planificación de ganancia, de tal forma que el controlador es capaz de ajustarse al estado actual del quadrotor.

El controlador del quadrotor se ha diseñado mediante el uso de controladores en cascada. El del bucle interno se encarga de controlar los estados correspondientes a los ángulos de Euler y la altura, por el otro lado el bucle externo proporciona las referencias a seguir por el interno a partir de unas referencias de posición o velocidad, según sea necesario.

El controlador de trayectoria es el que se encarga de generar las referencias de posición o velocidad para permitir el seguimiento de la trayectoria a partir de unos *waypoints* que han sido previamente definidos. En la generación de la trayectoria a seguir se deben de suavizar las intersecciones de cada tramo mediante el uso de *splines* o arcos con un determinado radio, de tal forma que el camino deseado presente continuidad por lo menos en la primera derivada.

El algoritmo implementado para conseguir el seguimiento de la trayectoria está basado en un control de posición del quadrotor. De esta forma se impone la referencia de posición del quadrotor y se modifica con el tiempo para que éste sea capaz de pasar por los puntos previamente definidos con unos ciertos estados. En concreto se pueden definir las velocidades de paso y sus ángulos de guiñada.

La validación del sistema de control que permite seguir una determinada trayectoria y del modelo dinámico desarrollado se ha hecho en el simulador de código abierto de Microsoft AirSim. Se ha utilizado el mismo quadrotor en ambas simulaciones obteniendo resultados próximos en ambos

casos. De esta forma se ha permitido validar cada una de las simulaciones presentadas para probar el control de trayectoria.

Posteriormente se comprueba la respuesta del sistema de seguimiento de trayectoria ante fuertes perturbaciones de viento. Cabe destacar que esta es una de las misiones que debe cumplir un sistema de control de trayectoria, asegurar su seguimiento incluso con fuertes perturbaciones que tiendan a sacarlo de ella. Las conclusiones que se obtienen del estudio es que el controlador diseñado es capaz de rechazar las diferentes perturbaciones y conseguir la estabilización del quadrotor para garantizar el seguimiento.

AirSim demuestra en el presente trabajo ser una herramienta que permite ser utilizada para simular la dinámica del movimiento de un quadrotor. De esta forma se ha podido utilizar para validar el sistema de control en el dominio discreto mediante la conexión con un programa de Python. Este simulador presenta un gran potencial en aplicaciones de control, concretamente en aquellas con visión artificial, aprendizaje reforzado o aprendizaje por imitación. Por otro lado se propone la implementación de detección de obstáculos en el seguimiento de la trayectoria que permita evaluar acciones correctoras para evitar colisiones.

Bibliografía

[1] G. Singhal, B. Bansod, and L. Mathew, "Unmanned Aerial Vehicle Classification , Applications and Challenges : A Review," *Preprint*, no. November, pp. 1–19, 2018.

[2] Q. Quan, *Introduction to multicopter design and control*. Springer Singapore, jun 2017.

[3] W. R. Young, *The Helicopters (The Epic of Flight)*. 1982.

[4] R. Gariepy, "Quadrotor Position Estimation using Low Quality Images," *Uwspace.Uwaterloo.Ca*, 2011.

[5] R. Amin, L. Aijun, and S. Shamshirband, "A review of quadrotor UAV: Control methodologies and performance evaluation," *International Journal of Automation and Control*, vol. 10, no. 2, pp. 87–103, 2016.

[6] M. A. Lotufo, L. Colangelo, and C. Novara, "Feedback Linearization for Quadrotors UAV," no. 1, pp. 1–4, 2019.

[7] C. Wang, Z. Chen, and M. Sun, "Sliding mode control of a quadrotor helicopter," *Zhongnan Daxue Xuebao (Ziran Kexue Ban)/Journal of Central South University (Science and Technology)*, vol. 48, no. 4, pp. 1006–1011, 2017.

[8] A. L. Salih, M. Moghavvemi, H. A. Mohamed, and K. S. Gaeid, "Modelling and PID controller design for a quadrotor unmanned air vehicle," in *2010 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2010 - Proceedings*, vol. 1, pp. 74–78, 2010.

[9] L. Ding and Z. Wang, "A Robust Control for an Aerial Robot Quadrotor under Wind Gusts," *Journal of Robotics*, vol. 2018, 2018.

[10] S. Khatoon, D. Gupta, and L. K. Das, "PID & LQR control for a quadrotor: Modeling and simulation," in *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, pp. 796–802, Institute of Electrical and Electronics Engineers Inc., nov 2014.

[11] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 279–284, Institute of Electrical and Electronics Engineers Inc., 2012.

[12] B. Rubí, R. Pérez, and B. Morcego, "A Survey of Path Following Control Strategies for UAVs Focused on Quadrotors," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 98, no. 2, pp. 241–265, 2020.

[13] I. Kaminer, O. Yakimenko, A. Pascoal, and R. Ghabcheloo, "Path Generation, Path Following and Coordinated Control for TimeCritical Missions of Multiple UAVs," in *2006 American Control Conference*, pp. 4906–4913, IEEE, 2006.

[14] D. Cabecinhas, R. Cunha, and C. Silvestre, "Rotorcraft path following control for extended flight envelope coverage," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 3460–3465, 2009.

[15] A. Akhtar, S. L. Waslander, and C. Nielsen, "Path following for a quadrotor using dynamic extension and transverse feedback linearization," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3551–3556, 2012.

[16] P. B. Sujit, S. Saripalli, and J. B. Sousa, "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, 2014.

[17] M. Elfeky, M. Elshafei, A. W. A. Saif, and M. F. Al-Malki, "Quadrotor helicopter with tilting rotors: Modeling and simulation," in *2013 World Congress on Computer and Information Technology, WCCIT 2013*, 2013.

[18] H. Efraim, A. Shapiro, and G. Weiss, "Quadrotor with a Dihedral Angle: on the Effects of Tilting the Rotors Inwards," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 80, no. 2, pp. 313–324, 2015.

[19] M. Bangura, "Aerodynamics and Control of Quadrotors," no. February, pp. 1–163, 2017.

[20] P. Sanchez-Cuevas, G. Heredia, and A. Ollero, "Characterization of the aerodynamic ground effect and its influence in multirotor control," *International Journal of Aerospace Engineering*, vol. 2017, 2017.

[21] D. Abeywardena, S. Kodagoda, G. Dissanayake, and R. Munasinghe, "Improved state estimation in quadrotor MAVs: A novel drift-free velocity estimator," *IEEE Robotics and Automation Magazine*, vol. 20, no. 4, pp. 32–39, 2013.

[22] P. Ventura Diaz and S. Yoon, "High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles," *AIAA Aerospace Sciences Meeting, 2018*, no. 210059, pp. 1–22, 2018.

[23] B. J. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annual Reviews in Control*, vol. 46, no. xxxx, pp. 165–180, 2018.

[24] C. D. McKinnon and A. P. Schoellig, "Unscented external force and torque estimation for quadrotors," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 5651–5657, Institute of Electrical and Electronics Engineers Inc., nov 2016.

[25] K. Ogata, "Modern control engineering," 2010.

[26] R. &. A. . Lectures, "LQR/LQG OPTIMAL CONTROL LECTURE 20 Linear Quadratic Regulation (LQR)," tech. rep., University of Queensland, Australia.

[27] T. Marada, R. Matoušek, and D. Zuth, "Design of linear quadratic regulator (LQR) based on genetic algorithm for inverted pendulum," *Mendel*, vol. 23, no. 1, pp. 149–156, 2017.

[28] A. Sanchez-Orta, V. Parra-Vega, C. Izaguirre-Espinosa, and O. Garcia, "Position-yaw tracking of quadrotors," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 137, jun 2015.

[29] T. Luukkonen, *Modelling and Control of a Quadcopter*. PhD thesis, Aalto University, 2011.

[30] A. Reizenstein, "Position and Trajectory Control of a Quadcopter Using PID and LQ Controllers," *Linköping University*, p. 81, 2017.

[31] H. P. Brown R.G., *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises-Wiley (2012)*.

[32] G. Perez, D. Gaydou, C. Paz, and L. Canali, "Experimental comparison of kalman and complementary filter for attitude estimation," *Proceedings of the AST*, no. August, pp. 205–215, 2011.

[33] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. De Lellis, and A. Pironti, "Path generation and tracking in 3-D for UAVs," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 4, pp. 980–988, 2009.

[34] Z. Chen, F. Luo, and C. Zhai, "Obstacle avoidance strategy for quadrotor uav based on improved particle swarm optimization algorithm," in *Chinese Control Conference, CCC*, vol. 2019-July, pp. 8115–8120, IEEE Computer Society, jul 2019.

[35] Á. Cuerva Tejero, J. L. Espino Granado, Ó. López, J. Meseguer Ruiz, and A. Sanz Andrés, *Teoría de los helicópteros*. Serie de ingeniería y tecnología aeroespacial, Madrid: Ibergarceta, 2ª ed. ed., 2012.

[36] A. C. Watts, V. G. Ambrosia, and E. A. Hinkley, "Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use," *Remote Sensing*, vol. 4, no. 6, pp. 1671–1692, 2012.

APÉNDICE A
Instalación de AirSim

La instalación del simulador AirSim puede realizarse de dos formas, siendo recomendable una de ellas u otra en función de las necesidades del usuario.

La primera opción es la descarga de los archivos binarios ya precompilados que permiten empezar inmediatamente a utilizar el simulador, sin embargo, no se podrán editar ni incluir nuevos escenarios o vehículos. Mediante la instalación haciendo uso de la primera opción no se tendrá capacidad alguna de modificación excepto algunas pequeñas configuraciones ambientales, de reloj o ruido. Por otro lado, es importante tener en cuenta que esta opción es únicamente válida para los sistemas operativos de Windows y Linux, en macOS la única opción válida de instalación es la segunda.

Para ambas opciones de instalación estas pequeñas configuraciones son comunes y pueden ser modificadas mediante el archivo 'settings.json' creado en la carpeta 'Documentos\AirSim' la primera vez que se lanza el programa. En la página de Github del simulador[1], se muestran todas las posibles modificaciones que se pueden realizar mediante este archivo. En el caso que se satisfagan todas las necesidades del usuario mediante el archivo 'settings.json', se podrá usar el simulador mediante los archivos precompilados sin necesidad de optar por la segunda opción de instalación que resulta un procedimiento más complejo. Además, cabe destacar que esta opción no requiere de instalación y es más sencilla para el usuario.

La descarga de los archivos binarios puede hacerse para dos motores gráficos distintos, Unreal Engine o Unity, aunque la segunda opción es antigua y todavía en fase experimental, por lo que no se asegura que funcionen todas las características ni las APIs y además no están disponibles todos los escenarios que si están para Unreal Engine. Es por ello que se recomienda la descarga de los archivos binarios del motor gráfico Unreal Engine.

De la página de Github[2] del simulador se pueden descargar todas las versiones antiguas y actuales para cada sistema operativo disponible. Existe un número elevado de escenarios donde poder simular cada vehículo y que van desde una ciudad hasta un campo de fútbol, por lo que el usuario puede seleccionar el que mejor se aproxime a sus necesidades. Es importante notar que únicamente es necesario descargar cada escenario en el que se desee simular y dentro de la carpeta descargada se encuentra un ejecutable que lanza el simulador. En el caso de que necesite un escenario que no tenga una versión precompilada deberá optar por la segunda opción de instalación e introducir el escenario de simulación deseado manualmente. Los vehículos que pueden ser utilizados están restringidos a un coche y un quadrotor con parámetros físicos ya definidos y que no pueden ser modificados, si se desean modificar las características físicas de alguno de los vehículos o introducir nuevas alternativas se deberá optar por la segunda opción de instalación e introducirlos manualmente.

[1] `https://microsoft.github.io/AirSim/settings/`
[2] `https://github.com/Microsoft/AirSim/releases`

En cuanto a la segunda opción, la recomendada si se desean realizar algunas de las modificaciones expuestas anteriormente o si se desea modificar el entorno de Unreal Engine, consiste en la compilación de AirSim y puede realizarse tanto en Windows como en Linux[3]. Para la compilación de AirSim en Unity se recomienda acceder a la página de Github del simulador[4]. En el procedimiento mostrado en el presente documento es únicamente válido para el motor gráfico de Unreal Engine en Windows[5].

A continuación se detallan los pasos para la correcta descarga y compilación del simulador AirSim.

1. En un primer lugar es necesaria la instalación del motor gráfico de Unreal Engine.
   a) Para ello es necesaria la descarga del *launcher* de 'Epic Games', compañía de videojuegos desarrolladora de este motor gráfico. Es importante considerar que necesita de un proceso de registro aunque sea de código abierto y de descarga gratuita.
   b) Una vez instalado el Epic Games Launcher y finalizado el proceso de registro se debe lanzar el programa, en la parte izquierda se encontrará la pestaña de Unreal Engine, tras hacer *click* en ella, aparecerán en la parte superior nuevas pestañas donde se deberá entrar en 'Library'. La ventana mostrará entonces las versiones del motor gráfico y los diferentes proyectos presentes en el sistema. Si no se ha hecho todavía es importante descargar la versión 4.24. En el caso que se tengan diferentes versiones instaladas, se debe marcar la versión 4.24 como 'current' de tal forma que si se inicia un proyecto será con esta versión.
2. A continuación se detallan los aspectos a considerar en la descarga y compilación de AirSim.
   a) Para el correcto seguimiento de la descarga e instalación de AirSim serán necesarios diversos *softwares*.
      1) Es necesaria la instalación de Visual Studio 2019. Es importante que en el proceso de instalación se seleccione las opciones 'Desktop Development with C++' y 'Windows 10 SDK 10.0.18362' que de normal son marcadas por defecto.
      2) Se debe instalar Git en Windows para ser capaces de clonar en una carpeta personal el repositorio de AirSim[6].
   b) Se debe iniciar la 'Developer Command Prompt for VS 2019'. Se recomienda para evitar posibles errores el modo administrador, además se recomienda el uso del comando 'cd' para moverse a un directorio de instalación conocido. Mediante 'cd ..' se puede mover hacia arriba una carpeta, mediante 'cd /' se puede acceder a la carpeta raíz. De esta forma una vez que se está en la carpeta raíz mediante 'cd Users/Josep/-Documents/AirSim' se puede acceder a la ubicación deseada de instalación.
   c) Una vez que se está en el directorio deseado de instalación se debe clonar el repositorio mediante el comando 'git clone https://github.com/Microsoft/AirSim.git' y acceder al directorio creado de AirSim mediante el comando 'cd AirSim'.
   d) Finalmente se debe lanzar 'build.cmd' desde la línea de comandos que creará el *plugin* en la carpeta 'Unreal\Plugins' y que será utilizado en cualquier proyecto de Unreal.
3. El último de los pasos a seguir consiste en compilación del proyecto de Unreal.
   a) Este consiste en la configuración del escenario y del ambiente. Existen dos opciones, la primera de ellas es utilizar el escenario que se encuentra construido dentro del simulador llamado 'Blocks' mientras que el segundo consiste en crear un escenario propio.

[3] `https://microsoft.github.io/AirSim/build_linux/`
[4] `https://microsoft.github.io/AirSim/Unity/`
[5] `https://microsoft.github.io/AirSim/build_windows/`
[6] `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`

Solamente se detalla la primera opción en este documento mientras que para la segunda se recomienda acceder a la página de Github del simulador [7]. Aunque el entorno de 'Blocks' no esté muy detallado, el tamaño es lo suficientemente bajo como para permitir una rápida compilación y ser la mejor forma de comenzar a introducirse.

b) Se debe navegar en un primer lugar al directorio donde se ha clonado el repositorio y una vez ahí navegar a la carpeta 'AirSim\Unreal\Environments\Blocks'. En esta ubicación se debe lanzar el archivo 'update_from_git.bat'.

c) A continuación se debe hacer *click* en el archivo '.sln' generado para abrir Visual Studio 2019. Se debe asegurar que el proyecto 'Blocks' está marcado como 'startup' y que la configuración de compilación es 'DebugGame_Editor' y 'Win64'. Finalmente apretar la tecla 'F5'.

d) Se habrá creado el proyecto en la ventana de Epic Games mencionada anteriomente y donde se podrá lanzar el simulador con el entorno de 'Blocks'. También es posible lanzar el simulador haciendo doble *click* en el archivo '.sln' en 'AirSim \Unreal \Environments \Blocks'.

Cualquier modificación que se haga en el código será necesario que se siga un flujo de trabajo para asegurar que lo cambios quedan guardados. Como se ha podido observar AirSim ha sido diseñado como un *plugin* y no puede correr por él solo si no que necesita ser puesto en un escenario de Unreal. De esta forma existen dos pasos, el primero de ellos consiste en la compilación del *plugin* y que se consigue con el comando 'build.cmd', mientras que el segundo consiste en ponerlo en un proyecto de Unreal y correrlo.

A continuación se detalla el flujo de trabajo a seguir para modificar el código de AirSim y verificar su funcionamiento[8].

1. Se deben utilizar los comandos en orden desde la 'Developer Command Window for VS 2019'. Una vez que se ha ejecutado el procedimiento se puede editar el código de AirSim desde Visual Studio y una vez que se termine apretar la tecla 'F5' y verificar los cambios.

   a) 'git pull'

   b) 'build.cmd'

   c) 'cd Unreal\Environments\Blocks'

   d) 'update_from_git.bat'

   e) 'start Blocks.sln'

2. Una vez que se ha terminado de modificar es posible que se deseen guardar los cambios en el repositorio local de AirSim. Con este propósito se deben ejecutar desde la línea de comandos de la 'Developer Command Window for VS 2019' los siguientes comandos:

   a) 'update_to_git.bat'

   b) 'build.cmd'

Es importante destacar que en el Github del simulador[9], se puede encontrar información más detallada del proceso de instalación aunque en el presente documento se han destacado las consideraciones y aspectos más importantes para conseguir el correcto compilado e instalación y facilitar el seguimiento al lector.

[7] `https://microsoft.github.io/AirSim/unreal_custenv/`

[8] `https://microsoft.github.io/AirSim/dev_workflow/`

[9] `https://microsoft.github.io/AirSim/`

APÉNDICE B
Códigos empleados en el control

En este capítulo del apéndice se muestran los códigos de los programas matlab y Python empleados para la validación del sistema dinámico y de control. Cabe destacar cada uno de los códigos de Python mostrados deberá ser ejecutado junto con el simulador de AirSim mientras que cada uno de los códigos de Matlab será ejecutado con el diagrama de bloques que representa la dinámica no lineal del sistema y de control en Simulink.

Para ello se adjuntan el código de Python y Matlab para el control de posición, el de velocidad no se incluye puesto que la estructura es idéntica, únicamente sería necesario variar los estados controlados en el controlador externo, sus referencias y las matrices Q y R del controlador.

En un primer lugar se adjunta el script de Matlab que debe de ser ejecutado para cargar cada uno de los parámetros físicos del problema mostrados en el capítulo 7, código B.1. La carga de este archivo con el nombre 'Physical_parameters_definitions' es común a todos los códigos de Matlab.

```
%% SYSTEM PARAMETERS

% Geometric Parameters
n_motor = 4;            % Number of motors
l       = 0.2275;       % Arm length [m]
l_arm_x = l*sqrt(2)/2;  % Arm length in X axis [m]
l_arm_y = l*sqrt(2)/2;  % Arm length in Y axis [m]
l_arm_z = 0.025;        % Arm length in Z axis [m]
l_box_x = 0.18;         % Central body length in X axis [m]
l_box_y = 0.11;         % Central body length in Y axis [m]
l_box_z = 0.04;         % Central body length in Z axis [m]
l_feet  = 0.1393;       % Feet length in Z axis [m]
Diam    = 0.2286;       % Rotor diameter [m]
Rad     = Diam/2;       % Rotor radius [m]

% Mass Parameters
m   = 1;                    % Mass [kg]
m_motor = 0.055;            % Motor mass [kg]
m_box = m-n_motor*m_motor; % Central body mass [kg]
g   = 9.8;                  % Gravity [m/s^2]
Ixx = m_box/12*(l_box_y^2+l_box_z^2)+(l_arm_y^2+l_arm_z^2)*m_motor*n_motor;
                                                % Inertia in X axis [kg m^2]
Iyy = m_box/12*(l_box_x^2+l_box_z^2)+(l_arm_x^2+l_arm_z^2)*m_motor*n_motor;
                                                % Inertia in Y axis [kg m^2]
Izz = m_box/12*(l_box_x^2+l_box_y^2)+(l_arm_x^2+l_arm_y^2)*m_motor*n_motor;
                                                % Inertia in Z axis [kg m^2]
Ir  = 2.03e-5; % Rotor inertia around spinning axis [kg m^2]

% Motor Parameters
max_rpm   = 6396.667;          % Rotor max RPM
```

```
max_omega = max_rpm/RADS2RPM; % Rotor max angular velocity [rad/s]
Tm        = 0.005;            % Motor low pass filter

% Aerodynamics Parameters
CT   = 0.109919; % Traction coefficient [-]
CP   = 0.040164; % Moment coefficient [-]
rho  = 1.225;    % Air density [kg/m^3]
k1   = CT*rho*Diam^4;
b1   = CP*rho*Diam^5/(2*pi);
Tmax = k1*(max_rpm/60)^2;  % Max traction [N]
Qmax = b1*(max_rpm/60)^2;  % Max moment [Nm]
k    = Tmax/(max_omega^2); % Traction coefficient
b    = Qmax/(max_omega^2); % Moment coefficient
c    = (0.04-0.0035);      % Lumped Drag constant
KB   = 2;                  % Fountain effect (¬2)

% Contact Parameters
max_disp_a = 0.001;                  % Max displacement in contact [m]
n_a = 4;                             % Number of contacts [-]
xi  = 0.95;                          % Relative damping [-]
ka  = m*g/(n_a*max_disp_a);          % Contact stiffness [N/m]
ca  = 2*m*sqrt(ka/m)*xi*1/sqrt(n_a); % Contact damping [Ns/m]
mua = 0.5;                           % Coulomb friction coefficient [-]

% Mixer    ( [F,taux,tay,tauz]->[omega1,omega2,omega3,omega4] )
% invMixer ( [omega1,omega2,omega3,omega4]->[F,taux,tay,tauz] )
Mixer = [ 1/(4*k) -(1/(2*sqrt(2)*l*k)) -(1/(2*sqrt(2)*l*k)) -(1/(4*b))
          1/(4*k)  (1/(2*sqrt(2)*l*k))  (1/(2*sqrt(2)*l*k)) -(1/(4*b))
          1/(4*k)  (1/(2*sqrt(2)*l*k)) -(1/(2*sqrt(2)*l*k))  (1/(4*b))
          1/(4*k) -(1/(2*sqrt(2)*l*k))  (1/(2*sqrt(2)*l*k))  (1/(4*b)) ];

invMixer = [             k              k              k              k
             -sqrt(2)*l*k/2  sqrt(2)*l*k/2  sqrt(2)*l*k/2 -sqrt(2)*l*k/2
             -sqrt(2)*l*k/2  sqrt(2)*l*k/2 -sqrt(2)*l*k/2  sqrt(2)*l*k/2
                         -b             -b              b              ...
                            b ];
```

Código B.1: Parámetros físicos del problema

B.1 Control de posición

B.1.1. Matlab

```
close all
clear

RADS2RPM = 60/(2*pi);
RAD2DEG  = 180/pi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% QUADROTOR POSITION CONTROL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% SYSTEM MODEL
run('Physical_parameters_definitions')

%% SIMULATION PARAMETERS
% Simulation time
Time = 12;
```

```
% Acquisition time, discrete Kalman algorithm time interval
T = 0.0003;
% Time vector
t = (0:T:Time)';
% Controlled variables references
R_X   =  5;  t_X   = 8;
R_Y   =  5;  t_Y   = 4;
R_Z   =  5;  t_Z   = 0;
R_psi =  pi; t_psi = 3;
% Disturbances
R_uwg   = 0; t_uwg   = 0;
R_vwg   = 0; t_vwg   = 0;
R_wwg   = 0; t_wwg   = 0;
R_Ax    = 0; t_Ax    = 0;
R_Ay    = 0; t_Ay    = 0;
R_Az    = 0; t_Az    = 0;
R_tauwx = 0; t_tauwx = 0;
R_tauwy = 0; t_tauwy = 0;
R_tauwz = 0; t_tauwz = 0;
% Initial conditions
x0 = [0;0;1_feet;0;0;0;0;0;0;0;0;0];

%% STATE SPACE SYSTEM
% States x = [x,y,z,phi,theta,psi,u,v,w,p,q,r]
% States in hover equilibrium x = [0 0 0 0 0 0 0 0 0 0 0 0]
% Control actions u = [F,taux,tauy,tauz]
% Control actions in hover equilibrium
F_0 = m*g; tauX_0 = 0; tauY_0 = 0; tauZ_0 = 0;
u0 = [F_0; tauX_0; tauY_0; tauZ_0];

A = [  0  0   0   0   0   0   1   0   0   0   0   0
       0  0   0   0   0   0   0   1   0   0   0   0
       0  0   0   0   0   0   0   0   1   0   0   0
       0  0   0   0   0   0   0   0   0   1   0   0
       0  0   0   0   0   0   0   0   0   0   1   0
       0  0   0   0   0   0   0   0   0   0   0   1
       0  0   0   0   g   0   0   0   0   0   0   0
       0  0   0  -g   0   0   0   0   0   0   0   0
       0  0   0   0   0   0   0   0   0   0   0   0
       0  0   0   0   0   0   0   0   0   0   0   0
       0  0   0   0   0   0   0   0   0   0   0   0
       0  0   0   0   0   0   0   0   0   0   0   0 ];

B = [  0      0       0       0
       0      0       0       0
       0      0       0       0
       0      0       0       0
       0      0       0       0
       0      0       0       0
       0      0       0       0
       0      0       0       0
     1/m      0       0       0
       0  1/Ixx       0       0
       0      0   1/Iyy       0
       0      0       0   1/Izz ];

C = diag(ones(size(A,1),1));

D = zeros(size(C,1),size(B,2));

% To access all states from state space block in Simulink
Cs = eye(size(C,2));
Ds = zeros(size(A,2),size(B,2));
```

```
80
81  %% LQR CONTROLLER DESIGN
82
83  % INTERNAL CONTROLLER
84  % Considered states in internal controller
85  x_ctr = [3,4,5,6,9,10,11,12]';
86  x_ctr_2_1 = zeros(size(x_ctr,1),size(A,1));
87  x_ctr_2_1(sub2ind(size(x_ctr_2_1),(1:size(x_ctr_2_1,1))',x_ctr)) = 1;
88  % Control Actions in internal controller
89  u_ctr = (1:4)';
90  n_per = 9; % Number of disturbances
91  u_ctr_2 = zeros(n_per+size(B,2),n_per+size(u_ctr,1));
92  u_ctr_2(sub2ind(size(u_ctr_2),(1:n_per)',(1:n_per)')) = 1;
93  u_ctr_2(sub2ind(size(u_ctr_2),n_per+u_ctr,n_per+...
94  (1:size(u_ctr_2,2)-n_per)')) = 1;
95  u_ctr_2 = u_ctr_2';
96  % Outputs in internal controller
97  y_ctr = [3,4,5,6]';
98  % State space system internal controller
99  A1 = A(x_ctr,x_ctr);
100 B1 = B(x_ctr,u_ctr);
101 C1 = C(y_ctr,x_ctr);
102 D1 = D(y_ctr,u_ctr);
103 % Extended system
104 Ab1 = [A1 zeros(size(A1,1),size(C1,1)); -C1 zeros(size(C1,1),size(C1,1))];
105 Bb1 = [B1; zeros(size(C1,1),size(B1,2))];
106 % Extended system controllability
107 Co = ctrb(Ab1,Bb1);
108 unco = length(Ab1) - rank(Co);
109 fprintf('Uncontrollable states internal extended system: %g\n',unco)
110 % Q y R LQR Matrices
111 Q  = blkdiag(0,1,1,0,0,0,0,0,0.6,60,60,1e-5);
112 R  = diag([1,1,1,1])*0.01;
113 % Internal LQR
114 [Kb,¬,¬] = lqr(Ab1,Bb1,Q,R);
115 K1  =  Kb(:,1:(size(A1,2)));
116 Ki1 = -Kb(:,end-size(C1,1)+1:end);
117 fprintf('Extended internal system poles:\n')
118 eig(Ab1-Bb1*Kb)
119
120 % EXTERNAL REGULATOR
121 % States considered in external regulator
122 x_ctr = [1,2,7,8]';
123 x_ctr_2_2 = zeros(size(x_ctr,1),size(A,1));
124 x_ctr_2_2(sub2ind(size(x_ctr_2_2),(1:size(x_ctr_2_2,1))',x_ctr)) = 1;
125 % Control actions in external regulator
126 u_ctr = [4,5]';
127 % LQR external outputs
128 y_ctr = [1,2]';
129 % State space system external controller
130 A2 = A(x_ctr,x_ctr);
131 B2 = A(x_ctr,u_ctr);
132 C2 = C(y_ctr,x_ctr);
133 D2 = zeros(size(y_ctr,1),size(u_ctr,1));
134 % Extended external system
135 Ab2 = [A2 zeros(size(A2,1),size(C2,1)); -C2 zeros(size(C2,1),size(C2,1))];
136 Bb2 = [B2; zeros(size(C2,1),size(B2,2))];
137 % Extended system controllability
138 Co = ctrb(Ab2,Bb2);
139 unco = length(Ab2) - rank(Co);
140 fprintf('Uncontrollable states internal extended system: %g\n',unco)
141 % Q y R LQR Matrices
142 Q  = blkdiag(0,0,0,0,0.07,0.07);
```

```
143 R  = diag([1,1]);
144 % External LQR
145 [Kb,¬,¬] = lqr(Ab2,Bb2,Q,R);
146 K2b  =  Kb(:,1:(size(A2,2))); K2xy = K2b(2,1); K2uv = K2b(2,3);
147 Ki2b = -Kb(:,end-size(C2,1)+1:end); Ki2 = Ki2b(2,1);
148 fprintf('Extended external system poles:\n')
149 eig(Ab2-Bb2*Kb)
150
151 %% EXTENDED KALMAN FILTER
152 Ck = [1,0,0,0,0,0,0,0,0,0,0,0;
153       0,1,0,0,0,0,0,0,0,0,0,0;
154       0,0,1,0,0,0,0,0,0,0,0,0;
155       0,0,0,1,0,0,0,0,0,0,0,0;
156       0,0,0,0,1,0,0,0,0,0,0,0;
157       0,0,0,0,0,1,0,0,0,0,0,0];
158
159 % Covariance error initial estimate
160 Pk0 = eye(size(A,2))*0;
161
162 % Process noise
163 Qr  = 0.000001*[10,10,10,0.1,0.1,0.1,100,100,100,1,1,1]; % Variance
164 qr  = [0,0,0,0,0,0,0,0,0,0,0,0]; % Mean
165 wk = [t 0*sqrt(Qr).*randn(length(t),size(A,2))+qr];
166 Qk = diag(Qr);
167 % Measure noise
168 Rr  = 0.01*[3,3,3,0.1,0.1,0.1]; % Variance
169 rr  = [0,0,0,0,0,0]; % Mean
170 vk = [t sqrt(Rr).*randn(length(t),size(Ck,1))+rr];
171 Rk = diag(Rr);
172
173 % Observability
174 Ob = obsv(A,Ck);
175 unob = length(A)-rank(Ob);
176 fprintf('Unobservable states Kalman Filter: %g\n',unob)
177
178 %% Matrices and physical are saved to be used in Python code
179 save('mat_controlposicion_params.mat','K1','K2b','Ki1','Ki2b','C1','C2',...
180     'Ck','u0','Mixer','x_ctr_2_1','x_ctr_2_2','max_omega','vk','x0',...
181     'Izz','Iyy','Ixx','m','g','invMixer','Pk0','Qk','Rk','K2xy','K2uv',...
182     'Ki2')
183
184 %% SIMULINK SIMULATION
185 out = sim('QR6DOF_NL_controlposicion_matlab_sim');
```

Código B.2: Implementación del control de posición en Matlab

La estructura seguida en el código B.2 consiste en un primer lugar en la carga de las variables físicas del sistema. Posteriormente se definen los parámetros referentes al tiempo, entre ellos el tiempo de simulación, el periodo de adquisición y de ejecución del algoritmo discreto de Kalman, las referencias de las variables controladas y las perturbaciones que en este caso se definen como escalones. A continuación, se definen las condiciones iniciales del quadrotor que se corresponde con los valores iniciales de todos los estados del sistema.

Posteriormente se definen las matrices del sistema dinámico linealizadas en torno al punto de vuelo en estacionario. Mediante ellas se permite el diseño de las leyes de control del bucle externo e interno donde se comprueba la controlabilidad del sistema y su estabilidad mediante el análisis de sus autovalores. Finalmente, se definen los parámetros necesarios para la ejecución del filtro de Kalman y se comprueba la observabilidad del sistema.

Antes de ejecutar el archivo de Simulink donde se simulará la evolución del sistema y el control, se guardarán las variables y parámetros de importancia en un archivo '.mat' que será leído por Python de tal forma que la misma configuración es utilizada en ambas simulaciones.

B.1.2. Python

```
import setup_path
import airsim
import numpy as np
import time
import matplotlib.pyplot as plt
import scipy.io as sio

# Function definitions
" Conversion Quaternion to Euler angles "
def quaternion_to_euler(x, y, z, w):

        t0 = +2.0 * (w * x + y * z)
        t1 = +1.0 - 2.0 * (x * x + y * y)
        X = np.arctan2(t0, t1)

        t2 = +2.0 * (w * y - z * x)
        t2 = +1.0 if t2 > +1.0 else t2
        t2 = -1.0 if t2 < -1.0 else t2
        Y = np.arcsin(t2)

        t3 = +2.0 * (w * z + x * y)
        t4 = +1.0 - 2.0 * (y * y + z * z)
        Z = np.arctan2(t3, t4)

        return X, Y, Z

" Conversion Speed in Global Frame to in Body frame "
def speed_global_to_body(x, y, z, phi, theta, psi):

        ...

        return u, v, w

" Get States and Time of QuadRotor "
def get_states_qr():

        s = client.getMultirotorState()
        [phi,theta,psi] = ...
            quaternion_to_euler(s.kinematics_estimated.orientation.x_val,\
                        s.kinematics_estimated.orientation.y_val,\
                        s.kinematics_estimated.orientation.z_val,\
                        s.kinematics_estimated.orientation.w_val)
        [u,v,w] = speed_global_to_body(\
                s.kinematics_estimated.linear_velocity.x_val,\
                -s.kinematics_estimated.linear_velocity.y_val,\
                -s.kinematics_estimated.linear_velocity.z_val, phi, ...
                    -theta, -psi)
        x = np.array([\
                s.kinematics_estimated.position.x_val,\
                -s.kinematics_estimated.position.y_val,\
                -s.kinematics_estimated.position.z_val,phi,\
                -theta,-psi,\
                u,v,\
                w,s.kinematics_estimated.angular_velocity.x_val,\
```

```
53                  -s.kinematics_estimated.angular_velocity.y_val,\
54                  -s.kinematics_estimated.angular_velocity.z_val])
55          t = s.timestamp/1e9
56
57          return x,t
58
59  " Non lineal dynamic model "
60  def dyn_model(xk,uk,T,Ixx,Iyy,Izz,m,g):
61
62          ...
63
64          return xkm
65
66  " Linearization around estimate "
67  def lin_model(xk,Ixx,Iyy,Izz,m,g):
68
69          ...
70
71          return Ak
72
73  " Gain Scheduling position controller "
74  def K2reg(psi,K2xy,K2uv,Ki2xy):
75
76          K2  = np.array([[K2xy*np.sin(psi),-K2xy*np.cos(psi),0,-K2uv],\
77                          [K2xy*np.cos(psi),K2xy*np.sin(psi),K2uv,0]])
78          Ki2 = np.array([[Ki2xy*np.sin(psi),-Ki2xy*np.cos(psi)],\
79                          [Ki2xy*np.cos(psi),Ki2xy*np.sin(psi)]])
80
81          return K2,Ki2
82
83  " Yaw state conditioning "
84  def yaw_cond(psi,psi_prev):
85
86          if np.sign(psi)!=np.sign(psi_prev):
87                  if (np.absolute(psi)>np.pi/4)and\
88                     (np.absolute(psi_prev)>np.pi/4):
89                          if psi<psi_prev:
90                                  add = 2*np.pi
91                          else:
92                                  add = -2*np.pi
93                  else:
94                          add = 0
95          else:
96                  add = 0
97
98          return add
99
100 " Yaw reference conditioning "
101 def yaw_ref_cond(psi,psi_ref):
102
103         while np.abs(psi-psi_ref)>np.pi:
104                 psi_ref = psi_ref+2*np.pi*np.sign(psi-psi_ref)
105
106         return psi_ref
107
108 # Connect to AirSim, Confirm Connection, Enable API Control, Arm QuadRotor
109 client = airsim.MultirotorClient()
110 client.confirmConnection()
111 client.enableApiControl(True)
112 client.armDisarm(True)
113
114 # Control time step, simulation time, physical Airsim clock speed
115 T = 0.0003
```

```
Time = 12
len = Time/T

# Create empty array to save simulation data
x   = np.empty((int(len)+1,12))
t   = np.empty(int(len)+1)
u   = np.empty((int(len)+1,4))
yk  = np.empty((int(len)+1,6))
xkm = np.empty((int(len)+1,12))
xk  = np.empty((int(len)+1,12))

# Load feeback, integrator, ... matrices
mat_contents = sio.loadmat('mat_controlposicion_params.mat')
C1           = mat_contents['C1']
C2           = mat_contents['C2']
Ck           = mat_contents['Ck']
K1           = mat_contents['K1']
# K2           = mat_contents['K2b']
K2xy         = float(mat_contents['K2xy'])
K2uv         = float(mat_contents['K2uv'])
Ki1          = mat_contents['Ki1']
# Ki2          = mat_contents['Ki2b']
Ki2xy        = float(mat_contents['Ki2'])
u0           = np.transpose(mat_contents['u0']).flatten('F')
x0           = np.transpose(mat_contents['x0'])
mixer        = mat_contents['Mixer']
invmixer     = mat_contents['invMixer']
x_ctr_2_1    = mat_contents['x_ctr_2_1']
x_ctr_2_2    = mat_contents['x_ctr_2_2']
Pk0          = mat_contents['Pk0']
Qk           = mat_contents['Qk']
Rk           = mat_contents['Rk']
max_omega    = float(mat_contents['max_omega'])
vk           = mat_contents['vk']
Izz          = float(mat_contents['Izz'])
Iyy          = float(mat_contents['Iyy'])
Ixx          = float(mat_contents['Ixx'])
m            = float(mat_contents['m'])
g            = float(mat_contents['g'])

# Control Loop
" Reference "
R_X   =  5; t_X   = 8
R_Y   =  5; t_Y   = 4
R_Z   =  5; t_Z   = 0
R_psi =  180*np.pi/180; t_psi = 3
Ref2                  = np.zeros((int(len),2))
Ref1                  = np.zeros((int(len),4))
Ref2[int(t_X/T):,0]   = R_X
Ref2[int(t_Y/T):,1]   = R_Y
Ref1[int(t_Z/T):,0]   = R_Z
Ref1[int(t_psi/T):,3] = R_psi
" Integrator "
xi1 = np.zeros(Ref1.shape[1])
xi2 = np.zeros(Ref2.shape[1])
"Kalman initial estimates"
xk[0] = x0
uk0   = u0
Pk    = Pk0
"Yaw reference conditioning"
adder = 0
psi_prev = x0[0,5]
" Loop "
```

```
for i in range(int(len)):
    [x[i],t[i]] = get_states_qr()
    'Check sign of yaw angle'
    adder    += yaw_cond(x[i,5],psi_prev)
    psi_prev = x[i,5]
    x[i,5]   = x[i,5]+adder
    'Extended Kalman Filter'
    yk[i]    = Ck@x[i]+vk[i,1:]
    xkm[i]   = np.transpose(dyn_model(xk[i],uk0,T,Ixx,Iyy,Izz,m,g))
    Ak       = lin_model(xk[i],Ixx,Iyy,Izz,m,g)
    PkmDot   = Ak@Pk+Pk@np.transpose(Ak)+Qk
    Pkm      = Pk+PkmDot*T
    Kk       = ...
        Pkm@np.transpose(Ck)@np.linalg.inv((Ck@Pkm@np.transpose(Ck)+Rk))
    xk[i+1] = xkm[i]+Kk@(yk[i]-Ck@xkm[i])
    Pk       = (np.identity(12)-Kk@Ck)@Pkm;
    'External Loop'
    [K2,Ki2]       = K2reg(xk[i,5],K2xy,K2uv,Ki2xy)
    dxi            = Ref2[i]-C2@x_ctr_2_2@xk[i]
    xi2            = xi2+dxi*T
    Ref1[i,(1,2)] = -K2@x_ctr_2_2@x[i]+Ki2@xi2
    'Yaw reference between range -180+psi, 180+psi'
    Ref1[i,3] = yaw_ref_cond(xk[i,5],Ref1[i,3])
    'Internal Loop'
    dxi  = Ref1[i]-C1@x_ctr_2_1@xk[i]
    xi1  = xi1+dxi*T
    u[i] = -K1@x_ctr_2_1@x[i]+Ki1@xi1+u0
    'PWM inputs'
    pwm  = np.clip(mixer@u[i]/(max_omega**2),0,1)
    u[i] = invmixer@np.clip(mixer@u[i],0,max_omega**2)
    uk0  = u[i].flatten('F')
    'Control API'
    client.moveByMotorPWMsAsync(pwm[0], pwm[1], pwm[2], pwm[3], 5)
    'Sellep the control algorithm until next iterations'
    time.sleep((i+1)*T-(t[i]-t[0]))

# Final states, control actions and times
[x[i+1],t[i+1]] = get_states_qr()
u[i+1]          = u[i]
yk[i+1]         = yk[i]
xkm[i+1]        = xkm[i]
t               = (t-t[0])

# Save simulation data
sio.savemat('mat_controlposicion_python_sim.mat', ...
    {'x':x,'t':t,'u':u,'xk':xk,'xkm':xkm,'yk':yk})
```

Código B.3: Implementación del control de posición en Python

La estructura seguida en el código de Python, que permite ejecutar AirSim para simular la respuesta del quadrotor y realizar la validación, consiste en una primera definición de las funciones que serán utilizadas. Entre ellas se encuentra la conversión de quaterniones a ángulos de Euler, la conversión de la velocidad en ejes globales a ejes cuerpo, la obtención de los estados del sistema y del tiempo de la simulación, la predicción de los estados del sistema mediante el filtro de Kalman, la linealización del sistema entorno a la estimación del filtro de Kalman, el regulador con planificación de ganancia para el control de posición y el acondicionamiento de la guiñada y su referencia para evitar la discontinuidad en $\pm 180°$.

Cabe destacar que algunas de ellas se han omitido puesto que ya han sido introducidas en el documento y de esta forma ahorrar espacio. La conversión de la velocidad de ejes globales a ejes cuerpo puede encontrarse en la ecuación (4.3). Las funciones que hacen referencia a la obtención

del modelo lineal entorno a la estimación del filtro de Kalman y la predicción inicial del estado pueden encontrarse en la subsección 5.3.2.

La función más importante de las definidas es aquella que permite la obtención de los estados del sistema junto con el tiempo de medición que será utilizada para realimentar el sistema tras observar las mediciones mediante el filtro de Kalman.

Posteriormente se conecta el programa de Python con el simulador de AirsSim mediante el uso de las APIs correspondientes y se definen los periodos de ejecución del algoritmo de control discreto y la duración del control. Tras definir algunos vectores utilizados para almacenar la información, se cargan las variables que habían sido guardadas anteriormente mediante el programa de Matlab.

Tras definir las referencias del sistema, se inicializan los integradores y las variables utilizadas en el filtro de Kalman y se inicia el bucle de control. Una vez que ha terminado la simulación se guardan los vectores que contienen los datos de la simulación en formato '.mat' que permita la lectura mediante Matlab.

Parte II

Planos

APÉNDICE C
Planos

En esta parte del documento se presentan los planos utilizados. Estos se corresponden con los diagramas de Simulink empleados para modelar el sistema dinámico y las estructuras de control utilizadas para conseguir unas determinadas especificaciones dinámicas del sistema en el seguimiento de una referencia y rechazo de perturbaciones.

En un primer lugar se comienza incluyendo los diagramas de control empleados. Posteriormente se introduce la estructura del observador de estados implementado, el sistema de control de trayectoria que incluye un control en cascada y finalmente se incluye el modelo no lineal que contiene la dinámica del quadrotor.

C.1 Estructura del controlador

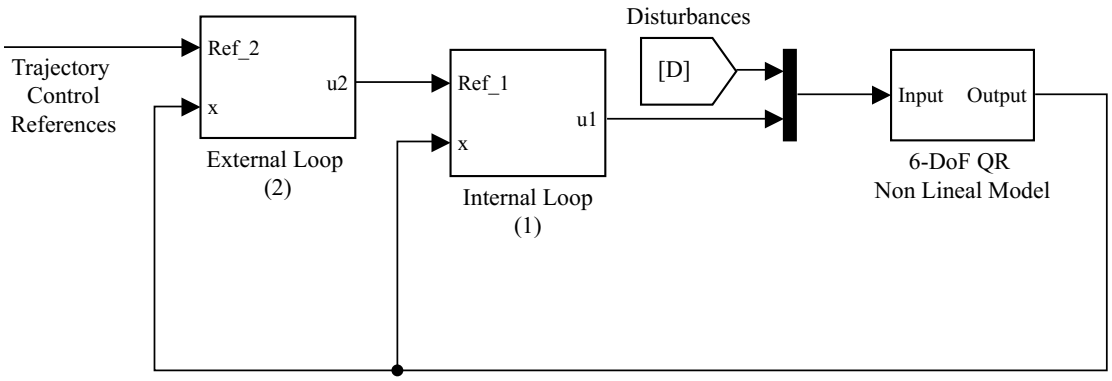


Figura C.1: Estructura del controlador: Bucle interno y externo

C.1.1. Controlador del bucle interno

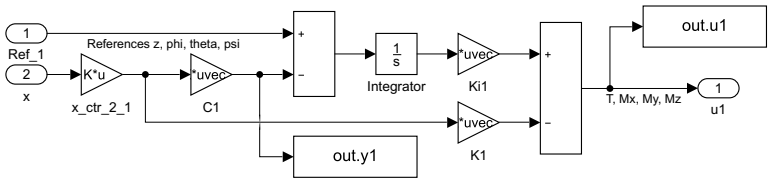


Figura C.2: Estructura del controlador: Bucle interno

C.1.2. Estructura del bucle externo

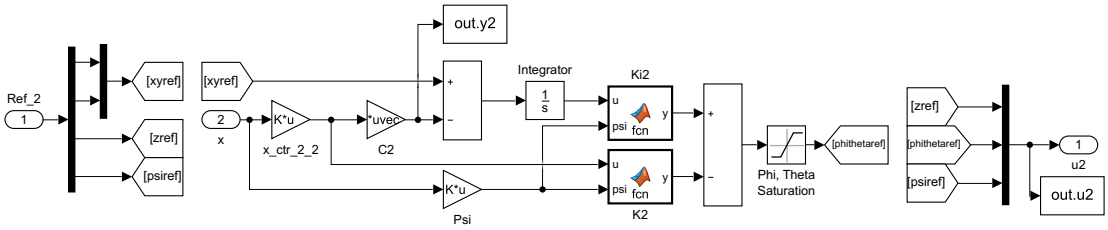


Figura C.3: Estructura del controlador: Bucle externo

C.2 Estructura del observador de estados en el sistema de control

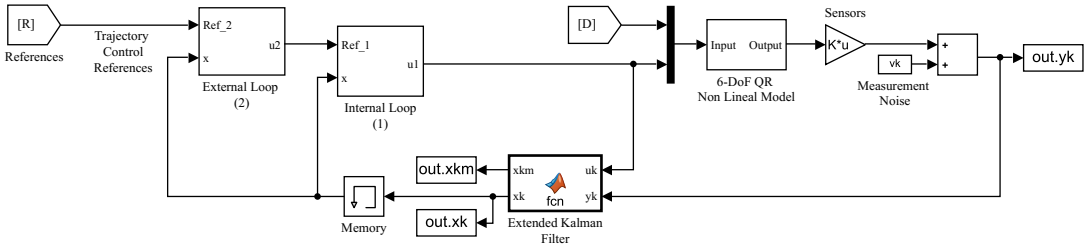


Figura C.4: Implementación del Filtro de Kalman Extendido (EKF) en la estructura del controlador.

C.3 Esquema del control completo de trayectoria de una quadrotor

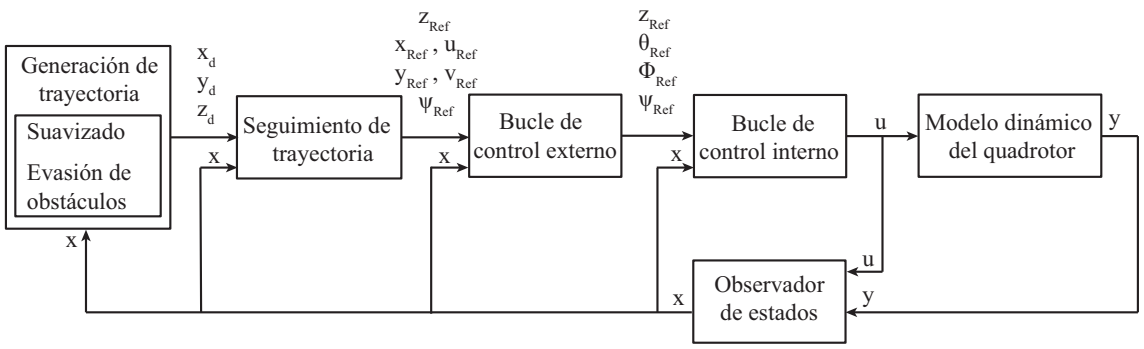


Figura C.5: Esquema de control completo para el control de trayectorias de un vehículo del tipo quadrotor

C.4 Modelo no lineal de la dinámica del quadrotor

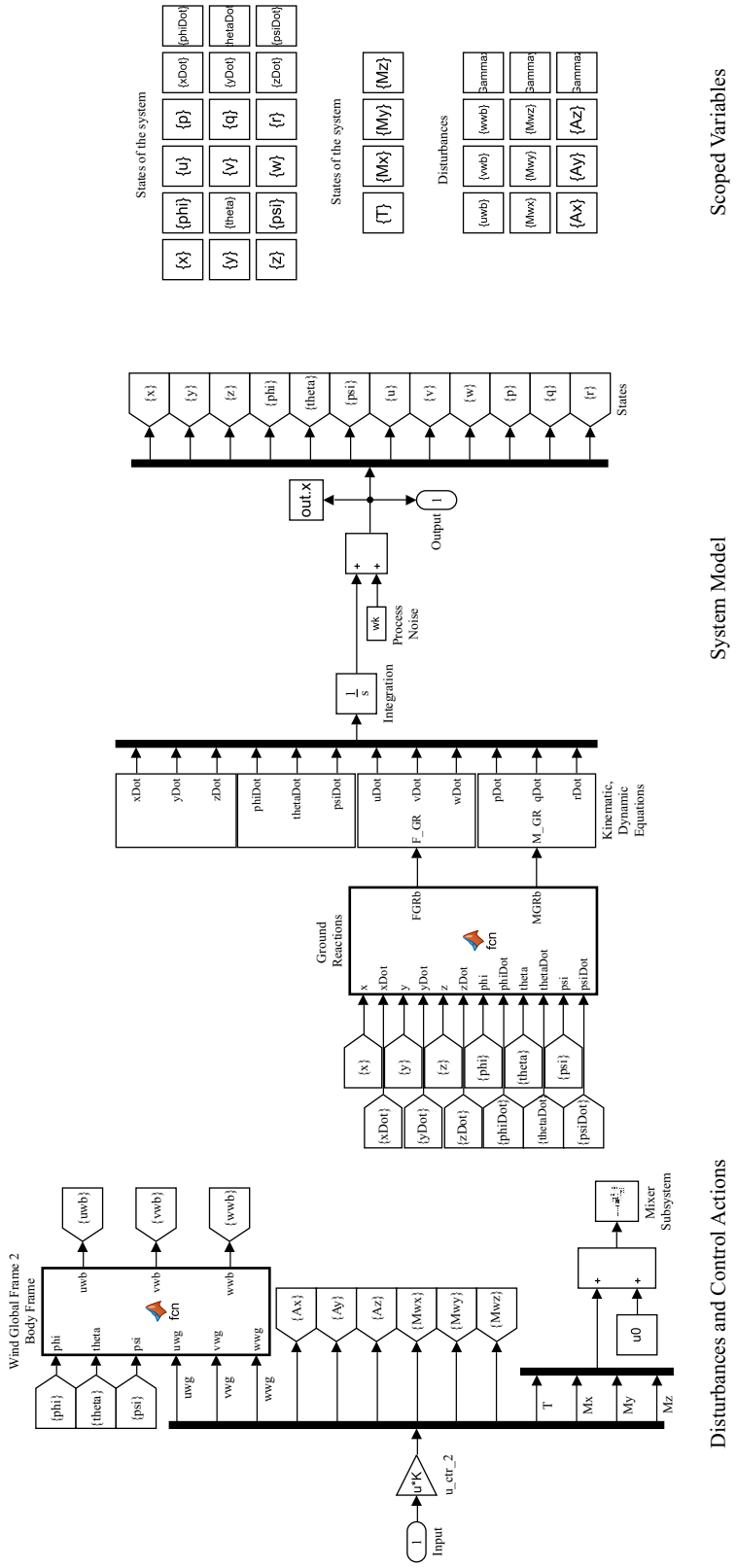


Figura C.6: Primer nivel del bloque del modelo no lineal de un quadrotor en Simulink

C.4.1. Subsistema de mezclado

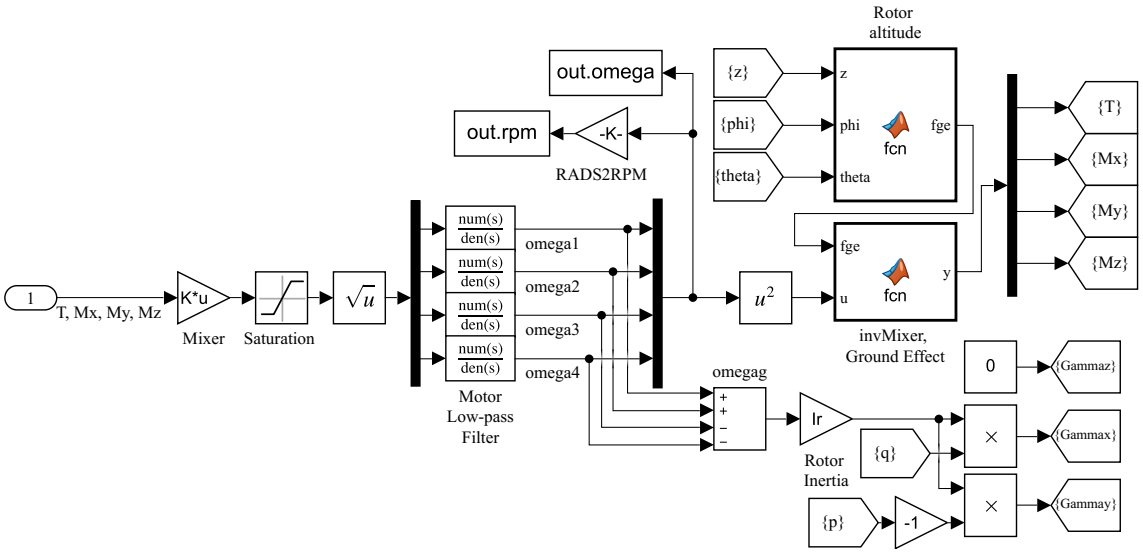
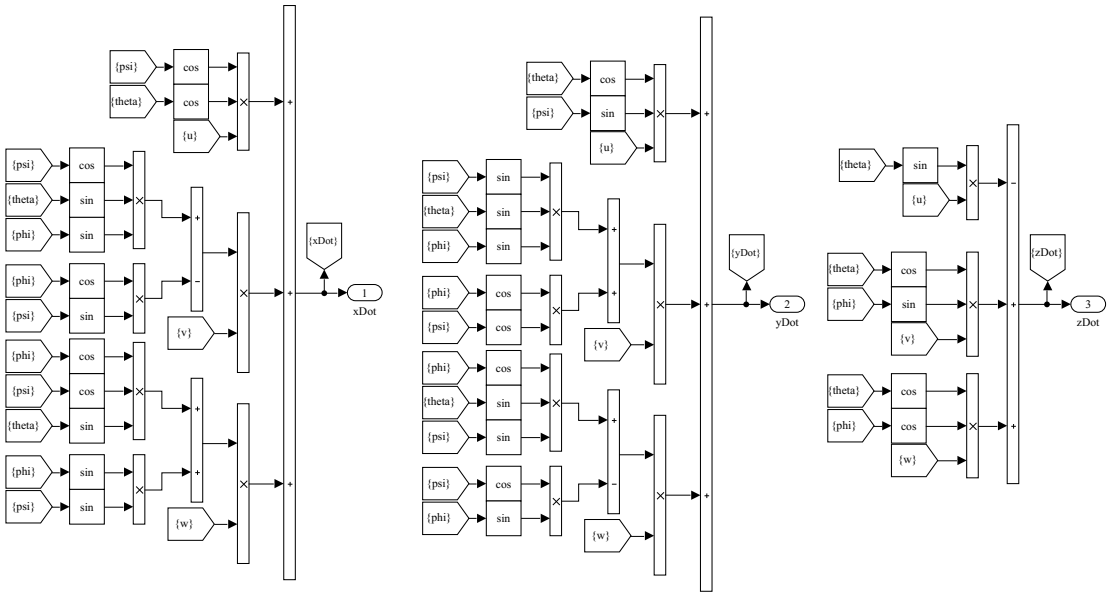


Figura C.7: Subsistema de mezclado de fuerzas y momentos a velocidades de giro

C.4.2. Ecuaciones cinemáticas y dinámicas

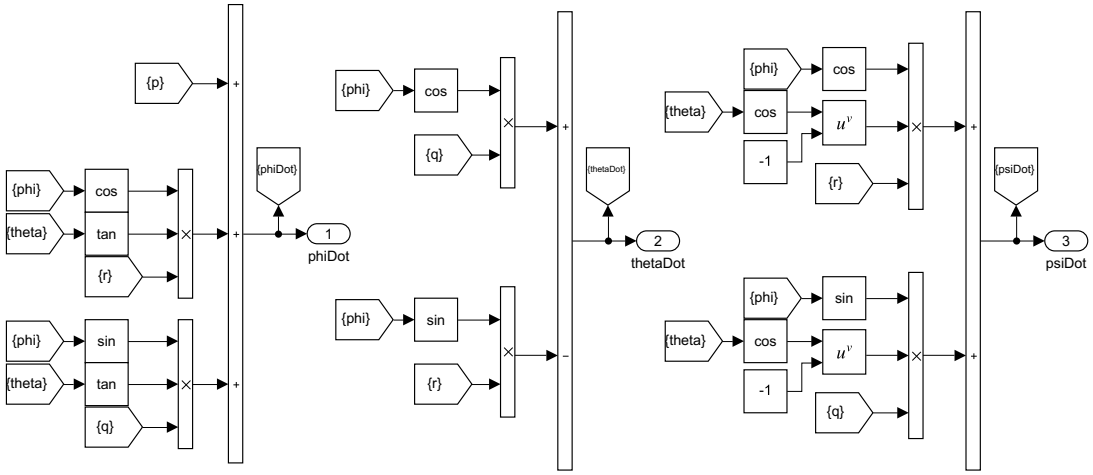
Ecuaciones cinemáticas de traslación



$$\dot{x} = uCos(\theta)Cos(\psi) + v(Cos(\psi)Sin(\theta)Sin(\phi) - Cos(\phi)Sin(\psi)) + w(Cos(\phi)Cos(\psi)Sin(\theta) + Sin(\phi)Sin(\psi))$$
$$\dot{y} = uCos(\theta)Sin(\psi) + v(Cos(\phi)Cos(\psi) + Sin(\theta)Sin(\phi)Sin(\psi)) + w(-Cos(\psi)Sin(\phi) + Cos(\phi)Sin(\theta)Sin(\psi))$$
$$\dot{z} = -uSin(\theta) + vCos(\theta)Sin(\phi) + wCos(\theta)Cos(\phi)$$

Figura C.8: Ecuaciones cinemáticas de traslación

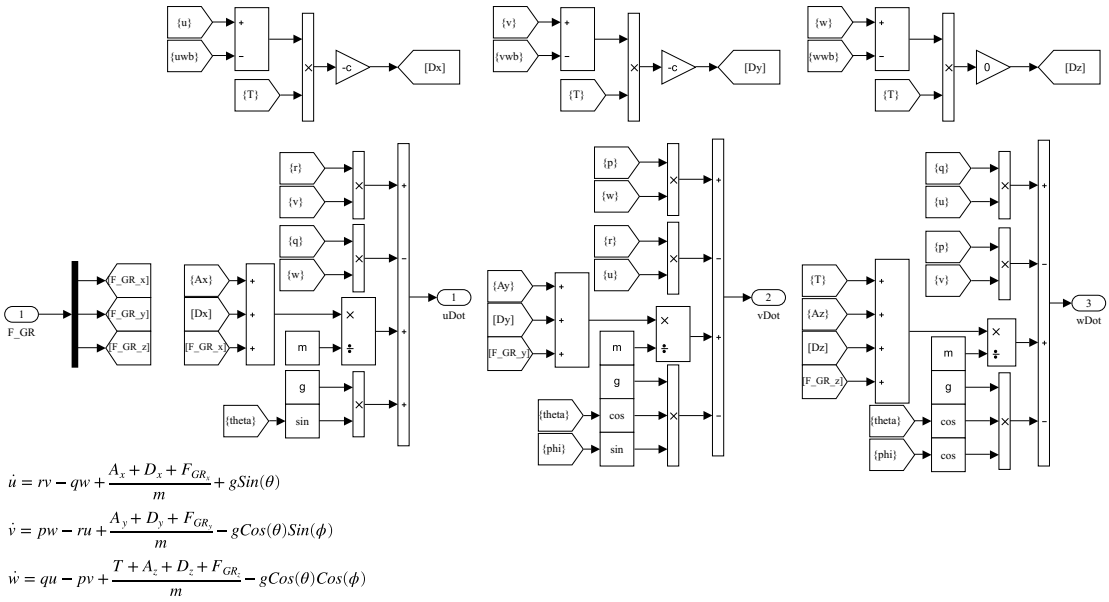
Ecuaciones cinemáticas de rotación



$$\dot{\phi} = p + rCos(\phi)Tan(\theta) + qSin(\phi)Tan(\theta)$$
$$\dot{\psi} = rCos(\phi)Sec(\theta) + qSec(\theta)Sin(\phi)$$
$$\dot{\theta} = qCos(\phi) - rSin(\phi)$$

Figura C.9: Ecuaciones cinemáticas de rotación

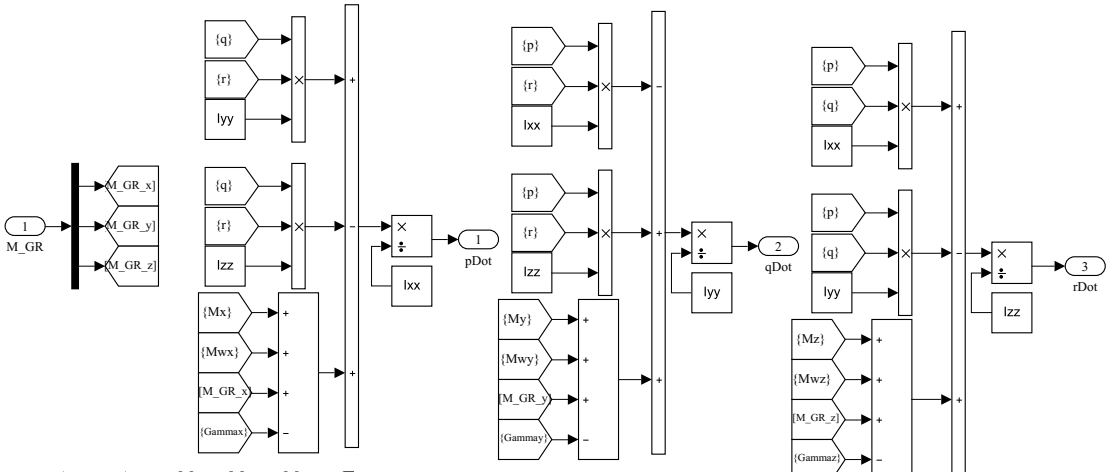
Ecuaciones dinámicas de traslación



$$\dot{u} = rv - qw + \frac{A_x + D_x + F_{GR_x}}{m} + gSin(\theta)$$
$$\dot{v} = pw - ru + \frac{A_y + D_y + F_{GR_y}}{m} - gCos(\theta)Sin(\phi)$$
$$\dot{w} = qu - pv + \frac{T + A_z + D_z + F_{GR_z}}{m} - gCos(\theta)Cos(\phi)$$

Figura C.10: Ecuaciones dinámicas de traslación

Ecuaciones dinámicas de rotación



$$\dot{p} = \frac{(I_{yy} - I_{zz})}{I_{xx}} qr + \frac{M_{T_x} + M_{wx} + M_{GR_x} - \Gamma_x}{I_{xx}}$$

$$\dot{q} = \frac{(I_{zz} - I_{xx})}{I_{yy}} pr + \frac{M_{T_y} + M_{wy} + M_{GR_y} - \Gamma_y}{I_{yy}}$$

$$\dot{r} = \frac{(I_{xx} - I_{yy})}{I_{zz}} pq + \frac{M_{T_z} + M_{wz} + M_{GR_z} - \Gamma_z}{I_{zz}}$$

Figura C.11: Ecuaciones dinámicas de rotación

Parte III

Pliego de condiciones

APÉNDICE D
Pliego de condiciones

A lo largo de la presente parte del documento, conocida como pliego de condiciones, se muestran las extensiones que deben existir en el contrato entre contratista y propiedad en la ejecución del trabajo de final de grado.

D.1 Condiciones de los materiales

A lo largo del presente documento se ha generado un contenido que permite la simulación, diseño y validación de una ley de control que permita el seguimiento de una determinada trayectoria que ha sido predefinida mediante una serie de *waypoints*. Tanto la documentación generada como los esquemas de control, modelo dinámico y códigos utilizados para la definición de los diferentes controladores serán accesibles a cualquier persona para su uso particular y la posterior ampliación del tema de estudio iniciado con este trabajo. La regulación que se utiliza en el rpesente trabajo de fin de grado se corresponde con las condiciones de uso de la plataforma Riunet.

El contenido que ha sido generado a parte de la documentación mostrada en la presente memoria se corresponden con:

- Modelo de la dinámica no lineal de un quadrotor implementado mediante diagrama de bloques en Simulink.
- Estructura del sistema de control LQR con planificación de ganancia para conseguir el seguimiento de unas determinadas referencias de posición o velocidad.
- Generación de las referencias que son suministradas al sistema de control para seguir una determinada referencia de trayectoria.
- Estructura del controlador en Python para permitir la conexión con AirSim para validar y simular la dinámica del quadrotor.

Por otro lado se explicitan las condiciones de utilización de los diferentes *softwares* que han sido utilizados en la generación del presente trabajo.

- Matlab y Simulink. La licencia utilizada en la realización del presente trabajo se corresponde con una licencia individual que puede ser instalada hasta en cuatro dispositivos.
- AirSim. La licencia que se ha utilizado en el simulador empleado se corresponde con la MIT con derechos de copyright de Microsoft. Esta licencia proporciona permisos, sin necesidad alguna de ningún coste, a cualquier persona para copiar, modificar, distribuir o publicar cualquier parte del código siempre y cuando se incluyan las notificaciones de *copyright*. Además

se proporciona el código sin ninguna garantía de ningún tipo y los autores no pueden ser culpados por cualquier incidente derivado del uso de este *software*.

- Python. La licencia de uso de este lenguaje de programación pueden encontrarse en su página web[1]. Cabe destacar que todas las versiones son de código libre y el usuario puede acceder a ellas y modificarlas.

D.2 Condiciones de utilización

A lo largo del presente documento se diseña un sistema de control que permite el seguimiento de trayectoria a un quadrotor siendo éste un vehículo aéreo no tripulado. Por ello se debe de cumplir con la normativa de la AESA en materia de aeronavegabilidad inicial y continuada en todos los puntos que hagan referencia al piloto automático. En la página web de la Agencia Española de Seguridad Aérea se puede encontrar la regulación existente [2].

[1] `https://docs.python.org/3/license.html#terms-and-conditions-for-accessing-\or-otherwise-using-python`

[2] `https://www.seguridadaerea.gob.es/lang_castellano/cias_empresas/trabajos/rpas/default.aspx`

Parte IV

Presupuesto

APÉNDICE E
Presupuesto

En la presente parte se encuentra el presupuesto necesario para llevar a cabo el trabajo que se ha mostrado a lo largo de todo el documento con título: 'Diseño de estrategias de control basadas en observadores de estados para el seguimiento de trayectorias en vehículos aéreos no tripulados tipo quadricóptero'.

El desglose de los gastos se hace en primer lugar en función de su naturaleza, es decir, divididos en costes asociados del uso de un determinado *hardware* o soporte material, derivados del uso de un determinado *software* o costes del personal. Posteriormente se presenta un desglose global de cada uno de los costes.

Para calcular correctamente el desglose de los gastos se debe considerar que un trabajo final de grado presenta 12 créditos ECTS (*European Credit Transfer and Accumulation System*), considerando que cada crédito supone una inversión de 25-30 horas de trabajo según el Plan Bolonia, el número de horas totales necesarias para la correcta realización del trabajo es de 300 horas. Asumiendo un horario de trabajo de 8 horas diarias, para la realización del trabajo final de grado se emplearán 37 días y medio.

E.1 Desglose de costes en función de su naturaleza

E.1.1. Costes materiales

En la realización del presente trabajo han sido necesarios diferentes soportes materiales que permitan entre ellos, elaborar la memoria, modelar las ecuaciones y poder lanzar las diferentes simulaciones.

El ordenador que ha sido utilizado en este caso se corresponde con un ordenador portátil *HP OMEN 15-DH0019NS*, con un procesador *Intel Core i9-9880H*, 32 GB de memoria RAM, almacenamiento de 1 TB de memoria SSD y con la gráfica dedicada *Nvidia RTX 2080*. Para mejorar la eficiencia del trabajo se ha hecho uso de tres periféricos principalmente. El primero de ellos consiste en un ratón modelo *Razer Abyssus Essential*, el segundo se corresponde con un monitor auxiliar, modelo *Samsung S22F350FHU*, conectado al ordenador portátil mediante un cable HDMI, y finalmente un mando de consola *Xbox One* utilizado para hacer las primeras pruebas con el simulador AirSim.

En la tabla E.1 se encuentra el coste de cada uno de los dispositivos utilizados. Además, en el desglose total de gastos solamente se ha considerado el coste de cada uno de los componentes durante el periodo de realización del trabajo, también llamado coste proporcional. Para ello se asumirá que le periodo de utilización de todos los componentes es de alrededor de 7 años y por tanto, considerando que solamente serán utilizados durante 37.5 días para la utilización del trabajo,

los costes materiales asociados a la utilización del *hardware* listado será calculado en proporción para cada componente.

En los costes derivados del uso de equipos materiales se incluye el coste debido al consumo energético. Principalmente este coste se encuentra constituido por la energía consumida por el monitor auxiliar y el ordenador portátil ya que el resto de componentes pueden ser despreciados. Para ello se asumirá un coste del kWh de 0.0929€, con un consumo medio de 29 W para el monitor auxiliar y 34.5 W para el ordenador portátil, por tanto el consumo energético será de 19.05 kWh con un coste de 1.77€.

Componente de hardware	Coste (€)	Coste proporcional (€)
HP OMEN 15-DH0019NS	2799.00	41.08
Samsung S22F350FHU	84.99	1.25
Razer Abyssus Essential	49.99	0.73
Xbox One Controller	49.99	0.73
Cable HDMI	3.25	0.05
Total	2987.22	43.84

Tabla E.1: Desglose de gastos derivados del uso del hardware

E.1.2. Costes de *software*

Los *softwares* que han sido utilizados en la realización del presente trabajo quedan listados en la tabla E.2. Cabe destacar que muchos de los programas presentes son de código libre que pueden ser utilizados sin la necesidad de pagar una licencia. En cuanto a aquellos que presentan una licencia perpetua se ha considerado que su periodo de utilización es de 7 años, igual que los componentes materiales. De esta forma, considerando que el tiempo de utilización de cada software para la realización del trabajo es de 37.5 días es posible calcular cuál será el coste proporciona de cada uno de ellos.

Software	Coste (€)	Coste proporcional (€)
Matlab & Simulink	1200 (Anual)	123.29
Mathematica (Premier Service)	710 (Anual)	72.95
Adobe Illustrator	29.99 (Mensual)	36.28
Microsoft 365	69.00 (Anual)	7.09
Microsft Windows 10 Professional	259.00 (Perpetua)	3.80
Python	0.00 (Perpetua)	0
AirSim	0.00 (Perpetua)	0
TeXstudio	0.00 (Perpetua)	0
Mendeley	0.00 (Perpetua)	0
Shotcut	0.00 (Perpetua)	0
Total		236.32

Tabla E.2: Desglose de gastos derivados del uso del software

E.1.3. Costes de humanos

Para la realización del presente trabajo se han empleado en total 300 horas de recursos humanos dedicadas por el autor del trabajo. Por otro lado, se han mantenido reuniones semanales con el profesor tutor del trabajo con una duración aproximada de 1 h a partir del mes de marzo del año de

redacción del trabajo, lo que se corresponde con 17 horas de trabajo destinadas a la tutorización. Considerando un coste aproximado de 20€/h para el autor del trabajo y 40€/h para el profesor tutor se muestra el desglose de costes asociados a los gastos humanos. Cabe destacar que se han separado las 300 horas totales del desarrollo del trabajo en función de las tareas realizadas junto con su coste temporal tal y como se muestra en la tabla E.3.

Personal	Tareas	Coste temporal (h)	Coste
Autor	Documentación	40	800
	Modelado del sistema dinámico	20	400
	Diseño del controlador y observador	35	700
	Diseño del controlador de trayectoria y generación	30	600
	Validación del sistema dinámico, de control y observador	40	800
	Simulaciones y validación del control de trayectoria	35	700
	Redacción de la memoria	100	2000
Tutor	Revisión y asesoramiento	17	680
	Total		6680

Tabla E.3: Costes humanos asociados a la realización del proyecto

E.2 Desglose de costes totales

Finalmente, en la tabla E.4 se resumen los costes totales necesarios para la realización del proyecto.

Desglose de costes totales del trabajo	
Costes de hardware	43.48 €
Coste energético	1.77 €
Costes de software	236.32 €
Costes humanos	6680 €
Costes totales	6961.57 €

Tabla E.4: Desglose de costes totales

Por tanto, la realización del trabajo supone un coste total de Seis Mil Novecientos Sesenta y Un Euros con Cincuenta y Siete Céntimos (6961.57€).