BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA, OPCIÓN EN AUTOMATIZACIÓN

"SIMULACIÓN DE LA ESTABILIZACIÓN Y EL SEGUIMIENTO DE TRAYECTORIAS DE UN CUADRI-ROTOR EN UN AMBIENTE VIRTUAL"

T E S I S

Presentada para obtener el título de:

Maestro en Ciencias de la Electrónica, Opción en Automatización

Presenta:

Ing. Edmundo Miguel Cortes Vazquez*

Directores:

Dra. Amparo Dora Palomino Merino (FCE-BUAP)

Dr. César Martínez Torres (EDEI-UDLAP)

Dr. Gibran Etcheverry Doger (EDEI-UDLAP)

Puebla, México **ENERO 2020**

* Becario CONACYT.

Agradecimientos

Gracias a la Benemérita Universidad Autónoma de Puebla (BUAP), por abrir sus puertas a mi educación. A la Facultad de Ciencias de la Electrónica (FCE) por brindarme la oportunidad de desarrollo académico y personal.

Gracias a la Maestría en Ciencias de la Electrónica, opción en Automatización (MCEA) y a todos los profesores que durante estos dos años compartieron sus conocimientos y experiencia. A la Coordinadora de la MCEA, Dra. Olga Guadalupe Félix Beltrán gracias por todo su apoyo.

Agradezco al Concejo Nacional de Ciencia y Tecnología (CONACYT), por la beca otorgada para la realización de mis estudios de maestría.

Mi más profundo agradecimiento para mis directores de tesis, Dra. Amparo Dora Palomino Merino, Dr. Cesar Martínez Torres y Dr. Gibran Etcheverry Doger, por su paciencia, guía y apoyo durante el desarrollo de esta tesis.

Agradezco al comité tutorial, Dr. Sergio Vergara Limon, Dra. Olga Guadalupe Félix Beltrán y Dr. Jaime Julián Cid Monjaraz, por sus correcciones y observaciones que contribuyeron a mejorar este trabajo de tesis.

Gracias con todo mi amor a mi esposa Abril, por compartir este reto de la maestría juntos, por sus palabras de ánimo y por ser mi compañera en la vida.

Un especial agradecimiento a mis padres, Sonia Vazquez Torres y José Segismundo Cortes Simón, por su amor y apoyo incondicional. A mis hermanos, Sonia, Yeniseik, Arantxa, Edson y Luz, por su apoyo y confianza.

Agradezco a mis suegros, Ramona Zurita Sánchez y Orlando Rivera Tenorio, por todo su apoyo.

Y finalmente, pero no menos importante agradecer a mis compañeros y amigos, Abril, Pájaro, Daniel, Edwin, Carlos, Caro y Danielote, mil gracias por su amistad y compañía.

DEDICATORIA

Dedico este trabajo:

A ti Abril, mi compañera de vida, LDN.

A mis padres Sonia y Segismundo, por su amor y apoyo incondicional, los amo.

Resumen

En este trabajo se presenta la implementación de un Cuadri-rotor en el software de simulación robótica de código abierto Webots, con el objetivo de verificar su funcionamiento con algoritmos de control para la estabilización y el seguimiento de trayectorias. El Cuadri-rotor se construye en el mundo virtual a partir de nodos basados en lenguaje de modelado de realidad virtual (VRLM), donde también es adicionado con sensores de posicionamiento global (GPS) y una unidad de medición inercial (IMU), entre otros. Para entender el funcionamiento del cuadri-rotor se estudia el modelo dinámico no lineal del cuadri-rotor, así, como su representación lineal y por planitud.

Se muestra una clasificación de los vehículos aéreos no tripulados (UAV) y la regulación del sistema de aeronave pilotada a distancia (RPAS) en México. Se introduce una descripción de la planificación de trayectorias basado en planitud. La planitud implica que es posible que cada variable del sistema se pueda expresar en términos de salidas planas y un número finito de sus derivadas, entonces, se puede construir una trayectoria de salida plana para obtener las entradas de control de bucle abierto que satisfagan el estado deseado y las trayectorias de entrada.

Para verificar el comportamiento de las simulaciones en Webots se utilizan los algoritmos de control Proporcional Derivativo (PD), Backstepping, Regulador Lineal Cuadrático (LQR) y el control basado en Planitud con estas tres estrategias. Todas las estrategias de control se prueban también en MATLAB, con fines de comparación.

Las estrategias de control son implementadas con éxito, encontrando una buena similitud entre Webots y MATLAB, además de que Webots aporta de manera visual el comportamiento del Cuadri-rotor en concordancia al mundo real.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

Los vehículos aéreos no tripulados (VANT) son objeto de interés debido a la amplia variedad de aplicaciones. Los vehículos de ala fija han sido utilizados para fines militares y meteorológicos frecuentemente debido a su duración, alcance y velocidad de vuelo [1][2], aun así, se considera que los vehículos de cuatro rotores, también conocidos como cuadri-rotor o comúnmente como drones, son preferibles para las aplicaciones de vigilancia, la entrega precisa, por su vuelo con agilidad y precisión [3][4][5].

El implementar las estrategias de control en cuadri-rotores, aun después de validado su rendimiento mediante simulación numérica [6][7][8][9], en los prototipos reales puede significar gasto de recursos y riego de daño por fallas, por lo que, un entorno de simulación que permita representar condiciones del mundo real, es una herramienta importante. Para abordar esta problemática en este trabajo de tesis se plantea construir un vehículo aéreo no tripulado tipo cuadri-rotor en un ambiente virtual, que permita la implementación de estrategias de control, para la estabilización de vuelo y seguimiento de trayectorias.

En este trabajo, se ha desarrollado una implementación del Cuadri-rotor basada en el software Webots [10], que es un simulador de robótica de código abierto que proporciona un entorno de desarrollo completo para modelar, programar y simular robots. Existen varios aspectos que hacen atractivo este software, por ejemplo, la programación se puede realizar en el lenguaje que se encuentre familiarizado el usuario (C / C ++, Java, Python y MATLAB), además este simulador es uno de los pocos que funcionan en multiplataforma, incluyendo Windows, entre otros. Webots sigue en continua revisión y desarrollo, anteriormente era un software comercial con licencia profesional y estudiantil, pero desde diciembre de 2018 se lanza la versión R2019a en código libre. La versión actual es R2020a, utilizada para el término del desarrollo de este trabajo.

Para poder resolver el problema antes mencionado se plantean los siguientes objetivos.

Objetivos

General:

Realizar la estabilización y el seguimiento de trayectorias de un cuadri-rotor mediante el software Webots.

Específicos:

- Estudiar las diferentes arquitecturas de los vehículos aéreos no tripulados (VANT) y su normatividad en México.
- Investigar el modelo dinámico del cuadri-rotor.
- Estudiar las ventajas y limitaciones de las estrategias de control aplicadas al seguimiento de trayectorias en un cuadri-rotor.
- Realizar la comparación de un controlador entre el modelo dinámico en MATLAB y Webots.
- Realizar simulaciones de estabilización y seguimiento de trayectorias de un cuadri-rotor mediante el software Webots.
- Publicación de resultados.
- Escritura de tesis.

Organización de la Tesis

En el capítulo 1 se introducen los antecedentes de los vehículos aéreos no tripulados y los fundamentos importantes para el desarrollo de la tesis.

En el capítulo 2 se presenta la cinemática del cuadri-rotor, continuando con el modelo dinámico mediante el método de Euler-Lagrange y otras representaciones de la dinámica del cuadri-rotor adecuadas para las técnicas de control propuestas.

En el capítulo 3 se presentan las estrategias de control PD, control Backstepping, control LQR y control del cuadri-rotor basado en Planitud, para fines de estabilización de vuelo y el seguimiento de trayectorias.

En el capítulo 4 se tiene la implementación de las simulaciones de las estrategias de control en MATLAB y Webots. También se muestra la construcción del cuadri-rotor en el ambiente de Webots.

En el capítulo 5 se muestran y discuten los resultados de las simulaciones de cada algoritmo de control en el seguimiento de dos trayectorias propuestas, en MATLAB y Webots.

Para fines prácticos en este trabajo y después del capítulo 1, los términos: vehículo aéreo no tripulado, vehículo aéreo, dron, helicóptero y cuadri-rotor tienen el mismo significado.

Capítulo 1

Antecedentes y Fundamentos

Hablar de vehículos aéreos no tripulados suena tan actual, que no se piensa en los sucesos que, a lo largo de la historia, han dado forma a lo que conocemos del gran auge de drones en la vida cotidiana en la sociedad. Por lo que en este trabajo se presenta una breve historia del año 425 a. C. a 1991.

De todos los diferentes UAV (del inglés Unmanned Aerial Vehicle) se encuentra dentro de los más comunes al cuadri-rotor, el cual se aborda en este trabajo, debido a sus características de despegue vertical y vuelo estacionario, aun así, existen características en otras arquitecturas, que pudieran ser de utilidad para otra aplicación. Se presentan algunas clasificaciones y características de los VANT.

La gran popularidad de estos vehículos, ha generado la necesidad de establecer una normatividad en el uso, con el fin de preservar la seguridad de usuarios y terceros. En México la SCT (Secretaria de Comunicaciones y transportes) presenta el proyecto PROY-NOM-107-SCT3-2016, donde se regula el uso de RPAS (del inglés Remotely Piloted Aircraft System), la cual presenta una actualización a septiembre de 2019.

Retomando la idea de planificación de movimiento por planitud de la tesis doctoral de C. Martínez [11], donde proporciona una técnica de control de tolerancia a fallas activa basada en la planitud, donde se usan los conceptos de planitud presentados en [12], para la planificación de movimiento. Uno de sus casos de aplicación es el cuadri-rotor en simulación numérica en MATLAB, por lo que se encuentra la motivación para implementar la generación de trayectorias con el sistema diferencialmente plano, para la realización de este trabajo de simulación en Webots.

La dinámica del cuadri-rotor es de naturaleza no lineal como veremos en el siguiente capítulo, por lo que es posible utilizar técnicas de control no lineales, pero también es de interés aplicar técnicas de control lineal. Para aplicar técnicas de control lineal es necesario linealizar el sistema.

1.1 Historia del UAV

La primera máquina voladora autónoma conocida fue en el año 425 a.C., un pájaro mecánico hecho de madera, llamado "The Pigeon", construido por Archytas de la ciudad de Tarantas o Tarentum en el sur de Italia. El pájaro era balanceado con pesos, usaba vapor encerrado en su estómago para volar hasta terminar su energía térmica [13].

Para 1898, Nikola Tesla inventó el “Teleautomaton”: un vehículo naval capaz de moverse, detenerse, girar a izquierda o derecha y enviar diferentes señales de radio. En 1912 la invención resurgió brevemente como prototipo de torpedo radio controlado.

Durante la Primera Guerra Mundial, la aviación no tripulada se estancó debido a la falta de desarrollo de la tecnología, de esta forma Elmer Ambrose Sperry realizó experimentos con giróscopos para aplicaciones marítimas, que lo llevaron a desarrollar un giroestabilizador para un avión en 1909. Posteriormente junto con Glenn Hammond Curtiss mejoró su invento en 1911, resultando ganador en 1914 de un premio en una exposición en Francia, incluyendo una invención anterior: Un horizonte artificial primitivo [14].

En 1915 Peter Cooper Hewitt y Sperry retomaron las ideas de Tesla. Los americanos Lawrence y Sperry desarrollaron un giroscopio, para estabilizar el fuselaje del avión, con la idea de fabricar un piloto automático. Así en 1916 fabrican el primer UAV. Lawrence y Sperry llamaron a su dispositivo “avión torpedo” mostrado en la figura 1.1a el cual logró superar una distancia de 30 millas de vuelo [15].

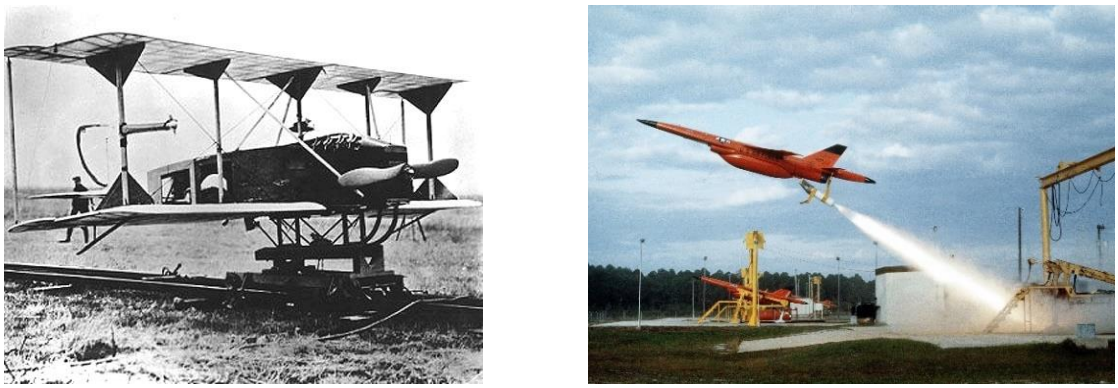


Fig. 1.1. a) Primer UAV del mundo, 1916. b) UAV en los 1960s (Firebee).

En 1917 la Armada de EE.UU. financió la idea y entregó cinco hidroaviones “Curtiss N-9” para desarrollar el experimento. Paralelamente la Curtiss Aeroplane & Motor Company se embarcaron en la fabricación de fuselajes para torpedos aéreos no tripulados, entregando los primeros seis llamados Speed Scout. El primer vuelo controlado con éxito de un avión no

tripulado tuvo lugar finalmente el 6 de marzo de 1918, 14 años después del de los hermanos Wright. En octubre de 1918 fue equipado con catapultas.

Los primeros sistemas desarrollados como armas de largo alcance (precursores de los actuales misiles de crucero) fueron dispositivos como Liberty Eagle, el torpedo aéreo americano de 1917, conocido como "Kettering Bug" de 1918 y el blanco aéreo británico A.T. (del inglés Aerial Target), iniciado en 1914 "Kettering bug" era un biplano más ligero. Por otro lado, el A.T. británico era un avión monoplano no tripulado radio-controlado y propulsado por un motor de 35 caballos de potencia.

El concepto de la serie A.T. sirvió para demostrar la viabilidad del uso de señales de radio como sistema de guiado para volar el avión a su destino pues eran sistemas de orientación toscos y poco fiables, por lo que ningún dispositivo fue desarrollado con éxito, aunque, marcaron una nueva era tecnológica.

Durante la década de 1920 el ejército británico desarrolló un avión monoplano capaz de transportar un cargamento militar de 114 kg, siendo capaz de volar a una distancia de 480 km, y realizando su primer vuelo en 1927. Este avión disponía de un sistema controlado por radio durante los primeros momentos del vuelo, pero luego era capaz de seguir un plan de vuelo específico [14].

Durante la guerra Fría el desarrollo de vehículos aéreos no tripulados continuo y para la década de 1960 se tiene un UAV llamado Firebee mostrado en la figura 1.1b Después de la guerra los EE.UU. e Israel comenzaron a desarrollar vehículos aéreos no tripulados más pequeños y más baratos. Estos eran aeronaves que adoptaron motores pequeños tales como los utilizados en motocicletas o motos de nieve que llevan cámaras de video y transmitían las imágenes a la ubicación del operador.

Los EE.UU. pusieron UAVs en práctica en la Guerra del Golfo en 1991, y vehículos aéreos no tripulados para aplicaciones militares se desarrollaron rápidamente después de esto. El más famoso UAV para uso militar es el Predador, que se muestra en la figura 1.2a.

Fig. 1.2. a) Predador en uso militar. b) UAVs de la Nasa.

Por otra parte, la NASA estaba en el centro de la investigación para uso civil durante este período. El ejemplo más típico de esta época fue el proyecto ERAST (tecnología de sensores y aeronaves de investigación ambiental). Se inició en la década de 1990, y fue un esfuerzo de investigación de síntesis para un UAV que incluyó el desarrollo de la tecnología de vuelo prolongado, motor, sensor, etc. Los aviones que se han desarrollado en este proyecto incluyeron Helios, Proteus, Altus y Perseus, etc., se muestran en la figura 1.2b.

1.2 Clasificación de los vehículos aéreos no tripulados

Son diversas las clasificaciones para los UAV, la cual puede estar basada en su peso, tamaño, configuración aerodinámica, etc., o combinaciones de ellas. En este trabajo se consideran importantes las siguientes clasificaciones.

1.2.1 Clasificación en México

La Secretaría de Comunicaciones y Transportes (SCT) de México clasifica a las aeronaves de acuerdo con su Peso máximo de despegue (MTOW Maximum Take Off Weight) [16], mostrado en la Tabla 1.1.

Tabla 1.1 Clasificación en Categorías de RPAS.

CLASIFICACIÓN DE SISTEMAS DE AERONAVES PILOTADAS A DISTANCIA	
PESO MÁXIMO DE DESPEGUE	**CATEGORÍA**
2.000 kg o menos	RPAS Micro
2.001 kg hasta 25.000 kg	RPAS Pequeño
Más de 25.001 kg	RPAS Grande

1.2.2 Clasificación por aplicación

También se puede clasificar de acuerdo con sus aplicaciones [17] [18]:

Aplicaciones civiles

- Monitoreo de tuberías.
- Recopilación de fotografías aéreas.
- Seguimiento y control vehicular.
- Vigilancia, localización y rescate de personas.

- Control de desastres naturales, medición y análisis de niveles de contaminación atmosférica.
- Análisis de caudal de los ríos e inundaciones.
- Control de incendios forestales
- Vigilancia aérea para el control del orden en eventos públicos.
- Cinematografía.
- Servicio de envío de paquetes.
- Inspección de redes eléctricas.
- Vigilancia marítima.

Aplicaciones militares

- Localización y destrucción de minas terrestres.
- Monitoreo de contaminación biológica, química y nuclear.
- Detección y desvío de misiles.
- Eliminación de explosivos.
- Vigilancia de actividad enemiga.

1.2.3 Clasificación por aerodinámica

Otra clasificación de las plataformas UAV en función de su configuración aerodinámica, es la siguiente [18] [19]:

Un solo rotor

Está compuesto por un solo rotor y alerones para compensar el torque del rotor, yaw. Si el rotor no tiene el plato cíclico, tiene alerones adicionales para producir los torques de roll y pitch.

La configuración de un solo rotor es mecánicamente más simple que un helicóptero estándar, aunque, es difícil de controlar. Una cantidad significativa de la energía se utiliza en el anti-torque, es decir, para detener al fuselaje de girar alrededor del eje vertical. Sin embargo, debido a su simplicidad mecánica es el la configuración más adecuado para microaviones. Como ejemplo de esta configuración podemos ver al avión llamado 3D en la figura 1.3a.

Doble rotor

Se puede distinguir dos tipos, aquellos que usan plato rotatorio y aquellos que utilizan el plato fijo. Entre los vehículos que utilizan placas cíclicas, podemos citar las siguientes: el clásico helicóptero, el helicóptero tándem y el helicóptero coaxial. Sin plato cíclico, tenemos al avión de doble rotor con alerones, es decir, dos rotores colocados en diferentes ejes y los alerones orientados en la dirección del rotor flujo de aire con el fin de obtener los torques necesarios para controlar el vehículo. Los rotores pueden girar en dirección opuesta o en la misma dirección. Un ejemplo de estos es el T-Wing de la Universidad de Sídney, se muestra

en figura 1.3b. También es posible tener dos rotores en el mismo eje y alerones en la dirección del flujo de aire. Esta última configuración es muy compacta, pero difícil de controlar. Por último, podemos tener dos rotores que se inclinan sobre dos ejes. En esta configuración, las hélices no tienen un plato cíclico y los rotores pueden ir en dos direcciones diferentes.

Fig. 1.3. a) El avión 3D. b) El T-Wing.

Fig. 1.4. a) Helicóptero con cuatro rotores b) Dirigible LSC.

Multi-rotores

Se tiene de tres, cuatro y con más de cuatro rotores. En el helicóptero de cuatro rotores o el cuadri-rotor, como el que se muestra en la figura 1.4a, se puede alcanzar la estabilidad y precisión de vuelo equilibrando las fuerzas producidas por los cuatro rotores. Una de las ventajas de utilizar un multirrotor es que se puede aumentar la capacidad de carga al aumentar más rotores.

Los cuadrirrotores son altamente maniobrables y permiten el despegue y el aterrizaje verticales, también tienen buen rendimiento volando en condiciones duras para alcanzar áreas específicas. Las principales desventajas son lo pesado de la aeronave y el alto consumo de energía debido a los motores extra.

Dirigible

Un dirigible es un avión más ligero que el aire, que puede ser dirigido y propulsado usando timones, hélices u otro empuje. En oposición a otras aeronaves aerodinámicas como aviones de ala fija y helicópteros que producen elevación moviendo un ala o una hélice a través del aire, los aviones aerostáticos (dirigibles, aire caliente globos, etc.) como el la figura 1.4b, permanecen en lo alto llenando una cavidad grande con algún gas de elevación.

Los tipos principales de dirigible son no rígidos, semi rígidos y rígidos. Los no rígidos son pequeños dirigibles sin esqueleto interno, los dirigibles semi rígidos son un poco más grandes y tienen cualquier forma de apoyo interno, los rígidos tiene estructuras rígidas que sostienen las bolsas de gas.

Avión

Un avión, o aeroplano como el de la figura 1.5a, es una clase de avión que utiliza las alas para generar fuerza de ascensión. Al cuerpo del avión se le llama fuselaje, suele ser en forma de un tubo largo. El aire que fluye sobre la parte superior del ala más rápido que por debajo, genera que la presión del aire sobre el ala sea más baja, creando el levantamiento ascendente. El diseño de las alas determina que tan rápido y alto puede volar el avión.

UAV de ala batiente

Una nueva tendencia en la comunidad UAV es tomar inspiración de insectos voladores o las aves para lograr capacidades de vuelo sin precedentes, como se puede ver en la figura 1.5b. Los sistemas biológicos son interesantes por su manera inteligente de depender de la aerodinámica inestable utilizando aleteo alas, son cada vez más inspiración para ingenieros entre otros aspectos como detección y actuación, fusión de sensores y procesamiento de información. Las aves demuestran que el vuelo del aleteo-ala es un modo de vuelo versátil, compatible con el flotar, vuelo de proa y deslizamiento para ahorrar energía. Sin embargo, el diseño es un reto porque la eficiencia aerodinámica está condicionada por los movimientos complejos de las alas.

Fig. 1.5. a) La configuración de avión. b) El Dragonfly.

1.3 Regulación de las aeronaves no tripuladas en México

A continuación, se abordan brevemente algunos puntos importantes que se incluyen en la Norma Oficial Mexicana [16] con la actualización [20] publicada el 10 de septiembre de 2019.

Objetivo y campo de aplicación

La presente Norma Oficial Mexicana establece los requerimientos del Sistema de Aeronave Pilotada a Distancia (RPAS) para operar dentro del espacio aéreo mexicano; de la misma manera para su comercialización en el territorio nacional. El campo de aplicación va dirigido a toda persona física/moral, operadores de estado que pretendan operar u operen un RPAS.

La presente Norma Oficial Mexicana no aplica a los RPAS de Estado que efectúen operaciones militares que son destinadas o en posesión del Ejército, Armada y Fuerza Aérea Nacionales; mismas que deben sujetarse a las disposiciones de tránsito aéreo señaladas en el artículo 37 de la Ley de Aviación Civil; asimismo no es aplicable a aeronaves no tripuladas clasificadas como autónomas, ni a los globos libres no tripulados

Clasificación del RPAS

Todo operador de RPAS que pretenda operar en espacio aéreo mexicano, debe dar cumplimiento a la presente Norma Oficial Mexicana, con base al peso máximo de despegue y uso del RPA, de conformidad con la Tabla 1.2:

Tabla 1.2 Clasificación de RPAS en México [20].

CLASIFICACIÓN DE SISTEMAS DE AERONAVES PILOTADAS A DISTANCIA			
PESO MÁXIMO DE DESPEGUE		**USO**	**Numeral NOM**
Igual o menor a 2 kg	RPAS Micro	Privado Recreativo	4.10, 4.11 y 5.1
		Privado No Comercial o Comercial	4.10, 4.11, 5.2 y 8
2.001 kg hasta 25.000 kg	RPAS Pequeño	Privado Recreativo	4.10, 4.11 y 6.1
		Privado No Comercial o Comercial	4.10, 4.11, 6.2 y 8
Más de 25.001 kg	RPAS Grande	Privado Recreativo	4.10, 4.11 y 7.1
		Privado No Comercial o Comercial	4.10, 4.11, 7.2 y 8

Requerimientos generales de operación

- El piloto debe operar el RPAS a una distancia de separación de al menos 9.2 km (5 MN) de cualquier aeródromo.

- El piloto del RPAS no debe dejar caer y/o arrojar (aunque tenga paracaídas) objetos o materiales que puedan causar daño a cualquier persona o propiedad.
- El piloto del RPAS debe mantener el control de la trayectoria de vuelo de la aeronave pilotada a distancia en todo momento.
- El operador y/o piloto del RPAS debe operar entre la salida y la puesta del sol.
- El piloto del RPAS debe dar en todo momento y sin excepción alguna, el derecho de paso a cualquier aeronave tripulada.
- El piloto del RPAS no debe operar más de una RPA al mismo tiempo.

Responsabilidades

- El operador y/o piloto del RPAS es el responsable de su operación, uso y en caso de incidente o accidente, de los daños y/o lesiones causados por la misma.
- El operador del RPAS es el responsable del uso que se dé a la información obtenida durante la operación de la aeronave.

Requerimientos

- Registrar las altas, bajas y cambios de los RPAS con un peso máximo de despegue mayor a 0.250 kg, en el sitio de internet de la SCT/DGAC.
- Operar la RPA a una altura máxima de 122 m.
- No operar la RPA más allá de una distancia horizontal de 457 m respecto al piloto.
- No exceder la velocidad máxima de operación, establecida por el fabricante.
- No se debe operar sobre personas a menos que participen directamente en la operación de la RPA o estén situadas debajo de una estructura que les provea de una protección razonable en caso de desplome de la RPA.
- Si el peso máximo de despegue de la RPA es igual o menor a 250 g, esta se puede operar sobre personas.

1.4 Sistemas diferencialmente planos

Un sistema es diferencialmente plano (o simplemente plano) si su comportamiento puede ser completamente descrito por un conjunto de funciones diferenciales independientes dependiendo de las variables del sistema y sus derivadas [21]. Los primeros trabajos con el objetivo de aplicaciones aeronáuticas se llevaron a cabo en [22].

La planitud diferencial de los sistemas no lineales y lineales podría describirse utilizando formalismos matemáticos, y específicamente de álgebra diferencial o de geometría diferencial [11].

1.4.1 Definición de planitud

Consideremos el sistema no lineal $\dot{x} = f(x, u)$, $x \in \mathbb{R}^n$ el vector de estados $u \in \mathbb{R}^m$ el vector de control y f a C^∞ función de x y u. El sistema es diferencialmente plano si, y solo si, existe un vector de salida plano $z \in \mathbb{R}^m$, tal que [23]:

- El vector de salida plana se expresa como una función del estado x y la entrada de control u y un número finito de sus derivadas de tiempo

$$z = \phi_z(x, u, \dot{u}, \dots, u^\gamma) \tag{1.1}$$

- El estado x y la entrada de control u y se expresan como funciones del vector z y un número finito de sus derivadas de tiempo

$$x = \phi_z(z, \dot{z}, \dots, z^a) \tag{1.2}$$

$$u = \phi_u(z, \dot{z}, \dots, z^{(a+1)}) \tag{1.3}$$

donde z^a es la derivada iésima de cada componente del vector de salida plano z.

Cada sistema plano es equivalente a uno lineal controlable por medio de difeomorfismo y retroalimentación dinámica endógena, en consecuencia, cada sistema lineal controlable es plano y viceversa. Además, con respecto a la observabilidad, un sistema plano siempre es observable desde las salidas planas.

1.5 Planificación de movimiento

El objetivo de la planificación de movimiento es proporcionar al sistema las trayectorias deseadas y viables para las cuales existe una entrada de control correspondiente que mueve el sistema en cuestión desde un estado inicial a una condición de objetivo, respetando las restricciones y evitando la colisión [11].

Planificación de ruta: una representación geométrica para pasar de una condición inicial a una final. El objetivo principal es encontrar un camino sin colisiones entre una colección de obstáculos estáticos y dinámicos.

Planificación de trayectoria: también conocida como generación de trayectoria. Incluye velocidades, aceleraciones y jerks a lo largo del camino. Normalmente, la tarea principal es encontrar trayectorias para rutas especificadas a priori. Esas trayectorias podrían verse obligadas a cumplir un cierto criterio.

Planificación de movimiento: es la unión de la planificación de trayectoria y de ruta.

1.5.1 Planificación de movimiento basado en planitud

El objetivo de la planificación del movimiento es calcular una trayectoria que satisfaga ciertas restricciones de ruta [24].

Definamos un sistema no lineal $\dot{x} = f(x, u)$. La planificación del movimiento consiste en cumplir las condiciones iniciales y finales que se presentan a continuación [24]:

$$x(t_i) = x_i, u(t_i) = u_i \tag{1.4}$$

$$x(t_f) = x_f, u(t_f) = u_f \tag{1.5}$$

Una vez que se define la ruta, el problema de generación de trayectoria consiste en encontrar una trayectoria $t \rightarrow x(t), u(t)$ para $t \in [t_i, t_f]$ que satisface las restricciones del sistema y las condiciones iniciales y finales (1.4) - (1.5). Las restricciones de trayectoria de tipo $\big(x(t), u(t)\big) \in A(t)$, donde $A(t)$ es una subvariedad de $X \times U$ podrían agregarse al problema inicial de planificación de movimiento. Esto da como resultado una complejidad creciente que requiere una solución iterativa mediante métodos numéricos para encontrar la entrada de control $u(t)$ que satisfaga las condiciones iniciales y finales (1.4) y (1.5). Este proceso iterativo se puede resolver utilizando técnicas de control óptimas, sin embargo, para sistemas no lineales, algunos problemas aún no se han resuelto.

Además, esta solución necesita integrar las ecuaciones del sistema para evaluar la solución propuesta.

La planificación de movimiento por planitud, no necesita integrar las ecuaciones del sistema y para una trayectoria de salida plana, las entradas de control se pueden calcular directamente. El vector $u(t)$ resultante siempre respeta la dinámica del sistema, ver ecuación (1.3). Como consecuencia, se encuentran las soluciones del conjunto de ecuaciones diferenciales [25].

La definición de planitud implica que para cada variable del sistema puede expresarse en términos de salidas planas y un número finito de sus derivadas. Como consecuencia, si queremos calcular una trayectoria cuyas condiciones iniciales y finales se especifican, es suficiente construir una trayectoria de salida plana para obtener las entradas de control de bucle abierto que satisfagan el estado deseado y las trayectorias de entrada.

Para calcular todas las variables del sistema, la trayectoria de salida plana creada debe ser al menos r veces diferenciable, donde r es la derivada de tiempo máximo de la salida plana que aparece en (1.1) y (1.3). En este trabajo, las trayectorias de las salidas planas se crean de la derivación numérica de Euler debido a su fácil implementación, tanto en simulación como para implementación en el futuro de este trabajo en tarjetas embebidas de bajo costo.

1.6 Linealización

Una herramienta útil para describir la dinámica aproximada del sistema, son las ecuaciones de estado lineales. La obtención de un modelo lineal a partir de uno no lineal se llama linealización. Esta generalmente consiste en una expansión en series de Taylor de la ecuación de estado (no-lineal) alrededor de un punto de operación definido naturalmente por el sistema o seleccionado arbitrariamente para satisfacer alguna necesidad de control.

1.6.1 Punto de equilibrio

Para calcular los puntos de equilibrio del sistema, se buscan los valores de las variables de estado que hacen nulo el vector de estados; para esto se iguala el vector de estado a cero y resolvemos el sistema de ecuaciones. Es conveniente elegirlo próximo al punto de funcionamiento del sistema para minimizar el error cometido en la linealización.

El cuadrirrotor tiene un punto de funcionamiento principal, que corresponde al punto donde el vehículo aéreo está fijo en una coordenada x, y, z. Este punto de funcionamiento se da cuando la aceleración es el eje z y los ángulos del cuadrirrotor sean cero, es decir se logra estabilización de vuelo, correspondiendo a un punto de equilibrio.

$$\psi_{eq} = \theta_{eq} = \phi_{eq} = 0 \tag{1.6}$$

$$\ddot{z}_{eq} = -g + \frac{1}{m}\left(cos\theta_{eq} cos\phi_{eq}\right)U_1 = 0 \tag{1.7}$$

Resolviendo las ecuaciones encontramos que el punto de equilibrio es:

$$U_{1eq} = mg \tag{1.8}$$

1.6.2 Linealización del modelo

El modelo no lineal del cuadri-rotor está formado por doce estados y cuatro entradas, las ecuaciones que describen la dinámica del sistema [11].

$$x_1 = x, x_2 = \dot{x}, x_3 = y, x_4 = \dot{y}, x_5 = z, x_6 = \dot{z}$$

$$x_7 = \psi, x_8 = \dot{\psi}, x_9 = \theta, x_{10} = \dot{\theta}, x_{11} = \phi, x_{12} = \dot{\phi} \tag{1.9}$$

$$U_1, U_2, U_3, U_4 \tag{1.8}$$

El modelo original en donde x, y, z son las coordenadas de traslación y ψ, θ y ϕ son las coordenadas de rotación, se puede describir en ecuaciones de estados.

$$\begin{aligned}\frac{\partial x_1}{\partial t} &= f_1(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, u_1, u_2, u_3, u_4)\\ &\vdots\\ \frac{\partial x_{12}}{\partial t} &= f_{12}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, u_1, u_2, u_3, u_4)\end{aligned} \quad (1.9)$$

La linealización se realiza en torno a un punto de operación, donde los valores nominales satisfacen al modelo no lineal. El punto de operación o punto de equilibrio donde se realizará la linealización corresponde a P_{eq}:

$$(x_{1eq}, x_{2eq}, x_{3eq}, x_{4eq}, x_{5eq}, x_{6eq}, x_{7eq}, x_{8eq}, x_{9eq}, x_{10eq}, x_{11eq}, x_{12eq}, u_{1eq}, u_{2eq}, u_{3eq}, u_{4eq})^T \quad (1.10)$$

Aplicando el teorema de Taylor en el punto P_{eq}, sobre (1.9), las ecuaciones de estado del sistema, se tiene:

$$\begin{aligned}f_1 \approx f_1(P_{eq}) &+ \left(\frac{\partial f_1}{\partial x_1}\right)_{P_{eq}}(x_1 - x_{1eq}) + .. + \left(\frac{\partial f_1}{\partial x_{12}}\right)_{P_{eq}}(x_{12} - x_{12eq}) + \left(\frac{\partial f_1}{\partial u_1}\right)_{P_{eq}}(u_1 - u_{1eq})\\ &+ \cdots + \left(\frac{\partial f_1}{\partial u_4}\right)_{P_{eq}}(u_4 - u_{4eq}) + T.O.S\end{aligned}$$

$$\vdots$$

$$\begin{aligned}f_{12} \approx f_{12}(P_{eq}) &+ \left(\frac{\partial f_{12}}{\partial x_1}\right)_{P_{eq}}(x_1 - x_{1eq}) + .. + \left(\frac{\partial f_{12}}{\partial x_{12}}\right)_{P_{eq}}(x_{12} - x_{12eq}) + \left(\frac{\partial f_{12}}{\partial u_1}\right)_{P_{eq}}(u_1 - u_{1eq})\\ &+ \cdots + \left(\frac{\partial f_{12}}{\partial u_4}\right)_{P_{eq}}(u_4 - u_{4eq}) + T.O.S\end{aligned} \quad (1.11)$$

donde T.O.S. denota términos de orden superior.

Reescribiendo el sistema del modelo original (1.9) como:

$$\begin{aligned}\frac{\partial x_1}{\partial t} = \frac{\partial(x_1 - x_{1eq})}{\partial t} \rightarrow \frac{\partial \bar{x}_1}{\partial t} &= f_1(\bar{x}_1, \dots, \bar{x}_{12}, \bar{u}_1, \dots, \bar{u}_4)\\ &\vdots\\ \frac{\partial x_{12}}{\partial t} = \frac{\partial(x_{12} - x_{12eq})}{\partial t} \rightarrow \frac{\partial \bar{x}_{12}}{\partial t} &= f_{12}(\bar{x}_1, \dots, \bar{x}_{12}, \bar{u}_1, \dots, \bar{u}_4)\end{aligned} \quad (1.12)$$

Sustituyendo en las ecuaciones (1.12) todas las funciones f_1 ... f_{12} despreciando los términos de orden superior y conociendo que en el punto de equilibrio $f_1(P_{eq}) = 0$ a $f_{12}(P_{eq}) = 0$, tenemos:

$$\begin{aligned}\frac{\partial \bar{x}_1}{\partial t} &= \left(\frac{\partial f_1}{\partial x_1}\right)_{P_{eq}}\bar{x}_1 + .. + \left(\frac{\partial f_1}{\partial x_{12}}\right)_{P_{eq}}\bar{x}_{12} + \left(\frac{\partial f_1}{\partial u_1}\right)_{P_{eq}}\bar{u}_1 + \cdots + \left(\frac{\partial f_1}{\partial u_4}\right)_{P_{eq}}\bar{u}_4\\ &\vdots\\ \frac{\partial \bar{x}_{12}}{\partial t} &= \left(\frac{\partial f_{12}}{\partial x_1}\right)_{P_{eq}}\bar{x}_1 + .. + \left(\frac{\partial f_{12}}{\partial x_{12}}\right)_{P_{eq}}\bar{x}_{12} + \left(\frac{\partial f_{12}}{\partial u_1}\right)_{P_{eq}}\bar{u}_1 + \cdots + \left(\frac{\partial f_{12}}{\partial u_4}\right)_{P_{eq}}\bar{u}_4\end{aligned} \quad (1.13)$$

El anterior sistema de ecuaciones se puede escribir de forma matricial como:

$$\frac{\partial \bar{x}}{\partial t} = A\bar{x} + B\bar{u} \tag{1.14}$$

donde

$$A = \begin{bmatrix} \left(\frac{\partial f_1}{\partial x_1}\right)_{P_{eq}} & \dots & \left(\frac{\partial f_1}{\partial x_{12}}\right)_{P_{eq}} \\ \vdots & \dots & \vdots \\ \left(\frac{\partial f_{12}}{\partial x_1}\right)_{P_{eq}} & \dots & \left(\frac{\partial f_{12}}{\partial x_{12}}\right)_{P_{eq}} \end{bmatrix} ; \; B = \begin{bmatrix} \left(\frac{\partial f_1}{\partial u_1}\right)_{P_{eq}} & \dots & \left(\frac{\partial f_1}{\partial u_4}\right)_{P_{eq}} \\ \vdots & \dots & \vdots \\ \left(\frac{\partial f_{12}}{\partial u_1}\right)_{P_{eq}} & \dots & \left(\frac{\partial f_{12}}{\partial u_4}\right)_{P_{eq}} \end{bmatrix} \tag{1.15}$$

La ecuación (1.14) representa la dinámica del sistema en variables x de estados y entradas u, pero también se puede obtener con el mismo procedimiento, anteriormente descrito, el vector de salidas y en función de las variables de estados x y entradas u.

$$y = g(x, u) \tag{1.16}$$

Aplicando el teorema de Taylor en el punto P_{eq}, sobre (1.9), las ecuaciones de estado del sistema, se obtiene:

$$\begin{aligned} g_1 \approx g_1(P_{eq}) + \left(\frac{\partial g_1}{\partial x_1}\right)_{P_{eq}} (x_1 - x_{1eq}) + .. + \left(\frac{\partial g_1}{\partial x_{12}}\right)_{P_{eq}} (x_{12} - x_{12eq}) + \left(\frac{\partial g_1}{\partial u_1}\right)_{P_{eq}} (u_1 - u_{1eq}) \\ + \cdots + \left(\frac{\partial g_1}{\partial u_4}\right)_{P_{eq}} (u_4 - u_{4eq}) + T.O.S \\ \vdots \\ g_{12} \approx g_{12}(P_{eq}) + \left(\frac{\partial f_{12}}{\partial x_1}\right)_{P_{eq}} (x_1 - x_{1eq}) + .. + \left(\frac{\partial g_{12}}{\partial x_{12}}\right)_{P_{eq}} (x_{12} - x_{12eq}) + \left(\frac{\partial g_{12}}{\partial u_1}\right)_{P_{eq}} (u_1 - u_{1eq}) \\ + \cdots + \left(\frac{\partial f_{12}}{\partial u_4}\right)_{P_{eq}} (u_4 - u_{4eq}) + T.O.S \end{aligned} \tag{1.17}$$

De la ecuación (1.10) sabemos que,

$$y_1 = g_1(P_{eq}) \; \dots \; y_{12} = g_{12}(P_{eq}) \tag{1.18}$$

Reescribiendo el sistema de ecuaciones (1.17), despreciando términos de orden superior:

$$\begin{aligned} g_1 = y_{1eq} + \left(\frac{\partial g_1}{\partial x_1}\right)_{P_{eq}} \bar{x}_1 + .. + \left(\frac{\partial g_1}{\partial x_{12}}\right)_{P_{eq}} \bar{x}_{12} + \left(\frac{\partial g_1}{\partial u_1}\right)_{P_{eq}} \bar{u}_1 + \cdots + \left(\frac{\partial g_1}{\partial u_4}\right)_{P_{eq}} \bar{u}_4 \\ \vdots \\ g_{12} = y_{12eq} + \left(\frac{\partial g_{12}}{\partial x_1}\right)_{P_{eq}} \bar{x}_1 + .. + \left(\frac{\partial g_{12}}{\partial x_{12}}\right)_{P_{eq}} \bar{x}_{12} + \left(\frac{\partial g_{12}}{\partial u_1}\right)_{P_{eq}} \bar{u}_1 + \cdots + \left(\frac{\partial g_{12}}{\partial u_4}\right)_{P_{eq}} \bar{u}_4 \end{aligned} \tag{1.19}$$

Sustituyendo (1.19) en (1.16) y escribiendo el sistema en forma matricial, resulta:

$$\frac{\partial \bar{y}}{\partial t} = C\bar{x} + D\bar{u} \tag{1.20}$$

donde

$$C = \begin{bmatrix} \left(\frac{\partial g_1}{\partial x_1}\right)_{P_{eq}} & \dots & \left(\frac{\partial g_1}{\partial x_{12}}\right)_{P_{eq}} \\ \vdots & \dots & \vdots \\ \left(\frac{\partial g_{12}}{\partial x_1}\right)_{P_{eq}} & \dots & \left(\frac{\partial g_{12}}{\partial x_{12}}\right)_{P_{eq}} \end{bmatrix} ; \; D = \begin{bmatrix} \left(\frac{\partial g_1}{\partial u_1}\right)_{P_{eq}} & \dots & \left(\frac{\partial g_1}{\partial u_4}\right)_{P_{eq}} \\ \vdots & \dots & \vdots \\ \left(\frac{\partial g_{12}}{\partial u_1}\right)_{P_{eq}} & \dots & \left(\frac{\partial g_{12}}{\partial u_4}\right)_{P_{eq}} \end{bmatrix} \tag{1.21}$$

El sistema linealizado queda:

$$\frac{\partial \bar{x}}{\partial t} = A\bar{x} + B\bar{u} \tag{1.22}$$

$$\frac{\partial \bar{y}}{\partial t} = C\bar{x} + D\bar{u} \tag{1.23}$$

El sistema linealizado (1.22) – (1.23) se utiliza para el diseño del control LQR.

Este capítulo presentó una breve historia de los vehículos aéreos no tripulados, su clasificación y su normativa en México para establecer los parámetros deseados, para la simulación del cuadri-rotor dentro del ambiente virtual. También se establecen algunos conceptos que serán utilizados en la descripción de los modelos dinámicos, abordados el siguiente capítulo.

Capítulo 2

Modelado del sistema

El vehículo aéreo no tripulado tipo cuadri-rotor está formado por un marco cruzado rígido y simétrico con cuatro rotores coplanarios en los extremos. La variación controlada de las velocidades en los rotores permite el movimiento del cuadri-rotor (ver figura 2.1).

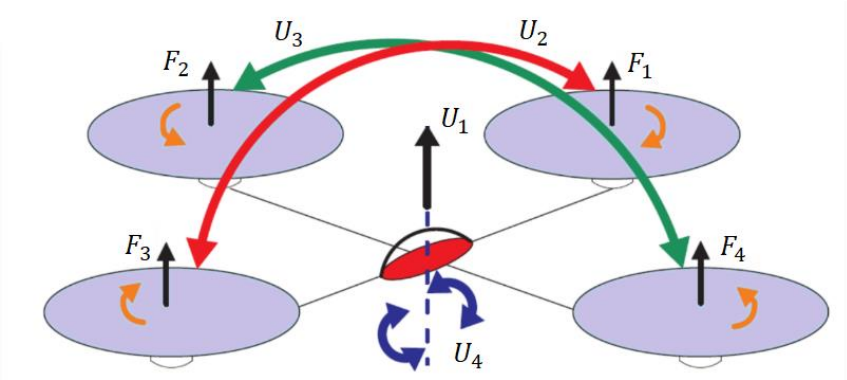


Fig. 2.1. Esquema del cuadri-rotor [26].

Las maniobras de vuelo según la figura 2.1, son: 1) el movimiento hacia adelante, girando sobre el eje Y, pitch se logra aumentando y disminuyendo las velocidades de los rotores trasero y delantero respectivamente numerados 1 y 3 que rotan en sentido horario; 2) el desplazamiento lateral, girando sobre X, roll se realiza aumentando y disminuyendo las velocidades con los rotores izquierda y derecha numerados 2 y 4 que giran en sentido antihorario; 3) el movimiento de yaw, girando sobre Z se obtiene acelerando los dos rotores en sentido horario mientras se desaceleran los rotores con sentido antihorario y viceversa para el movimiento hacia el otro sentido, y 4) para el vuelo vertical en necesario aumentar o disminuir en la misma proporción de las velocidades de los cuatro rotores, esto aumentará o disminuirá el empuje total aplicado al vehículo permitiendo ascender o descender. Es importante recordar que los rotores solo pueden girar en el sentido indicado.

2.1 Modelo cinemático

Los ángulos de Tait-Bryan son usados para describir la orientación en el espacio respecto al marco de referencia fijo, estos ángulos son roll, pitch y yaw (ver figura 2.2). La matriz de rotación se obtiene por rotaciones sucesivas alrededor de cada uno de los ejes del cuerpo rígido.

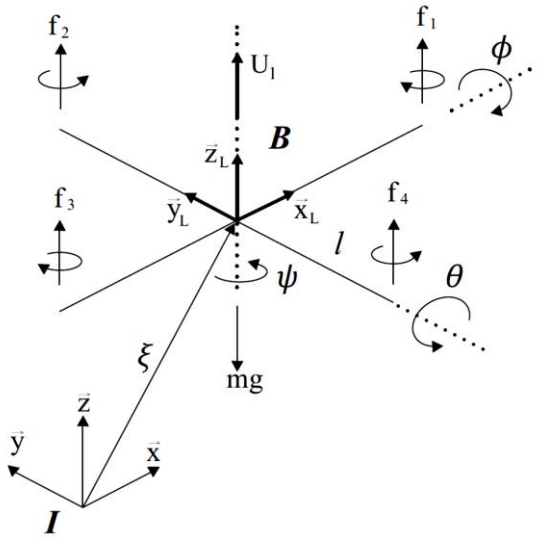


Fig. 2.2 Ejes de coordenadas y fuerzas aplicadas al cuadri-rotor [26].

$$R_{(x,\phi)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sen\phi \\ 0 & sen\phi & cos\phi \end{bmatrix} \tag{2.1}$$

$$R_{(y,\theta)} = \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix} \tag{2.2}$$

$$R_{(z,\psi)} = \begin{bmatrix} cos\psi & -sen\psi & 0 \\ sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Matriz de rotación de cosenos directores del eje B respecto al eje inercial I [27].

$$R_{\mathcal{J}} = R_{(z,\psi)} \cdot R_{(y,\theta)} \cdot R_{(x,\phi)} \tag{2.4}$$

$$R_{\mathcal{J}} = \begin{bmatrix} cos\psi cos\theta & cos\psi sen\theta sin\phi - sen\psi cos\phi & cos\psi sen\theta sen\phi - sen\psi sen\phi \\ sen\psi cos\theta & sen\psi sen\theta sen\phi + cos\psi cos\phi & sen\psi sen\theta cos\phi - cos\psi sen\phi \\ -sen\theta & cos\theta sen\phi & cos\theta cos\phi \end{bmatrix} \tag{2.5}$$

Considerando la propiedad de ortogonalidad, la matriz de rotación respecto al eje de coordenadas del cuerpo B, es la transpuesta de R_J

$$R_{\mathrm{B}} = \begin{bmatrix} cos\psi cos\theta & sen\psi cos\theta & -sen\theta \\ cos\psi sen\theta sen\phi - sen\psi cos\phi & sen\psi sen\theta sen\phi + cos\psi cos\phi & cos\theta sen\phi \\ cos\psi sen\theta sen\phi - sen\psi sen\phi & sen\psi sen\theta cos\phi - cos\psi sen\phi & cos\theta cos\phi \end{bmatrix} \quad (2.6)$$

Las ecuaciones cinemáticas de rotación del vehículo que establecen las relaciones angulares, se obtienen mediante el análisis siguiente:

$$\dot{R}_J = R_J \cdot S(\omega) \quad (2.7)$$

$$S(\omega) = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \quad (2.8)$$

La cinemática rotacional se puede representar mediante la siguiente expresión

$$\eta = W^{-1}\omega \quad (2.9)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sen\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sen\phi \\ 0 & sen\phi sec\theta & cos\phi sec\theta \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.10)$$

donde, $\eta = [\phi \quad \theta \quad \psi]^T$ y ω=$[p \quad q \quad r]^T$, son los ángulos y velocidades angulares respectivamente en los ejes del cuerpo.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & cos\phi & sen\phi cos\theta \\ 0 & -sen\phi & cos\phi cos\theta \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (2.11)$$

Cinemática traslacional

$$v = R_J \cdot V \quad (2.12)$$

donde las velocidades lineales $v = [u_0 \quad v_0 \quad w_0]^T$ y V=$[u_L \quad v_L \quad w_L]^T$ están expresadas en el eje inercial y el eje del cuerpo.

2.2 Modelo dinámico no lineal

El cuadri-rotor es un sistema complejo del vuelo. Esto es debido en parte al número de los efectos físicos que actúan simultáneamente [27].

• Efectos aerodinámicos: rotación de los rotores ($C\Omega^2$)

• Torques inerciales: Cambios de velocidad en la hélice ($J_R\dot{\Omega}$).

• Efectos giroscópicos: Cambio de orientación del cuerpo ($J\theta\psi, J_R\Omega\theta$).

• Gravedad y fricciones ($mg, C\dot{\phi}, \dot{\theta}, \dot{\psi}$).

Este sistema tiene seis grados de libertad y cuatro entradas, siendo un sistema subactuado.

El modelo dinámico del cuadri-rotor se obtiene mediante el método de Euler-Lagrange como lo describe [16], se considera una buena aproximación de la realidad, ya que incluye los fenómenos físicos descritos anteriormente.

Las ecuaciones de movimiento de Euler-Lagrange

$$\Gamma_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \qquad i = 1, \dots, k \tag{2.13}$$

$$L = E_c - E_p \tag{2.14}$$

donde, L es el Lagrangiano, E_c la energía cinética total, E_p la energía potencial total, q_i es la coordenada generalizada, $\dot{q}_i$ es la primera derivada respecto al tiempo de la coordenada generalizada y Γ_i son las fuerzas/pares generados por las fuerzas/pares no conservativos, k el número de coordenadas generalizadas.

Para el planteamiento de las ecuaciones de Euler-Lagrange, se toma en consideración el sistema de coordenadas inercial $I = [\vec{x}\ \vec{y}\ \vec{z}\,]$ y el sistema de coordenadas ligado al cuadri-rotor $B = [\overrightarrow{x_L}\ \overrightarrow{y_L}\ \overrightarrow{z_L}\,]$ como se muestra en la figura 2.2.

Las coordenadas generalizadas están expresadas por [28]:

$$q = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T \ \in \mathbb{R}^6 \tag{2.15}$$

donde $\xi = [x \quad y \quad z]^T \in \mathbb{R}^3$ es la posición del centro de masa expresada en I, y $\eta = [\phi \quad \theta \quad \psi] \in \mathbb{R}^3$.

La expresión del Lagrangiano para el cuadri-rotor está dado por

$$L(q, \dot{q}) = E_{c_{traslación}} + E_{c_{rotación}} - E_p \tag{2.16}$$

A partir de las hipótesis planteadas se tiene la matriz de inercia

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.17}$$

Entonces la energía cinética traslacional [26].

$$E_{c_{traslación}} = \frac{1}{2}\int \dot{\xi}^2(x,y,z)dm = \frac{m}{2}\dot{\xi}^T\dot{\xi} \tag{2.18}$$

La energía cinética rotacional [27].

$$E_{c_{rotación}} = \frac{1}{2}I_{xx}\left(\dot{\phi} - \dot{\psi}sen\theta\right)^2 + \frac{1}{2}I_{yy}(\dot{\theta}cos\phi - \dot{\psi}sen\phi cos\theta)^2 +$$

$$\frac{1}{2}I_{zz}(\dot{\theta}sen\phi - \dot{\psi}cos\phi cos\theta)^2 \tag{2.19}$$

Reescribiendo

$$E_{c_{rotación}} = \frac{1}{2}I_{xx}p^2 + \frac{1}{2}I_{yy}q^2 + \frac{1}{2}I_{zz}r^2 = \frac{1}{2}\omega^T J\omega \tag{2.20}$$

Con W_η como el Jacobiano que relaciona ω con $\dot{\eta}$ en la ecuación se puede definir la siguiente matriz:

$$\mathcal{J} = \mathcal{J}(\eta) = W_\eta^T J W_\eta \tag{2.21}$$

Por lo que la ecuación de la energía cinética se puede escribir en función de las coordenadas generalizadas η.

$$E_{c_{rotación}} = \frac{1}{2}\,\dot{\eta}^T\mathcal{J}\,\dot{\eta} \tag{2.22}$$

La energía potencial

$$E_p = mgz \tag{2.23}$$

Aplicando las ecuaciones (2.13) y (2.14), se obtienen las ecuaciones del movimiento completo dada por la siguiente expresión.

$$\begin{bmatrix} F_\xi \\ \tau_n \end{bmatrix} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \tag{2.24}$$

donde τ_n representa los momentos de roll, pitch y yaw, y $F_\xi = R_I\hat{F}$ es la fuerza de traslación aplicada al quadrotor debido a la entrada del control principal U_1 en dirección del eje Z.

Debido a que el Lagrangiano no contiene energía cinética en términos combinados de $\dot{\xi}$ y $\dot{\eta}$, las ecuaciones de Euler Lagrange pueden ser divididas en la dinámica de traslación y la dinámica de rotación, quedando la ecuación de Euler Lagrange para el movimiento de traslación

$$L(\xi, \dot{\xi}) = T_{traslación} - V \tag{2.25}$$

$$\frac{\partial L(\xi,\dot{\xi})}{\partial \xi} = -mgE_3, \quad \frac{\partial L(\xi,\dot{\xi})}{\partial \dot{\xi}} = m\dot{\xi}, \ \frac{d}{dt}\left(\frac{\partial L(\xi,\dot{\xi})}{\partial \dot{\xi}}\right) = m\ddot{\xi} \tag{2.26}$$

$$\frac{d}{dt}\left(\frac{\partial L(\xi,\dot{\xi})}{\partial \dot{\xi}}\right) - \frac{\partial L(\xi,\dot{\xi})}{\partial \xi} = F_\xi \tag{2.27}$$

$$m\ddot{\xi} + mgE_3 = F_\xi \tag{2.28}$$

Reescribiendo la ecuación (2.28) en función del vector de estados ξ y el vector de efectos aerodinámicos $\alpha T = [A_x \quad A_y \quad A_z]^T$considerando como perturbaciones externas para efectos de diseño, teniendo:

$$\ddot{x} = \frac{1}{m}(cos\psi sen\theta cos\phi + sen\psi sen\phi)U_1 + \frac{A_x}{m} \tag{2.29}$$

$$\ddot{y} = \frac{1}{m}(sen\psi sen\theta cos\phi - cos\psi sen\phi)U_1 + \frac{A_y}{m} \tag{2.30}$$

$$\ddot{z} = -g + \frac{1}{m}(cos\theta cos\phi)U_1 + \frac{A_z}{m} \tag{2.31}$$

donde $U_1 = F_1 + F_2 + F_3 + F_4$.

Para las coordenadas de η las ecuaciones de Euler Lagrange son:

$$\frac{d}{dt}\left(\frac{\partial L(\eta,\dot{\eta})}{\partial \dot{\eta}}\right) - \frac{\partial L(\eta,\dot{\eta})}{\partial \eta} = \tau_\eta \tag{2.32}$$

$$\begin{bmatrix} \frac{d}{dt}\left(\frac{\partial L(\eta,\dot{\eta})}{\partial \dot{\phi}}\right) - \frac{\partial L(\eta,\dot{\eta})}{\partial \phi} \\ \frac{d}{dt}\left(\frac{\partial L(\eta,\dot{\eta})}{\partial \dot{\theta}}\right) - \frac{\partial L(\eta,\dot{\eta})}{\partial \theta} \\ \frac{d}{dt}\left(\frac{\partial L(\eta,\dot{\eta})}{\partial \dot{\psi}}\right) - \frac{\partial L(\eta,\dot{\eta})}{\partial \psi} \end{bmatrix} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{2.33}$$

El modelo matemático de la dinámica rotacional se puede presentar en la forma general [28].

$$M(\eta)\ddot{\eta} + C(\eta,\dot{\eta})\dot{\eta} = \tau_\eta \tag{2.34}$$

con $M(\eta) = \mathcal{I}(\eta)$

$$\boldsymbol{M}(\eta) = \begin{bmatrix} I_{xx} & 0 & -I_{xx}sen\theta \\ 0 & I_{yy}cos^2\phi + I_{zz}sen^2\phi & (I_{yy} - I_{zz})cos\phi sen\phi cos\theta \\ -I_{xx}sen\theta & (I_{yy} - I_{zz})cos\phi sen\phi cos\theta & I_{xx}sen^2\theta + I_{yy}sen^2\phi cos^2\theta + I_{zz}cos^2\phi cos^2\theta \end{bmatrix} \quad (2.35)$$

donde I_{xx}, I_{yy}, I_{zz}, son los momentos principales.

La matriz antisimétrica $C(\eta, \dot{\eta})$

$$C(\eta, \dot{\eta}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \quad (2.36)$$

donde

- $c_{11} = 0$
- $c_{12} = \left(I_{yy} - I_{zz}\right)\left(\dot{\theta}cos\phi sen\phi + \dot{\psi}sen^2\phi cos\theta\right) + \left(I_{zz} - I_{yy}\right)\dot{\psi}cos^2\phi cos\theta - I_{xx}\dot{\psi}cos\theta$
- $c_{13} = \left(I_{zz} - I_{yy}\right)\dot{\psi}cos\phi sen\phi cos^2\theta$
- $c_{21} = \left(I_{zz} - I_{yy}\right)\left(\dot{\theta}cos\phi sen\phi + \dot{\psi}sen^2\phi cos\theta\right) + \left(I_{yy} - I_{zz}\right)\dot{\psi}cos^2\phi cos\theta - I_{xx}\dot{\psi}cos\theta$
- $c_{22} = \left(I_{zz} - I_{yy}\right)\dot{\phi}cos\phi sen\phi$
- $c_{23} = -I_{xx}\dot{\psi}sen\theta cos\theta + I_{yy}\dot{\psi}sen^2\phi cos\theta sen\theta + I_{zz}\dot{\psi}cos^2\phi cos\theta sen\theta$
- $c_{31} = \left(I_{yy} - I_{zz}\right)\dot{\psi}cos^2\theta sin\phi cos\phi - I_{xx}\dot{\theta}cos\theta$
- $c_{32} = \left(I_{zz} - I_{yy}\right)\left(\dot{\theta}cos\phi sen\phi sen\theta + \dot{\phi}sen^2\phi cos\theta\right) + \left(I_{yy} - I_{zz}\right)\dot{\phi}cos^2\phi cos\theta + I_{xx}\dot{\psi}sen\theta cos\theta - I_{yy}\dot{\psi}sen^2\phi cos\theta sen\theta - I_{zz}\dot{\psi}cos^2\phi cos\theta sen\theta$
- $c_{33} = \left(I_{yy} - I_{zz}\right)\dot{\phi}cos\phi sen\phi cos^2\theta + I_{xx}\dot{\theta}sen\theta cos\theta - I_{yy}\dot{\theta}sen^2\phi cos\theta sen\theta - I_{zz}\dot{\theta}cos^2\phi cos\theta sen\theta$

Los pares aplicados al sistema τ_η están dados por:

$$\tau_\eta = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lU_2 \\ lU_3 \\ T_q lU_4 \end{bmatrix} \quad (2.37)$$

$$\tau_\eta = \begin{bmatrix} l(F_1 - F_3) \\ l(F_2 - F_4) \\ T_q l(F_1 - F_2 + F_3 - F_4) \end{bmatrix} \quad (2.38)$$

donde T_q es una constante para convertir la fuerza de empuje en fuerza de guiñada.

Las ecuaciones (2.29) y (2.31) modelan la dinámica traslacional del cuadri-rotor, en este trabajo las perturbaciones externas no se consideran. Para el sistema rotacional (2.34), se plantea que el punto de operación se encuentra alrededor del punto de equilibrio para $\phi = 0$ y $\theta = 0$, es decir $cos\,\phi \approx cos\,\theta \approx cos\,\psi \approx 1$, y $sen\,\phi \approx sen\,\theta \approx 0$ $sen\,\psi \approx \psi$, donde se tiene

la estabilización del sistema. De esta manera el modelo no lineal se ve reducido a las siguientes ecuaciones:

$$\ddot{x} = \frac{1}{m}(cos\psi sen\theta cos\phi + sen\psi sen\phi)U_1 \tag{2.39}$$

$$\ddot{y} = \frac{1}{m}(sen\psi sen\theta cos\phi - cos\psi sen\phi)U_1 \tag{2.40}$$

$$\ddot{z} = -g + \frac{1}{m}(cos\theta cos\phi)U_1 \tag{2.41}$$

$$\ddot{\phi} = \frac{(I_{yy}-I_{zz})}{I_{xx}}\dot{\psi}\dot{\theta} - \frac{J_R\Omega}{I_{xx}}\dot{\theta} + \frac{l}{I_{xx}}U_2 \tag{2.42}$$

$$\ddot{\theta} = \frac{(I_{zz}-I_{xx})}{I_{yy}}\dot{\psi}\dot{\phi} + \frac{J_R\Omega}{I_{yy}}\dot{\phi} + \frac{l}{I_{yy}}U_3 \tag{2.43}$$

$$\ddot{\psi} = \frac{(I_{xx}-I_{yy})}{I_{zz}}\dot{\phi}\dot{\theta} + \frac{1}{I_{zz}}U_4 \tag{2.44}$$

donde (2.45) – (2.48) son las entradas del sistema, que representan las fuerzas aplicadas por los cuatro rotores Ω_1, Ω_2, Ω_3, y Ω_4.

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{2.45}$$

$$U_2 = bl(\Omega_1^2 - \Omega_3^2) \tag{2.46}$$

$$U_3 = bl(\Omega_4^2 - \Omega_2^2) \tag{2.47}$$

$$U_4 = k_T(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{2.48}$$

La ecuación (2.58) representa las perturbaciones producidas por los rotores.

$$\Omega = \Omega_1 + \Omega_2 + \Omega_3 + \Omega_4 \tag{2.58}$$

Este modelo se ocupará para probar las técnicas de control PD y Backstepping en el capítulo siguiente.

2.3 Modelo dinámico no lineal simplificado

Otro modelo dinámico del cuadri-rotor es llamado modelo simplificado es el presentado en [28], donde la acción del yaw no influye en el movimiento traslacional. Este modelo dinámico servirá para obtener el modelo por planitud. Este modelo no tiene la misma

orientación del modelo no lineal (2.39) - (2.44) de la sección anterior, sin embargo, es fácil identificar la relación entre ellos, por lo que se dejará en sus términos originales.

$$m\ddot{x} = -U_1 sen\theta \tag{2.59}$$

$$m\ddot{y} = U_1 cos\theta sen\phi \tag{2.60}$$

$$m\ddot{z} = U_1 cos\theta cos\phi - mg \tag{2.61}$$

$$\ddot{\psi} = U_2 \tag{2.62}$$

$$\ddot{\theta} = U_3 \tag{2.63}$$

$$\ddot{\phi} = U_4 \tag{2.64}$$

2.4 Modelo dinámico por planitud

El objetivo en el enfoque de planitud es expresar explícitamente todos los estados y todas las entradas de control como funciones de las salidas planas y un número finito de sus derivadas temporales. De (2.59) – (2.64), y definiendo las salidas planas como $z = [\mathrm{x\ y\ z}]^T$ [29], porque tenemos cuatro entradas de control. Podemos escribir todo el sistema de estados como función de las salidas planas z y sus derivadas de tiempo de la siguiente manera [11]:

$$x = z_1 \tag{2.63}$$

$$\dot{x} = \dot{z}_1 \tag{2.64}$$

$$y = z_2 \tag{2.65}$$

$$\dot{y} = \dot{z}_2 \tag{2.66}$$

$$z = z_3 \tag{2.67}$$

$$\dot{z} = \dot{z}_3 \tag{2.68}$$

$$\psi = z_4 \tag{2.69}$$

$$\dot{\psi} = \dot{z}_4 \tag{2.70}$$

$$\theta = \text{asen}\left(\frac{m\,\ddot{z}_1}{-u_1}\right) \tag{2.71}$$

$$\dot{\theta} = -m\left(\frac{\dddot{z}\,u_1 - u_1^2\ddot{z}_1}{u_1^2\sqrt{\alpha}}\right) \tag{2.72}$$

$$\phi = \text{atan}\left(\frac{\ddot{z}_2}{\ddot{z}_3+g}\right) \tag{2.73}$$

$$\dot{\phi} = \frac{\dddot{z}_2\,(\ddot{z}_3+g) - \dddot{z}_3\ddot{z}_2}{(\ddot{z}_3+g)^2+\ddot{z}_2{}^2} \tag{2.74}$$

De manera similar, las entradas de control se expresan en función de las salidas planas y sus derivadas [11].

$$U_1 = m\sqrt{\ddot{z}_1{}^2 + \ddot{z}_2{}^2 + (\ddot{z}_3 + g)^2} \tag{2.75}$$

$$U_2 = \ddot{z}_4 \tag{2.76}$$

$$U_3 = -m\left(\frac{\left(z_1^{(4)}u_1 - \ddot{u}_1\,\ddot{z}_1\right)u_1^2\sqrt{A} - (\dddot{z}_1u_1 - \ddot{u}_1\ddot{z}_1)C + 2u_1\dot{u}_1\sqrt{A}}{u_1^4 A}\right) \tag{2.77}$$

$$U_4 = \frac{\left(z_2^{(4)}(\ddot{z}_3+g) + \ddot{z}_2\dddot{z}_3 - z_3^{(4)}\ddot{z}_2 - \dddot{z}_3\dddot{z}_2\right)(\dddot{z}_2(\ddot{z}_3+g) - \dddot{z}_3\ddot{z}_2) - B}{(\ddot{z}_3+g)^4 + \ddot{z}_2{}^4} \tag{2.78}$$

donde

$$A = 1 - \left(\frac{m\ddot{z}_1}{u_1}\right)^2 \tag{2.79}$$

$$B = (2(\ddot{z}_3 + g)\dddot{z}_3 + 2\ddot{z}_2\dddot{z}_2)(\dddot{z}_2(\ddot{z}_3 + g) - \ddot{z}_2\dddot{z}_3) \tag{2.80}$$

$$C = \frac{-2m\ddot{z}_1(m\dddot{z}_1u_1 - \dot{u}_1m\ddot{z}_1)}{u_1\sqrt{\alpha}} \tag{2.81}$$

2.5 Modelo dinámico lineal

Con el método de linealización de la sección 1.6, el modelo lineal del cuadri-rotor servirá para la aplicación de algoritmos de control lineal.

El modelo lineal del vehículo aéreo es:

$$\frac{\partial x}{\partial t} = Ax + Bu \tag{2.82}$$

$$\frac{\partial y}{\partial t} = Cx + Du \tag{2.83}$$

Con el punto de equilibrio P_{eq} de (2.84).

$$P_{eq} = (x, y, z, 0,0,0,0,0,0,0,0,0, mg, 0,0,0)^T \tag{2.84}$$

y donde:

$$A = \begin{bmatrix} 0_{3x3} & I_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & \alpha_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & I_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \end{bmatrix} \tag{2.85}$$

con

$$\alpha = \begin{bmatrix} 0 & -u_1/m & 0 \\ 0 & 0 & u_1/m \\ 0 & 0 & 0 \end{bmatrix} \tag{2.86}$$

$$B = \begin{bmatrix} 0_{5x1} & 0_{5x3} \\ 1/m & 0_{1x3} \\ 0_{3x1} & 0_{3x3} \\ 0_{3x1} & I_{3x3} \end{bmatrix} \tag{2.87}$$

$$C = [I_{12x12}] \tag{2.88}$$

$$D = [0_{4x4}] \tag{2.89}$$

Este capítulo presentó la descripción del funcionamiento básico del cuadri-rotor y la representación de algunos modelos dinámicos que describen su comportamiento. Estos modelos dinámicos presentan características adecuadas que permiten utilizarlos con las estrategias de control para la estabilización de vuelo y el seguimiento de trayectorias planteadas en el siguiente capítulo.

Capítulo 3

Estructuras de control

Debido a la naturaleza de la dinámica del cuadri-rotor, se han probado varios algoritmos de control y como era de esperarse cada uno de ellos presenta sus ventajas y desventajas.

Tabla 3.1. Comparación de los algoritmos de control de cuadrirrotor [30].

ALGORITMO DE CONTROL	**CARACTERÍSTICA**												
	Robustez	Adaptivo	Óptimo	Inteligente	Capacidad de seguimiento	Rápida respuesta	Precisión	Sencillez	Rechazo de perturbaciones	Gestión de parámetros no modelados	Sintonización manual	Susceptible al ruido	Presenta cascabeleo
1.PID/PD	1	0	0	0	1	1	1	2	0	0	2	2	0
2.PID INTELIGENTE	1	0	0	2	1	1	1	1	0	0	0	1	0
3.LQR	0	2	1	0	1	1	0	1	1	0	1	1	0
4.LQG	0	2	2	0	1	1	0	0	2	0	1	0	0
5.L_1	0	2	2	0	1	2	2	0	1	0	0	0	0
6.H_1	2	1	2	0	2	0	1	0	1	1	0	0	0
7.SMC	1	2	1	0	2	2	2	1	2	1	0	0	2
8.FBL	1	1	0	0	2	2	2	1	1	1	0	1	0
9.BACKSTEPPING	0	2	0	0	2	0	1	0	2	1	0	0	0
10.LÓGICA DIFUSA	1	1	1	2	1	1	1	1	1	0	1	0	0
11.REDES NEURONALES	1	2	2	2	1	1	1	0	1	1	0	0	0
12.GENÉTICO	1	2	2	2	1	1	1	0	1	2	0	0	0

0-BAJO O NINGUNO; 1-MEDIO; 2-ALTO. ADEMÁS DEL 1 AL 5 (LINEAL); 6 A 12 (NO LINEAL).

La tabla 3.1 resume la comparación de los diferentes algoritmos que generalmente se aplican a cuadri-rotores [30]. El rendimiento de un algoritmo particular depende de muchos factores, algunos de los cuales ni siquiera pueden ser modelados. Por lo tanto, esta tabla funciona como una guía básica para la selección del algoritmo. Para este trabajo se ha seleccionado el algoritmo de Control PD, debido a la sencillez de aplicación, el control Backstepping por su capacidad de seguimiento, que es adaptivo y que no presenta cascabeleos y al LQR por ser uno de los controles lineales con sencillez y con sintonización manual.

El objetivo de la planificación de trayectoria es proporcionar al sistema las trayectorias deseadas y viables para las cuales existe una entrada de control correspondiente que mueve el sistema en cuestión, desde un estado inicial a una condición de objetivo, respetando las restricciones y evitando la colisión [11]. Para lo cual, se utilizará el control basado en Planitud con las estrategias PD, Backstepping y LQR.

3.1 Control PD

El enfoque de control Proporcional Derivativo (PD) es uno de los enfoques de control más populares porque es muy simple de diseñar. Los resultados de la implementación de este controlador se encuentran publicados en [31], con una trayectoria helicoidal.

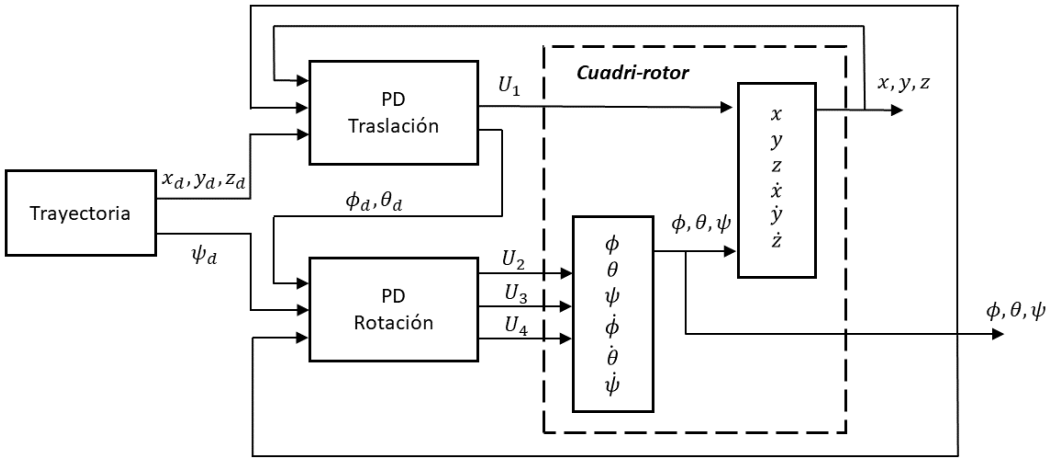


Fig. 3.1. Esquema de control PD del cuadri-rotor.

La implementación del control PD siguiendo el esquema de la figura 3.1, puede ser definida por:

$$U = K_p e(t) + K_d \frac{d}{dt} e(t) \tag{3.1}$$

donde K_p es la ganancia proporcional y K_d es la ganancia derivativa, $e(t)$ es la diferencia entre el valor deseado y el valor actual.

Para el control PD de rotación U es resultado del cambio del ángulo, quedando:

$$U_2 = K_{p,\phi}(\phi_d - \phi) + K_{d,\phi}(\dot{\phi}_d - \dot{\phi}) \tag{3.2}$$

$$U_3 = K_{p,\phi}(\theta_d - \theta) + K_{d,\theta}(\dot{\theta}_d - \dot{\theta}) \tag{3.3}$$

$$U_4 = K_{p,\psi}(\psi_d - \psi) + K_{d,\psi}(\dot{\psi}_d - \dot{\psi}) \tag{3.4}$$

Las ecuaciones (3.2) – (3.4) representan los torques para cada eje debido al empuje de los rotores τ_ϕ, τ_θ y τ_ψ respectivamente, $K_{p,\phi}$, $K_{p,\phi}$ y $K_{p,\psi}$ son las ganancias proporcionales, $K_{d,\phi}$, $K_{d,\theta}$ y $K_{d,\psi}$ ganancias derivativas, ϕ_d, θ_d y ψ_d son pitch, roll y yaw deseados. ϕ, θ y ψ corresponden a los valores de orientación.

Para el control Pd de traslación U es resultado del cambio posición, considerando la posición deseada x_d, y_d y z_d son valores conocidos, mientras que x, y, z son valores que son medidos.

Las ecuaciones (2.39) – (2.41) se pueden resolver para determinar las variables de control.

$$U_1 = \frac{1}{cos\theta cos\phi}(r_3 + mg) \tag{3.5}$$

$$d_\phi = arcsen\left(\frac{m}{U_1}(r_1 sen\psi - r_2 cos\psi)\right) \tag{3.6}$$

$$d_\theta = arcsen\left(\frac{m}{U_1}(r_1 cos\psi + r_2 sen\psi)\right) \tag{3.7}$$

$$r_1 = K_{px}(x_d - x) + K_{d,x}(\dot{x}_d - \dot{x}) \tag{3.8}$$

$$r_2 = K_{py}(y_d - y) + K_{dy}(\dot{y}_d - \dot{y}) \tag{3.9}$$

$$r_3 = K_{pz}(z_d - z) + K_{dz}(\dot{z}_d - \dot{z}) \tag{3.10}$$

donde la ecuación (3.5) es la fuerza de control del desplazamiento vertical z, (3.6) y (3.7) son los ángulos deseados para colocar el cuadri-rotor en x_d, y_d del plano cartesiano. r_1, r_2 y r_3 son las señales de control.

3.2 Control Backstepping

El control Backstepping es un tipo de control adecuado para sistemas no lineales que presentan una estructura en cascada. Descompone el problema original en una secuencia de subsistemas, auxiliándose en la recursividad del mismo [32]. Resultados de la implementación de este controlador están publicados en [33], donde se utiliza la técnica de Backstepping para controlar el cuadri-rotor con objetivos de estabilización de vuelo y seguimiento de una trayectoria helicoidal.

Para diseñar el controlador se reescribe el sistema dinámico no lineal de (2.39) – (2.44), en la forma de espacio de estados $\dot{X} = f(X, U)$ introduciendo $X = (x_1 \dots x_{12})$ como el vector de estados del sistema, siendo:

$$x_1 = \phi \tag{3.11}$$

$$x_2 = \dot{x}_1 = \dot{\phi} \tag{3.12}$$

$$x_3 = \theta \tag{3.13}$$

$$x_4 = \dot{x}_3 = \dot{\theta} \tag{3.14}$$

$$x_5 = \psi \tag{3.15}$$

$$x_6 = \dot{x}_5 = \dot{\psi} \tag{3.16}$$

$$x_7 = z \tag{3.17}$$

$$x_8 = \dot{x}_7 = \dot{z} \tag{3.18}$$

$$x_9 = x \tag{3.19}$$

$$x_{10} = \dot{x}_9 = \dot{x} \tag{3.20}$$

$$x_{11} = y \tag{3.21}$$

$$x_{12} = \dot{x}_{11} = \dot{y} \tag{3.22}$$

Con el nuevo vector de estados de (3.11) - (3.22) se puede escribir el sistema de la siguiente forma:

$$\dot{\mathrm{X}} = f(\mathrm{X},\mathrm{U}) = \begin{bmatrix} x_2 \\ a_1 x_4 x_6 + a_2 x_4 \Omega + b_1 U_2 \\ x_4 \\ a_3 x_2 x_6 + a_4 x_2 \Omega + b_2 U_3 \\ x_6 \\ a_5 x_2 x_4 + b_3 U_4 \\ x_8 \\ -g + (cos x_1 cos x_3)\frac{U_1}{m} \\ x_{10} \\ u_x \frac{U_1}{m} \\ x_{12} \\ u_y \frac{U_1}{m} \end{bmatrix} \tag{3.23}$$

con:

$$a_1 = \frac{\left(I_{yy} - I_{zz}\right)}{I_{xx}} \tag{3.24}$$

$$a_2 = -\frac{J_R}{I_{xx}} \tag{3.25}$$

$$a_3 = \frac{(I_{zz} - I_{xx})}{I_{yy}} \tag{3.26}$$

$$a_4 = \frac{J_R}{I_{yy}} \tag{3.27}$$

$$a_5 = \frac{\left(I_{xx} - I_{yy}\right)}{I_{zz}} \tag{3.28}$$

$$b_1 = \frac{l}{I_{xx}} \tag{3.29}$$

$$b_2 = \frac{l}{I_{yy}} \tag{3.30}$$

$$b_3 = \frac{1}{I_{xx}} \tag{3.31}$$

$$u_x = cos(x_1)\, sen(x_3)\, cos(x_5) + sen(x_1)\, sen(x_5) \tag{3.32}$$

$$u_y = cos(x_1)\, sen(x_3)\, sen(x_5) - sen(x_1)\, cos(x_5) \tag{3.33}$$

Esto es útil para ver que el sistema de rotación no depende de las componentes de traslación, como ya se había comentado anteriormente. Por otro lado, las traslaciones dependen de los ángulos. Se puede imaginar el sistema completo descrito por la ecuación (3.11) – (3.22), constituido por dos subsistemas, el de rotaciones angulares y el de traslaciones lineales, conectados conforme a la figura 3.2.

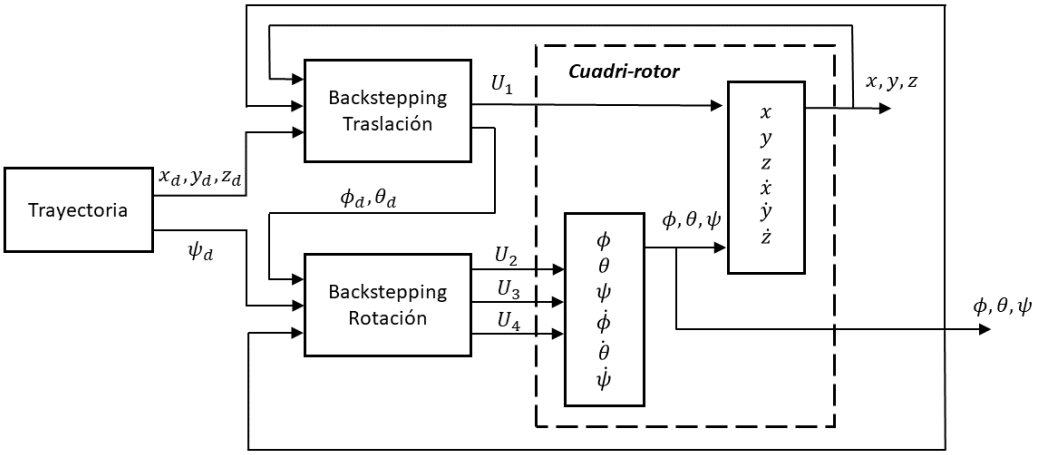


Fig. 3.2. Esquema de control Backstepping del cuadri-rotor.

Utilizando la técnica de Backstepping se puede diseñar la ley de control forzando al sistema a seguir la trayectoria de referencia. Para ello, como primer paso se define el error de seguimiento como sigue [27]:

$$n_1 = x_{1d} - x_1 \tag{3.34}$$

Según el teorema de Lyapunov, y considerando que la función Lyapunov $V(z_1)$ presentada a continuación, es definida positiva y su derivada con respecto al tiempo es semidefinida negativa:

$$V(n) = \frac{1}{2} {n_1}^2 \tag{3.35}$$

$$\dot{V}(n_1) = n_1(\dot{x}_{1d} - x_2) \tag{3.36}$$

la estabilización de n_1 puede ser obtenida introduciendo la siguiente entrada de control virtual x_2:

$$x_2 = \dot{x}_{1d} + \alpha_1 n_1 \tag{3.37}$$

con $\alpha_1 > 0$.

La derivada de la función de Lyapunov se escribe como:

$$\dot{V}(n_1) = -\alpha_1 {n_1}^2 \tag{3.38}$$

permitiendo proseguir con el cambio de variables realizando:

$$n_2 = x_2 - \dot{x}_{1d} - \alpha_1 n_1 \tag{3.39}$$

Para anular este nuevo error, se realizará un segundo paso considerando la siguiente función de Lyapunov aumentada:

$$V(n_1, n_2) = \frac{1}{2}{n_1}^2 + \frac{1}{2}{n_2}^2 \tag{3.40}$$

donde su derivada temporal viene dada por:

$$\dot{V}(n_1, n_2) = n_2(a_1 x_4 x_6) + a_2 x_4 \Omega + b_1 U_2 - n_2(\ddot{x}_{1d} - \alpha_1(n_2 + \alpha_1 n_1)) - n_1 n_2 - \alpha_1 {n_1}^2 \tag{3.41}$$

Por lo tanto, si se considera la referencia en aceleración nula ($\ddot{x}_{1d} = 0$), se obtiene la siguiente señal de control U_2, satisfaciendo $\dot{V}(n_1, n_2) < 0$:

$$U_2 = \frac{1}{b_1}(n_1 - a_1 x_4 x_6 - a_2 x_4 \Omega - \alpha_1(n_2 + \alpha_1 n_1) - \alpha_2 n_2) \tag{3.42}$$

con

$$n_1 = x_{1d} - x_1 \tag{3.43}$$

$$n_2 = x_2 - \dot{x}_{1d} - \alpha_1 n_1 \tag{3.44}$$

que estabiliza el sistema aumentado, ya que con esta señal de control se obtiene que:

$$\dot{V}(n_1, n_2) = -\alpha_1 {n_1}^2 - \alpha_2 {n_2}^2 < 0 \tag{3.45}$$

Nótese cómo el término $\alpha_2 n_2$ con $\alpha_2 > 0$ se añade para estabilizar n_1.

Se realizan los mismos pasos para obtener U_3 y U_4:

$$U_3 = \frac{1}{b_2}(n_3 - a_3 x_2 x_6 - a_4 x_2 \Omega - \alpha_3(n_4 + \alpha_3 n_3) - \alpha_4 n_4) \tag{3.46}$$

$$U_4 = \frac{1}{b_3}(n_5 - a_5 x_2 x_4 - \alpha_5(n_6 + \alpha_5 n_5) - \alpha_6 n_6) \tag{3.47}$$

con

$$n_3 = x_{3d-}x_3 \tag{3.48}$$

$$n_4 = x_4 - \dot{x}_{3d} - \alpha_3 n_3 \tag{3.49}$$

$$n_5 = x_{5d-}x_5 \tag{3.50}$$

$$n_6 = x_6 - \dot{x}_{5d} - \alpha_5 n_5 \tag{3.51}$$

La entrada de control de la altura U_1 se logra usando la misma técnica presentada en la anteriormente:

$$U_1 = \frac{m}{\cos(x_1)\cos(x_3)}(n_7 + g - \alpha_7(n_8 + \alpha_7 n_7) - \alpha_8 n_8) \tag{3.52}$$

con

$$n_7 = x_{7d-}x_7 \tag{3.53}$$

$$n_8 = x_8 - \dot{x}_{7d} - \alpha_7 n_7 \tag{3.54}$$

El modelo (2.39) – (2.41) refleja que el movimiento a través de los ejes X y Y depende de la entrada de control U_1. De hecho, U_1 es el vector de empuje total diseñado para obtener el movimiento lineal deseado. Si se considera u_x y u_y las orientaciones de U_1 responsables del movimiento a través de los ejes X y Y, respectivamente, se puede obtener de (2.42) – (2.43) los ángulos de roll, ϕ, y de pitch, θ, necesarios para calcular las señales de control u_x y u_y satisfaciendo $\dot{V}(z_1, z_2) < 0$.

$$u_x = \frac{m}{U_1}(n_9 - \alpha_9(n_{10} + \alpha_9 n_9) - \alpha_{10} n_{10}) \tag{3.55}$$

$$u_y = \frac{m}{U_1}(n_{11} - \alpha_{11}(n_{12} + \alpha_{11} n_{11}) - \alpha_{12} n_{12}) \tag{3.56}$$

3.3 Control LQR

Si se tiene un sistema dinámico definido por el sistema de ecuaciones

$$\dot{x} = f(x, u) \tag{3.57}$$

donde las condiciones iniciales son $[x(0), x(\infty)]$ y la función de costo está dada por [34]:

$$J = \int_0^\infty (L(x, u, t) + S(x(T), T))dt \tag{3.58}$$

El objetivo primordial del control LQR consiste en buscar una señal $u(t)$ que minimice J. Al realizar esto estamos consiguiendo minimizar la función lineal cuadrática y ajustarla a la referencia. A este tipo de control se le conoce como regulador lineal cuadrático o problema LQR, que consiste en minimizar una función cuadrática y mantenerla lo más próximo al origen ($x = 0$).

Sea el sistema dado en espacio de estados:

$$\dot{x} = Ax + Bu \tag{3.59}$$

$$y = Cx + Du \tag{3.60}$$

Así que la función de costo se define como:

$$J = \int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t))\, dt \tag{3.61}$$

Las matrices Q y R, son matrices definidas positivas y simétricas que son definidas por el diseñador. Las matrices son diagonales y se escogen en función de las necesidades de control.

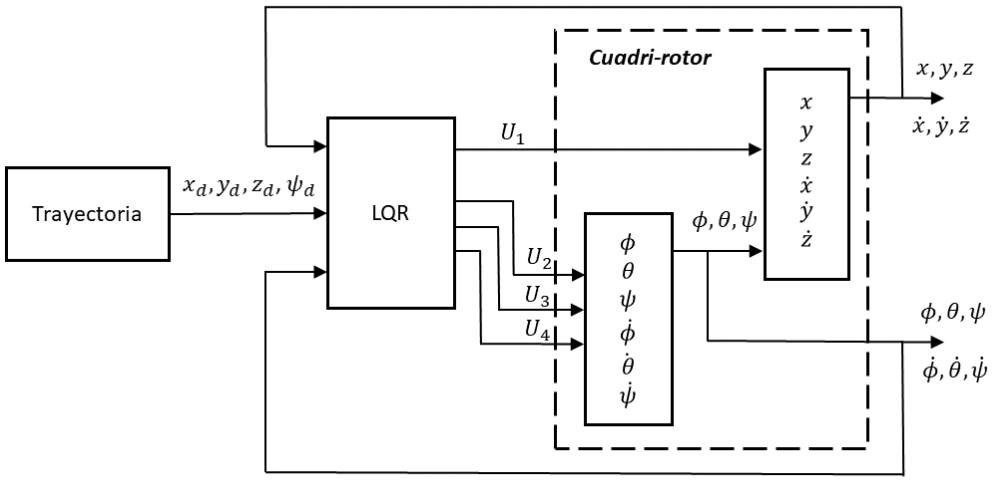


Fig. 3.3. Esquema de control LQR del cuadri-rotor.

Para conseguir esto de acuerdo a la figura 3.3, la ley de control por retroalimentación utilizada es:

$$u = -Kx \tag{3.62}$$

donde K es la matriz de control LQR y se define como:

$$K = R^{-1} B^T P \tag{3.63}$$

donde P es la matriz solución de la ecuación algebraica de Riccati [34]:

$$P^T A + A^T P + Q - PBR^{-1}B^T P = 0 \tag{3.64}$$

Esta es la teoría detrás del control LQR, pero en este trabajo con el fin de facilitar la implementación, se usa la instrucción de MATLAB '$[K] = lqr(A, B, Q, R)$' para calcular el control LQR con el modelo lineal calculado en el capítulo anterior (2.82) – (2.88) y así obtener la matriz de ganancias K de la ley de control.

Para comprobar si el controlador LQR está funcionando como estrategia de control para el sistema se prueba con el sistema lineal simplificado (2.59) – (2.64). En este caso al tener definidas las matrices del sistema, solo queda elegir los valores para las matrices Q y R. Por facilidad, se definen ambas matrices como la identidad con el rango de cada una.

Las matrices obtenidas son:

$$Q = [I_{12\times 12}], \qquad R = [I_{4\times 4}] \tag{3.65}$$

De esta manera, al conocer los valores se obtiene la matriz K utilizada en la ley de control.

$$K = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1.7 & 0 & 0 \\ -1 & 0 & 0 & -1.9 & 0 & 0 & 0 & 13.5 & 0 & 0 & 5.2 & 0 \\ 0 & 1 & 0 & 0 & 1.9 & 0 & 0 & 0 & 13.5 & 0 & 0 & 5.2 \end{bmatrix} \tag{3.66}$$

Con esta matriz de retroalimentación, se simula el sistema no lineal simplificado (2.59) – (2.64) dándole como entrada una posición inicial y obligando al sistema a llegar a un valor de referencia, comprobando el funcionamiento del controlador.

3.4 Control del cuadri-rotor basado en planitud

La planificación de movimiento por planitud, no necesita integrar las ecuaciones del sistema y para una trayectoria de salida plana las entradas de control se pueden calcular directamente. El vector u_r resultante siempre respeta la dinámica del sistema.

Como se presenta en el capítulo 1, la definición de planitud implica que cada variable del sistema puede expresarse: en términos de salidas planas y un número finito de sus derivadas de tiempo. Por lo tanto, para calcular una trayectoria cuyas condiciones iniciales y finales se especifican, se construye una trayectoria de salida plana para obtener las entradas de control de bucle abierto que satisfagan el estado deseado y las trayectorias de entrada.

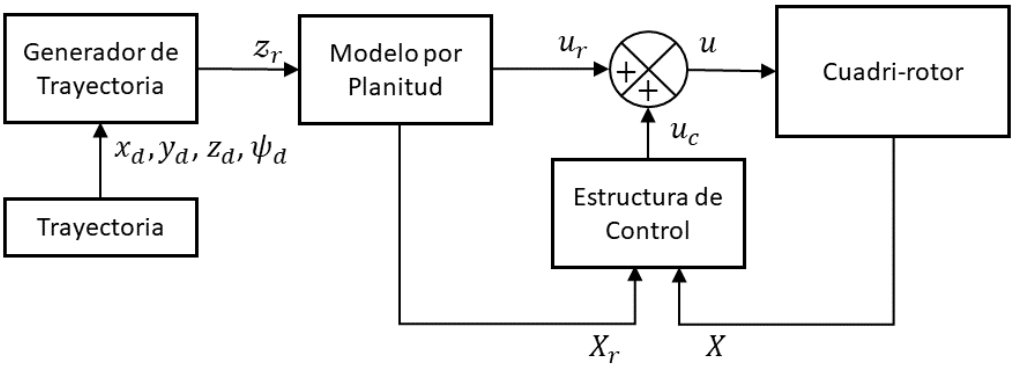


Fig.3.4. Esquema de control del cuadri-rotor basado en Planitud.

La figura 3.4 muestra el diagrama de bloques del control del cuadri-rotor basado en planitud. El bloque de Generacion de Trayectoria genera un vector z_r que consiste en una posición deseada y sus derivadas. El bloque de Modelo por Planitud toma estos parámetros de trayectoria y calcula una entrada de referencia u_r de (2.75) – (2.78) y también calcula los estados de referencia X_r de la (2.63) – (2.74). Se utiliza el controlador PD, Backstepping y LQR para el control de retroalimentación.

Por lo que para el control del cuadri-rotor basado en Planitud la entrada de control es:

$$u = u_r + u_c \tag{3.67}$$

3.4.1 Control PD basado en Planitud

Para el control PD basado en Planitud las entradas de control se reformulan:

$$u = u_{r1} + (K_p e(t) + K_d \frac{d}{dt} e(t)) \tag{3.68}$$

donde u_r es el torque de referencia del modelo por planitud, K_p es la ganancia proporcional y K_d es la ganancia derivativa, $e(t)$ es la diferencia entre el valor de referencia del modelo por planitud y el valor actual.

$$u_{c1} = \frac{1}{cos\theta cos\phi}\left(K_{p,z}(z_r - \phi) + K_{d,\phi}(z_r - \dot{\phi})\right) \tag{3.69}$$

$$u_{c2} = K_{p,\phi}(\phi_r - \phi) + K_{d,\phi}(\phi_r - \dot{\phi}) \tag{3.70}$$

$$u_{c3} = K_{p,\phi}(\theta_r - \theta) + K_{d,\theta}(\dot{\theta}_r - \dot{\theta}) \tag{3.71}$$

$$u_{c4} = K_{p,\psi}(\psi_r - \psi) + K_{d,\psi}(\dot{\psi}_r - \dot{\psi}) \tag{3.72}$$

3.4.2 Control Backstepping basado en Planitud

Para el control Backstepping basado en Planitud con el nuevo error de seguimiento:

$$z_1 = x_{r1} - x_1 \tag{3.73}$$

Las entradas de control se reformulan:

$$u_{c1} = \frac{m}{\cos(x_1)\cos(x_3)}(z_7 - \alpha_7(z_8 + \alpha_7 z_7) - \alpha_8 z_8) \tag{3.74}$$

$$u_{c2} = \frac{1}{b_1}(z_1 - a_1 x_4 x_6 - a_2 x_4 \Omega - \alpha_1(z_2 + \alpha_1 z_1) - \alpha_2 z_2) \tag{3.75}$$

$$u_{c3} = \frac{1}{b_2}(z_3 - a_3 x_2 x_6 - a_4 x_2 \Omega - \alpha_3(z_4 + \alpha_3 z_3) - \alpha_4 z_4) \tag{3.76}$$

$$u_{c42} = \frac{1}{b_3}(z_5 - a_5 x_2 x_4 - \alpha_5(z_6 + \alpha_5 z_5) - \alpha_6 z_6) \tag{3.77}$$

3.4.3 Control LQR basado en Planitud

Ahora, suponiendo una trayectoria de referencia dependiente del tiempo x_{ref}, el control LQR se puede aplicar como seguidor de trayectoria para minimizar pequeños errores entre x el estado medido completo y el estado de referencia x_{ref}, de modo que el control aplicado se convierta:

$$u = u_{ref} - K(x - x_{ref}) \tag{3.78}$$

En este capítulo se mostró las expresiones de los controladores PD, Backstepping y LQR, además de la técnica basada en planitud diferencial del modelo del cuadrirrotor para la generación de trayectorias y entradas de control de referencia, los cuales serán implementados en el software Webots, tanto como en MATLAB en el siguiente capítulo.

Capítulo 4

Simulación del Cuadri-rotor

En esta sección se presentan las simulaciones de un cuadri-rotor siguiendo dos trayectorias planteadas, en cada simulación se implementa un algoritmo de control (PD, LQR y Backstepping) en lazo cerrado, de la misma manera se implementa planitud en los mismos tres controladores. En el capítulo 3, se presentó el desarrollo de cada controlador para su implementación.

Las simulaciones en MATLAB son ampliamente utilizadas en el área de ingeniería y control, sin embargo, los resultados no permiten visualizar de una manera más clara, el comportamiento real del cuadri-rotor. Para esto existen simuladores de realidad virtual, que muestran un escenario donde interactúa el robot de acuerdo con el algoritmo de control programado. Con el fin de verificar el funcionamiento de los algoritmos de control se realiza la simulación del seguimiento de trayectoria de un cuadri-rotor tanto en MATLAB como en Webots.

Para este trabajo, el objetivo central es realizar la estabilización en vuelo y el seguimiento de trayectorias en el software Webots, para esto, fue necesario construir primeramente el modelo del cuadri-rotor dentro del mundo virtual, ya que dicho sistema no existe como tal en Webots, para después aplicar las técnicas de control. De esta manera en este capítulo se presentan los detalles y requerimientos para las simulaciones del cuadri-rotor, para MATLAB y Webots.

4.1 Simulación en MATLAB

MATLAB es acrónimo que proviene de Matrix Laboratory (laboratorio matricial), de esta manera, MATLAB es un lenguaje de programación matemático de alto nivel con entornos gráficos amigables, visualización de datos, funciones y computación numérica para desarrollar algoritmos matemáticos con aplicaciones de ciencias exactas. Permite realizar simulaciones de sistemas mecatrónicos, aunque, la simulación depende de un buen modelo matemático que reproduzca fielmente todos los fenómenos físicos del sistema. Las versiones recientes son multiplataforma, disponible para sistemas operativos como Unix, Windows y Apple Mac OS X. MATLAB dispone de dos herramientas adicionales: Simulink y GUIDE [35].

La estructura de las simulaciones es: un Programa principal, en un archivo .m donde se inicializan las variables y se llaman tanto funciones de MATLAB como los Bloques de Simulink. La versión utilizada en este trabajo es MATLAB R2018b. Para cada controlador se realiza una simulación en MATLAB.

4.1.1 Simulación control PD

Para esta simulación se implementa la estructura de control PD siguiendo el diagrama de la figura 4.1. El bloque Trayectoria se encuentra formado por bloques que describen la trayectoria a seguir, de este modo se generan las posiciones deseadas. Los bloques del control: PD traslacional y PD rotacional, contienen la ley de control descritas en las ecuaciones presentadas en el capítulo anterior. El bloque del cuadri-rotor contiene el modelo dinámico no lineal (2.39) – (2.44) del capítulo 2.

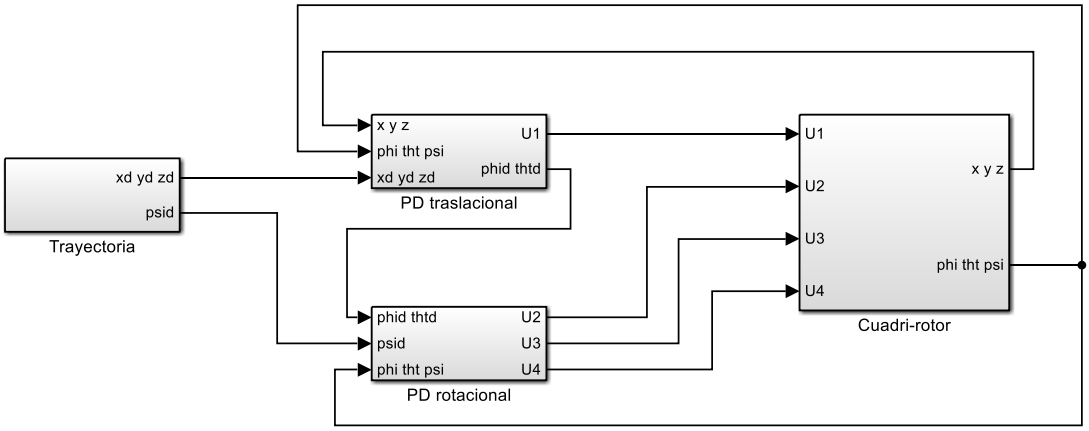


Fig. 4.1. Diagrama a bloques en Simulink del control PD.

Las ganancias usadas en la simulación fueron seleccionadas de manera experimental, buscando tiempo de llegada rápido sin sobretiro y que se encuentran en la Tabla 4.1:

Tabla 4.1. Ganancias de simulación PD

Parámetro	Valor	Parámetro	Valor
$K_{p,\phi}$	1	$K_{d,\phi}$	0.3
$K_{p,\theta}$	1	$K_{d,\theta}$	0.3
$K_{p,\psi}$	0.8	$K_{d,\psi}$	0.2
K_{px}	3	$K_{d,x}$	2
K_{py}	3	K_{dy}	2
K_{pz}	4	K_{dz}	3

Los códigos y demás bloques utilizados se encuentran el apéndice C.1.

4.1.2 Simulación control Backstepping

En esta simulación se implementa la estructura de control Backstepping siguiendo el diagrama de la figura 4.2. El bloque Generador de Trayectoria entrega las posiciones deseadas. Los bloques del control Backstepping rotacional y traslacional contienen las ecuaciones presentadas en el capítulo anterior. El modelo dinámico implementado del cuadri-rotor es el no lineal (2.39) – (2.44).

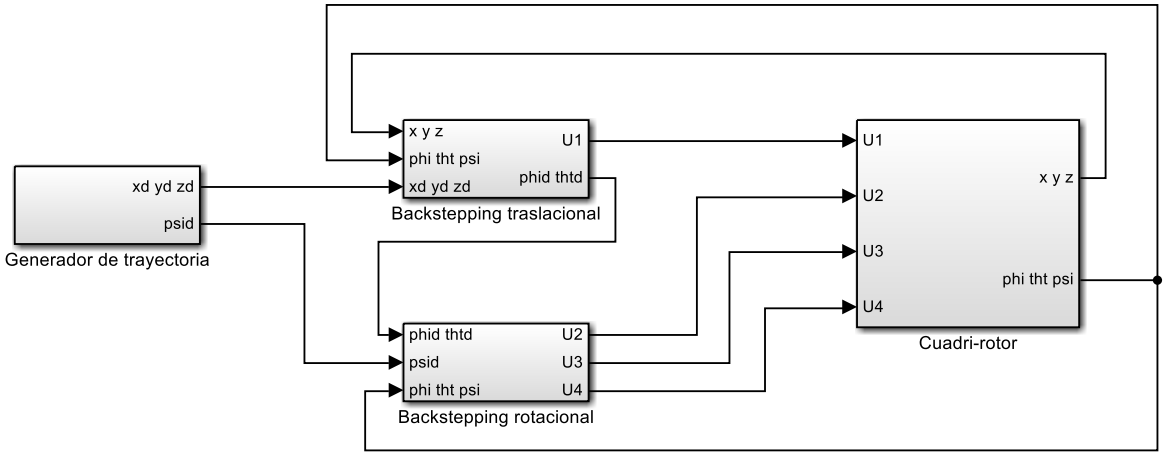


Fig. 4.2. Diagrama a bloques en Simulink del control Backstepping.

La Tabla 4.2 presenta las ganancias usadas en la simulación obtenidas experimentalmente, buscando tiempo de llegada rápido sin sobretiro.

Tabla 4.2. Ganancias de simulación Backstepping.

Parámetro	Valor	Parámetro	Valor	Parámetro	Valor
a_1	0	α_1	500	α_7	4
a_2	-0.1	α_2	1	α_8	1
a_3	0	α_3	500	α_9	4
a_4	0.1	α_4	1	α_{10}	1
a_5	0	α_5	500	α_{11}	4
b	3500	α_6	1	α_{12}	1

Los códigos y demás bloques utilizados se encuentran el apéndice C.2.

4.1.3 Simulación control LQR

Para esta simulación se implementa la estructura de control LQR siguiendo el diagrama de la figura 4.3. Las posiciones deseadas de Trayectoria se restan con las posiciones retroalimentadas dentro de LQR y los demás estados entran directo. K es la matriz calculada en el capítulo anterior. El modelo dinámico implementado del cuadri-rotor es el no lineal simplificado (2.59) – (2.64).

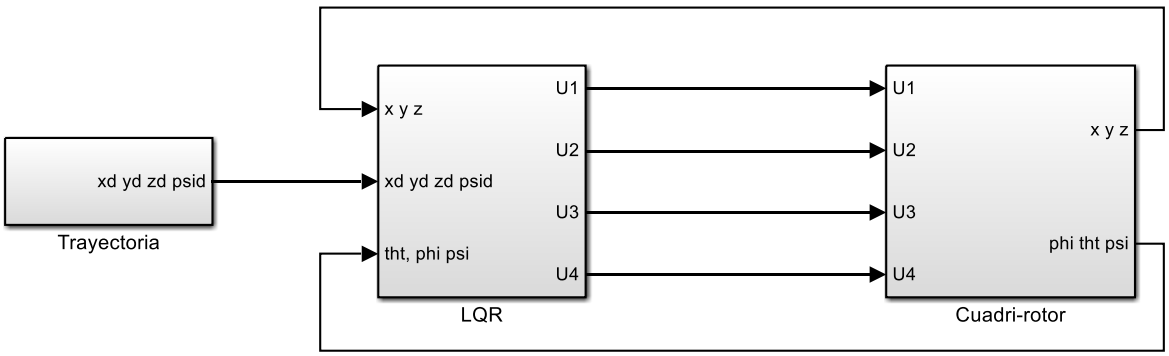


Fig. 4.3. Diagrama a bloques en Simulink del control LQR.

Después de encontrar los valores deseados de Q y de R que entregan una respuesta satisfactoria, el valor de K es:

$$K = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1.9 & 0 & 0 & 13.5 & 0 & 0 & 5.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1.7 & 0 \\ 0 & 0 & 0 & 0 & 1.9 & 0 & 0 & 0 & 13.5 & 0 & 0 & 5.2 \end{bmatrix} \tag{4.1}$$

Los códigos y demás bloques utilizados se encuentran el apéndice C.3.

4.1.4 Simulación control del cuadri-rotor basado en planitud

Para esta simulación se implementa la estructura de control basado en planitud siguiendo el diagrama de la figura 4.4. La trayectoria deseada y el generador de trayectoria se generan en el programa principal, con ciclos *For* que incrementa con el tiempo de muestreo hasta el tiempo final de la simulación para generar al sistema dinámico plano del capítulo 2. El bloque del Sistema plano entrega los estados y los torques de referencia. Las estrategias de control a utilizar son el PD, Backstepping y LQR según la reformulación mostrada en control basado en planitud del capítulo 3. El modelo dinámico del cuadri-rotor implementado, es el no lineal simplificado, para las tres siguientes simulaciones.

Los códigos y demás bloques utilizados se encuentran el apéndice C.4.

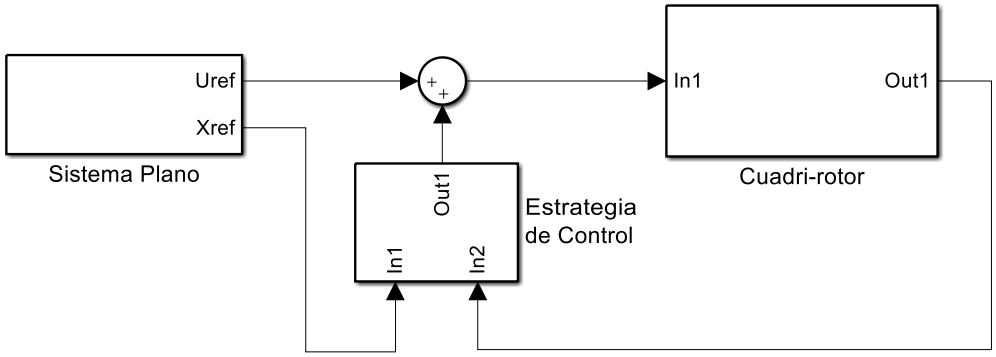


Fig. 4.4. Diagrama a bloques en Simulink del control basado en planitud.

4.1.4.1 Simulación control PD basado en planitud

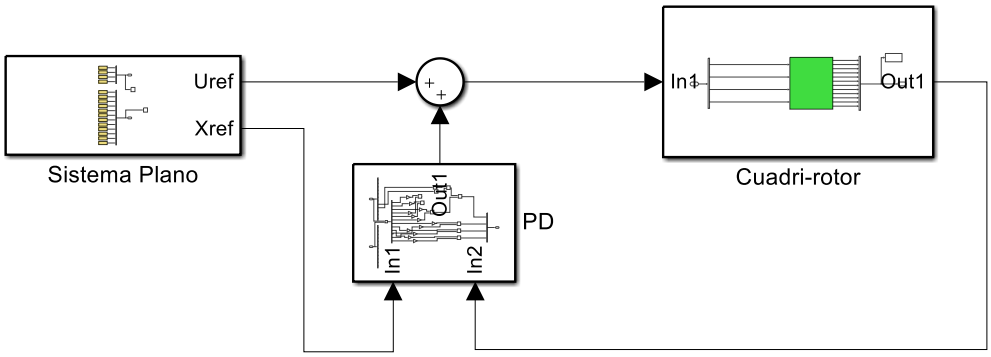


Fig. 4.5. Diagrama a bloques en Simulink del control PD basado en planitud.

Las ganancias utilizadas se presentan en la Tabla 4.3.

Tabla 4.3. Ganancias de simulación PD basado en planitud

Parámetro	Valor	Parámetro	Valor
$K_{p,\phi}$	1	$K_{d,\phi}$	0.3
$K_{p,\theta}$	1	$K_{d,\theta}$	0.3
$K_{p,\psi}$	0.8	$K_{d,\psi}$	0.2
K_{px}	3	$K_{d,x}$	2
K_{py}	3	K_{dy}	2
K_{pz}	4	K_{dz}	3

Los códigos y demás bloques utilizados se encuentran el apéndice C.4.1.

4.1.4.2 Simulación control Backstepping basado en planitud

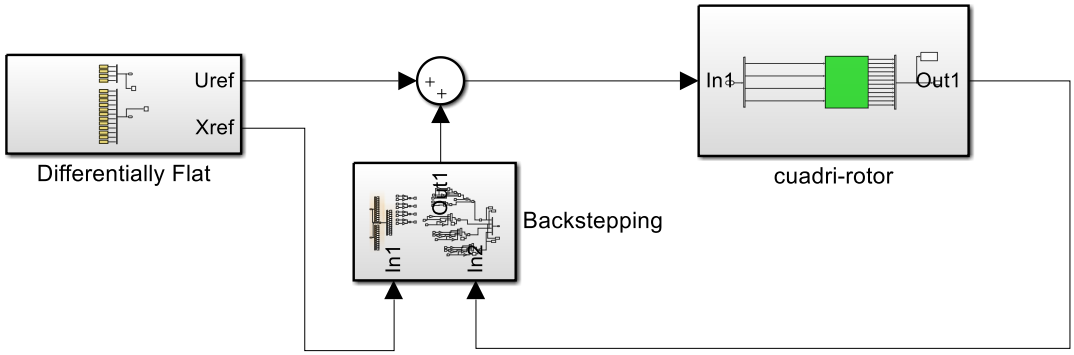


Fig. 4.6. Diagrama a bloques en Simulink del control Backstepping basado en planitud.

Las ganancias usadas en la simulación son las de la Tabla 4.4.

Tabla 4.4. Ganancias de simulación Backstepping basado en planitud.

Parámetro	Valor	Parámetro	Valor	Parámetro	Valor
a_1	0	α_1	500	α_7	4
a_2	-0.1	α_2	1	α_8	1
a_3	0	α_3	500	α_9	4
a_4	0.1	α_4	1	α_{10}	1
a_5	0	α_5	500	α_{11}	4
b	3500	α_6	1	α_{12}	1

Los códigos y demás bloques utilizados se encuentran el apéndice C.4.2.

4.1.4.3 Simulación control LQR basado en planitud

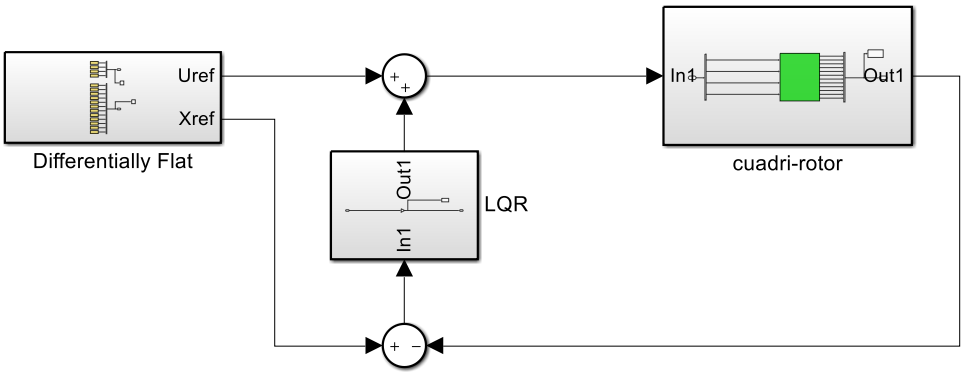


Fig. 4.7. Diagrama a bloques en Simulink del control LQR basado en planitud.

Después de probar los valores deseados de Q y de R que proporcionan resultados satisfactorios, el valor de K es:

$$K = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1.9 & 0 & 0 & 13.5 & 0 & 0 & 5.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1.7 & 0 \\ 0 & 0 & 0 & 0 & 1.9 & 0 & 0 & 0 & 13.5 & 0 & 0 & 5.2 \end{bmatrix} \quad (4.2)$$

Los códigos y demás bloques utilizados se encuentran el apéndice C.4.3.

4.2 Simulación en Webots

Webots es un simulador de robots que proporciona un entorno de desarrollo completo para modelar, programar y simular robots en un entorno virtual en 3D, como se muestra en la figura 4.8. Con este software, se puede diseñar configuraciones complejas de robots, con uno o varios robots similares o diferentes, en un entorno compartido. Se pueden elegir las propiedades de cada objeto, como forma, color, textura, masa, fricción, además de contar con una variedad de sensores y actuadores con la que poder equipar nuestros robots, se pueden ejecutar controladores escritos en lenguajes compilados (C / C ++, Java) o interpretados (Python, MATLAB), o software de terceros que permitan la capacidad por medio del protocolo TCP/IP. El comportamiento de los robots se puede probar en mundos físicamente realistas.

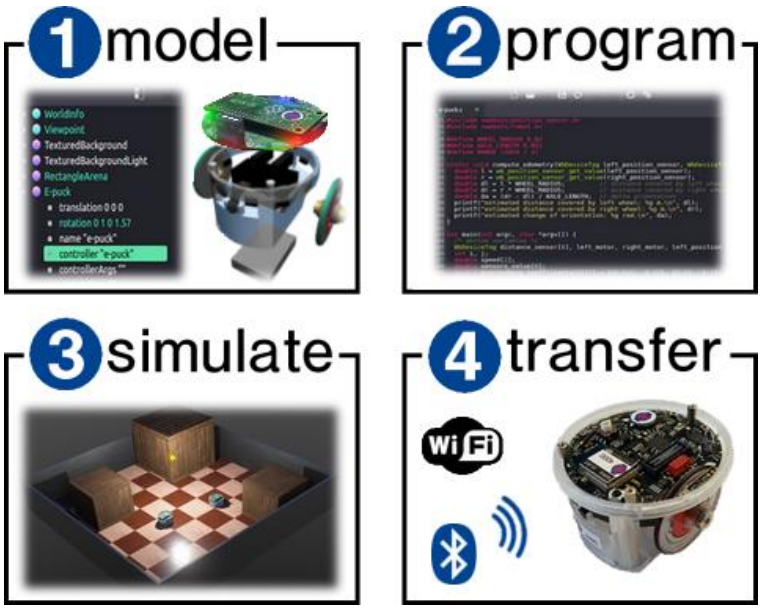


Fig. 4.8. Simulador de Robots Webots [36].

Este trabajo se comenzó con la versión 2018b con licencia comercial estudiantil. Para diciembre de 2018 se actualiza a la versión R2019a, primera version en código libre, este trabajo se finaliza en la versión R2020a con licencia código libre. El software se ejecuta en Windows, con procesador Intel core i5 2.30GHz, 8GB de RAM y una tarjeta gráfica NVIDIA GTX 1050 con 4GB.

El lenguaje de modelado de realidad virtual (VRML de sus siglas en inglés de Virtual Reality Modeling Language), es el lenguaje de programación que se utiliza en Webots para modelar el mundo virtual donde se simulará el comportamiento de los robots. Este lenguaje posibilita la descripción de objetos 3D a partir de prototipos basados en formas geométricas básicas o estructuras especificadas a partir de vértices y aristas. Además, se pueden añadir características a los objetos como color o materiales. Webots no incluye todos los nodos de VRML97, pero sí que añade algunos muy importantes para trabajar con simulación de robots.

Los nodos se estructuran como un árbol, donde de manera jerárquica de un mundo virtual en VRML como se muestra en la figura 4.8, donde cada círculo representa un nodo con cierta funcionalidad y los nodos padres agrupan a los nodos hijos.

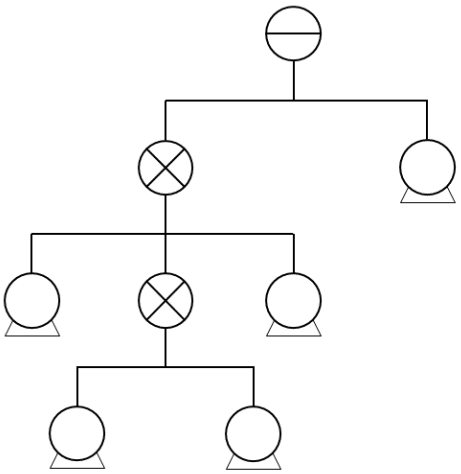


Fig. 4.8. Jerarquía de un mundo VRML [37].

Algunos de los nodos que aporta Webots se muestran en la figura 4.9. De los cuales, el nodo Propeller que servirá para crear las hélices del cuadrirrotor, es de los más importantes para este trabajo. Tambien se cuenta con algunos sensores, como por ejemplo un sensor de distancia con el nodo DistanceSensor, e incluso un receptor GPS (del inglés Global Positioning System) agregando el nodo GPS al robot. Para establecer algún tipo de comunicación, podemos incluir en la simulación los nodos Emitter y Receiver, los cuales no están contemplados para este trabajo. Además de otros nodos como pueden ser Charger,

Gripper, Joint, LED, Light Sensor, Physics o TouchSensor, tenemos el nodo Supervisor. Este nodo se puede utilizar para controlar el mundo y todos los robots que éste contiene.

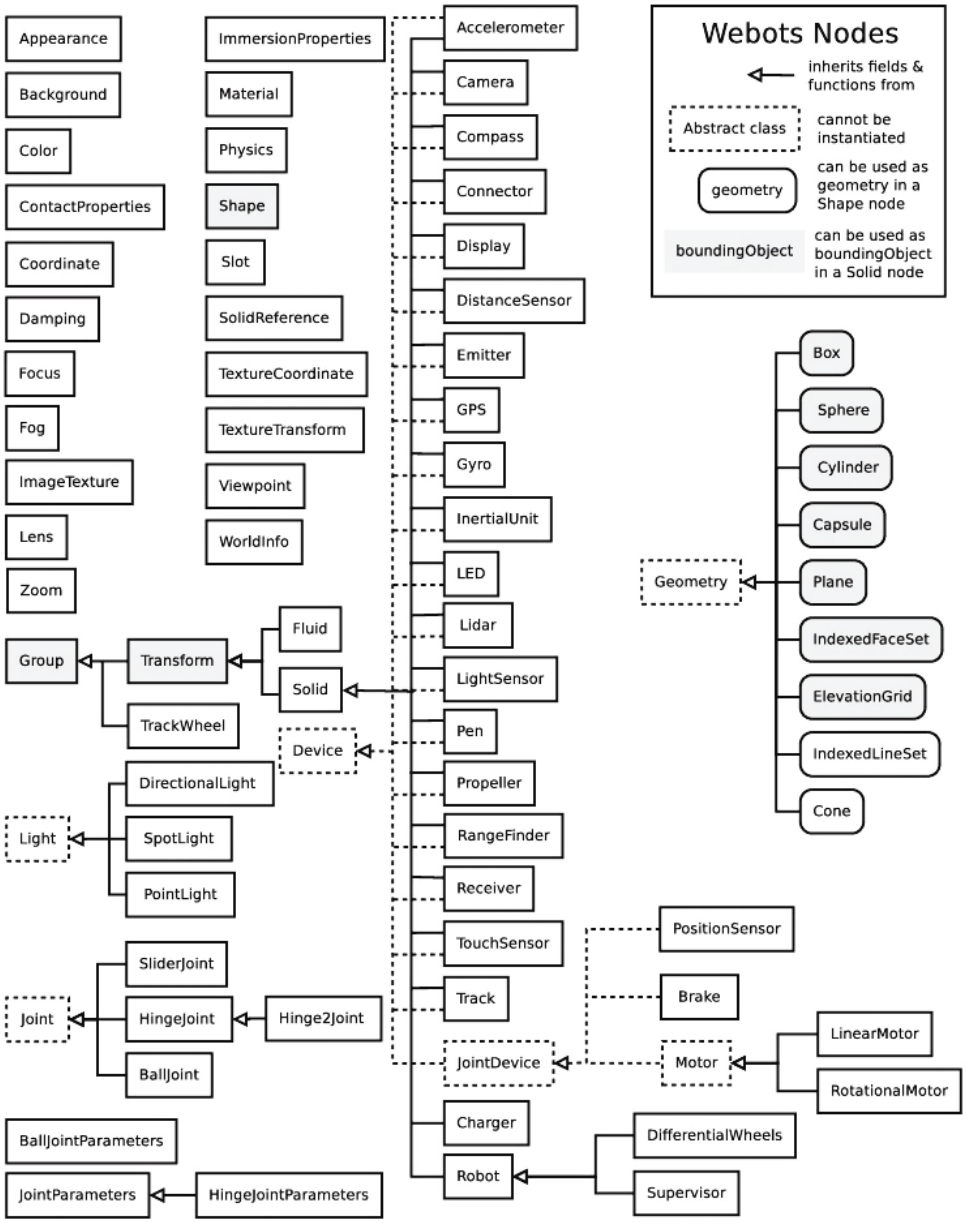


Fig. 4.9. Nodos en Webots [36].

El Nodo Propeller se utilizó para modelar una hélice, este nodo tiene cinco propiedades las cuales se configuraron con los mismos parámetros considerados en la simulación en MATLAB. ShaftAxis define el eje sobre el cual se ejercerá el empuje y torque generado, centerOfThrust es el punto donde se aplica el empuje, thrustConstants y torqueConstants son las constantes de empuje y torque respectivamente, fastHelixthreshold define la representación gráfica de la hélice de acuerdo a su velocidad.

La construcción de una hélice se realizó agregando nodos según se muestra en las figura 4.10, donde después de configurar sus parámetros, se agrego el nodo Device y a este último un nodo RotationalMotor al cual se le asigna la variable Motor 1 con la cual se controla con su valor de velocidad desde el controlador. Para FastHelix y SlowHelix se generan nodos solidos para asignar Hélice fast y Hélice slow que son la representación de las hélices, dependientes de fastHelixthreshold. Ya contruida esta hélice se puede replicar para generar las otras tres para el cuadri-rotor.

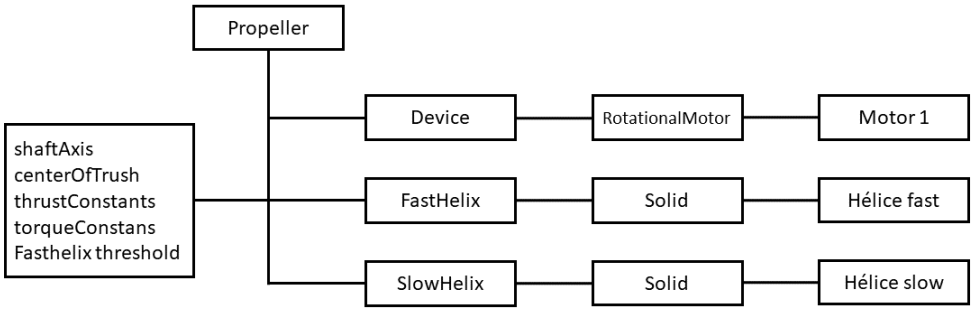


Fig. 4.10. Hélice en el entorno de Webots.

La primera aproximación a la construcción del cuadri-rotor, fue utilizar cuatro nodos propeller anclados a una masa en el centro de ellos, se experimentó para observar el funcionamiento del GPS y de la Unidad de medicion inercial (IMU del inglés Inertial Measurement Unit), quedando el modelo como en la figura 4.11.

En esta primera simulación se observó el efecto de girar sobre su eje Z debido a que el sentido de giro de las cuatro hélices fue en mismo. Corrigiendo, como se explica en el capitulo 2, el funcionamiento del cuadri-rotor es el que dos de las hélices giren en sentido horario y las otras dos en sentido antihorario, esto se especifica, cambiando el signo del vector de shaftAxis en las dos de las hélices. Ya con el sentido de las helices correcto, se realizan experimentos donde se aplica la misma velocidad a cada motor, con lo cual se logra sustentación y se logra verificar el funcionamiento del sensor GPS y de la IMU.

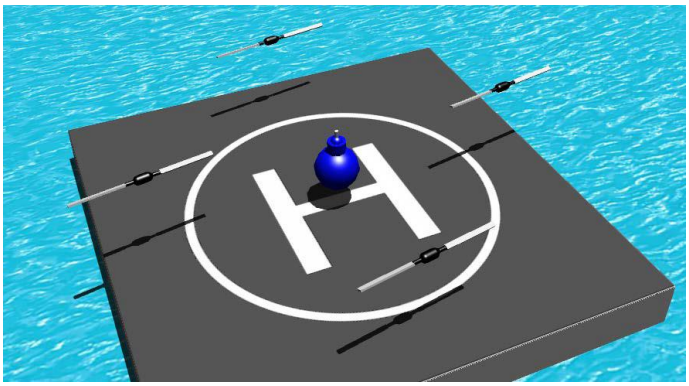


Fig. 4.11. Primera aproximación del cuadri-rotor en el entorno de Webots.

Aplicando lo aprendido con las simulaciones anteriores, se realiza un modelo mas realista basado en el cuerpo de un cuadrirrotor con las medidas que se muestran en la figura 4.12.

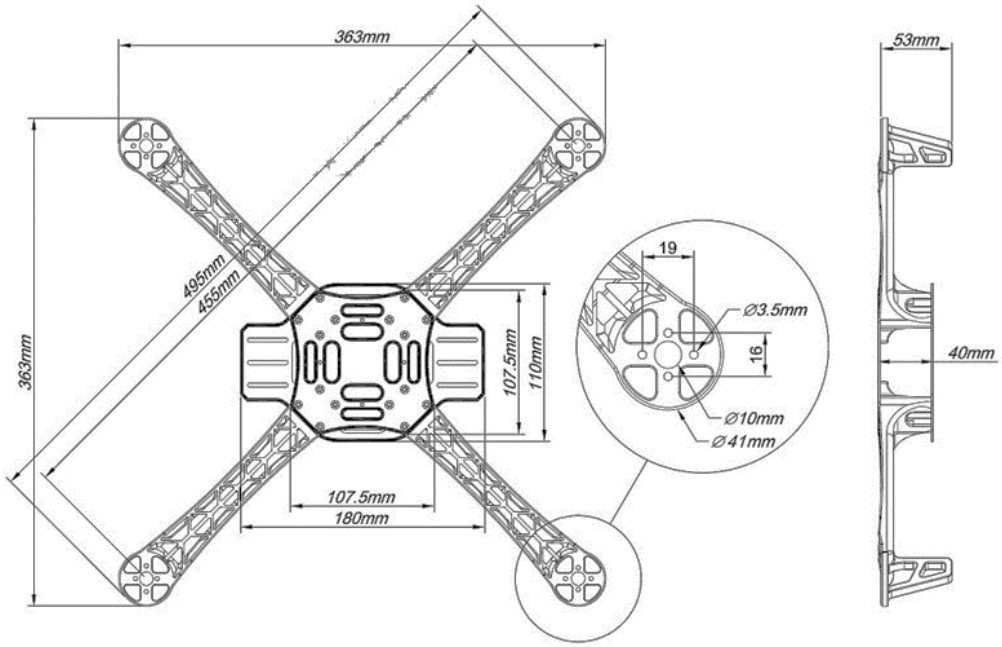


Fig. 4.12. Esquema del cuerpo del cuadrirrotor DJI F450.

Se optó, por crear el cuerpo en SolidWorks, software CAD para modelado mecánico, quedando como se ve en la figura 4.13, el cual se exporta con formato VRML97, para su importación en Webots. Se muestra en el apéndice D.

Fig. 4.13. Cuerpo del cuadrirrotor DJI F450 en SolidWorks.

La estructura del cuadri-rotor siguiendo la estructura jerárquica, mostrada en la figura 4.14, se tiene el primer nodo padre, que es del tipo Robot. Al cual se le agregan los nodos hijos, GPS, IMU, Propeller y Transform. El nodo Propeller se agregó cuatro veces, uno para cada rotor. El nodo Motor se agrega al nodo Device quien es un nodo hijo de Propeller. A los nodos Transform se agregan nodos de la importación de la estructura de un archivo VRLM97 creado en SolidWorks. El nodo IMU y GPS entregan la orientación y posición del cuadri-rotor y el nodo Motor recibe el valor de velocidad generado en el controlador.

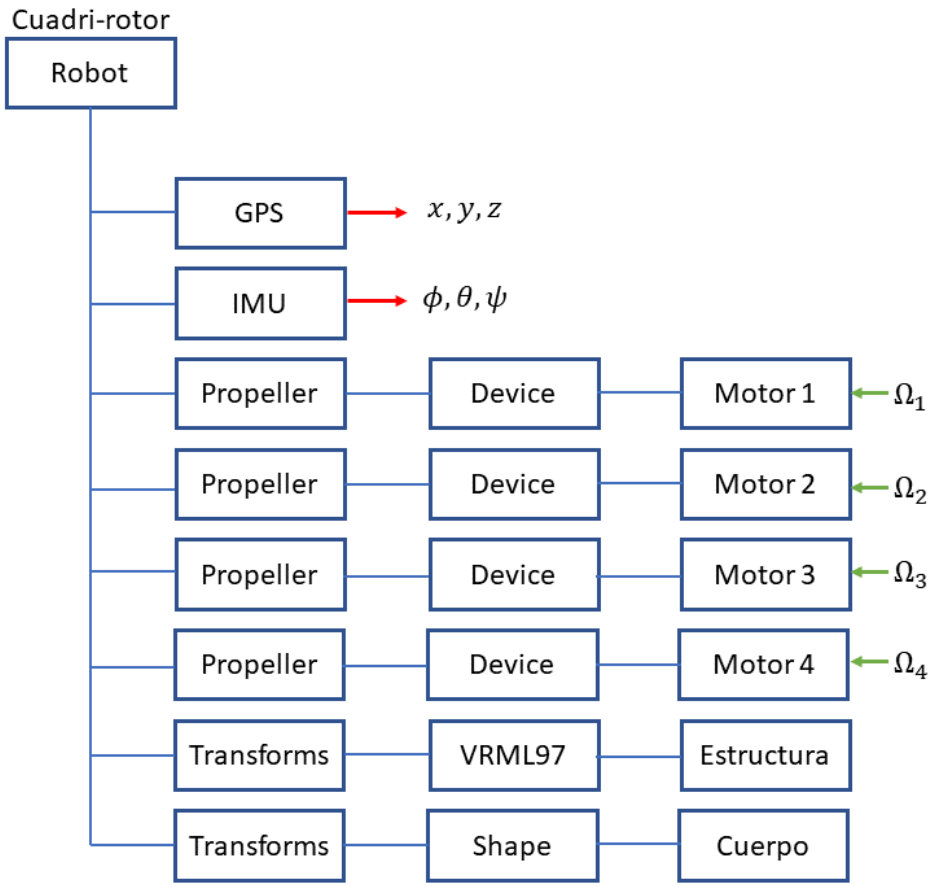


Fig. 4.14. Estructura del cuadri-rotor en Webots.

El modelo del cuadri-rotor contempla la masa de la estructura más un aproximado de la masa del equipo electrónico necesario para su implementación, estableciendo una masa de 1.2 kg en total, para una mejor aproximación de la simulación a la realidad y también respetando los esquemas de los valores permitidos (sección 1.3). Teniendo el cuadri-rotor construido en el ambiente de Webots como se muestra en la figura 4.15.

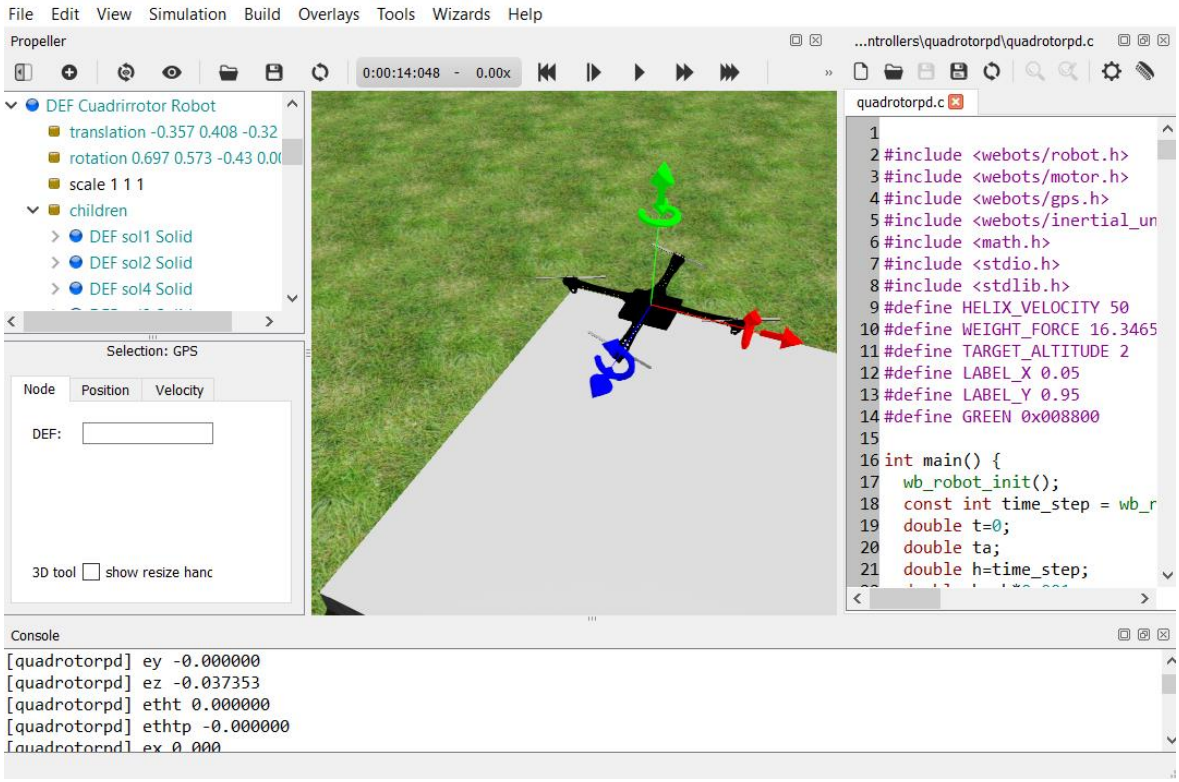


Fig. 4.15. Modelo de cuadrirrotor en entorno Webots.

La implementación de los programas de los controladores se creó en lenguaje C, donde se programó según el diagrama de la figura 4.16, dentro del controlador se utilizó el método de Euler para la obtención de las derivadas. Para el seguimiento de trayectorias se parte de que cuadri-rotor se encuentra estabilizado en su punto de partida de la trayectoria deseada, para esto en las simulaciones se asignó un tiempo de 10 s donde el cuadri-rotor llega a su condición inicial mediante un control PD. Fue necesario desarrollar los códigos en lenguaje C, para especificar la trayectoria deseada, calcular los errores de seguimiento, la implementación de la derivada, la implementación de los algoritmos de control y la obtención de las velocidades a partir de las entradas de control.

Las ganancias para cada controlador son las mismas que en las simulaciones en MATLAB con el fin de que la comparación se haga bajo las mismas condiciones. Para facilitar la comparación con las simulaciones en MATLAB, los datos se guardan en un fichero.txt, el cual se genera en la misma carpeta del controlador de la simulación en Webots. Para su visualización se crea un programa en MATLAB que genera las gráficas requeridas. Los códigos de los controladores se realizan en similitud al código del controlador PD que se encuentra en el apéndice E.

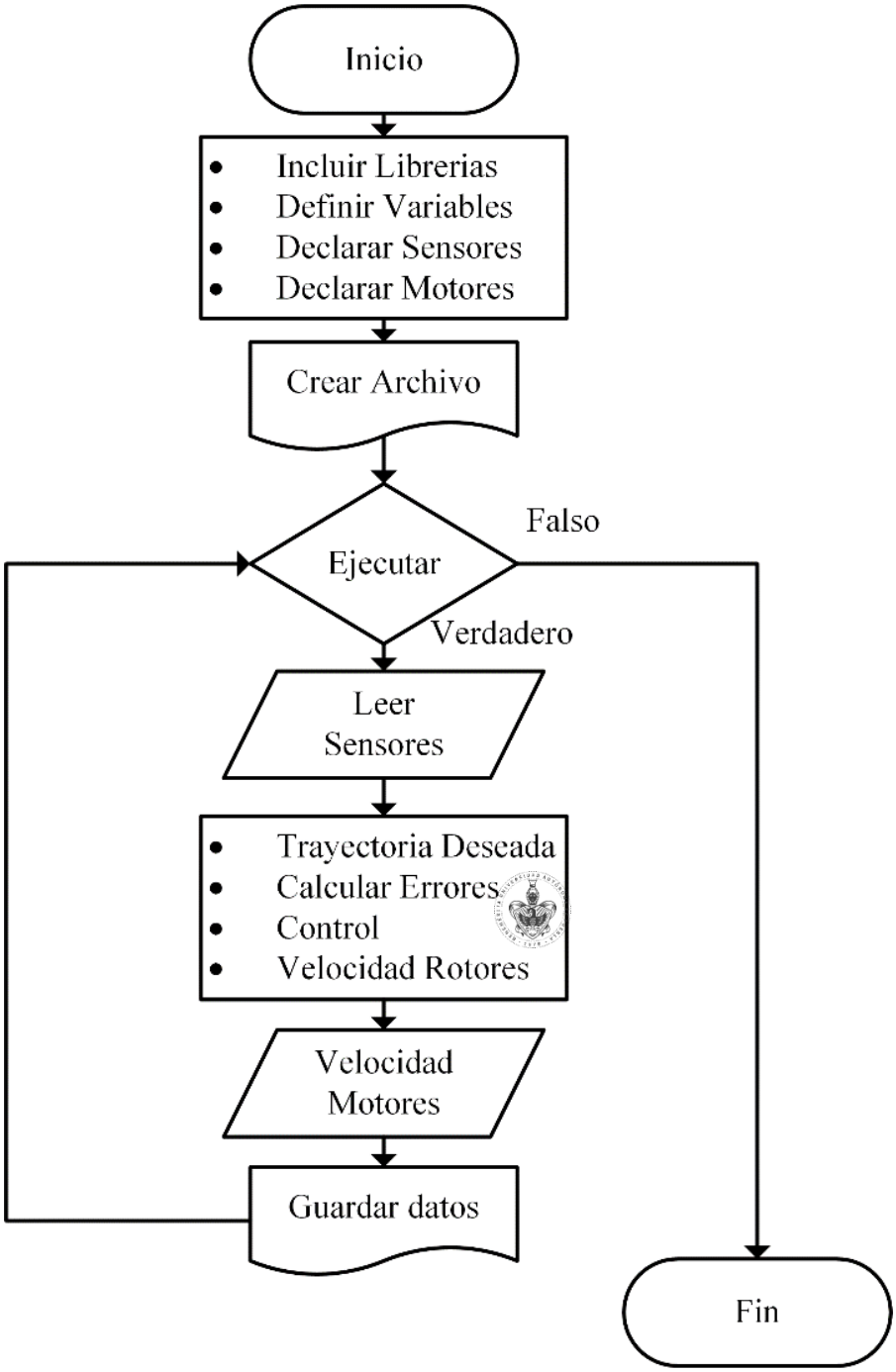


Fig. 4.16. Modelo de cuadri-rotor en entorno Webots.

Capítulo 5

Resultados

Con el fin de visualizar y tener un punto de comparación de los resultados arrojados por Webots, se realizan las simulaciones tanto en Webots como en MATLAB, considerando los mismos parámetros en ambos, estos parámetros se muestran en la Tabla 5.1; La trayectoria deseada se describe con las coordenadas x, y, z (posiciones lineales) y ψ (posición angular en yaw).

Tabla 5.1. Parámetros de simulación

Parámetro	Valor	Parámetro	Valor
I_{xx},I_{yy},I_{zz}	0.001kg m^2	m	1.2 kg
b	0.0001N s^2	l	0.35 m
k_T	0.0001 Nm s^2	h	0.035 s
J_R	0.001kg m^2	g	9.81 m/s^2

Las simulaciones se prueban siguiendo dos trayectorias según la Tabla 5.2 y 5.3.

Tabla 5.2. Trayectoria de simulación A

Trayectoria A: Circulo	
Posición deseada 1	**Valor**
x_d	$0.8\cos(0.2t)$m
y_d	$0.8sen(0.2t)$m
z_d	0.5 m
ψ_d	0 rad
x_i	0.8m
y_i	0 m
z_i	0.5 m
ψ_i	0 rad

Tabla 5.3. Trayectoria de simulación B

Trayectoria B: Ocho	
Posición deseada	**Valor**
x_d	$\cos(0.1t)$m
y_d	$sen(0.2t)$m
z_d	1 m
ψ_d	0 rad
x_i	1 m
y_i	0 m
z_i	1 m
ψ_i	0 rad

5.3 Simulación en MATLAB y Webots

En todas las simulaciones el punto de inicio de la trayectoria en el tiempo $t = 0$, corresponde donde el cuadri-rotor se encuentra en las posiciones iniciales, y se encuentra en vuelo estacionario. Para las simulaciones en Webots se implementa un control PD que lleva al cuadri-rotor a las condiciones iniciales de las trayectorias en 10 s, (figura 5.1) y (figura 5.2).

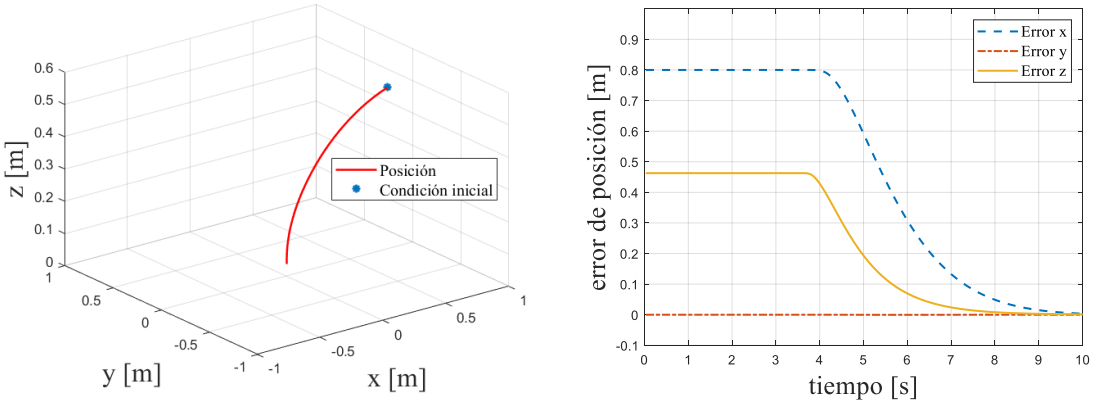


Fig. 5.1. Condición inicial y error de posición para las simulaciones de la trayectoria A en Webots.

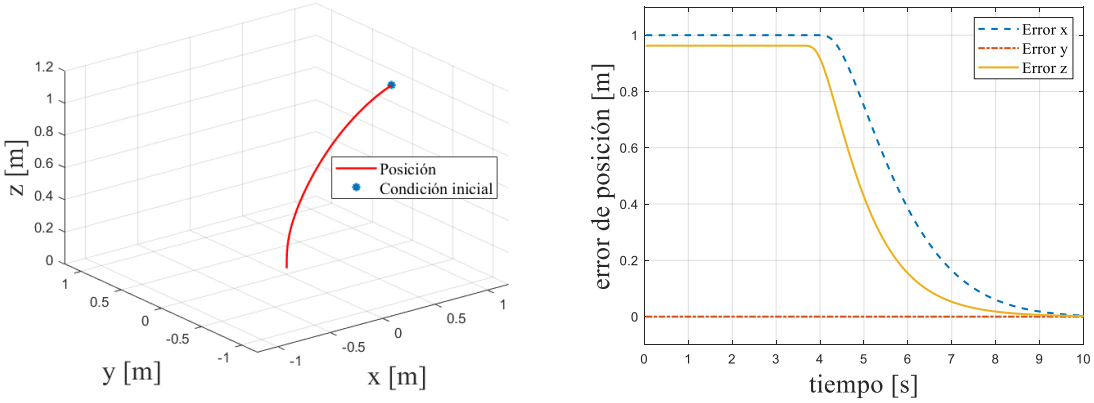


Fig. 5.2. Condición inicial y error de posición para las simulaciones de la trayectoria B en Webots.

5.3.1 Simulación control PD

El resultado de las simulaciones en MATLAB (figura 5.3) y en Webots (figura 5.4) del control PD, a pesar de la diferencia entre la referencia y la posición, se tiene que en ambas se logra seguir la trayectoria A, logrando mantener la altura z y con errores en x, y más grandes en la simulación en MATLAB que en Webots.

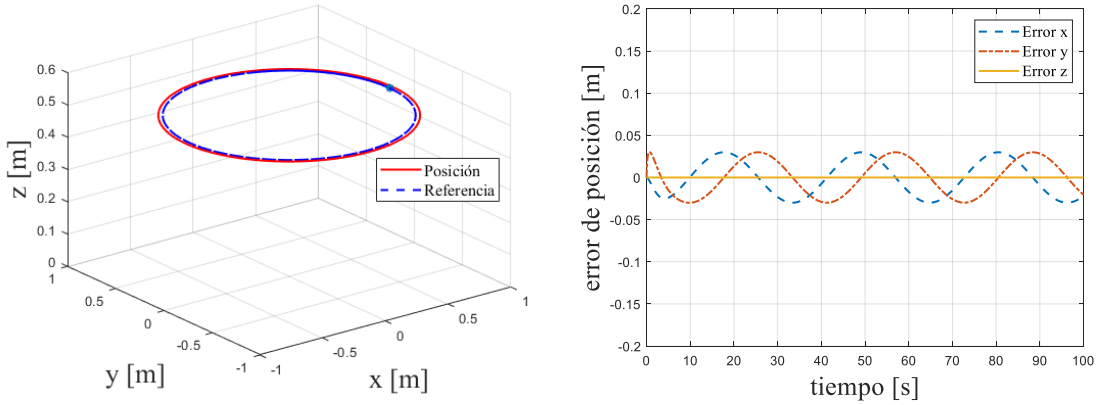


Fig. 5.3. Trayectoria del cuadri-rotor y error en la posición, aplicando el control PD siguiendo la trayectoria A en MATLAB.

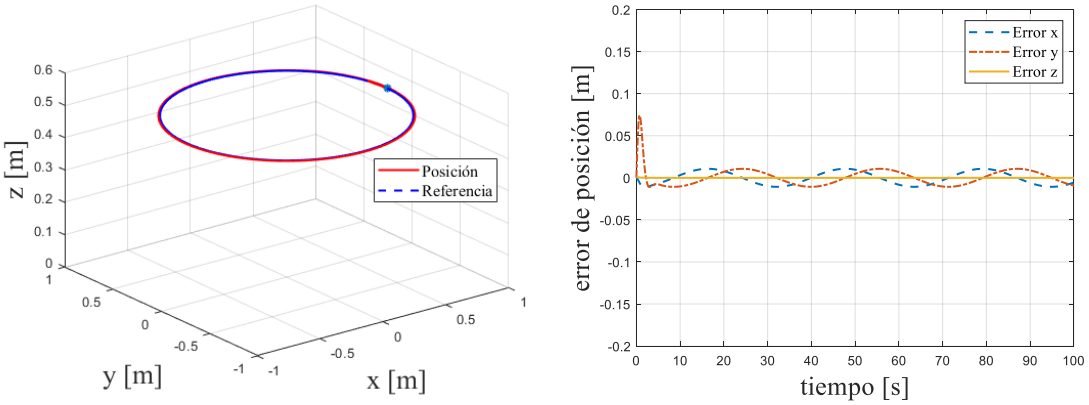


Fig. 5.4. Trayectoria del cuadri-rotor y error en la posición, aplicando el control PD siguiendo la trayectoria A en Webots.

El resultado de las simulaciones en MATLAB (figura 5.5) y en Webots (figura 5.6), muestran que se logra seguir la trayectoria B, el error en y es más grande que el de x esto debido a que la referencia de y cambia más rápido que x.

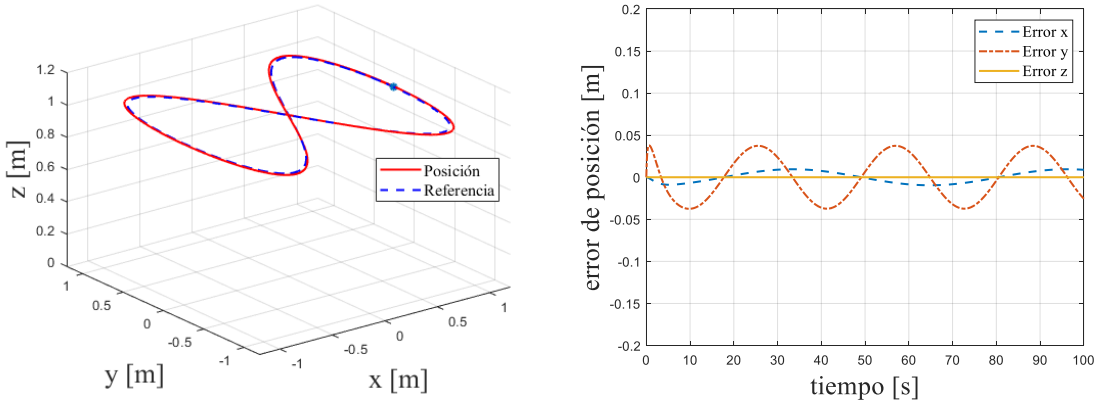


Fig. 5.5. Trayectoria del cuadri-rotor y error en la posición, aplicando el control PD siguiendo la trayectoria B en MATLAB.

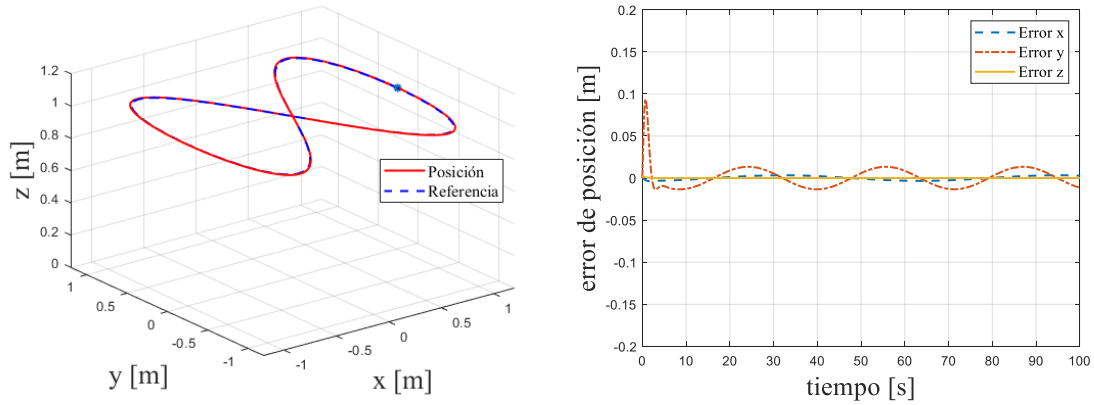


Fig. 5.6. Trayectoria del cuadri-rotor y error en la posición, aplicando el control PD siguiendo la trayectoria B en Webots.

Para el control de PD se muestra un mejor desempeño en la simulación en Webots, en el seguimiento de las dos trayectorias, aun así el comportamiento del cuadri-rotor en Webots se considera compatible con el modelo dinámico (2.39) – (2.44), implementado en MATLAB.

5.3.2 Simulación control Backstepping

El resultado de las simulaciones en MATLAB (figura 5.7) y en Webots (figura 5.8), muestra que se logra seguir la trayectoria A de referencia. Los errores de la simulación en MATLAB tienen similar forma a los de Webots, pero en MATLAB se observa una vibración en la posición.

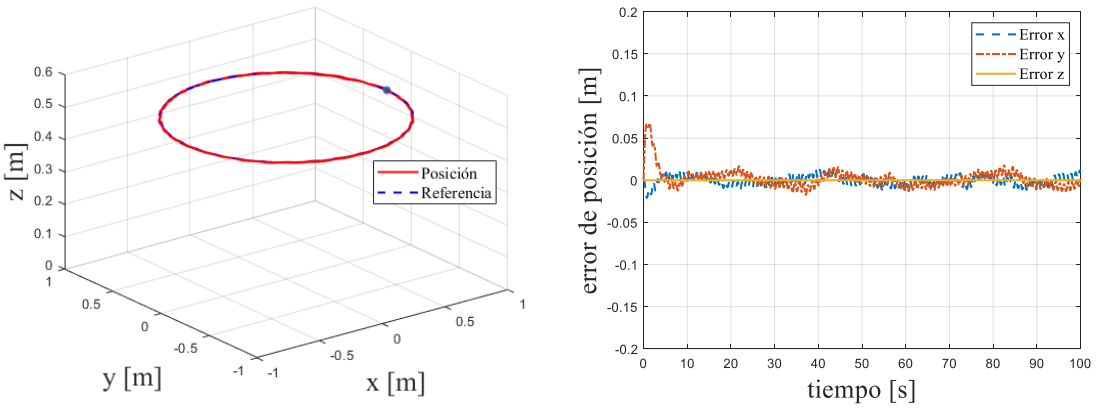


Fig. 5.7. Trayectoria del cuadri-rotor y error en la posición, aplicando el control Backstepping siguiendo la trayectoria A en MATLAB.

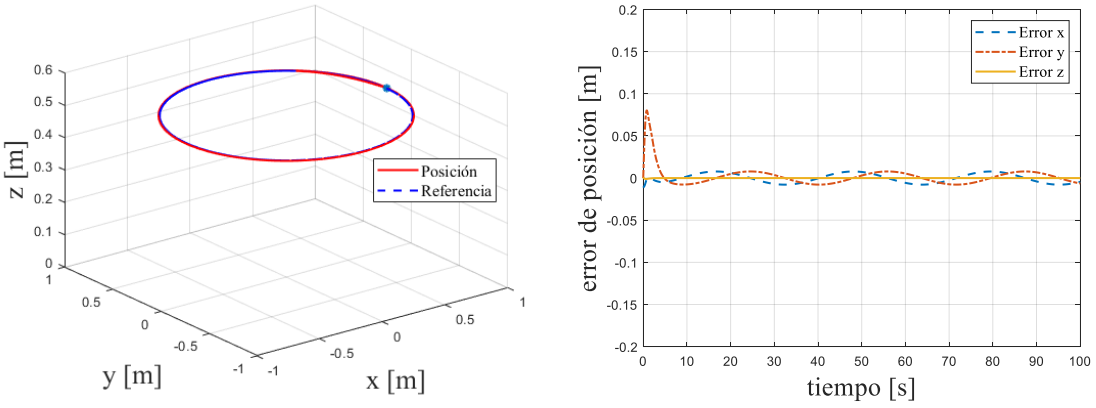


Fig. 5.8. Trayectoria del cuadri-rotor y error en la posición, aplicando el control Backstepping siguiendo la trayectoria A en Webots.

El resultado de las simulaciones en MATLAB (figura 5.9) y en Webots (figura 5.10), muestra que en ambas se logra seguir la trayectoria B, manteniendo la altura z y con errores en x, y, similares entre ambas simulaciones.

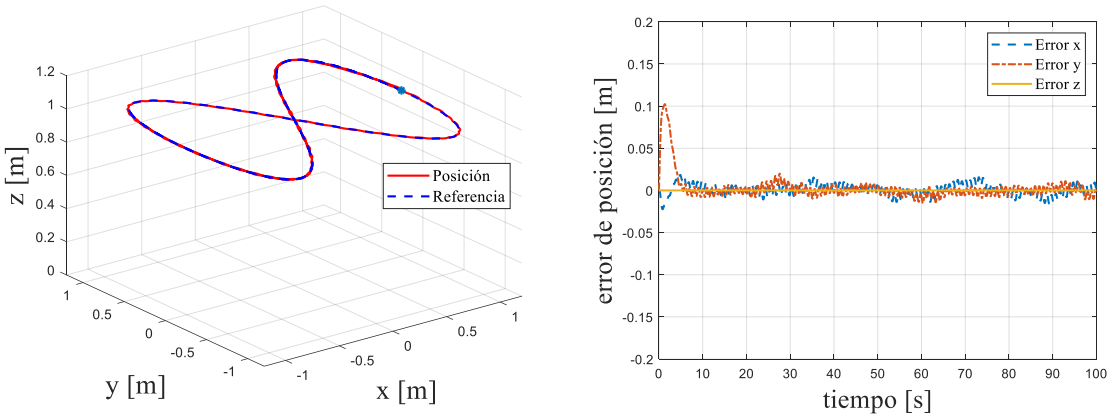


Fig. 5.9. Trayectoria del cuadri-rotor y error en la posición, aplicando el control Backstepping siguiendo la trayectoria B en MATLAB.

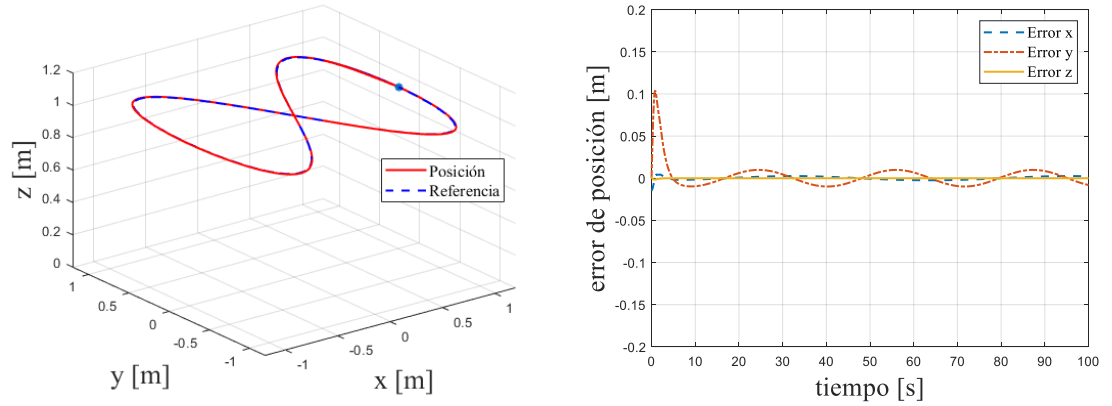


Fig. 5.10. Trayectoria del cuadri-rotor y error en la posición, aplicando el control Backstepping siguiendo la trayectoria B en Webots.

Para el control de Backstepping se muestra que el desempeño en la simulación en Webots para el seguimiento de las dos trayectorias, es similar al comportamiento del cuadri-rotor implementado en MATLAB. Por lo que se encuentra que el modelo dinámico implementado internamente por Webots tiene correspondencia con el del modelo dinámico (2.39) – (2.44).

5.3.3 Simulación control LQR

El resultado de las simulaciones en MATLAB (figura 5.11) y en Webots (figura 5.12), a pesar de la diferencia entre la referencia y la posición, se tiene que en ambas se logra seguir la trayectoria A, logrando mantener la altura z y con errores en x, y de 0.18m y de 0.16m en MATLAB como en Webots.

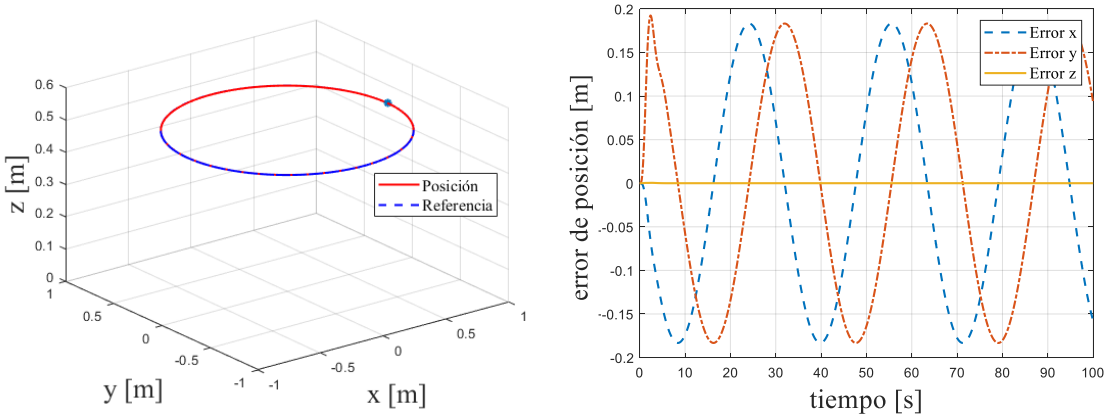


Fig. 5.11. Trayectoria del cuadri-rotor y error en la posición, aplicando el control LQR siguiendo la trayectoria A en MATLAB.

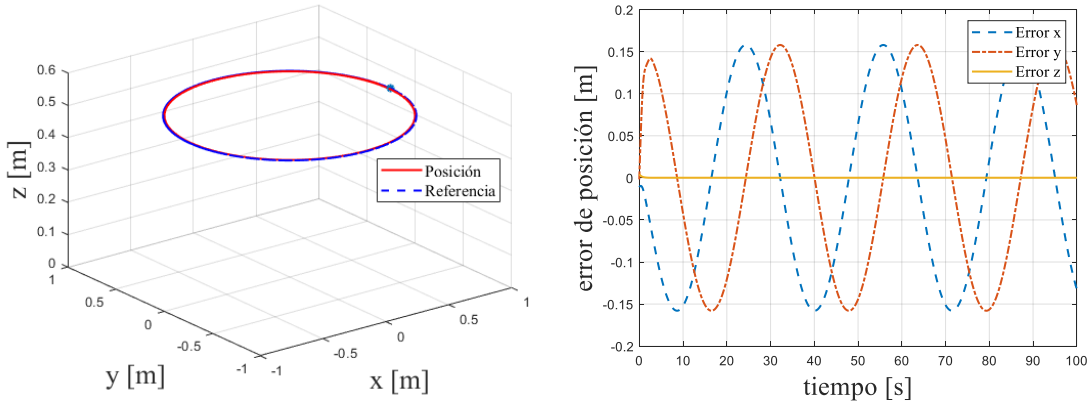


Fig. 5.12. Trayectoria del cuadri-rotor y error en la posición, aplicando el control LQR siguiendo la trayectoria A en Webots.

El resultado de las simulaciones en MATLAB (figura 5.13) y en Webots (figura 5.14), para la trayectoria B, las dos logran seguir la trayectoria, logrando mantener la altura z y con errores en $x\ de$ 0.1 y en y de 0.2m, muy similares entre la simulación en MATLAB y en Webots.

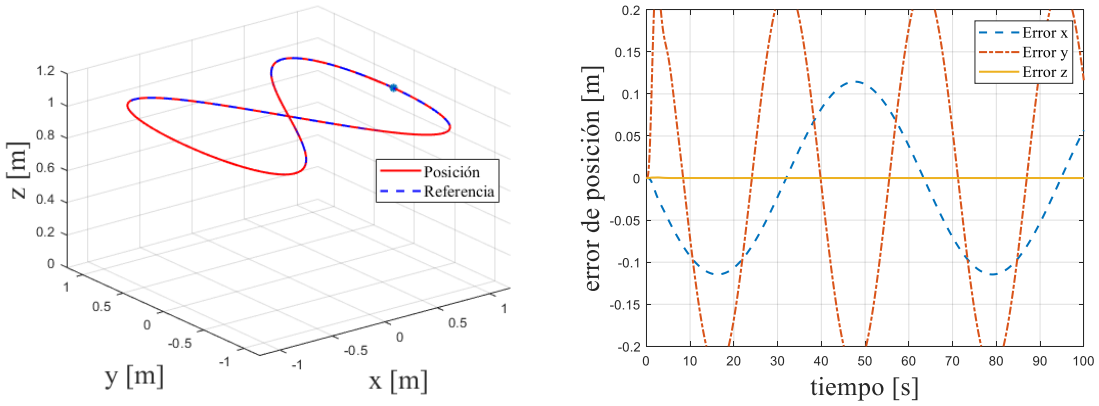


Fig. 5.13. Trayectoria del cuadri-rotor y error en la posición, aplicando el control LQR siguiendo la trayectoria B en MATLAB.

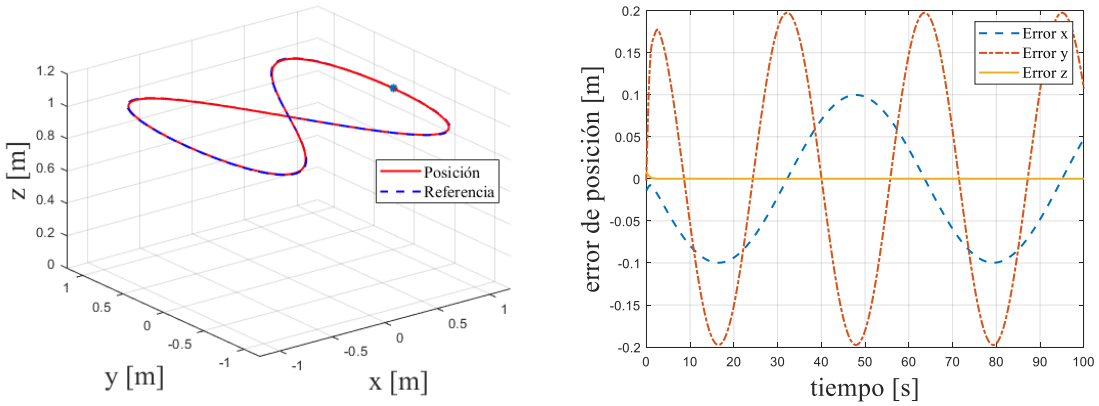


Fig. 5.14. Trayectoria del cuadri-rotor y error en la posición, aplicando el control LQR siguiendo la trayectoria B en Webots.

Para el control de LQR se muestra un desempeño muy similar entre las dos simulaciones, en el seguimiento de las dos trayectorias, por lo que se considera compatible el modelo en Webots, con el modelo dinámico simplificado (2.59) – (2.64) implementado en MATLAB.

5.3.4 Simulación control del cuadri-rotor basado en planitud

Se muestran en las figuras 5.15 y 5.16 las posiciones y sus derivadas calculadas en la simulación usando el método de derivación numérica de Euler, para la trayectoria A y B respectivamente, las cuales sirven para la obtención del modelo plano (2.63) – (2.74).

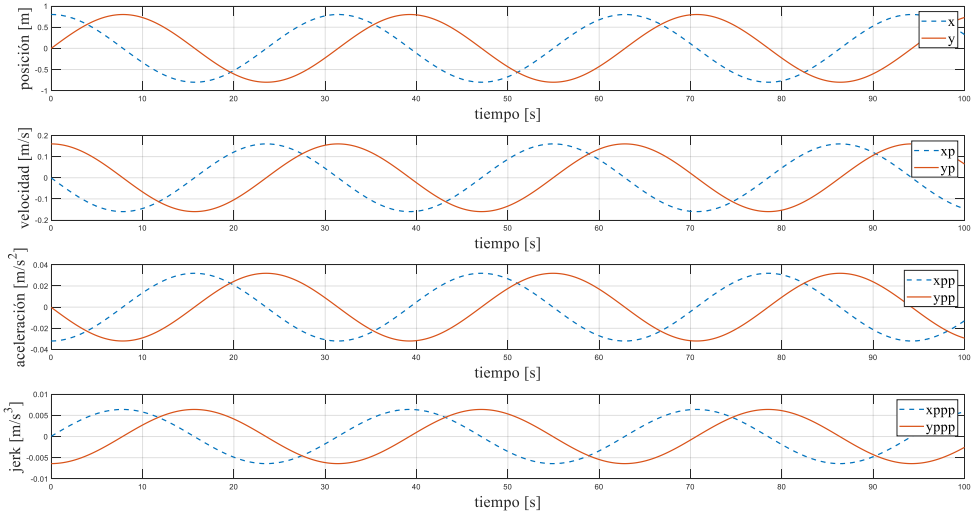


Fig. 5.15. Trayectoria de referencia y sus derivadas calculadas para la generación de trayectorias para la trayectoria A.

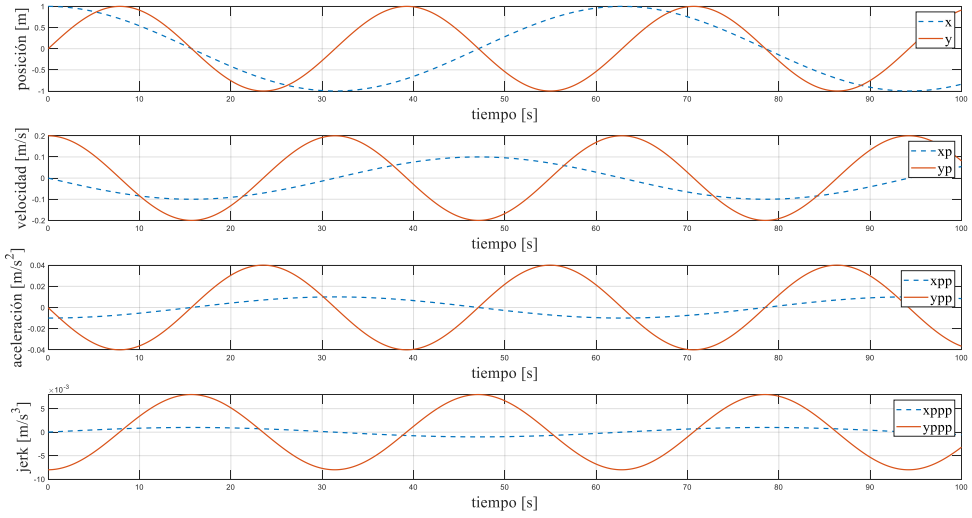


Fig. 5.16. Trayectoria de referencia y sus derivadas calculadas para la generación de trayectorias para la trayectoria B.

5.3.4.1 Simulación control PD basado en planitud

El resultado de las simulaciones en MATLAB (figura 5.17) y en Webots (figura 5.18), a pesar de la diferencia entre la referencia y la posición, se tiene que en ambas se logra seguir la trayectoria A, logrando mantener la altura z y con errores en x, y más grandes en la simulación en Webots que en MATLAB.

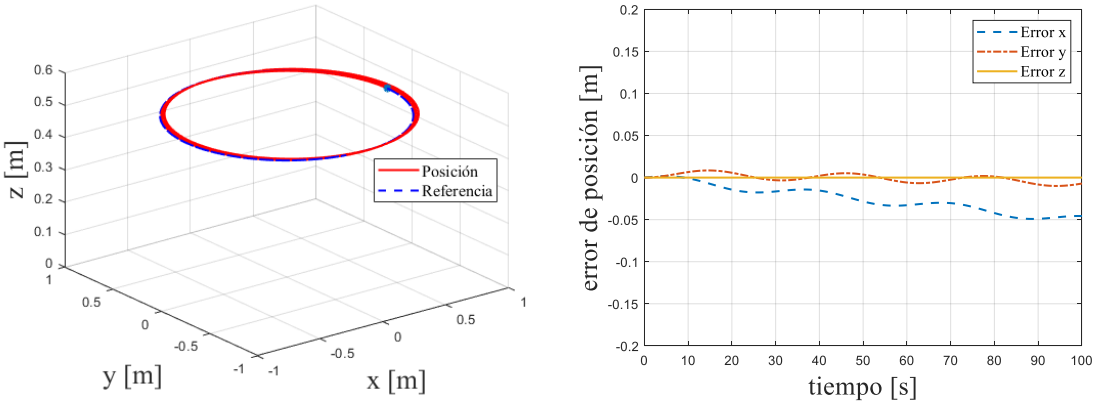


Fig. 5.17. Trayectoria del cuadri-rotor y error en la posición, aplicando el control PD basado en planitud siguiendo la trayectoria A en MATLAB.

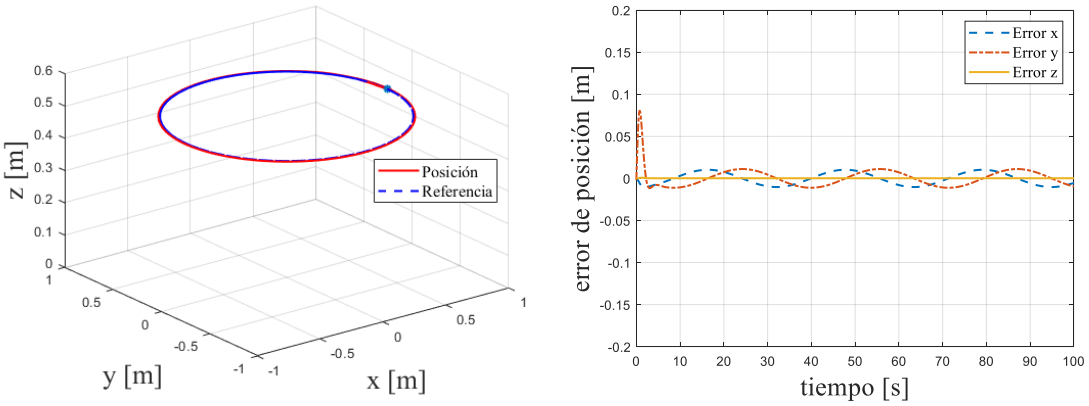


Fig. 5.18. Trayectoria del cuadri-rotor y error en la posición, aplicando el control PD basado en planitud siguiendo la trayectoria A en Webots.

El resultado de las simulaciones en MATLAB (figura 5.19) y en Webots (figura 5.20), se tiene que en ambas se logra seguir la trayectoria B, logrando mantener la altura z y con errores en x, y más grandes en la simulación en MATLAB que en Webots, como en la otra trayectoria.

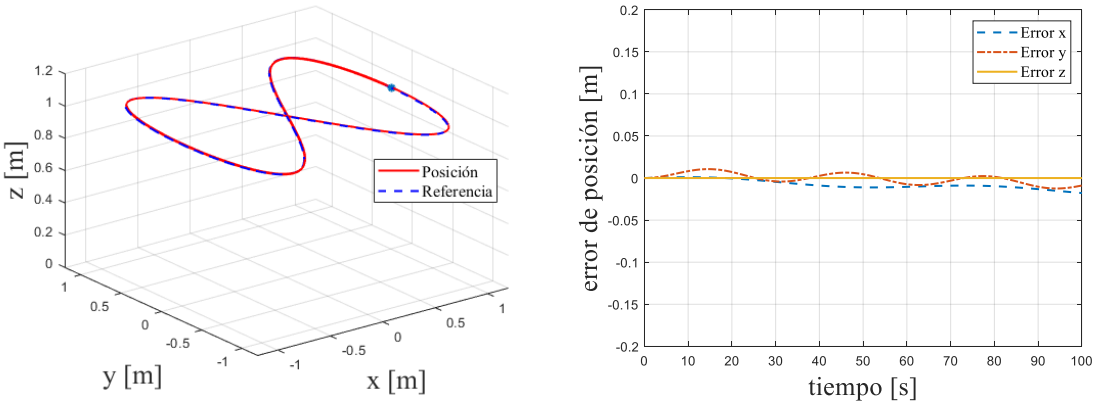


Fig. 5.19. Trayectoria del cuadri-rotor y error en la posición, aplicando el control PD basado en planitud siguiendo la trayectoria B en MATLAB.

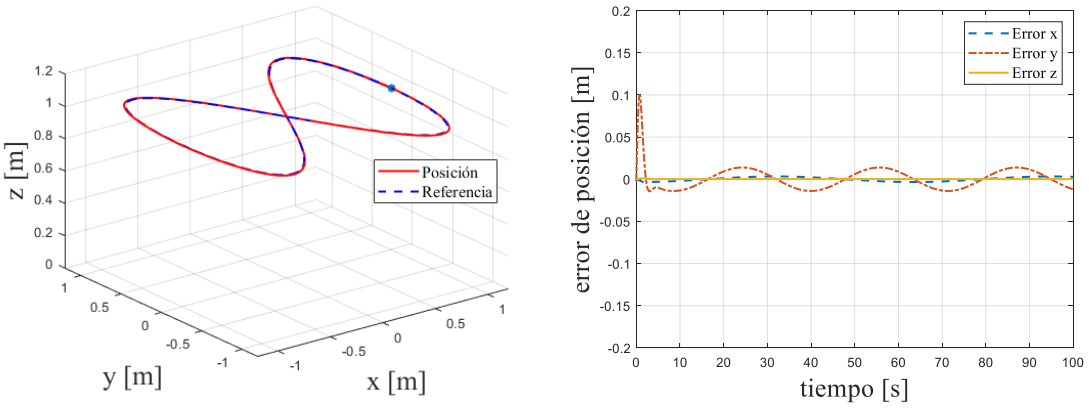


Fig. 5.20. Trayectoria del cuadri-rotor y error en la posición, aplicando el control PD basado en planitud siguiendo la trayectoria B en Webots.

Para el control de PD basado en planitud, la implementación en MATLAB muestra un mejor desempeño en el seguimiento de las dos trayectorias, aun así, el comportamiento del cuadri-rotor en Webots se considera compatible con el modelo dinámico (2.59) – (2.64), implementado en MATLAB.

5.3.4.2 Simulación control Backstepping basado en planitud

El resultado de las simulaciones en MATLAB (figura 5.21) y en Webots (figura 5.22), a se observa que en ambas se logra seguir la trayectoria A, logrando mantener la altura z y con error en x más grande en la simulación en MATLAB que en Webots.

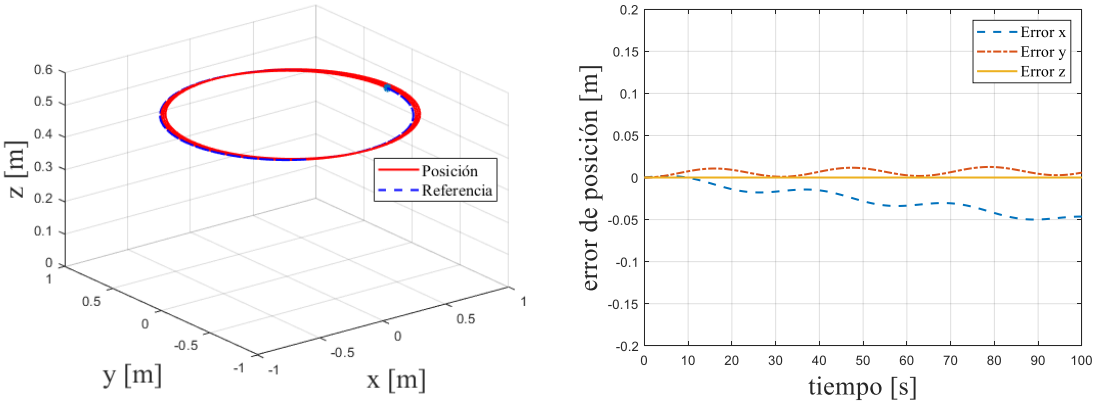


Fig. 5.21. Trayectoria del cuadri-rotor y error en la posición, aplicando el control Backstepping basado en planitud siguiendo la trayectoria A en MATLAB.

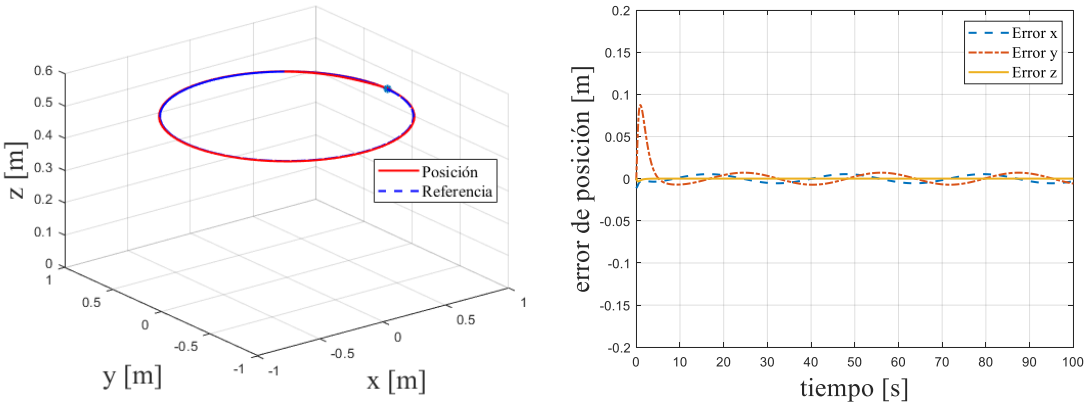


Fig. 5.22. Trayectoria del cuadri-rotor y error en la posición, aplicando el control Backstepping basado en planitud siguiendo la trayectoria A en Webots.

El resultado de las simulaciones en MATLAB (figura 5.23) y en Webots (figura 5.24), a pesar de la diferencia entre la referencia y la posición, se tiene que en ambas se logra seguir la trayectoria B, logrando mantener la altura z y con errores en x, y un poco más grandes en la simulación en MATLAB que en Webots.

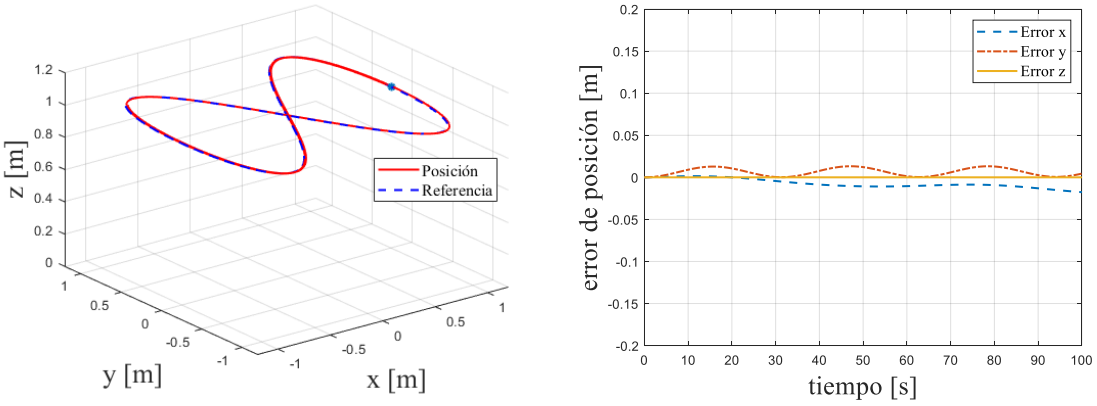


Fig. 5.23. Trayectoria del cuadri-rotor y error en la posición, aplicando el control Backstepping basado en planitud siguiendo la trayectoria B en MATLAB.

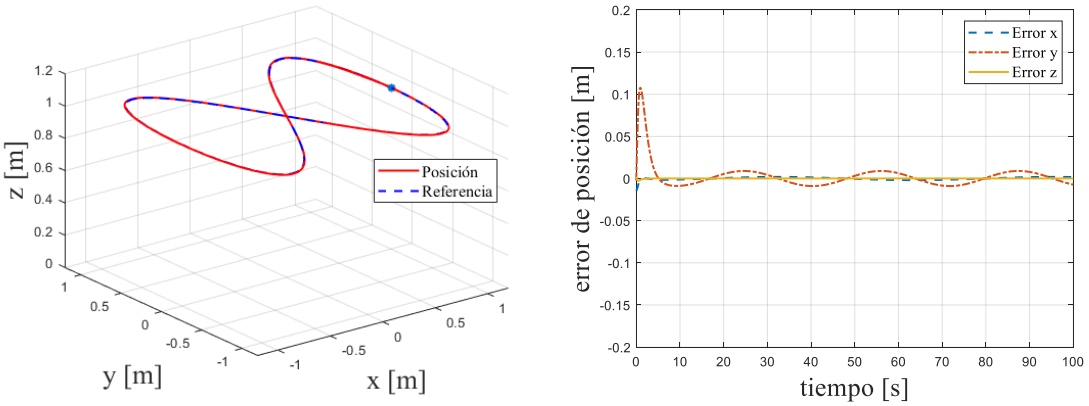


Fig. 5.24. Trayectoria del cuadri-rotor y error en la posición, aplicando el control Backstepping basado en planitud siguiendo la trayectoria B en Webots.

Para el control de Backstepping basado en planitud se muestra un mejor desempeño en la simulación en Webots, en el seguimiento de las dos trayectorias, aun así, el comportamiento del cuadri-rotor en Webots se considera compatible con el modelo dinámico (2.59) – (2.64), implementado en MATLAB.

5.3.4.3 Simulación control LQR basado en planitud

El resultado de las simulaciones en MATLAB (figura 5.25) y en Webots (figura 5.26), a pesar de la diferencia entre la referencia y la posición, se tiene que en ambas se logra seguir la trayectoria A, logrando mantener la altura z y mostrando un error en y más grande en la simulación de MATLAB que en Webots.

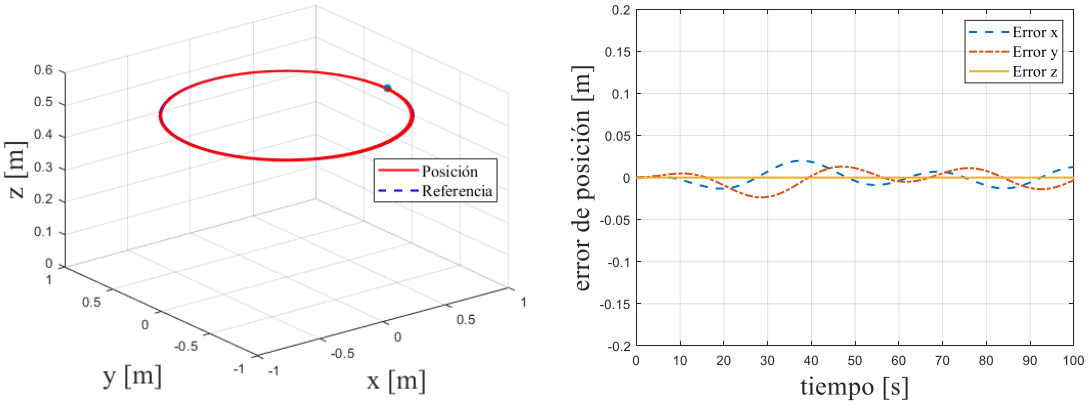


Fig. 5.25. Trayectoria del cuadri-rotor y error en la posición, aplicando el control LQR basado en planitud siguiendo la trayectoria A en MATLAB.

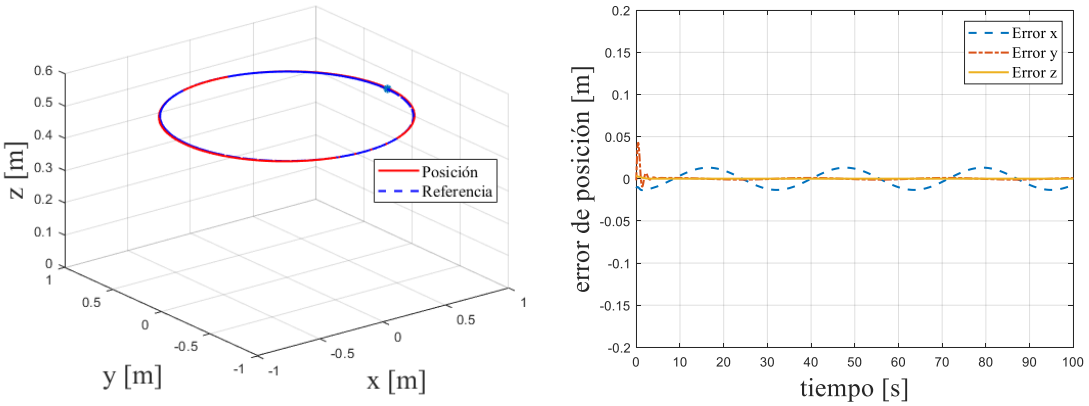


Fig. 5.26. Trayectoria del cuadri-rotor y error en la posición, aplicando el control LQR basado en planitud siguiendo la trayectoria A en Webots.

El resultado de las simulaciones en MATLAB (figura 5.27) y en Webots (figura 5.28) se tiene que en ambas se logra seguir la trayectoria B, logrando mantener la altura z y con un error en y más grande en la simulación en MATLAB que en Webots.

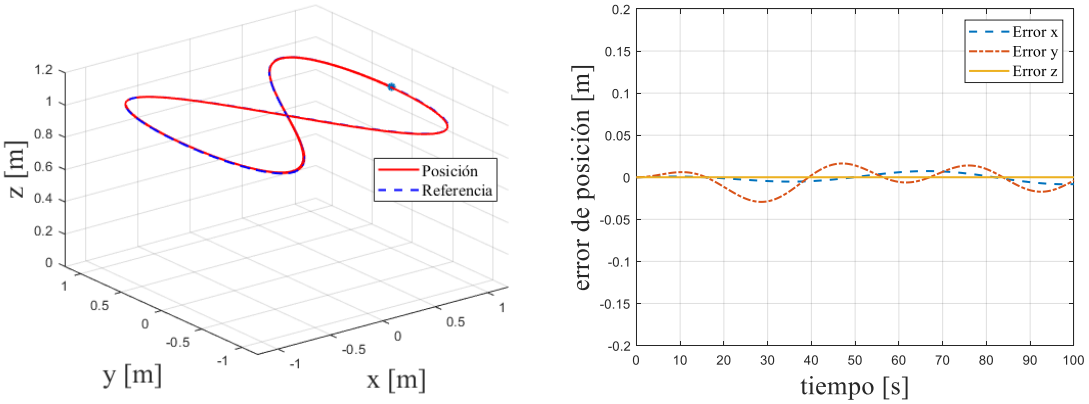


Fig. 5.27. Trayectoria del cuadri-rotor y error en la posición, aplicando el control LQR basado en planitud siguiendo la trayectoria B en MATLAB.

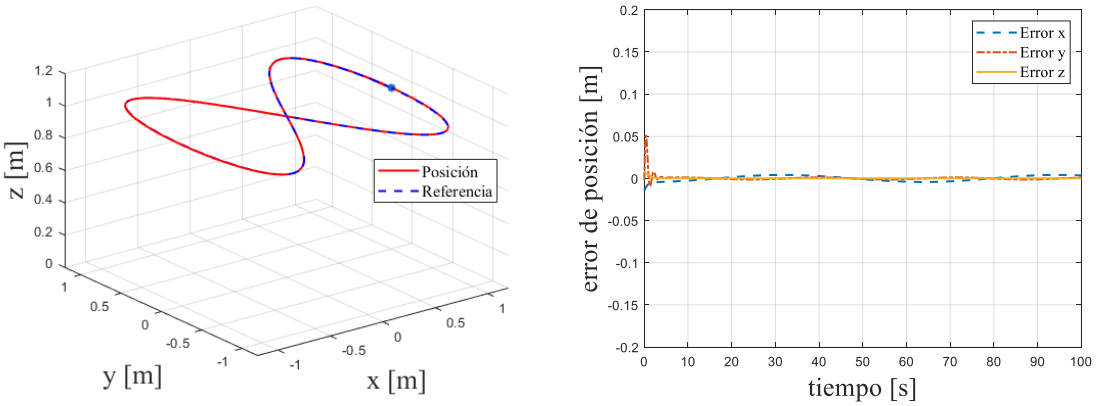


Fig. 5.28. Trayectoria del cuadri-rotor y error en la posición, aplicando el control LQR basado en planitud siguiendo la trayectoria B en Webots.

Para el control LQR basado en planitud se muestra un mejor desempeño en la simulación en Webots, en el seguimiento de las dos trayectorias, así el comportamiento del cuadri-rotor en Webots se considera compatible con el modelo dinámico (2.39) – (2.44), implementado en MATLAB.

Conclusiones

De acuerdo con los resultados presentados en este trabajo, se concluye que es posible usar técnicas de control para la estabilización de vuelo y seguimiento de trayectorias para un cuadri-rotor en el ambiente virtual de Webots. Además, la comparación de los resultados de las simulaciones en MATLAB, software ampliamente usado para estos fines, muestran que, el software Webots es perfectamente válido para los mismos propósitos; es decir, el modelo del cuadri-rotor generado en la plataforma Webots se puede considerar como un modelo aceptable que representa la dinámica del sistema con una buena aproximación.

Es importante conocer y entender a fondo el modelo dinámico del cuadri-rotor su representación no lineal, lineal y plano para su uso con las estrategias de control. También se ha estudiado el método de control basado en la dinámica plana del sistema, para generar las entradas de control de referencia que respetan la dinámica del sistema. Esta técnica es muy importante ya que se muestra una mejora significativa en el seguimiento de trayectoria en la simulación en Webots, respecto a los casos que no son prealimentados con esta entrada.

La creación del modelo dinámico del cuadri-rotor en el ambiente de simulación Webots ha sido un gran reto, ya que el software no contaba con algún modelo del cuadri-rotor, por lo cual, esta actividad llevó más tiempo de lo planeado, encontrando algunas problemáticas como el no saber cómo calcula la dinámica el software y la creación de estructuras más detalladas con los nodos de formas. El modelo dinámico del cuadri-rotor creado en Webots se ha generado a partir de elementos básicos (cilindros, prismas, propelas, motores, etc.), ya que el software no cuenta con esta información. Para esto fue necesario entender a detalle la concepción del sistema del cuadri-rotor.

Al principio de este trabajo de tesis en el software Webots no existía un Robot con las características de un cuadri-rotor de arquitectura abierta, que permitiera al usuario simular su dinámica en función de parámetros deseados; es por esta razón que en este trabajo de tesis se considera una valiosa aportación el sistema creado el cual permite al usuario probar diferentes estrategias de control. También se aporta con la creación de código en lenguaje C para la implementación de los controladores en Webots.

Se escribieron, fueron aceptados y publicados dos artículos relacionados con el tema de tesis y presentados en los SOMIXXXIV y CNCA2019 respectivamente, logrando compartir y obtener una perspectiva de los trabajos actuales en aeronaves no tripuladas.

Por lo anterior se tiene que el objetivo principal y los objetivos particulares de este trabajo de tesis se han cumplido satisfactoriamente.

Trabajo a futuro

Realizar mejoras al detallado del modelo del cuadri-rotor, el cual se presenta como una primera aproximación en camino a realizar una calibración con un modelo en el mundo real.

La generación de elementos para una biblioteca con bloques de construcción de vehículos aéreos no tripulados, por ejemplo, un Nodo que contenga un motor, una propela y que permita al usuario solo ingresar sus parámetros sin tener que iniciar desde cero.

A partir de este modelo es posible reproducir este Robot tantas veces como se desee (bajo el rendimiento de procesamiento de datos en la computadora) para realizar aplicaciones de vuelo de robots cooperativos empleando algoritmos de consenso.

A la fecha el software cuenta con una demostración de un cuadri-rotor DJI Mavic 2PRO, pero no permite hacer modificaciones ni mucho menos implementar leyes de control a gusto del usuario; es decir este modelo no es arquitectura abierta.

Apéndices

Apéndice A: Ponencias

Apéndice A.1: Congreso SOMI XXXIV

AMCA

Congreso Nacional de Control Automático | **2019**

FCE

No. Folio: CNCA2019/Mie2T3.2

Se otorga el presente

Certificado

a

Edmundo Miguel Cortes Vazquez, Amparo PALOMINO, César MARTINEZ TORRES, Gibran Etcheverry

Por su invaluable presentación del artículo "*Simulación de un Cuadri-Rotor en el Software Webots*" en el **Congreso Nacional de Control Automático 2019**, llevado a cabo del 23 al 25 de octubre, en el Centro de Convenciones de la Benemérita Universidad Autónoma de Puebla.

Ciudad Universitaria, octubre 2019

Dr. José Fermí Guerrero Castellanos

Dra. Luz del Carmen Gómez Pavón

Dr. Jesús Manuel Muñoz Pacheco

Presidentes del Congreso

Dr. Martín Velasco Villa

Presidente de la AMCA

BUAP

Facultad de Ciencias de la Electrónica

A través de la Maestría en Ciencias de la Electrónica, Opción en Automatización

otorga la presente

CONSTANCIA

al: **Ing. Edmundo Miguel Cortés Vázquez**

Por su participación en el evento:

"Simulación y Control de un Quadrotor en el Software Webots "

"Pensar bien, para vivir mejor"

H. Puebla de Zaragoza a 14 de Enero de 2020

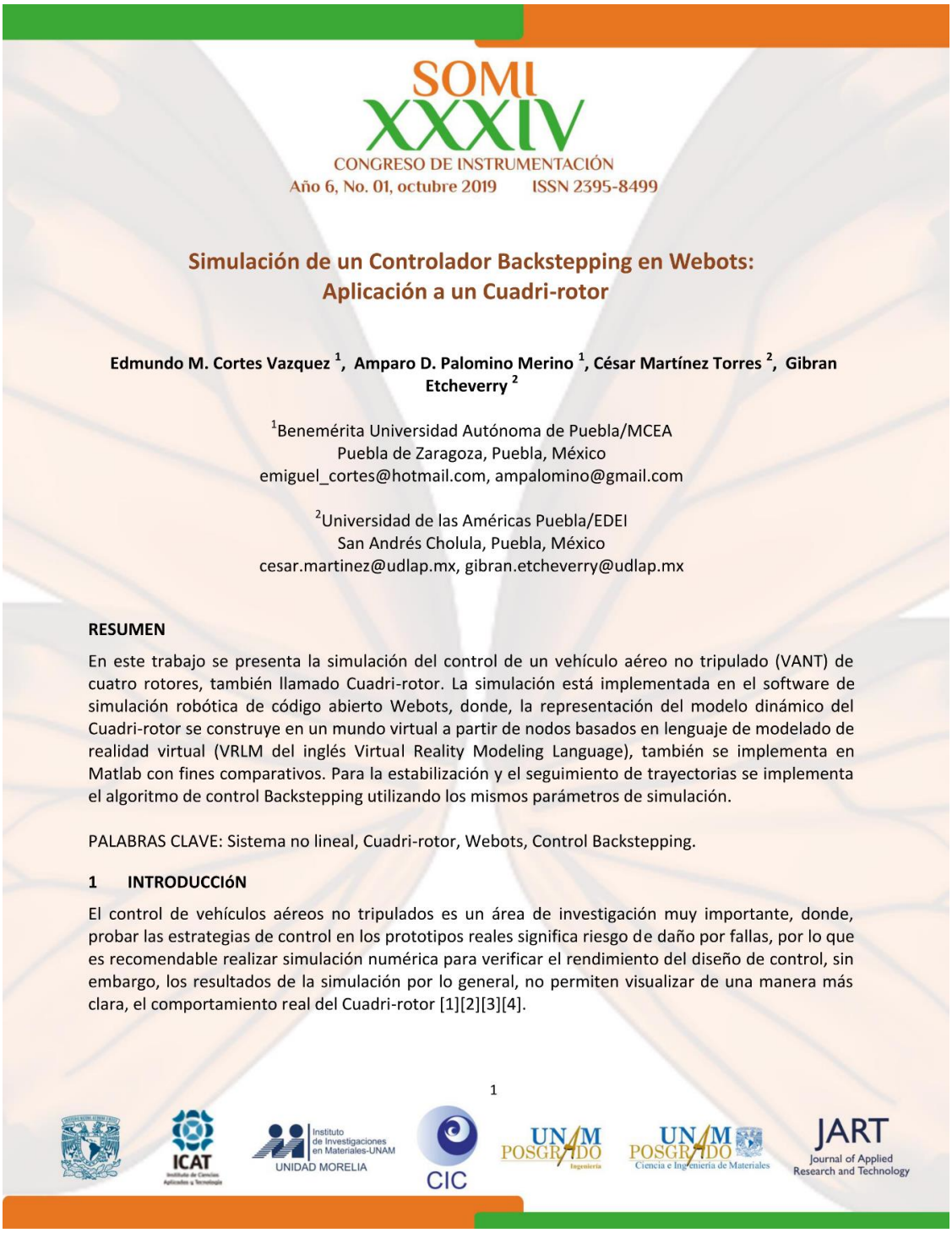
Dra. Olga Guadalupe Félix Beltrán
Coordinadora de la Maestría

Dra. Luz del Carmen Gómez Pavón
Directora de la Facultad

Apéndice B: Artículos

Apéndice B.1: Publicación de artículo 1

Se anexa el artículo publicado en SOMI Congreso de Instrumentación, publicación en formato digital con ISSN: 2395-8499.



SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

Simulación de un Controlador Backstepping en Webots: Aplicación a un Cuadri-rotor

Edmundo M. Cortes Vazquez [1], Amparo D. Palomino Merino [1], César Martínez Torres [2], Gibran Etcheverry [2]

[1]Benemérita Universidad Autónoma de Puebla/MCEA
Puebla de Zaragoza, Puebla, México
emiguel_cortes@hotmail.com, ampalomino@gmail.com

[2]Universidad de las Américas Puebla/EDEI
San Andrés Cholula, Puebla, México
cesar.martinez@udlap.mx, gibran.etcheverry@udlap.mx

RESUMEN

En este trabajo se presenta la simulación del control de un vehículo aéreo no tripulado (VANT) de cuatro rotores, también llamado Cuadri-rotor. La simulación está implementada en el software de simulación robótica de código abierto Webots, donde, la representación del modelo dinámico del Cuadri-rotor se construye en un mundo virtual a partir de nodos basados en lenguaje de modelado de realidad virtual (VRLM del inglés Virtual Reality Modeling Language), también se implementa en Matlab con fines comparativos. Para la estabilización y el seguimiento de trayectorias se implementa el algoritmo de control Backstepping utilizando los mismos parámetros de simulación.

PALABRAS CLAVE: Sistema no lineal, Cuadri-rotor, Webots, Control Backstepping.

1 INTRODUCCIóN

El control de vehículos aéreos no tripulados es un área de investigación muy importante, donde, probar las estrategias de control en los prototipos reales significa riesgo de daño por fallas, por lo que es recomendable realizar simulación numérica para verificar el rendimiento del diseño de control, sin embargo, los resultados de la simulación por lo general, no permiten visualizar de una manera más clara, el comportamiento real del Cuadri-rotor [1][2][3][4].

SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

En este trabajo, se ha desarrollado una simulación del VANT basada en el software de simulación Webots, el cual es de código abierto que proporciona un entorno de desarrollo completo para modelar, programar y simular robots. Aunque existen ambientes de simulación similares, como lo son Gazebo, V-Rep entre otros, algunos de los aspectos que hace atractivo este software es el hecho de contar con librerías estables, el soporte que ofrece la empresa y que este simulador es uno de los pocos que funcionan en multiplataforma, incluyendo Windows. Anteriormente Webots era un software comercial con licencias profesional y estudiantil, pero recientemente en diciembre de 2018 se lanza la versión de R2019a en código libre, en la cual se desarrolla este trabajo.

2 MODELO DEL CUADRI-ROTOR

Con el fin de poder representar el modelo matemático del Cuadri-rotor en Webots, dadas las restricciones que presenta el software, en este trabajo se asumieron para el desarrollo del modelo las siguientes hipótesis: la estructura es rígida y simétrica, el centro de masa y el origen de las coordenadas coinciden, las hélices son rígidas y el empuje es proporcional al cuadrado de la velocidad de las hélices [5] [6].

El funcionamiento del vehículo es sencillo, la posición ($\xi = x, y, z$) y la orientación ($\eta = \phi, \theta, \psi$) deseadas se logran al variar la velocidad y el par de los cuatro rotores [7], como se muestra en la Figura 1.

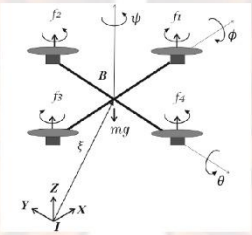


Figura 1. Posición y orientación del Cuadri-rotor.

Usando el formalismo de Euler-Lagrange se obtiene el modelo del VANT, el cual tiene seis ecuaciones que describen su dinámica y cuatro que describen las entradas de control [6] [7] [8]. Los símbolos utilizados se describen en la Tabla 1.

SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

Tabla 1. Definición de los símbolos

Símbolo	Definición	Símbolo	Definición
ϕ	ángulo de roll	b	factor de empuje de la hélice
θ	ángulo de pitch	l	distancia de los rotores al centro de la estructura
ψ	ángulo yaw	k_T	factor de arrastre de la hélice
$\Omega_1, \Omega_2, \Omega_3, \Omega_4$	velocidad de los rotores	m	masa de la estructura
I_x, I_y, I_z	momentos de inercia de la estructura	J_R	momento de inercia del rotor
Ω	velocidad total	l	distancia de los rotores al centro de la estructura

Las ecuaciones que describen el modelo dinámico son:

$$\ddot{x} = \frac{1}{m}(cos\psi sen\theta cos\phi + sen\psi sen\phi)U_1 \quad (1)$$

$$\ddot{y} = \frac{1}{m}(sen\psi sen\theta cos\phi - cos\psi sen\phi)U_1 \quad (2)$$

$$\ddot{z} = -g + \frac{1}{m}(cos\theta cos\phi)U_1 \quad (3)$$

$$\ddot{\phi} = \frac{(I_y - I_z)}{I_x}\dot{\psi}\dot{\theta} - \frac{J_R\Omega}{I_x}\dot{\theta} + \frac{l}{I_x}U_2 \quad (4)$$

$$\ddot{\theta} = \frac{(I_z - I_x)}{I_y}\dot{\psi}\dot{\phi} + \frac{J_R\Omega}{I_y}\dot{\phi} + \frac{l}{I_y}U_3 \quad (5)$$

$$\ddot{\psi} = \frac{(I_x - I_y)}{I_z}\dot{\phi}\dot{\theta} + \frac{1}{I_z}U_4 \quad (6)$$

con:

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \quad (7)$$

$$U_2 = bl(\Omega_1^2 - \Omega_3^2) \quad (8)$$

$$U_3 = bl(\Omega_4^2 - \Omega_2^2) \quad (9)$$

$$U_4 = k_T(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \quad (10)$$

$$\Omega = \Omega_1 + \Omega_2 + \Omega_3 + \Omega_4 \quad (11)$$

Año 6, No. 01, octubre 2019 ISSN 2395-8499

3 MODELO EN WEBOTS

El lenguaje utilizado en Webots es VRML, que es el lenguaje de programación utilizado para modelar el mundo virtual donde se simulará el comportamiento de los robots. Este lenguaje posibilita la descripción de objetos 3D a partir de prototipos basados en formas geométricas básicas o estructuras descritas a partir de vértices y aristas [9]. Es de gran importancia cuidar las dimensiones de los elementos creados ya que el software calcula las propiedades dinámicas de cada elemento al generar la simulación. El Cuadri-rotor se construye a partir de un nodo Robot en el árbol de escena, al cual se añaden los nodos necesarios para generar la estructura y los nodos de las propelas, además de adicionar una unidad de medición inercial (IMU del inglés Inertial Measurement Unit) y un sistema de posicionamiento global (GPS del inglés Global Positioning System), quedando como se muestra en la Figura 2.

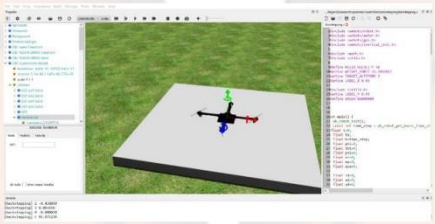


Figura 2. Interfaz gráfica de Webots.

.

4 ESTRATEGIA DE CONTROL

El control Backstepping surge como la necesidad de simplificar el desarrollo de una ley de control para sistemas de orden superior y donde el sistema puede considerarse como una conexión en cascada de subsistemas, auxiliándose en la recursividad de los mismos para reconstruir el sistema original [10][11] [12].

Para diseñar el controlador se reescribe el sistema de las ecuaciones (1)-(6), en la forma de espacio de estados $\dot{X} = f(X,U)$ [5][6][12], siendo:

SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

$$\dot{\mathbf{X}} = \mathrm{f}(\mathbf{X},\mathbf{U}) = \begin{bmatrix} x_2 \\ a_1 x_4 x_6 + a_2 x_4 \Omega + b_1 U_2 \\ x_4 \\ a_3 x_2 x_6 + a_4 x_2 \Omega + b_2 U_3 \\ x_6 \\ a_5 x_2 x_4 + b_3 U_4 \\ x_8 \\ -g + (\cos x_1 \cos x_3)\frac{U_1}{m} \\ x_{10} \\ u_x \frac{U_1}{m} \\ x_{12} \\ u_y \frac{U_1}{m} \end{bmatrix} \quad (12)$$

con:

$$a_1 = \frac{(I_{yy} - I_{zz})}{I_{xx}} \quad (13)$$

$$a_2 = -\frac{J_R}{I_{xx}} \quad (14)$$

$$a_3 = \frac{(I_{zz} - I_{xx})}{I_{yy}} \quad (15)$$

$$a_4 = \frac{J_R}{I_{yy}} \quad (16)$$

$$a_5 = \frac{(I_{xx} - I_{yy})}{I_{zz}} \quad (17)$$

$$b_1 = \frac{l}{I_{xx}} \quad (18)$$

$$b_2 = \frac{l}{I_{yy}} \quad (19)$$

$$b_3 = \frac{1}{I_{xx}} \quad (20)$$

$$u_x = \cos(x_1)\,\mathrm{sen}(x_3)\cos(x_5) + \sin(x_1)\,\mathrm{sen}(x_5) \quad (21)$$

$$u_y = \cos(x_1)\,\mathrm{sen}(x_3)\,\mathrm{sen}(x_5) - \mathrm{sen}(x_1)\cos(x_5) \quad (22)$$

SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

El sistema completo descrito por la ecuación (12), está constituido por dos subsistemas, el de rotaciones angulares que es independiente de los componentes de traslación y el de traslaciones lineales que dependen de los ángulos, controlados por la estrategia de control conforme al esquema de la Figura 3.

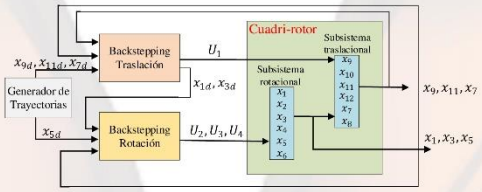


Figura 3. Esquema de control Backstepping del Cuadri-rotor.

Utilizando la técnica de Backstepping como en [5], se obtiene la ley de control que fuerza al sistema a seguir la trayectoria de referencia como se muestra en las ecuaciones (23)-(28):

$$U_2 = \frac{1}{b_1}(z_1 - a_1 x_4 x_6 - a_2 x_4 \Omega - \alpha_1(z_2 + \alpha_1 z_1) - \alpha_2 z_2) \quad (23)$$

$$U_3 = \frac{1}{b_2}(z_3 - a_3 x_2 x_6 - a_4 x_2 \Omega - \alpha_3(z_4 + \alpha_3 z_3) - \alpha_4 z_4) \quad (24)$$

$$U_4 = \frac{1}{b_3}(z_5 - a_5 x_2 x_4 - \alpha_5(z_6 + \alpha_5 z_5) - \alpha_6 z_6) \quad (25)$$

$$U_1 = \frac{m}{\cos(x_1)\cos(x_3)}(z_7 + g - \alpha_7(z_8 + \alpha_7 z_7) - \alpha_8 z_8) \quad (26)$$

$$u_x = \frac{m}{U_1}(z_9 - \alpha_9(z_{10} + \alpha_9 z_9) - \alpha_{10} z_{10}) \quad (27)$$

$$u_y = \frac{m}{U_1}(z_{11} - \alpha_{11}(z_{12} + \alpha_{11} z_{11}) - \alpha_{12} z_{12}) \quad (28)$$

Para el diseño de la ecuación (23) se define el error de seguimiento en (29), y según el teorema de Lyapunov, y considerando que la función Lyapunov $V(z_1)$ presentada en (30), es definida positiva y su derivada (31) con respecto al tiempo es semidefinida negativa:

$$z_1 = x_{1d} - x_1 \quad (29)$$

$$(z_1) = \frac{1}{2} {z_1}^2 \quad (30)$$

$$\dot{V}(z_1) = z_1(\dot{x}_{1d} - x_2) \tag{31}$$

La estabilización de z_1 se obtiene introduciendo la entrada de control virtual x_2 con $\alpha_1 > 0$:

$$x_2 = \dot{x}_{1d} + \alpha_1 z_1 \tag{32}$$

La derivada de la función de Lyapunov se describe en (33), permitiendo realizar un cambio de variable con (34):

$$\dot{V}(z_1) = -\alpha_1 {z_1}^2 \tag{33}$$

$$z_2 = x_2 - \dot{x}_{1d} - \alpha_1 z_1 \tag{34}$$

Para anular este nuevo error, se realizará un segundo paso considerando la siguiente función de Lyapunov aumentada (35) y donde su derivada temporal viene dada por (36):

$$V(z_1, z_2) = \frac{1}{2}{z_1}^2 + \frac{1}{2}{z_2}^2 \tag{35}$$

$$\dot{V}(z_1, z_2) = z_2(a_1 x_4 x_6) + a_2 x_4 \Omega + b_1 U_2 - z_2(\ddot{x}_{1d} - \alpha_1(z_2 + \alpha_1 z_1)) - z_1 z_2 - \alpha_1 {z_1}^2 \tag{36}$$

Por lo tanto, si se considera la referencia en aceleración nula ($\ddot{x}_{1d} = 0$), se obtiene la señal de control U_2 de la ecuación (23), satisfaciendo $\dot{V}(z_1, z_2) < 0$ con los valores de todas las ganancias positivas. De manera análoga se diseña para las ecuaciones (24)-(26). Si se considera u_x y u_y las orientaciones de U_1 responsables del movimiento a través de los ejes x e y, se puede obtener de (21)-(22) los ángulos de balanceo, ϕ, y de cabeceo, θ, necesarios para calcular las señales de control u_x y u_y (27)-(28).

5 SIMULACIÓN Y RESULTADOS

Con el fin de visualizar y tener un punto de comparación de los resultados arrojados por Webots, se realizan las simulaciones tanto en Webots como en Matlab, considerando los mismos parámetros en ambos los cuales se muestran en la Tabla 2; en Matlab se simulo el modelo dinámico del Cuadri-rotor a partir de bloques en Simulink del modelo dinámico descrito por (1)-(6), y en ambos entornos de simulación el controlador es el de las ecuaciones (23)-(28). La trayectoria deseada se describe con las coordenadas x, y, z (posiciones lineales) y ψ (posición angular en yaw).

SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

Tabla 2. Parámetros de simulación

Parámetro	Valor	Parámetro	Valor	Parámetro	Valor	Parámetro	Valor
I_x, I_y, I_z	$0.001 Nms$	m	1.2 kg	a_4	0.1	α_3	4000
b	$0.0001 Ns$	J_R	$0.0001 Nm$	b_1, b_2, b_3	350	α_5	4000
l	0.35 m	a_1, a_3, a_5	0	α_1	5000	$\alpha_7, \alpha_9, \alpha_{11}$	4
k_T	$0.0001\ ms$	a_2	-0.1	$\alpha_2, \alpha_4, \alpha_6$	0.1	$\alpha_8, \alpha_{10}, \alpha_1$	1

La primera trayectoria es un escalón unitario en la coordenada z_d, y para las demás coordenadas deseadas x_d, y_d, ψ_d en cero como se muestra en la Tabla 3. En la figura 4 se muestra el resultado de la simulación en Webots y en la figura 5 se muestra el resultado de la simulación en Matlab; en ambas simulaciones se sigue la trayectoria deseada y los errores en la posición son similares, llegando y manteniéndose en la posición cerca de los 5s.

Tabla 3. Trayectoria Escalón

Posición deseada	Valor
x_d	0 m
y_d	0 m
z_d	1 m
ψ_d	0 rad

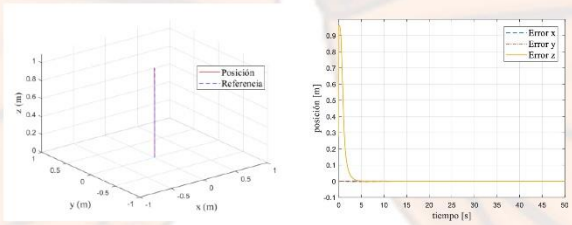


Figura 4. Posición y Error de posición de la simulación en Webots para la trayectoria escalón.

SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

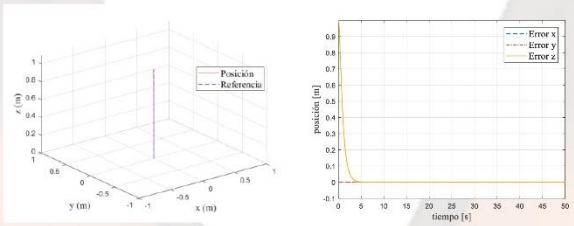


Figura 5. Posición y Error de posición de la simulación en Matlab para la trayectoria escalón.

La segunda trayectoria es una espiral ascendente, con las coordenadas deseadas como se observa en la Tabla 4. En la figura 6 se muestra el resultado de la simulación en Webots y en la figura 7 se muestra el resultado de la simulación en Matlab, en ambas simulaciones se sigue la trayectoria deseada y respecto a los errores en la posición se nota una diferencia significativa, ya que en Webots alcanza en 5s a la trayectoria y en Matlab cerca de los 30s. Esto se debe a como se había mencionado anteriormente que el modelo dinámico en Webots no es introducido mediante ecuaciones, sino se calcula en base a las dimensiones y propiedades de las estructuras creadas.

Tabla 4 Trayectoria Espiral

Posición deseada	Valor
x_d	$0.5\cos(0.3t)$m
y_d	$0.5sen(0.3t)$m
z_d	0.1t m
ψ_d	$\pi/3$ rad

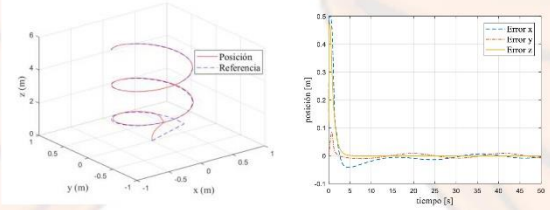


Figura 6. Posición y Error de posición de la simulación en Webots para la trayectoria espiral.

SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

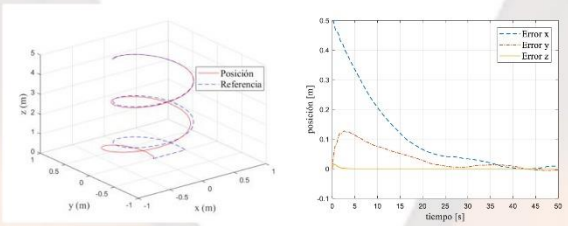


Figura 7. Posición y Error de posición de la simulación en Matlab para la trayectoria espiral.

La tercera trayectoria es una lemniscata, con las expresiones de las coordenadas deseadas como se observa en la Tabla 5. En la figura 8 se muestra el resultado de la simulación de en Webots y en la figura 9 se muestra el resultado de la simulación en Matlab, en ambas simulaciones se sigue la trayectoria deseada cerca de los 30 segundos y en los errores en la posición se nota poca diferencia entre ellas. La simulación de Matlab presenta algunas oscilaciones, esto se puede remediar cambiando ajustando un poco las ganancias del controlador, pero, para fines comparativos se utilizan las mismas en todas las simulaciones.

Tabla 5. Trayectoria Lemniscata

Posición deseada	Valor
x_d	$0.3\frac{\sqrt{2}\cos(0.2t)}{sen^2(0.2t)+1}$ m
y_d	$0.3\frac{\sqrt{2}\cos(0.2t)\,\mathrm{sen}(0.2t)}{sen^2(0.2t)+1}$ m
z_d	0.5 m
ψ_d	0 rad

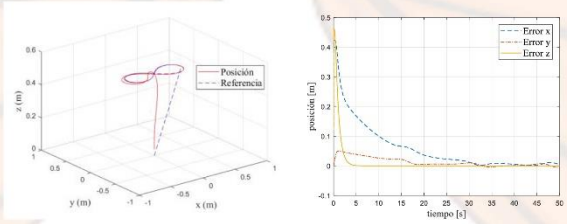


Figura 8. Posición y Error de posición de la simulación en Webots para la trayectoria Lemniscata.

SOMI XXXIV
CONGRESO DE INSTRUMENTACIÓN
Año 6, No. 01, octubre 2019 ISSN 2395-8499

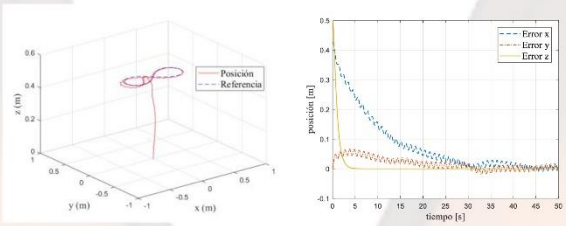


Figura 9. Posición y Error de posición de la simulación en Matlab para la trayectoria Lemniscata.

Se calcula el índice de desempeño mediante la integral de valor absoluto del error para cada una de las simulaciones, además se realiza una nueva simulación en Webots modificando la resolución de la IMU a 0.001 rad ya que de manera predeterminada los sensores en Webots son ideales.

La ecuación del índice de desempeño IVAE es:

$$IVAE = \int_0^T |e(t)|\, dt \tag{37}$$

En la Tabla 6 se muestra la evaluación comparativa del índice de desempeño del algoritmo de control variando la trayectoria en ambos ambientes de simulación y modificando la resolución de la IMU en Webots.

Tabla 6. Índices de desempeño

Trayectoria	Matlab	Webots	Webots IMU
Escalón	1.0005	1.0824	1.8595
Espiral	5.6475	1.4937	2.8587
Lemniscata	4.3126	3.2162	2.4942

Mientras el índice de desempeño sea más pequeño, representa un mejor resultado, por lo que las simulaciones en Webots en general presentan un mejor desempeño contra las de Matlab, al modificar la resolución de la IMU a 0.001 rad el desempeño baja, pero aún sigue la trayectoria deseada como se observa en la Figura 10.

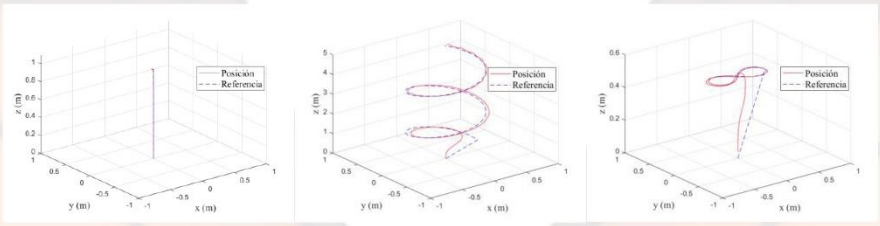


Figura 10. Resultados de las simulaciones en Webots cambiando la resolución de la IMU, de las trayectorias Escalón, Espiral y Lemniscata.

6 CONCLUSIONES

Al lograr implementar la simulación en Webots del Cuadri-rotor con el control Backstepping y compararla con la simulación en Matlab, se logra verificar que la simulación en Webots realizada en este trabajo, entrega resultados satisfactorios, generando con esto, la confianza de usar el ambiente de simulación Webots, para la evaluación de estrategias de control en este vehículo. Además, se tiene que es posible modificar los parámetros de los sensores para igualarlos con las de un sensor real, algo que en Matlab es complicado de realizar. Con lo obtenido se tiene que es posible adecuar la simulación en Webots para que funcione como un laboratorio virtual para la prueba de algoritmos de control.

REFERENCIAS

[1] Nguyen, M., Nguyen, T., Sheng, Y. The Quadrotor MAV System using PID Control. Proceedings of IEEE International Conference on Mechatronics and Automation. Beijín, China, Agosto 2-5, 2015.
[2] Parra, M., Feitosa, E., Alves, J. Mathematical Modeling and Control of a Four-Engine Helicopter. Scientia et Technica 2013; 18: 672-781.
[3] Shaik, M., Whidborne, J. Robust Sliding Mode Control of a Quadrotor. Proceedings of UKACC 11th International Conference on Control. Belfast, UK, Agosto 31- Septiembre 2, 2016.
[4] Salih, A., Moghavvemi, M. Mohamed, H. et al. Modelling and PID Controller Design for a Quadrotor Unmanned Air Vehicle, Proceedings of IEEE International Conference on Automation, Quality and Testing, Robotics. Cluj-Napoca, Rumania, Mayo 28-30, 2010.
[5] Bouabdallah, S. Design and control of quadrotors with application to autonomous flying. Tesis Doctoral, Escuela Politécnica Federal de Lausana, Suiza, Febrero 2007.
[6] Vianna, G. Modelado y Control de un Helicóptero Quadrotor. Tesis de Maestría, Universidad de Sevilla, Dept. Ing. Sistemas y Automática, Sevilla, España, Diciembre 2007.

Año 6, No. 01, octubre 2019 ISSN 2395-8499

[7] Bouabdallah, S., Murrieri, P., y Siegwart, R. Towards Autonomous Indoor Micro VTOL, Autonomous Robots 18. Paises Bajos: Springer 2005: pp. 171-183.
[8] Zulu, A., John S. A Review of Control Algorithms for Autonomous Quadrotors. Open Journal of Applied Sciences 2014; 4: 547-556.
[9] Cyberbotics: Webots Reference Manual. Disponible en https://www.cyberbotics.com (Accedido en: Mayo 15, 2019).
[10] Márquez, H. Nonlinear Control Systems: Analysis and Desing. New Jersey: Jhon Wiley & Sons 2003.
[11] Krstic, M., Kanellakopoulos, I., Kokotovic, P. Nonlinear and Adaptive Control Desing. New York: Jhon Wiley & Sons 1995.
[12] Bautista, I. Análisis y Comparación de Rendimiento en Algoritmos de Control para Seguimiento de Trayectorias Aplicados en un Drone Cuadricóptero. Tesis de Maestría, Instituto Politécnico Nacional, México, Diciembre 2006.

Apéndice B.2 Publicación de artículo 2

Se anexa el artículo publicado en Memorias del Congreso Nacional Automático, publicación en formato digital con ISSN: 2594-2492.

AMCA

Memorias del Congreso Nacional de Control Automático
ISSN: 2594-2492

Simulación de un Cuadri-rotor en el Software Webots

Edmundo M. Cortes Vazquez* Amparo D. Palomino Merino*
César Martínez Torres Gibran Etcheverry****

** Benemérita Universidad Autónoma de Puebla, Puebla de Zaragoza, Puebla, México*
(e-mail: emiguel_cortes@hotmail.com, ampalomino@gmail.com).
*** Universidad de las Américas Puebla, San Andrés Cholula, Puebla, México*
(e-mail: cesar.martinez@udlap.mx, gibran.etcheverry@udlap.mx)

Resumen: En este trabajo se presenta la simulación de un Cuadri-rotor en el software de simulación robótica de código abierto Webots, con el objetivo de verificar el desempeño de un algoritmo de control para seguimiento de trayectorias. El modelo del Cuadri-rotor se construye en el mundo virtual a partir de nodos basados en lenguaje de modelado de realidad virtual (VRLM). Para verificar la simulación en Webots se utiliza el algoritmo de control Proporcional Derivativo (PD) usando los mismos parámetros que una simulación en Matlab. Adicionalmente se modifica la resolución del sensor de posicionamiento global(GPS) y de la unidad de medición inercial (IMU) para observar su influencia en el seguimiento de trayectoria.

Palabras clave: Cuadri-rotor, Webots, Control PD, Simulación 3D, Seguimiento de trayectoria, Simulador Código abierto.

1. INTRODUCCIÓN

La investigación sobre el control de vehículos aéreos no tripulados (VANT), en particular de cuatro rotores, también conocido como Cuadri-rotor, figura 1, es una importante área de investigación de los vehículos aéreos no tripulados, donde probar las estrategias de control en los prototipos reales significa riesgo de daño por fallas, por lo que es de lo más recomendable realizar simulación numérica para verificar el rendimiento del diseño de control; sin embargo, los resultados de la simulación por lo general, no permiten visualizar de una manera más clara, el comportamiento real del Cuadri-rotor.

Los entornos de simulación son siempre una herramienta importante en investigación, es de gran importancia el contar con un instrumento que permita simular condiciones del mundo real, así como probar previamente el uso de sensores que pudieran ser de alto costo antes de implementarlos en el prototipo. En el área es comúnmente utilizada la simulación en Matlab como en Peng y Qingbo (2011) y en Paiva et al (2015), en estas simulaciones sería complicado dotar al Cuadri-rotor con sensores. En Bouabdallah et al. (2005) se ocupa una simulación en Webots para validar un PID, desde esa fecha el software se ha dotado de mejoras en la simulación de la dinámica y la integración de sensores.

En este trabajo, se ha desarrollado una simulación del Cuadri-rotor basada en el software Webots. Podemos ver que en Cyberbotics (2018a) Webots es un simulador de robótica de código abierto que proporciona un entorno de desarrollo completo para modelar, programar y simular robots. Aunque existen ambientes de simulación similares, como lo son Gazebo, V-Rep entre otros, existen varios aspectos que hacen atractivo este software, por ejemplo, la programación se puede realizar en el lenguaje que se encuentre familiarizado el usuario (C / C ++, Java, Python y *MATLAB*), además este simulador es uno de los pocos que funcionan en multiplataforma, incluyendo Windows. Webots sigue en continua revisión y desarrollo, anteriormente era un software comercial con licencia profesional y estudiantil, pero desde diciembre de 2018 se lanza la versión R2019a en código libre, en la cual se desarrolla este trabajo.

Fig. 1. Vehículo aéreo no tripulado tipo Cuadri-rotor.

2. MODELO DEL CUADRI-ROTOR

En este trabajo se asumieron las siguientes hipótesis para el desarrollo del modelo, la estructura se supone rígida y simétrica, el centro de masa y el origen de las coordenadas coinciden, las hélices son rígidas, el empuje y la resistencia al avance son proporcionales al cuadrado de la velocidad de las hélices, el modelo se obtiene de la misma manera que en Bouabdallah (2007) y Vianna (2007).

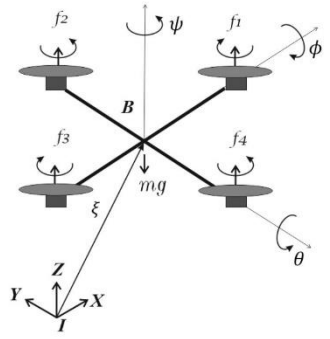


Fig. 2. Posición y orientación del Cuadri-rotor.

El funcionamiento del Cuadri-rotor es sencillo. La posición ($\xi = x, y, z$) y la orientación ($\eta = \phi, \theta, \psi$) deseada se logra al variar la velocidad y par de los cuatro rotores, como se muestra en la figura 2 y en Bouabdallah et al (2005).

Usando el formalismo de Euler-Lagrange se obtiene el modelo del Cuadri-rotor, el cual tiene seis ecuaciones que describen su dinámica, como en Zulu y John (2014) y Bouabdallah (2007):

$$\ddot{x} = \frac{1}{m}(cos\psi sen\theta cos\phi + sen\psi sen\phi)U_1 \quad (1)$$

$$\ddot{y} = \frac{1}{m}(sen\psi sen\theta cos\phi - cos\psi sen\phi)U_1 \quad (2)$$

$$\ddot{z} = -g + \frac{1}{m}(cos\theta cos\phi)U_1 \quad (3)$$

$$\ddot{\phi} = \frac{(I_y - I_z)}{I_x}\dot{\psi}\dot{\theta} - \frac{J_R\Omega}{I_x}\dot{\theta} + \frac{l}{I_x}U_2 \quad (4)$$

$$\ddot{\theta} = \frac{(I_z - I_x)}{I_y}\dot{\psi}\dot{\phi} + \frac{J_R\Omega}{I_y}\dot{\phi} + \frac{l}{I_y}U_3 \quad (5)$$

$$\ddot{\psi} = \frac{(I_x - I_y)}{I_z}\dot{\phi}\dot{\theta} + \frac{1}{I_z}U_4 \quad (6)$$

Donde U_1 es el empuje total de la suma de cada uno de los empujes f_1, f_2, f_3 y f_4. En términos de las velocidades de los rotores:

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \quad (7)$$

Y los pares U_2, U_3, U_4 que representan las entradas al sistema.

$$U_2 = bl(\Omega_4^2 - \Omega_2^2) \quad (8)$$

$$U_3 = bl(\Omega_1^2 - \Omega_3^2) \quad (9)$$

$$U_4 = k_T(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \quad (10)$$

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \quad (11)$$

Tabla 1. Definición de los símbolos

Símbolo	Definición
ϕ	ángulo de roll
θ	ángulo de pitch
ψ	ángulo yaw
Ω	velocidad general de los rotores
I_x, I_y, I_z	inercias de la estructura
J_R	inercia del rotor
b	factor de empuje
l	distancia de los rotores al centro de la estructura
k_T	factor de arrastre
m	masa de la estructura

3. MODELO EN WEBOTS

El lenguaje utilizado en Webots es el de modelado de realidad virtual (VRML de sus siglas en inglés de Virtual Reality Modeling Language), es el lenguaje de programación utilizado para modelar el mundo virtual donde se simulará el comportamiento de los robots. Este lenguaje posibilita la descripción de objetos 3D a partir de prototipos basados en formas geométricas básicas o estructuras descritas a partir de vértices y aristas como se explica en Cyberbotics (2018b).

La figura 3 muestra la interfaz gráfica del software de simulación Webots que está conformada por:

- El Árbol de escenas (izquierda): Muestra la información de describe un mundo simulado en forma jerárquica. El árbol de escenas esta estructurado como un archivo VRML97, compuesto por una lista de nodos y cada nodo contiene campos.
- Ventana 3D (centro): Permite interactuar con la simulación 3D. En esta ventana se puede, navegar en

la escena, mover un objeto, aplicar fuerza a un objeto y aplicar torque a un objeto.

- Editor de texto (derecha): Es un editor de texto de múltiples pestañas que permite desarrollar controladores. El editor presenta resaltado de sintaxis para el lenguaje compatible con Webots.

- Consola (abajo): Muestra mensajes de compilación y salidas de ejecución.

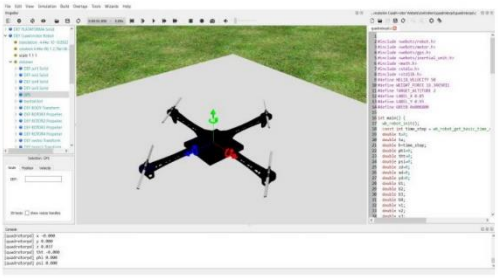


Fig. 3. Interfaz gráfica de Webots.

El Cuadri-rotor se construye a partir de un Nodo *Robot* en el árbol de escena como se muestra en la figura 4, en *children* del Nodo *Robot* como se ve en la figura 5, se añaden los Nodos *Transform* para generar la estructura, agregando formas geométricas para las carcasas de los motores y para mejorar el detalle de los brazos de la aeronave se importa un archivo VRLM97 de los brazos a *Transform*. Para que le sea posible sustentarse se agregan los nodos *Propeller* como se muestra en la figura 6, donde se agregan las características de las propelas como empuje y arrastre.

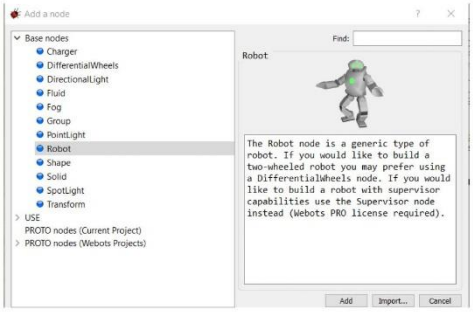


Fig. 4. Nodo *Robot* en Webots.

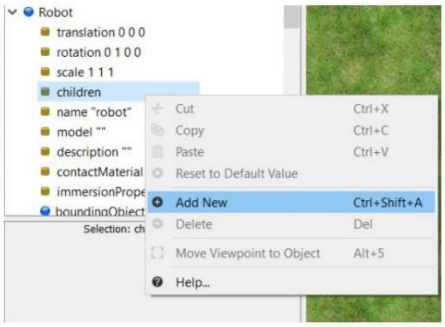


Fig. 5. Pestaña *children* del nodo *Robot*.

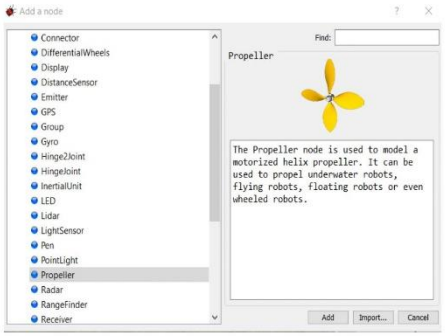


Fig. 6. Nodos admitidos por Nodo *Robot*. Los Nodos *GPS*, *InertialUnit* y *Propeller* son utilizados en el Cuadri-rotor.

Además de adicionar los sensores *GPS* y *InertialUnit* mostrados en la lista de nodos de la figura 6, colocándolos en el centro geométrico del vehículo, quedando terminado el Cuadri-rotor como se ve en la figura 7 y al centro de la figura 3. En los sensores en Webots es posible modificar algunas de sus características como lo es la resolución, el ruido, el ángulo de detección, tiempo de muestreo entre algunas otras. Adicionalmente en este trabajo se plantea la interesante opción de modificar la resolución con la idea de poder probar el comportamiento del Cuadri-rotor a una resolución de un sensor comercial.

Es de gran importancia cuidar las dimensiones de los elementos creados y propiedades de material ya que el software calcula sus centros de masa y los momentos de inercias de cada elemento para generar la simulación dinámica.

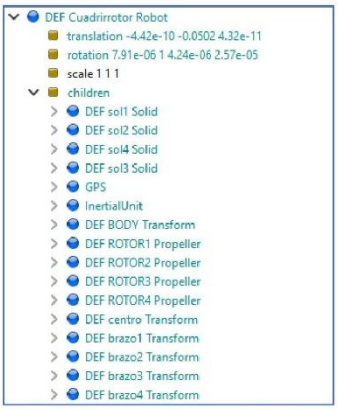


Fig. 7. Árbol de nodos del Cuadri-rotor en Webots.

La implementación del controlador para este trabajo se realiza en código C, en el editor de texto de Webots se programa siguiendo el diagrama de flujo de la figura 8, para después ser ligado al Nodo *Robot* del Cuadri-rotor.

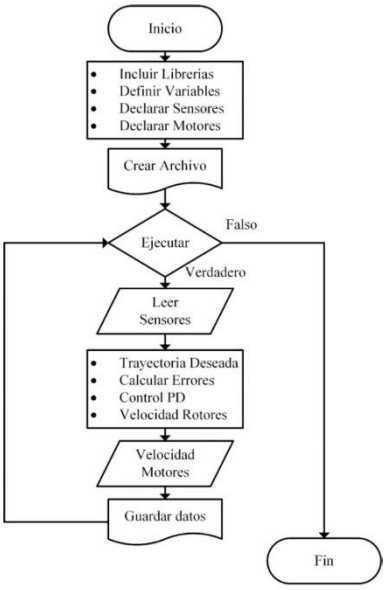


Fig. 8. Diagrama de flujo del Controlador en Webots.

4. ESTRATEGIA DE CONTROL

Para probar que el modelo creado en Webots funciona en congruencia al modelo simulado en Matlab, en este trabajo se utiliza un controlador Proporcional Derivativo (PD).

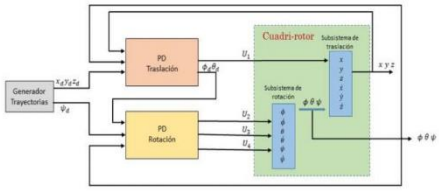


Fig. 9. Diagrama a bloques de la estrategia de control PD.

La implementación del control PD, siguiendo el esquema de la figura 9, puede ser definida por:

$$U = K_p e(t) + K_d \frac{d}{dt} e(t) \tag{12}$$

donde K_p es la ganancia proporcional y K_d es la ganancia derivativa, $e(t)$ es la diferencia entre el valor deseado y el valor medido.

Las entradas de control U_2, U_3, U_4 se tienen similar a Dikmen et al (2009) con:

$$U_2 = K_{p,\phi}(d_\phi - \phi) + K_{d,\phi}(d_{\dot{\phi}} - \dot{\phi}) \tag{13}$$

$$U_3 = K_{p,\theta}(d_\theta - \theta) + K_{d,\theta}(d_{\dot{\theta}} - \dot{\theta}) \tag{14}$$

$$U_4 = K_{p,\psi}(d_\psi - \psi) + K_{d,\psi}(d_{\dot{\psi}} - \dot{\psi}) \tag{15}$$

Las ecuaciones (13)-(15) representan los torques para cada eje debido al empuje de los rotores U_2, U_3 y U_4 respectivamente y con $K_{p,\phi}$, $K_{p,\phi}$ y $K_{p,\psi}$ que son las ganancias proporcionales. $K_{d,\phi}$, $K_{d,\theta}$ y $K_{d,\psi}$ son las ganancias derivativas. d_ϕ, d_θ y d_ψ son pitch, roll y yaw deseados, ϕ, θ y ψ corresponden a los valores actuales de orientación.

Las ecuaciones (1)-(3) se pueden resolver para determinar las variables de control:

$$U_1 = \frac{1}{cos\theta cos\phi}(z_3 + mg) \tag{16}$$

$$d_\phi = arcsen\left(\frac{m}{U_1}(z_1 sen\psi - z_2 cos\psi)\right) \tag{17}$$

$$d_\theta = arcsen\left(\frac{m}{U_1}(z_1 cos\psi + z_2 sen\psi)\right) \quad (18)$$

$$z_1 = K_{px}(d_x - x) + K_{d,x}(d_{\dot{x}} - \dot{x}) \quad (19)$$

$$z_2 = K_{py}(d_y - y) + K_{dy}(d_{\dot{y}} - \dot{y}) \quad (20)$$

$$z_3 = K_{pz}(d_z - z) + K_{dz}(d_{\dot{z}} - \dot{z}) \quad (21)$$

Considerando la posición deseada d_x, d_y y d_z son los valores conocidos, mientras que x, y, z son los valores medidos.

Donde (16) es la fuerza de control del desplazamiento vertical z, (17) y (18) son los ángulos deseados para colocar el cuadrirrotor en x, y deseadas del plano cartesiano, z_1, z_2 y z_3 son las señales de salidas de control.

5. SIMULACION Y RESULTADOS

Las simulaciones se realizan con los mismos parámetros en Webots y en Matlab; en Matlab está construido el modelo del Cuadri-rotor a partir de bloques en Simulink del modelo dinámico descrito por (1) a (6).

Tabla 2. Parámetros de simulación

Parámetro	Valor	Parámetro	Valor
I_x, I_y, I_z	0.001 Nms^2	d_ψ	$\pi/3$ rad
b	0.0001 Ns^2	d_z	0.1 tm
l	0.35 m	m	1.2 kg
k_T	0.0001 Nms^2	$K_{p,\phi}, K_{p,\theta}, K_{p,\psi}$	1
g	9.81 ms^{-2}	$K_{d,\phi}$,$K_{d,\theta}$,$K_{d,\psi}$	0.3
J_R	0.0001 Nms^2	$K_{p,x}$,$K_{p,y}$, $K_{p,z}$	1
d_x	$0.5\cos(0.2t)$ m	$K_{d,x}$,$K_{d,y}$, $K_{d,z}$	0.8
d_y	$0.5sen(0.2t)$ m		

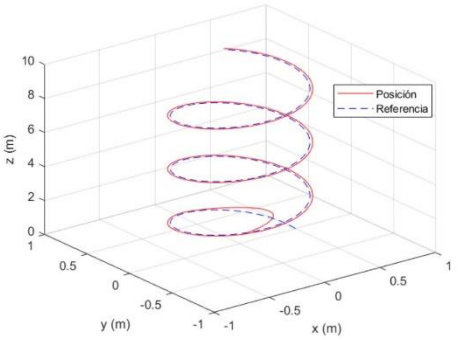


Fig. 10. Variables de posición en trayectoria 3D en Matlab.

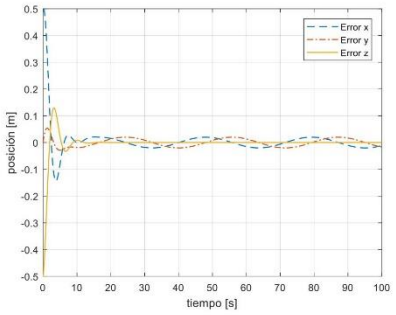


Fig. 11. Errores de posición en seguimiento de trayectoria en Matlab.

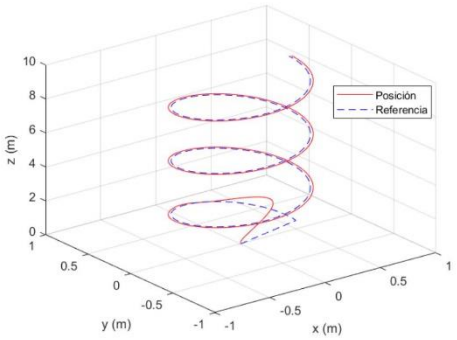


Fig. 12. Variables de posición en trayectoria 3D en Webots.

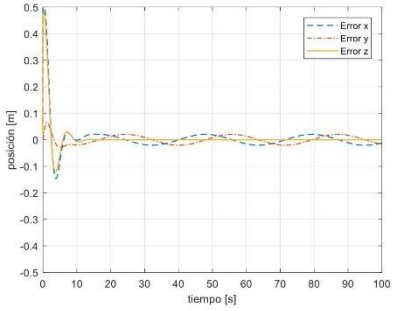


Fig. 13. Errores de posición en seguimiento de trayectoria en Webots.

La simulación en Matlab; del seguimiento de la trayectoria se observa en la figura 10 en línea puntuada la trayectoria deseada y en solido la trayectoria recorrida, se observa que Cuadri-rotor es capaz de seguir la trayectoria deseada. El error de las posiciones está acotado, esto se observa en la figura 11.

La simulación en Webots; del seguimiento de la trayectoria se observa en la figura 12 en línea puntuada la trayectoria deseada y en solido la trayectoria recorrida, se observa que Cuadri-rotor es capaz de seguir la trayectoria deseada y el error en las posiciones en la figura 13 donde se observa que el error es muy similar a la simulación en Matlab.

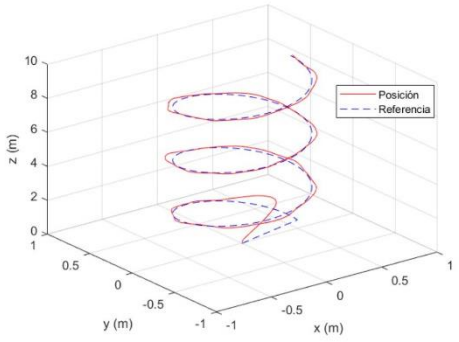


Fig. 14. Variables de posición en trayectoria 3D en Webots modificando la resolución de los sensores.

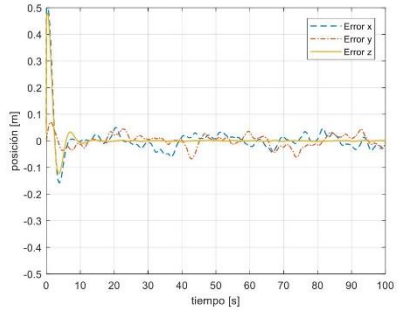


Fig. 15. Errores de posición en seguimiento de trayectoria en Webots, con resolución modificada.

También se realiza la simulación en Webots modificando la resolución de la IMU y del GPS 0.001rad y 0.001 m respectivamente, debido a que en la anterior están considerados como instrumentos ideales, con una resolución máxima y sin ruido en Webots. En la figura 14 en línea puntuada la trayectoria deseada y en solido la trayectoria recorrida y el error en las posiciones en la figura 15 donde se observa que el error es mayor a la simulación anterior, pero ejemplificando que es posible modificar los parámetros de los sensores y observar que para esta resolución logra seguir la trayectoria el control PD, pero, aumenta el error de posición.

6. CONCLUSIONES

Al implementar la simulación en Webots del Cuadri-rotor con el control PD y compararla con la simulación en Matlab, se logra verificar que la simulación en Webots realizada en este trabajo, entrega resultados satisfactorios, generando con esto, la confianza de usar el ambiente de simulación Webots, para la evaluación de estrategias de control. Como trabajo futuro se planea adecuar la simulación en Webots para que funcione como un laboratorio virtual para la prueba de algoritmos de control.

REFERENCIAS

Bouabdallah, S. (2007). Design and Control of Quadrotor with Application to Autonomus Flying. Ph.D. Dissertation, Lausanne Polytechnic University. Suiza.

Bouabdallah, S., Murrieri, P., y Siegwart, R. (2005). Towards Autonomous Indoor Micro VTOL. 18(2), 171-183.

Cyberbotics. (2018a). Webots User Guide. Lausana, Suiza. Disponible en https://www.cyberbotics.com.

Cyberbotics. (2018b). Webots Reference Manual. Lausana, Suiza. Disponible en https://www.cyberbotics.com.

Dikmen, I., Arısoy, A., y Temelta, H. (2009). Attitude control of a quadrotor. 4th International Conference on Recent Advances in Space Technologies, pp. 722–727. Turquia.

Paiva, E., Soto, J., Salinas, J., y Ipanequé. W. (2015). Modeling and PID Cascate Control of a Quadcopter for Trajectory Tracking. Conference on Electrical, Electronic Engineering, Information and Communication Technologies, 809-815. Chile.

Peng, L., Qingbo, G. (2011). Real-Time Simulation System for UAV Based on Matlab/Simulink. Conference on Computing, Control and Industrial Engineering. China.

Vianna, G. (2007). Modelado y Control de un Helicoptero Quadrotor. Tesis M.S., Dept. Ingenieria de Sistemas y Automática, Universidad de Sevilla, Sevilla.

Zulu, A., John S. (2014). A Review of Control Algorithms for Autonomous Quadrotors. Open Journal of Applied Sciences, 4, 547-556.

Apéndice C: Programas en MATLAB

C.1: Control PD

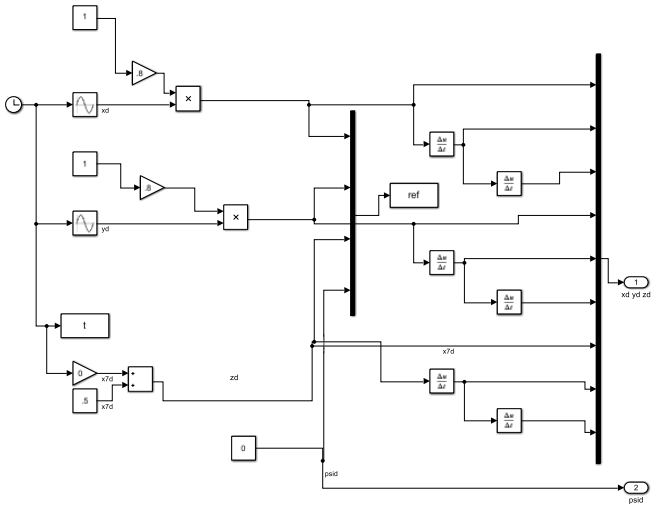


Fig. C.1.1. Generador de Trayectorias.

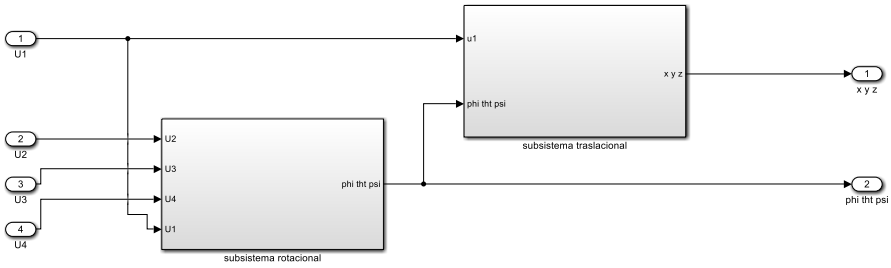


Fig. C.1.2. Cuadri-rotor.

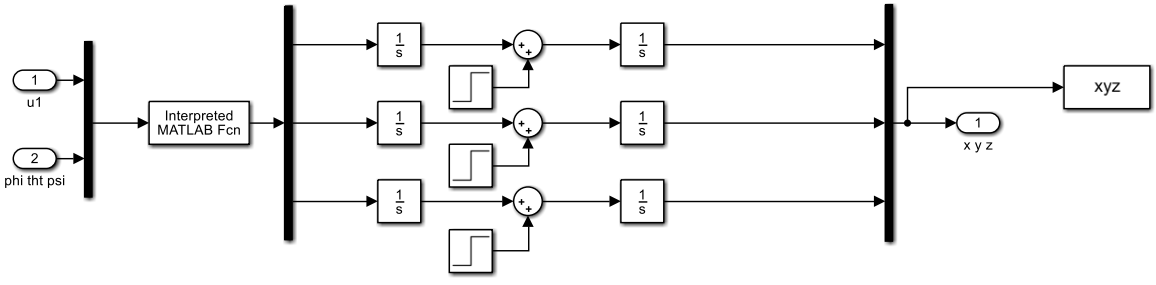


Fig. C.1.3. Subsistema traslacional.

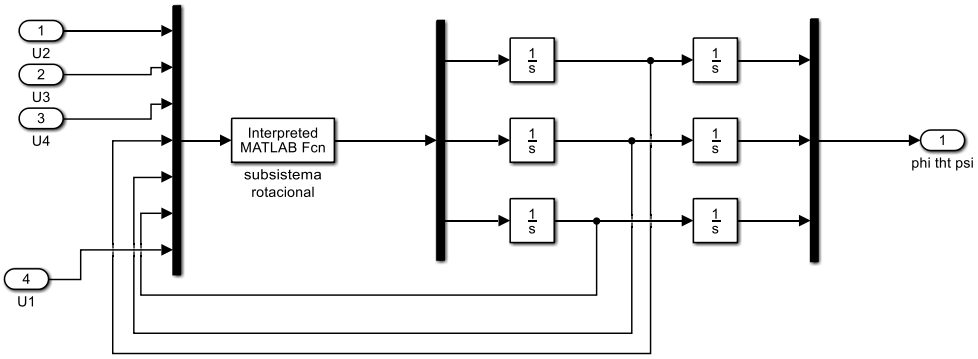


Fig. C.1.4. Subsistema rotacional.

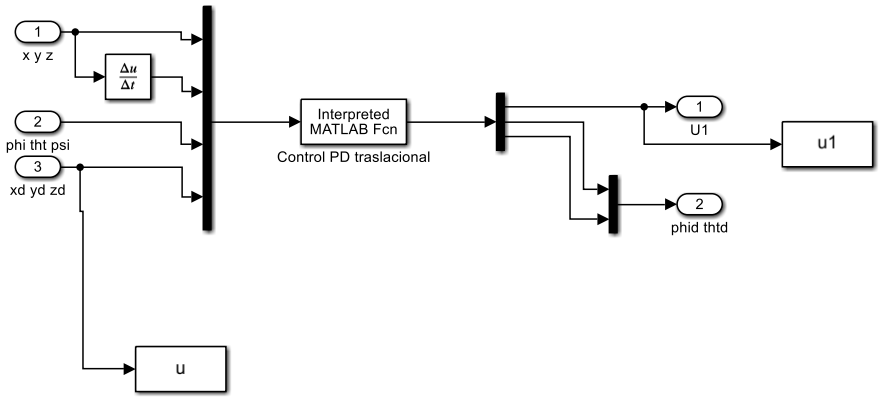


Fig. C.1.5. PD traslacional.

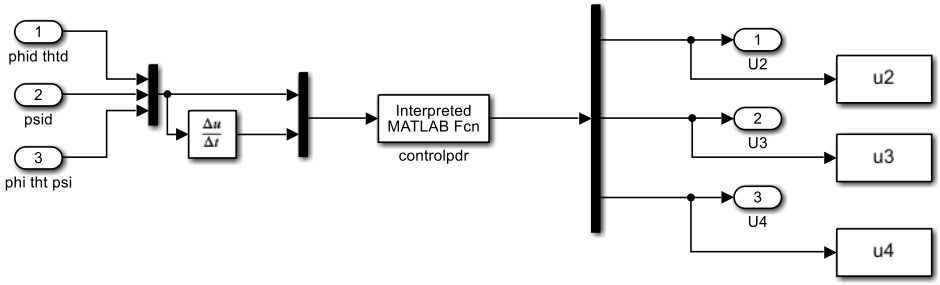


Fig. C.1.6. PD rotacional.

Programa C.1.1: inicio.m

```
clear all; close all;
global m I g Jr l
m = 1.2;I = [0.001,0,0;0,0.001,0;0,0,0.001 ];g=9.81;Jr=.0001;l=0.35;
sim('controlpd');
x=xyz(:,1);y=xyz(:,2);z=xyz(:,3);xd=ref(:,1);yd=ref(:,2);zd=ref(:,3);
figure(1)
plot(t,xd-x,'--',t,yd-y,'-.',t,zd-z,'-','Linewidth',1.5);grid;
leg1=legend('Error x','Error y','Error z')
set(leg1,'FontName','Times New Roman','FontSize', 12);
axis([0 +100 -.2 .2]);
xlabel('tiempo [s]','FontName','Times New Roman','FontSize', 20);
ylabel('posición [m]','FontName','Times New Roman','FontSize', 20);
```

Programa C.1.2: modelr.m

```
function op=modelr(u)
global I Jr l
u2=u(1);u3=u(2);u4=u(3);dphi=u(4);dtht=u(5);dpsi=u(6);u1=u(7);
k=.0001; k2=.00035;k1=.001;Ixx=I(1);Iyy=I(5);Izz=I(9);
omega1=sqrt(abs((u1/k)-(u4/k1)+(u3/k2)));
omega2=sqrt(abs((u1/k)+(u4/k1)-(u2/k2)));
omega3=sqrt(abs((u1/k)-(u4/k1)-(u3/k2)));
omega4=sqrt(abs((u1/k)+(u4/k1)+(u2/k2)));
omega=omega2+omega1+omega4+omega3;
ddphi=((Iyy-Izz)/Ixx)*dpsi*dtht-Jr*(omega/Ixx)*dtht+(l/Ixx)*u2;
ddtht=((Izz-Ixx)/Iyy)*dpsi*dphi+Jr*(omega/Iyy)*dphi+(l/Iyy)*u3;
ddpsi=((Ixx-Iyy)/Izz)*dphi*dtht+(1/Izz)*u4;
op=[ddphi;ddtht;ddpsi];
end
```

Programa C.1.3: modelt.m

```
function op=model(u)
global m g
u1=u(1);phi=u(2);tht=u(3);psi=u(4);
ddx=(u1/m)*(cos(phi)*sin(tht)*cos(psi)+sin(phi)*sin(psi));
ddy=(u1/m)*(cos(phi)*sin(tht)*sin(psi)-sin(phi)*cos(psi));
ddz=-g+(u1/m)*cos(phi)*cos(tht);
op=[ddx;ddy;ddz];
end
```

Programa C.1.4: controlpdr.m

```
function op=controlpdr(u)
phid=u(1);thtd=u(2);psid=u(3);phi=u(4);tht=u(5);psi=u(6);dphid=u(7);
dthtd=u(8);dpsid=u(9);dphi=u(10);dtht=u(11);dpsi=u(12);
Kpphi=1;Kdphi=.3;Kptht=1;Kdtht=.3;Kppsi=1;Kdpsi=.3;
u2=Kpphi*(phid-phi)+Kdphi*(dphid-dphi);
u3=Kptht*(thtd-tht)+Kdtht*(dthtd-dtht);
u4=Kppsi*(psid-psi)+Kdpsi*(dpsid-dpsi);
op=[u2;u3;u4];
```

Programa C.1.5: controlpdt.m

```
function op=controlpdt(u)
global m g
x=u(1);y=u(2);z=u(3);dx=u(4);dy=u(5);dz=u(6);phi=u(7);tht=u(8);
psi=u(9);xd=u(10);yd=u(13);zd=u(16);dxd=u(11);dyd=u(14);dzd=u(17);
```

```
Kpz=1;Kdz=1.5;Kpx=1;Kdx=2;Kpy=1;Kdy=2;
r1=Kpx*(xd-x)+Kdx*(dxd-dx);
r2=Kpy*(yd-y)+Kdy*(dyd-dy);
r3=Kpz*(zd-z)+Kdz*(dzd-dz);
u1=(g*m+m*r3)/(cos(tht)*cos(phi));
phid=asin((m/u1)*(r1*sin(psi)-r2*cos(psi)));
thtd=asin((m/u1)*(r1*cos(psi)+r2*sin(psi)));
op=[u1;phid;thtd];
end
```

C.2: Control Backstepping

La representación del modelo del cuadri-rotor es igual a la de D.1, por lo que ya no se presentan.

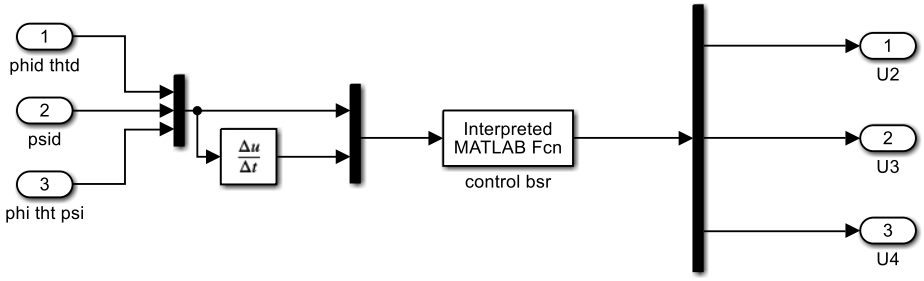


Fig. C.2.1. Backstepping rotacional.

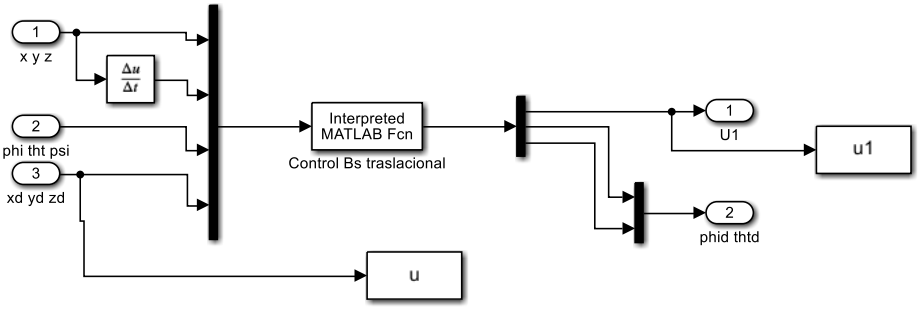


Fig. C.2.2. Backstepping traslacional.

Programa C.2.1: inicio.m

```
clear all; close all
format long
global omega m I g Jr l a1 a2 a3 a4 a5 b1 b2 b3
a1=0;a2=-.1;a3=0;a4=.1;a5=0;b1=3500;b2=3500;b3=3500;m = 1.20908;
I = [0.001,0,0;0,0.001,0;0,0,0.001 ];g=9.8;Jr=.0001;l=0.35;omega=256;
sim('controlbs');
x=xyz(:,1);y=xyz(:,2);z=xyz(:,3);xd=ref(:,1);yd=ref(:,2);zd=ref(:,3);
ex=xd-x;ey=yd-y;ez=zd-z;
```

Programa C.2.2: controlbsr.m

```
function op=controlbsr(u)
global a1 a2 a3 a4 a5 b1 b2 b3 omega
x1d=u(1);x3d=u(2);x5d=u(3);x1=u(4);x3=u(5);x5=u(6);x2d=u(7);x4d=u(8);
```

```
x6d=u(9);x2=u(10);x4=u(11);x6=u(12);
al1=500;al2=1;al3=500;al4=1;al5=500;al6=1;
z1=x1d-x1;z2=x2-x2d-al1*z1;z3=x3d-x3;z4=x4-x4d-al3*z3;
z5=x5d-x5;z6=x6-x6d-al5*z5;
u2=(1/b1)*(z1-a1*x4*x6-a2*x4*omega-al1*(z2+al1*z1)-al2*z2);
u3=(1/b2)*(z3-a3*x2*x6-a4*x2*omega-al3*(z4+al3*z3)-al4*z4);
u4=(1/b3)*(z5-a5*x2*x4-al5*(z6+al5*z5)-al6*z6);
op=[u2;u3;u4];
```

Programa C.2.3: controlbst.m

```
function op=controlbst(u)
global m g
x9=u(1);x11=u(2);x7=u(3);x10=u(4);x12=u(5);x8=u(6);x1=u(7);x3=u(8);
x5=u(9);x9d=u(10);x11d=u(13);x7d=u(16);dx9d=u(11);dx11d=u(14);
dx7d=u(17);al7=4;al8=1;al9=4;al10=1;al11=4;al12=1;
z7=x7d-x7;z8=x8-dx7d-al7*z7;z9=x9d-x9;z10=x10-dx9d-al9*z9;
z11=x11d-x11;z12=x12-dx11d-al11*z11;
u1=(m/(cos(x3)*cos(x1)))*(z7+g-al7*(z8+al7*z7)-al8*z8);
x1d=asin((m/u1)*(z9-al9*(z10+al9*z9)-al10*z10));
x3d=asin((m/u1)*(z11-al11*(z12+al11*z11)-al12*z12));
op=[u1;x1d;x3d];
end
```

C.3: Control LQR

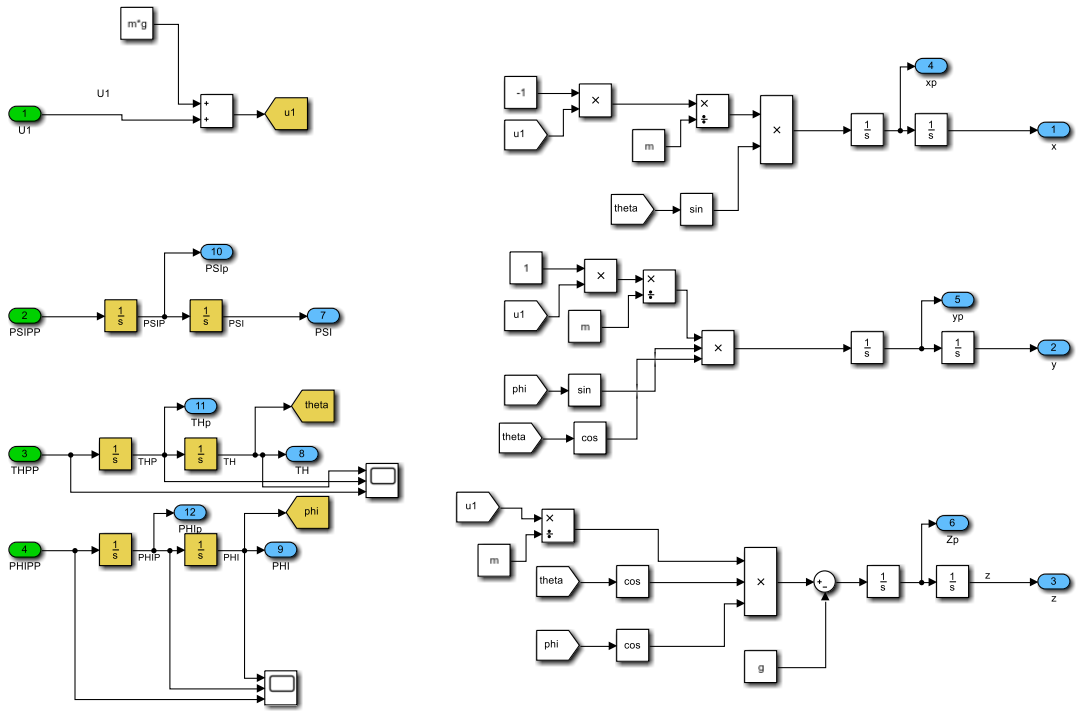


Fig. C.3.1. Modelo del cuadri-rotor.

Programa D.3.1: inicio.m

```
clear all
close all
m=1.20908;g=9.81;tf=100;h=0.032;n=1;tf1=tf+4*h;
for ta=0:h:tf1;
    t(n)=ta;
xd(n)=1*cos(.1*ta);
yd(n)=1*sin(.2*ta);
zd(n)=1;
psid(n)=0;
n=n+1;
end
nf=length(t);
for n=1:1:nf-1;
xdp(n)=(xd(n+1)-xd(n))/h;
ydp(n)=(yd(n+1)-yd(n))/h;
zdp(n)=(zd(n+1)-zd(n))/h;
psidp(n)=(psid(n+1)-psid(n))/h;
end
for n=1:1:nf-2;
xdpp(n)=(xdp(n+1)-xdp(n))/h;
ydpp(n)=(ydp(n+1)-ydp(n))/h;
zdpp(n)=(zdp(n+1)-zdp(n))/h;
psidpp(n)=(psidp(n+1)-psidp(n))/h;
end
for n=1:1:nf-3;
xdppp(n)=(xdpp(n+1)-xdpp(n))/h;
ydppp(n)=(ydpp(n+1)-ydpp(n))/h;
zdppp(n)=(zdpp(n+1)-zdpp(n))/h;
psidppp(n)=(psidpp(n+1)-psidpp(n))/h;
end
for n=1:1:nf-2;
u1(n)=m*sqrt((xdpp(n)^2)+(ydpp(n)^2)+(zdpp(n)+g)^2);
end
for n=1:1:nf-2;
thtd(n)=asin(m*xdpp(n)/-u1(n));
phid(n)=atan(ydpp(n)/(g+zdpp(n)));
end
for n=1:1:nf-3;
phidp(n)=(phid(n+1)-phid(n))/h;
thtdp(n)=(thtd(n+1)-thtd(n))/h;
end
for n=1:1:nf-4;
phidpp(n)=(phidp(n+1)-phidp(n))/h;
thtdpp(n)=(thtdp(n+1)-thtdp(n))/h;
end
for  n=1:1:nf-4;
xr(n)=xd(n);yr(n)=yd(n); zr(n)=zd(n);psir(n)=psid(n);thtr(n)=thtd(n);
phir(n)=phid(n);xpr(n)=xdp(n);ypr(n)=ydp(n);zpr(n)=zdp(n);
psipr(n)=psidp(n);thtpr(n)=thtdp(n);phipr(n)=phidp(n);xppr(n)=xdpp(n);
yppr(n)=ydpp(n);zppr(n)=zdpp(n); psippr(n)=psidpp(n); ts(n)=t(n);
thtppr(n)=thtdpp(n);phippr(n)=phidpp(n);xpppr(n)=xdppp(n);
ypppr(n)=ydppp(n); zpppr(n)=zdppp(n); psipppr(n)=psidppp(n);
u1r(n)=u1(n);u2r(n)=psidpp(n);u3r(n)=thtdpp(n);u4r(n)=phidpp(n);
end
K=[0,0,2.51,0,0,2.66,0,0,0,0,0,0;0,0,0,0,2.41,0,0,2.41,0,0;
-2.66,0,0,-0.05,0,0,0,17.58,0,0,6.01,0;
```

```
0,0,0,0,3.05,0,0,0,17.58,0,0,6.01];
ciphi=phir(1);citht=thtr(1);ciphip=phipr(1);cithtp=thtpr(1);cix=xr(1);
ciy=yr(1);ciz=zr(1);cixp=xpr(1);ciyp=ypr(1);cizp=zpr(1);
sim ('LQR');
ts=ts.';xa=Xa(:,1);ya=Xa(:,2);za=Xa(:,3);xpa=Xa(:,4);ypa=Xa(:,5);
zpa=Xa(:,6);psia=Xa(:,7);thta=Xa(:,8);phia=Xa(:,9);psipa=Xa(:,10);
thtpa=Xa(:,11);phipa=Xa(:,12);
figure(1)
plot(ts,xr.'-xa,'--',ts,yr.'-ya,'-.',ts,zr.'-za,'-','Linewidth',1.5);
grid;
leg1=legend('Error x','Error y','Error z')
set(leg1,'FontName','Times New Roman','FontSize', 12);
axis([0 +100 -.2 .2]);
xlabel('tiempo [s]','FontName','Times New Roman','FontSize', 20);
ylabel('posición [m]','FontName','Times New Roman','FontSize', 20);
figure(2)
plot3(xa,ya,za,'r',xr.',yr.',zr.','--b','Linewidth',1.5);
grid;
leg=legend('Posición','Referencia','Location','east')
set(leg,'FontName','Times New Roman','FontSize', 12);
axis([-1.2 1.2 -1.2 1.2 0 1.2]);
xlabel('x [m]','FontName','Times New Roman','FontSize', 20);
ylabel('y [m]','FontName','Times New Roman','FontSize', 20);
zlabel('z [m]','FontName','Times New Roman','FontSize', 20);
```

C.4: Control basado en planitud

Programa C.4: inicio.m

```
clear all
close all
m=1.20908;g=9.81;tf=100;h=0.032;n=1;tf1=tf+4*h;
for ta=0:h:tf1;   t(n)=ta;
xd(n)=1*cos(.1*ta);
yd(n)=1*sin(.2*ta);
zd(n)=1;
psid(n)=0;
n=n+1;
end
nf=length(t);
for n=1:1:nf-1;
xdp(n)=(xd(n+1)-xd(n))/h;
ydp(n)=(yd(n+1)-yd(n))/h;
zdp(n)=(zd(n+1)-zd(n))/h;
psidp(n)=(psid(n+1)-psid(n))/h;
end
for n=1:1:nf-2;
xdpp(n)=(xdp(n+1)-xdp(n))/h;
ydpp(n)=(ydp(n+1)-ydp(n))/h;
zdpp(n)=(zdp(n+1)-zdp(n))/h;
psidpp(n)=(psidp(n+1)-psidp(n))/h;
end
for n=1:1:nf-3;
xdppp(n)=(xdpp(n+1)-xdpp(n))/h;
ydppp(n)=(ydpp(n+1)-ydpp(n))/h;
zdppp(n)=(zdpp(n+1)-zdpp(n))/h;
```

```
psidppp(n)=(psidpp(n+1)-psidpp(n))/h;
end
for n=1:1:nf-2;
u1(n)=m*sqrt((xdpp(n)^2)+(ydpp(n)^2)+(zdpp(n)+g)^2);
end
for n=1:1:nf-2;
thtd(n)=asin(m*xdpp(n)/-u1(n));
phid(n)=atan(ydpp(n)/(g+zdpp(n)));
end
for n=1:1:nf-3;
phidp(n)=(phid(n+1)-phid(n))/h;
thtdp(n)=(thtd(n+1)-thtd(n))/h;
end
for n=1:1:nf-4;
phidpp(n)=(phidp(n+1)-phidp(n))/h;
thtdpp(n)=(thtdp(n+1)-thtdp(n))/h;
end
for  n=1:1:nf-4;
xr(n)=xd(n);yr(n)=yd(n);zr(n)=zd(n); psir(n)=psid(n);thtr(n)=thtd(n);
phir(n)=phid(n); xpr(n)=xdp(n);ypr(n)=ydp(n);zpr(n)=zdp(n);
psipr(n)=psidp(n);thtpr(n)=thtdp(n);phipr(n)=phidp(n);xppr(n)=xdpp(n);
yppr(n)=ydpp(n); zppr(n)=zdpp(n); psippr(n)=psidpp(n); ts(n)=t(n);
thtppr(n)=thtdpp(n);phippr(n)=phidpp(n);xpppr(n)=xdppp(n);
ypppr(n)=ydppp(n);zpppr(n)=zdppp(n); psipppr(n)=psidppp(n);
u1r(n)=u1(n);u2r(n)=psidpp(n);u3r(n)=thtdpp(n)u4r(n)=phidpp(n);
end
K=[0,0,2.51,0,0,2.66,0,0,0,0,0,0;0,0,0,0,2.41,0,0,2.41,0,0;
-2.66,0,0,-0.05,0,0,0,17.58,0,0,6.01,0;
0,0,0,0,3.05,0,0,0,17.58,0,0,6.01];
ciphi=phir(1);citht=thtr(1);ciphip=phipr(1);cithtp=thtpr(1);cix=xr(1);
ciy=yr(1);ciz=zr(1);cixp=xpr(1);ciyp=ypr(1);cizp=zpr(1);
sim ('LQR');
ts=ts.';xa=Xa(:,1);ya=Xa(:,2);za=Xa(:,3);xpa=Xa(:,4);ypa=Xa(:,5);
zpa=Xa(:,6);psia=Xa(:,7);thta=Xa(:,8);phia=Xa(:,9);psipa=Xa(:,10);
thtpa=Xa(:,11);phipa=Xa(:,12);
figure(1)
plot(ts,xr.'-xa,'--',ts,yr.'-ya,'-.',ts,zr.'-za,'-','Linewidth',1.5);
grid;
leg1=legend('Error x','Error y','Error z')
set(leg1,'FontName','Times New Roman','FontSize', 12);
axis([0 +100 -.2 .2]);
%axis([0 +100 -2 2]);
xlabel('tiempo [s]','FontName','Times New Roman','FontSize', 20);
ylabel('posición [m]','FontName','Times New Roman','FontSize', 20);
figure(2)
plot3(xa,ya,za,'r',xr.',yr.',zr.','--b','Linewidth',1.5);
grid;
leg=legend('Posición','Referencia','Location','east')
set(leg,'FontName','Times New Roman','FontSize', 12);
%axis([-1 1 -1 1 0 .6]);
axis([-1.2 1.2 -1.2 1.2 0 1.2]);
xlabel('x [m]','FontName','Times New Roman','FontSize', 20);
ylabel('y [m]','FontName','Times New Roman','FontSize', 20);
zlabel('z [m]','FontName','Times New Roman','FontSize', 20);
```

C.4.1: Control PD basado en planitud

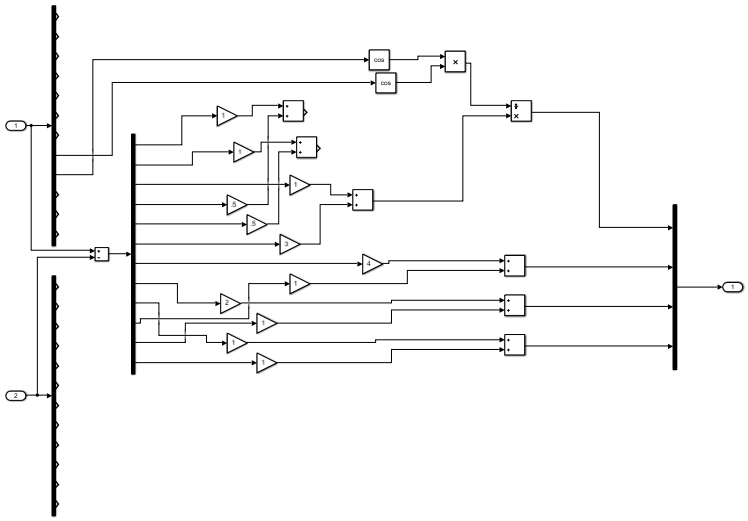


Fig. C.4.1. Control PD basado en planitud.

C.4.2: Control Backstepping basado en planitud

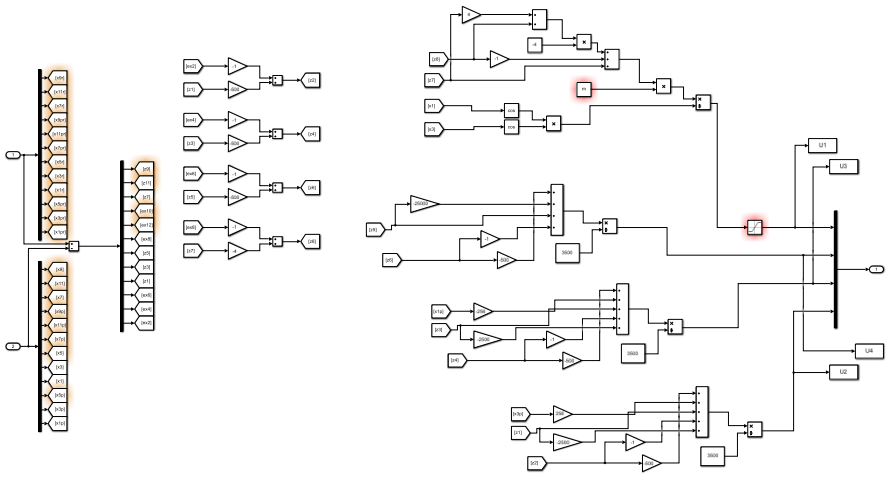


Fig. C.4.2. Control Backstepping basado en planitud.

C.4.3: Control LQR basado en planitud

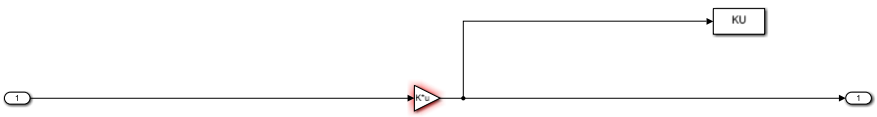


Fig. C.4.3. Control LQR basado en planitud.

Apéndice D: Estructura Cuadri-rotor SolidWorks

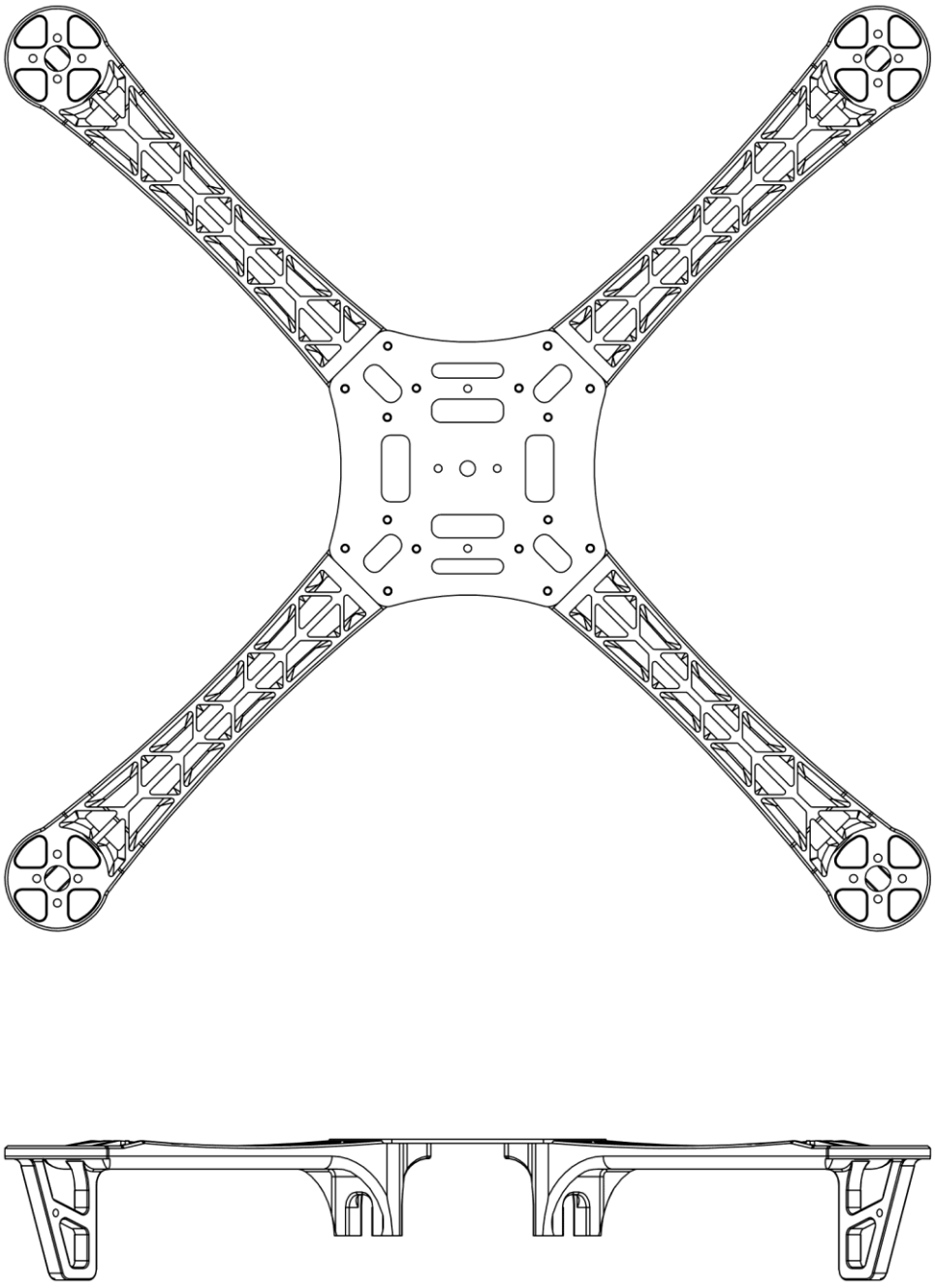


Fig. D.1. Vista superior y horizontal del Cuadri-rotor en SolidWorks.

Apéndice E: Programa en Webots

Programa E.1: PD.c

```
int main() {
  //Posición inicial
    t=t+h*.001;
    if (t<10)
    {
    xd=.8;
    yd=0;
    zd=.5;
    psi=0;
    //errores de posición
    ex=xd-xa;
    ey=yd-ya;
    ez=zd-za;
   //errores de orientación
    er=phi-roll;
    ep=tht-pitch;
    eya=psi-yaw;
   //Derivación numérica
    dx=(ex-exaux)/(hr);
    dy=(ey-eyaux)/(hr);
    dz=(ez-ezaux)/(hr);
    dr=(er-eraux)/(hr);
    dp=(ep-epaux)/(hr);
    dya=(eya-eyaaux)/(hr);
    exaux=ex;
    eyaux=ey;
    ezaux=ez;
    eraux=er;
    epaux=ep;
    eyaaux=eya;
   //Control PD traslacional
    r1=kpx*ex+kdx*dx;
    r2=kpy*ey+kdy*dy;
    r3=kpz*ez+kdz*dz;
    U1=(g*m +r3)/(cos(roll)*cos(pitch));
    phi=asin((m/U1)*(r2*sin(yaw)+r1*cos(yaw)));
    tht=asin((m/U1)*(-r1*sin(yaw)+r2*cos(yaw)));
    //Control PD rotacional
    U2=kpr*er+kdr*dr;
    U3=kpp*ep+kdp*dp;
    U4=kpya*eya+kdya*dya;
        //Velocidades de rotores
   v1=sqrt(U1/(4*k)+(U3/(2*k*l))-(U4/(4*b)));
   v2=sqrt(U1/(4*k)-(U2/(2*k*l))+(U4/(4*b)));
   v3=sqrt(U1/(4*k)-(U3/(2*k*l))-(U4/(4*b)));
   v4=sqrt(U1/(4*k)+(U2/(2*k*l))+(U4/(4*b)));
    }
  //Trayectoria deseada
   if (t>10)
    {
   ta=t-10.016;
   zd=.5;
   xd=.8*cos(.2*ta);
```

```
    yd=.8*sin(.2*ta);
    psi=3.1416/3;
   //errores de posición
     ex=xd-xa;
     ey=yd-ya;
     ez=zd-za;
    //errores de orientación
     er=phi-roll;
     ep=tht-pitch;
     eya=psi-yaw;
    //Derivación numérica
     dx=(ex-exaux)/(h*.001);
     dy=(ey-eyaux)/(h*.001);
     dz=(ez-ezaux)/(h*.001);
     dr=(er-eraux)/(h*.001);
     dp=(ep-epaux)/(h*.001);
     dya=(eya-eyaaux)/(h*.001);
     exaux=ex;
     eyaux=ey;
     ezaux=ez;
     eraux=er;
     epaux=ep;
     eyaaux=eya;
    //Control PD traslacional
     r1=kpx*ex+kdx*dx;
     r2=kpy*ey+kdy*dy;
     r3=kpz*ez+kdz*dz;
     U1=(g*m +m*r3)/(cos(roll)*cos(pitch));
     phi=asin((m/U1)*(r2*sin(yaw)+r1*cos(yaw)));
     tht=asin((m/U1)*(-r1*sin(yaw)+r2*cos(yaw)));
     //Control PD rotacional
     U2=kpr*er+kdr*dr;
     U3=kpp*ep+kdp*dp;
     U4=kpya*eya+kdya*dya;
     //Velocidades de rotores
    v1=sqrt(U1/(4*k)+(U3/(2*k*l))-(U4/(4*b)));
    v2=sqrt(U1/(4*k)-(U2/(2*k*l))+(U4/(4*b)));
    v3=sqrt(U1/(4*k)-(U3/(2*k*l))-(U4/(4*b)));
    v4=sqrt(U1/(4*k)+(U2/(2*k*l))+(U4/(4*b)));
    //Guardar datos
     fp=fopen("datos.txt","a");
    fprintf(fp,"%f %f %f %f %f %f %f %f %f %f %f %f %f
%f\n",xd,xa,yd,ya,zd,za,ta,U1,U2,U3,U4,er,ep,eya);
   fclose(fp);
    }
   }
   return 0;
}
```

Referencias

[1] U. Army. (2010). *Unmanned Aircraft Systems Roadmap*. US Army UAS Center of Excellence. USA.

[2] Valavanis, K. (2008). *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*. Springer Science & Business Media.

[3] Fahlstrom, P., Gleason, T. (2012). *Introduction to UAV Systems*. Jhon Wiley & Son.

[4] Valavanis, K., Vachtsevanos, G. (2015). *Handbook of Unmanned Aerial Vehicles*. Springer.

[5] Miranda, R. (2018). *Drones, Modelado y Control de Cuadrotores.* México. Alfaomega.

[6] Dikmen, I., Arısoy, A., y Temelta, H. (2009). A*ttitude control of a quadrotor*. 4th International Conference on Recent Advances in Space Technologies. Turquia.

[7] Paiva, E., Soto, J., Salinas, J., y Ipanequé. W. (2015). *Modeling and PID Cascade Control of a Quadcopter for Trajectory Tracking.* Conference on Electrical, Electronic Engineering, Information and Communication Technologies. Chile.

[8] Peng, L., Qingbo, G. (2011). *Real-Time Simulation System for UAV Based on Matlab/Simulink*. Conference on Computing, Control and Industrial Engineering. China.

[9] Bouaddallah, S., Siegwart, R. (2005). *Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor*. International Conference on Robotics and Automation.

[10] Cyberbotics. (2020). *Webots User Guidel*. Lausana, Suiza. Disponible en https://www.cyberbotics.com.

[11] Martínez, C. (2014). *Fault Tolerant Control by Flatness Approach*. Tesis Doctoral, Universidad Autónoma de Nuevo León. Nuevo León.

[12] Fliess, M., Levine, et al. (1995). *Flatness and Defect of Non-linear Systems: Introductory Theory and Examples*. International Journal of Control. 61(6), 1327 - 1361.

[13] Reyes, F., Cid, J. (2019). *Drones. Cinemática, Dinámica y Control de Cuadricópteros*. México. Alfaomega.

[14] Newcome, L. (2004). Unmanned aviation. *A brief history of Unmanned Aerial Vehicles.* Reston, Virginia. AIAA.

[15] Organización de Aviación Civil Internacional (2011). *Circular 328, Sistemas de aeronaves no tripuladas (UAS)*. Montreal.

[16] Secretaría de Comunicaciones y Transportes de México (2017). *PROYECTO de Norma Oficial Mexicana PROY-NOM-107-SCT3-2016, Que establece los requerimientos para operar un sistema de aeronave pilotada a distancia (RPAS) en el espacio aéreo mexicano*. Ciudad de México.

[17] Austin, R. (2010). *Unmanned Aircraft Systems. UAVS Desing, Development and Deployment*. Jonh Wiley & Sons.

[18] Nonami, K., Kendoul, F., Suzuki, S., et al. (2010). *Autonomous Flying Robots. Unmanned Aerial Vehicles and Micro Aerial Vehicles*. Springer.

[19] Lozano, R.(Ed.) (2010). *Unmanned Aerial Vehicles. Embedded Control*. London. Iste.

[20] Secretaría de Comunicaciones y Transportes de México (2019). *RESPUESTA a los comentarios al Proyecto de Norma Oficial Mexicana PROY-NOM-107-SCT3-2016, Que establece los requerimientos para operar un sistema de aeronave pilotada a distancia (RPAS) en el espacio aéreo mexicano, publicado el 20 de septiembre de 2017*. Ciudad de México.

[21] Martin, P. (1992). *Contribution à L'étude des Systèmes Différentiellement Plats*. Tesis Doctoral. École Nationale Supérieure des Mines de Paris, Francia.

[22] Charlet, B., Lévine y Marino, R. (1991). *Sufficient conditions for dynamic state feedback linearization, SIAM Journal on Control and Optimization*. 29(1), 38–57.

[23] Martínez, C., Lavigne, L., et al. (2013). *Control reconfiguration for differentially flat systems*. Congreso Nacional de Control Automático (AMCA). Ensenada, México. 477-482.

[24] L'evine, J. (2009). *Analysis and Control of Nonlinear Systems*. Springer. Berlin.

[25] Louembet, C. (2007). *Génération de Trajectoires Optimales pour Systèmes Différentiellement Plats: Application aux Manoeuves D'attitude sur Orbite*. Tesis Doctoral. Université Bordeaux 1. Francia.

[26] Vianna, G. (2007). *Modelado y Control de un Helicóptero Quadrotor*. Tesis de Maestría, Universidad de Sevilla, Dept. Ing. Sistemas y Automática, Sevilla, España.

[27] Bouabdallah, S. (2007). *Design and Control of Quadrotors with Application to Autonomous Flying*. Tesis Doctoral, Escuela Politécnica Federal de Lausana, Suiza.

[28] Castillo, P., García, P., et al. (2007). *Modelado y Estabilización de un Helicóptero con Cuatro Rotores*. RIAI Revista Iberoamericana de Automática e Informática Industrial.

[29] Cowling, I., Yakimenko, O., et al. (2007). *Prototype of an Autonomous Controller for a Quadrotor UAV*. European Control Conference.

[30] Zulu, A., Jonh, S. (2014). *A Review of Control Algorithms for Autonomous Quadrotors*. Open Journal of Applied Science.

[31] Cortes, E., Palomino, A., Martínez, C. y Etcheverry, G. (2019). *Simulación de un Cuadri-rotor en el Software Webots*. Memorias del Congreso Nacional de Control Automático. México.

[32] Khalil, H., (2002*). Nonlinear Systems*. Prentice Hall.

[33] Cortes, E., Palomino, A., Martinez, C. y Etcheverry, G. (2019). *Simulación de un Controlador Backstepping en Webots: Aplicación a un Cuadri-rotor*. Memorias SOMIXXIV Congreso de Instrumentación, México.

[34] Hespanha, J. (2009) *Linear Systems Theory*. Princeton University Press

[35] Reyes, F., (2011). *MATLAB, Aplicado a Robótica y Mecatrónica*. Alfaomega. México.

[36] Cyberbotics. (2020). *Webots Reference Manual*. Lausana, Suiza. Disponible en https://www.cyberbotics.com.

[37] Stolfi, D., Gálvez, S. (2010). *Mundos Virtuales 3D con VRML97*. Universidad de Málaga.