UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Ingeniería de Sistemas y Automática

Estudio y comparación entre metodologías de control para el seguimiento de trayectorias en sistemas no lineales.

Trabajo Fin de Máster

Máster Universitario en Automática e Informática Industrial

AUTOR/A: Del Río Rodríguez, Pablo

Tutor/a: Cuenca Lacruz, Ángel Miguel

Director/a Experimental: ALBERTOS PEREZ, PEDRO

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

MÁSTER EN ÁUTOMÁTICA E INFORMÁTICA INDUSTRIAL

Curso académico 2021/2022

TRABAJO FIN DE MÁSTER

Estudio y comparación entre metodologías de control para el seguimiento de trayectorias en sistemas no lineales

Autor:

PABLO DEL RÍO RODRÍGUEZ

Tutores:

ÁNGEL CUENCA LACRUZ

PEDRO ALBERTOS PÉREZ

València, julio de 2022

Agredecimientos

A mi familia por el apoyo desde la distancia.

A mis amigos por acompañarme y hacerme feliz todos los días.

A Pedro por las horas en su despacho con sus clases particulares y por haberme ayudado a disfrutar durante el desarrollo de este proyecto.

Resumen

El presente documento recoge el estudio teórico del diseño de controladores para el seguimiento de trayectorias en sistemas no lineales. Se hace hincapié en la simulación y comparación de una metodología basada en la linealización por realimentación de estados y de una metodología basada en álgebra lineal. Se comienza el trabajo con una breve descripción del estado del arte actual donde se exponen las características principales de este tipo de sistemas, así como una primera descripción teórica de las metodologías mencionadas. Posteriormente, se propone el caso de un dron para comprobar el desempeño de ambas metodologías y se realiza el estudio teórico para este caso particular de cara a su implementación práctica. Finalmente, se simulan los controladores obtenidos para distintas situaciones con el fin de obtener unas conclusiones de sobre las ventajas e inconvenientes del uso de las metodologías analizadas.

Abstract

This thesis collates the theoretical study regarding the design of controllers for tracking trajectories in non-linear systems. It places emphasis on the simulation and comparison between a methodology based on full state feedback linearisation and a linear algebra based methodology. The project begins with a brief description of the current state of the art, highlighting the most important characteristics of these systems, as well as a theoretical description of the aforementioned methodologies. Secondly, the case of a quadricopter drone is proposed in order to evaluate the performance in both cases and a theoretical study of this case in particular is carried out for its practical implementation. Finally, the controllers designed are simulated under different conditions in order to the advantages and disadvantages of the studied methodologies.

Resum

Aquest document recull l'estudi teòric del disseny de controladors per al seguiment de trajectòries en sistemes no lineals. Es posa l'accent en la simulació i la comparació d'una metodologia basada en la linealització per realimentació d'estats i d'una metodologia basada en àlgebra lineal. Es comença el treball amb una descripció breu de l'estat de l'art actua on s'exposen les característiques principals d'aquest tipus de sistemes, així com una primera descripció teòrica de les metodologies esmentades. Posteriorment, es proposa el cas d'un dron per comprovar l'acompliment d'ambdues metodologies i es realitza l'estudi teòric per a aquest cas particular amb vista a l'implementació pràctica. Finalment, se simulen els controladors obtinguts per a diferents situacions per obtenir unes conclusions sobre els avantatges i els inconvenients de l'ús de les metodologies analitzades.

Índice general

Índice de Figuras

Índice de Tablas

1. Introducción

De cara a facilitar la lectura y simplificar el orden de los contenidos del presente documento, se describe a continuación la manera en la que está estructurado.

En primer lugar, el presente trabajo busca la satisfacción de los objetivos que se presentan en el apartado 2.

Posteriormente, en el apartado 3, se hace una presentación del estado del arte que implica a ciertos sistemas no lineales y a las metodologías de control utilizadas en estos sistemas. En los subapartados 3.2 y 3.3 se describen teóricamente las dos metodologías *Linear Algebra Based* (LAB) y *Feedback-linearization control* (se la denominará metodología FL en ciertas ocasiones para abreviar) sobre las que se desarrolla este trabajo.

En el apartado 4 se desarrolla toda la formulación matemática relacionada con la implementación práctica de las metodologías comentadas y el desarrollo completo de ambos modelos para la realización de las simulaciones haciendo uso de *Matlab*&*Simulink*. Se particulariza la metodología LAB en el subapartado 4.2 y la metodología FL en el subapartado 4.3.

A continuación, en el apartado 5, se recogen las descripciones de los entornos empleados para las simulaciones, los parámetros obtenidos de la calibración de los reguladores y se describen los resultados conseguidos. En el subapartado 5.3, se presenta la comparación final y el análisis final de los resultados obtenidos con ambos controladores.

Se plantean una serie de posibles líneas futuras de investigación en relación con el tema tratado en este proyecto en el apartado 6.

Finalmente, se dedica el apartado 7 a la redacción de las conclusiones a las cuales se ha llegado durante el desarrollo de este proyecto.

2. Objetivos

El objetivo primordial sobre el que se basa el desarrollo y redacción de este documento es la realización de una investigación básica sobre el control de trayectorias de sistemas no lineales para la obtención del título del Máster en Automática e Informática Industrial impartido en la Universitat Politècnica de València durante el curso 2021/2022. De manera adicional, a continuación se resumen el resto de objetivos intrínsecos del trabajo:

- Analizar el estado del arte vigente para el modelado y control de sistemas dinámicos no lineales considerando dos metodologías: una bien establecida y documentada, como es la linealización por realimentación del estado, y otra de reciente implantación como es la metodología basada en álgebra lineal.
- Estudiar e implementar de manera pseudo-experimental las metodologías LAB y FL para el control de sistemas no lineales (se particulariza en el caso de un dron cuadricóptero) con el fin de realizar una comparación entre ambas.
- Realizar un análisis cuantitativo y cualitativo de las ventajas y desventajas que presentan cada una de ellas de cara a la implementación real en sistemas no lineales intentando justificar cual de las dos realiza mejor el seguimiento de la trayectoria y al mismo tiempo supone un ahorro en términos de energía. Este último aspecto resulta sumamente relevante en vehículos autónomos para conseguir aumentar la autonomía de los mismos.

3. Estado del Arte

3.1. Introducción a los sistemas dinámicos no lineales

Cuando se desea controlar un sistema, es necesario partir de un modelo del propio sistema para conocer cómo las variables están relacionadas entre sí y su evolución temporal. Los sistemas en los cuales la evolución de las variables depende de sus valores anteriores en el tiempo reciben el nombre de sistemas dinámicos. La complejidad en la construcción del modelo depende del propio sistema y de las simplificaciones que se desean realizar, determinando en este punto la sofisticación del control que se empleará.

Esta variación de las magnitudes del sistema con respecto del tiempo, en términos espaciales, se conoce como movimiento. Por lo general, a nivel dinámico existen tres tipos de movimientos: movimientos estacionarios (las características del movimiento no varían con respecto del tiempo), movimientos periódicos (las características del movimiento se repiten en el tiempo cada cierto periodo) y movimientos caóticos (la complejidad del movimiento es tal que sus características son imposibles de predecir). Debido al amplio espacio de posibilidades que implica el término movimiento, se suele considerar como una característica de los sistemas dinámicos su interdisciplinariedad. De esta manera, se puede abordar el control de muy diversos sistemas dinámicos con una serie limitada de metodologías de control.

Los sistemas no lineales son aquellos que no siguen una dinámica lineal, es decir, que se definen por una o más variables que evolucionan en el tiempo y cuya respuesta ante entradas no es proporcional a dichas entradas ni se cumple el principio de superposición. Se emplea la terminología «no-lineal» en contraposición al término «lineal», ya que esta última aproximación es la más empleada tradicionalmente debido a su relativa sencillez matemática. La aproximación lineal para el modelado de sistemas lleva implícita ciertas hipótesis:

- Se cumplen condiciones de proporcionalidad (pequeñas entradas generan pequeñas salidas).
- Son sistemas aditivos, es decir, se cumplen principios de superposición (la respuesta total es la suma de la respuesta de las partes).
- Son sistemas replicables (una misma entrada siempre generará la misma salida).
- La relación entre el comportamiento de las variables de entrada y de salida permiten conocer el comportamiento del sistema por completo.

Por otra parte, cuando la relación entre variables de un sistema no responde a alguno de los criterios anteriores nos encontramos ante una situación bastante diferente. Una relación proporcional entre variables es aquella que cumple que $y = k \cdot x$. Siempre que un sistema no satisfaga esa condición, se tratará de un sistema no-lineal. Llegados a este punto, es fácil suponer que la mayor parte de los sistemas físicos no sigan una

dinámica lineal [1].

En contraposición a los sistemas lineales, algunas características de la no linealidad son las siguientes:

- Al no ser sistemas proporcionales, pequeñas entradas pueden generar grandes salidas.
- No se cumple la aditividad, por lo que la respuesta del todo es distinta a la resultante de la suma de las partes.
- Son sistemas muy sensibles a las condiciones iniciales, por lo que en la práctica, es muy difícil replicarlos y prácticamente imposible reproducir resultados exactos experimentales.
- Las no linealidades vuelven al sistema imprevisible hasta cierto punto, pudiendo incurrir en inestabilidades.

La representación de sistemas dinámicos puede dividirse en sistemas discretos o continuos dependiendo de la manera de medir el tiempo. Los sistemas discretos son aquellos en los que el estado de sus variables (incluido el tiempo) queda definido en instantes concretos del tiempo, mientras que en los sistemas continuos, el estado de sus variables no se limita a mediciones muestreadas. Ambos tipos de sistemas, discretos o continuos, pueden clasificarse dependiendo de la cantidad de entradas y salidas que definan su comportamiento, siendo los modelos *Single Input Single Ouput* (SISO) los más simples y *Multiple Input Multiple Output* (MIMO) los más complejos. Esta categorización de los tipos de modelos puede realizarse tanto para sistemas lineales como no lineales.

Un sistema discreto puede definirse matemáticamente de distintas maneras (transformada $\mathcal{Z}$, ecuación en diferencias...). Se presenta a continuación un ejemplo de sistema discreto SISO definido por su ecuación en diferencias

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + b_1 u(k) + b_2 u(k-1) \tag{3.1}$$

donde k es un número natural que hace referencia al instante discreto del tiempo, y es la salida, u la entrada y los coeficientes a y b dependen de la propia naturaleza del sistema.

Los sistemas continuos, al modelar el tiempo de un modo continuo, pueden representarse dinámicamente por ecuaciones diferenciales. Se presenta como ejemplo de un sistema continuo un sistema típico de masa-resorte-amortiguador como el de la figura 3.1

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0 \tag{3.2}$$

Una notación muy empleada en el modelado matemático de estos sistemas (empleada en este trabajo) sustituye las derivadas $\frac{dx}{dt}$ por el equivalente $\dot{x}$, donde el orden de la derivada se corresponde con la cantidad de puntos encima de la variable. Según

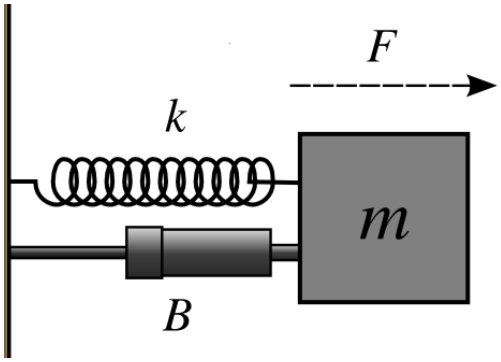

Figura 3.1: Sistema dinámico de masa-resorte-amortiguador. Fuente: [2]

esta notación, el sistema continuo anterior resulta

$$m\ddot{x} + b\dot{x} + kx = 0 \tag{3.3}$$

Cabe destacar que los dos sistemas expuestos, discreto (3.1) y continuo (3.3), son sistemas lineales.

La clasificación de los problemas de control se ha dividido tradicionalmente en dos categorías: los *Sistemas de regulación*, en los cuales se considera una referencia constante y el objetivo principal es la reducción del efecto de las perturbaciones, y los *Servosistemas* o sistemas de control de trayectorias, donde el objetivo principal es controlar un sistema para que las variables de salida del mismo sigan con un mínimo error una referencia externa. Ambos casos son similares en términos de realimentación de la información del proceso. Por otra parte, en el caso del control de trayectorias se conoce adicionalmente la información de la referencia y, posiblemente, su dinámica. Esto permite considerar técnicas de prealimentación que puede mejorar las prestaciones del control.

El seguimiento de trayectorias en sistemas dinámicos es uno de los principales problemas en la teoría de control. El término «referencia» se define como la base o apoyo sobre el cual se hace una comparación, mientras que el término «trayectoria» se corresponde con la línea en el espacio que describe el movimiento de un cuerpo. El control de trayectorias se basa en la realización de comparaciones periódicas entre la trayectoria y la referencia y la reducción del error entre ambas utilizando técnicas de control [3]. En algunos casos, dentro de la trayectoria se consideran también términos de orientación.

3.2. Metodología LAB

Como se ha mencionado, el control de la trayectoria se puede realizar siguiendo distintas estrategias. En este proyecto, se empleará una estrategia basada en la prealimentación de la referencia y el procesado del control utilizando herramientas de álgebra lineal. Esta metodología se denomina LAB [3,4].

De manera general, el objetivo principal del sistema de control es proporcionar las señales de entrada apropiadas para obtener del sistema físico las respuestas deseadas. En el ámbito del control de trayectorias, las salidas del sistema físico deseadas serán el seguimiento de la trayectoria desde una posición inicial hasta la posición final, a pesar de las posibles perturbaciones externas.

Independientemente de la metodología de control que se emplee, resulta indispensable contar con un modelo matemático que aproxime el comportamiento del sistema y de las perturbaciones que lo afecten. Dependiendo del modelo obtenido del sistema hay técnicas que se vuelven más ventajosas que otras.

A continuación, se describe el método de diseño de controladores basado en álgebra lineal.

3.2.1. Planteamiento del problema

Se asume que se conoce tanto el modelo matemático del proceso a controlar como el de la trayectoria que se desea seguir. La manera básica de expresar el modelo en espacio de estados es la siguiente

$$\begin{aligned} \dot{x}(t) &= F(x(t), u(t), d(t), t) \\ y(t) &= H(x(t), u(t), t) \end{aligned} \tag{3.4}$$

donde $x \in R^n$ denota el estado del sistema, $u \in R^m$ denota la entrada, $y \in R^p$ denota la salida del sistema y $d \in R^r$ denota las perturbaciones externas. Para el desarrollo del control en este proyecto se han tomado las siguientes hipótesis:

1. El modelo es afín en el control.
2. El modelo es de fase mínima.
3. El modelo es invariante en el tiempo.
4. El estado es accesible.
5. El modelo es exacto y no hay perturbaciones.
6. Tanto la referencia como sus derivadas son conocidas.

De esta manera, el modelo inicial está dado por

$$\begin{aligned} \dot{x}(t) &= f(x(t)) + g(x(t))u(t) \\ y(t) &= x(t) \end{aligned} \tag{3.5}$$

Como se ha mencionado, al considerarse todas las variables medibles, el vector de estado se puede dividir en dos componentes

$$\begin{bmatrix} \dot{\xi}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} f_\xi(\xi(t), z(t)) \\ f_z(\xi(t), z(t)) \end{bmatrix} + \begin{bmatrix} g_\xi(\xi(t), z(t)) \\ g_z(\xi(t), z(t)) \end{bmatrix} u(t) \tag{3.6}$$

donde $\xi \in R^{n_1}$ está formado por las variables de seguimiento (aquellas cuya trayectoria se desea seguir) y $z \in R^{n-n_1}$ es el vector de variables auxiliares, que llamaremos variables sacrificadas. Estas son las variables cuyo seguimiento temporal no es requerido. La trayectoria de referencia definida se denominará $\xi_r(t)$ y la referencia de las variables auxiliares $z_r(t)$. Este último término será irrelevante y se obtendrá del proceso de cómputo del sistema de control.

3.2.2. Planteamiento del control

Ahora bien, se puede expresar el problema de control de la siguiente manera: conociendo el proceso 3.6 y la referencia $\xi_r(t)$, se ha de determinar la entrada de control $u(t)$ que fuerza al subestado $\xi(t)$ a seguir la referencia.

En la metodología LAB se propone obtener la acción de control directamente del modelo 3.6. Para ello, se asume en primer lugar que la derivada de las variables de seguimiento se corresponde con la de las variables de referencia suponiendo una aproximación suave como puede ser una aproximación proporcional al error de cada variable

$$\begin{bmatrix} \dot{\xi}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{\xi}_r(t) - k_\xi[\xi_r(t) - \xi(t)] \\ \dot{z}_r(t) - k_z[z_r(t) - z(t)] \end{bmatrix} \tag{3.7}$$

donde k_ξ, k_z son dos matrices diagonales positivas de dimensiones $(n_1, n - n_1)$ y contienen los parámetros de control.

El seguimiento de trayectorias propuesto supone que los modelos 3.6 y 3.7 serán iguales, por lo que agrupándolos, la dinámica del problema vendrá dada por

$$\begin{bmatrix} \dot{\xi}_r(t) - k_\xi[\xi_r(t) - \xi(t)] - f_\xi(\xi(t), z(t)) \\ \dot{z}_r(t) - k_z[z_r(t) - z(t)] - f_z(\xi(t), z(t)) \end{bmatrix} = \begin{bmatrix} g_\xi(\xi(t), z(t)) \\ g_z(\xi(t), z(t)) \end{bmatrix} u(t) \tag{3.8}$$

que a su vez se puede expresar como

$$b(t) = A(t)u(t) \tag{3.9}$$

Para poder despejar la acción de control del sistema es necesario que el vector $b(t)$ y la matriz $A(t)$ sean compatibles, lo que se traduce en que el vector $b(t)$ debe pertenecer al espacio vectorial de la columna de $A(t)$. Siendo ese el caso, $u(t)$ podría despejarse directamente de 3.9 de la siguiente manera

$$u(t) = A^\dagger(t)b(t) \tag{3.10}$$

donde $A^\dagger(t)$ hace referencia a la matriz pseudoinversa de $A(t)$.

3.2.3. Diseño del control basado en álgebra lineal

Teniendo claros los conceptos descritos sobre la metodología, a continuación se establecerán los pasos para el diseño del control, incidiendo en las opciones posibles en cada paso:

1. Obtener representación matemática interna del proceso a controlar (3.5). El modelo ha de ser afín en la señal de control, lo que significa que la entrada de control no ha de estar contenida en ninguna de las funciones no lineales.

2. Descomponer el vector de estados en las componentes o subvectores: $\xi(t)$ para las variables de seguimiento y $z(t)$ para las variables sacrificadas.

3. Aproximar las derivadas de las variables de estado a las de la referencia. Normalmente se usa un enfoque proporcional (véase 3.7) donde la selección de k_i definirá el comportamiento del sistema controlado. Este tipo de aproximación puede adaptarse dependiendo de la dinámica del sistema.

4. Determinar el valor de la referencia para las variables sacrificadas para satisfacer $b \in A$ de forma que la primera expresión en 3.8 sea compatible con el resto, o lo que es lo mismo, que

$$\dot{\xi}_r(t) - k_\xi[\xi_r(t) - \xi(t)] - f_\xi(\xi(t), z(t)) = g_\xi(\xi(t), z(t))u(t) \tag{3.11}$$

   sea compatible con la segunda expresión del sistema de ecuaciones, pudiendo despejar de aquí la referencia $z_r(t)$.

5. Calcular la acción de control resolviendo el sistema 3.9 por el método de los mínimos cuadrados.

3.3. Metodología de linealización por realimentación

Como alternativa a la metodología LAB se va a utilizar la linealización por realimentación del estado, analizándose las similitudes y diferencias . Esta metodología busca la implementación de un controlador para sistemas no lineales realizando una linealización de los mismos por realimentación. Es una técnica aplicable para sistemas de la siguiente forma

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{3.12}$$

donde $f(x)$ y $g(x)$ son vectores de $\mathbf{R}^n$, u es la entrada e y es la salida del sistema.

El proceso de diseño del control es encontrar un entero ρ y una realimentación de estados

$$u = \alpha(x) + \beta(x) \cdot v \tag{3.13}$$

donde v es la nueva variable de control, α y β son funciones definidas alrededor de un punto $x_0 \in \mathbf{R}^n$ y $\beta(x_0) = 0$. Con estas consideraciones se puede reescribir 3.12 de la siguiente manera

$$\begin{aligned} \dot{x} &= f(x) + g(x)(\alpha(x) + \beta(x) \cdot v) \\ y &= h(x) \end{aligned} \tag{3.14}$$

para que la derivada de orden ρ sea dada por

$$y^\rho = v\ , \quad t \in \Gamma \tag{3.15}$$

Donde Γ es un intervalo abierto que contiene $t = 0$. Este problema se denomina como *input-output feedback linearization*. El punto x_0 alrededor del cual se realiza la linealización se denomina como punto de análisis.

La idea anterior se puede implementar como un sistema donde los estados son la salida y y una sucesión de sus ρ primeras derivadas

$$y^0 = h(x) \tag{3.16}$$

$$\begin{aligned} y^1 &= L_f h(x) \\ &\vdots \end{aligned} \tag{3.17}$$

$$y^{\rho-1} = L_f^{\rho-1} \cdot h(x) \tag{3.18}$$

$$y^{\rho} = L_f^{\rho} h(x) + L_g L_f^{\rho-1} h(x) u \tag{3.19}$$

Para entender mejor la estructura de este tipo de sistemas se usan las derivadas de Lie. Teniendo en cuenta que el sistema es de la forma que se muestra en 3.12 y que $y = h(x)$

$$\dot{y} = \frac{dh(x)}{dt} = \frac{dh(x)}{dx}\dot{x} = \frac{dh(x)}{dx} f(x) + \frac{dh(x)}{dx} g(x) u \tag{3.20}$$

y según la notación utilizada habitualmente en la literatura

$$L_f h(x) = \frac{dh(x)}{dx} f(x) \tag{3.21}$$

$$L_g h(x) = \frac{dh(x)}{dx} g(x) \tag{3.22}$$

donde 3.21 es la derivada de Lie de $h(x)$ a lo largo de $f(x)$ y 3.22 la derivada de Lie de $h(x)$ a lo largo de $g(x)$. Con esta nueva notación, se puede expresar $\dot{y}$ como

$$\dot{y} = L_f h(x) + L_g h(x) u \tag{3.23}$$

Esta notación resulta conveniente cuando se toman múltiples derivadas con respecto al mismo campo vectorial $f(x)$ o distinto $g(x)$

$$L_f^2 h(x) = \frac{d(L_f h(x))}{dx} f(x) \tag{3.24}$$

$$L_g L_f h(x) = \frac{d(L_f h(x))}{dx} g(x) \tag{3.25}$$

En el sistema linealizado por realimentación se ha construido un vector de estados de la salida $y = h(x)$ y sus ρ primeras derivadas. Se introduce ahora el concepto del grado relativo, que en este tipo de problemas se corresponde con el número de veces que es necesario derivar y para que la entrada u aparezca explícitamente en la expresión. Asimismo, este grado relativo es la diferencia entre el número de polos y ceros del sistema en el caso de un sistema lineal.

Siguiendo la nomenclatura de las expresiones 3.16 a 3.19, el término $L_f^k h(x)$, donde $k \in [2, \rho]$, como se ha mencionado, se denomina derivada de Lie de $L_f^{k-1} h(x)$ a lo largo del campo f. Asumiendo un grado relativo de ρ, los términos $L_g L_f^i h(x)$ para $i \in [1, \rho - 2]$ son cero y la entrada u no contribuye en el sistema en las primeras $\rho - 1$ derivadas. Nótese que si se emplea un control para 3.19 tal que así

$$u = \frac{v - L_f^\rho h(x)}{L_g L_f^{\rho-1} h(x)} \tag{3.26}$$

y dado que el entero ρ existe y $L_g L_f^{\rho-1} h(x) \neq 0$ en las proximidades de x_0, las funciones $\alpha(x)$ y $\beta(x)$ de la expresión 3.14 se pueden obtener directamente de 3.26 como

$$\alpha(x) = -\frac{L_f^\rho h(x)}{L_g L_f^{\rho-1} h(x)} \tag{3.27}$$

$$\beta(x) = \frac{1}{L_g L_f^{\rho-1} h(x)} \tag{3.28}$$

De la expresión 3.15 podemos observar que el control basado en la inversión de 3.26 tiene la capacidad de dar una forma a la respuesta mediante el diseño del control de v para obtener la salida deseada. Por otra parte, debido a que la ley de control por inversión está basada únicamente en la dinámica de entradas-salidas del sistema, puede fallar en un sistema de bucle cerrado estable. Esto puede suceder cuando el sistema es de fase no mínima y tiene algún polo o cero en el semiplano real [5].

Esta idea de cancelar las no-linealidades de un sistema mediante la realimentación del estado no es aplicable para todos los sistemas dinámicos no lineales. Se tienen que dar ciertas condiciones en la estructura del sistema que nos permita realizar la cancelación. No es difícil concluir que para cancelar mediante substracción la no-linealidad $\alpha(x)$, el control u y dicha no-linealidad tienen que aparecer juntos como una suma $u + \alpha(x)$. Asimismo, para cancelar un término no-lineal $\gamma(x)$ mediante una división, dicho término ha de encontrarse multiplicado por u tal que $\gamma(x)u$. Si la matriz $\gamma(x)$ no es singular en el dominio de interés, se puede cancelar por $u = \beta(x)v$, donde $\beta(x) = \gamma(x)^{-1}$. Por lo tanto, la capacidad de linealizar un sistema mediante realimentación del estado requiere que el sistema sea de la forma

$$\dot{x} = Ax + B\beta(x)^{-1}(u - \alpha(x)) \tag{3.29}$$

donde la entrada de control que linealiza el sistema es la que se muestra en 3.13. El sistema linealizado final resultaría de la forma

$$\dot{x} = Ax + Bv \tag{3.30}$$

Si esto no se puede dar de manera directa en el problema, es necesario realizar un cambio de variable. Este cambio de variable tiene que linealizar la relación entre las

entradas y salidas del sistema y no solo el modelo de estados [6].

Otra opción es realizar una expansión dinámica que fuerce que la matriz $\beta(x)$ sea invertible. Esto se puede hacer introduciendo «salidas virtuales» en el sistema mediante la inclusión de integradores delante de las entradas de control [7] como se verá más adelante en el apartado 4.3 para la aplicación concreta al caso de control de trayectorias en un dron.

4. Implementación del control

Como se ha mencionado previamente, prácticamente todas las metodologías de control de trayectorias basan su estudio en modelos dinámicos conocidos de los procesos a controlar. Para el estudio práctico de ambos métodos de control planteados, se ha elegido un sistema dinámico no lineal basado en un dron cuadricóptero, para el cual se ha desarrollado el modelo matemático que se expone a continuación.

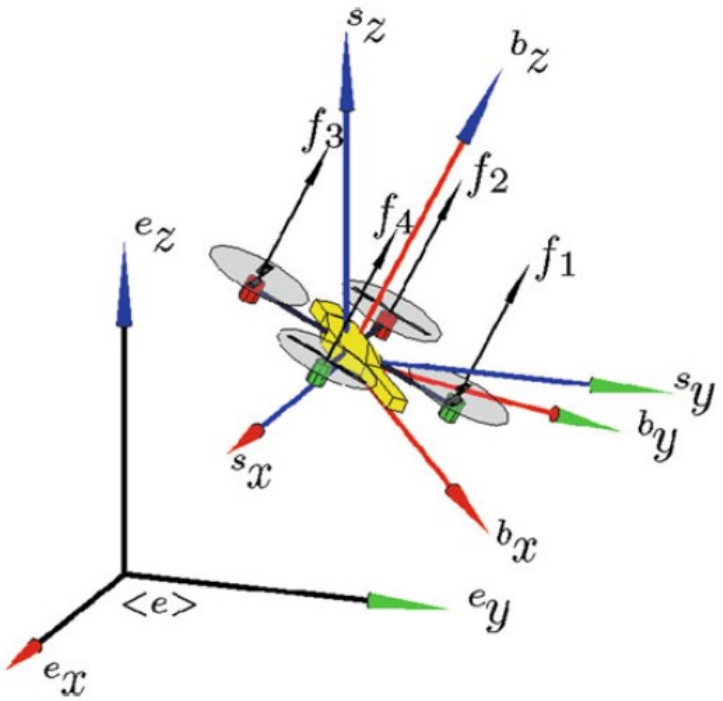

Figura 4.1: Diagrama de un cuadricóptero con 6 grados de libertad y los distintos marcos de referencia. Fuente: [8]

Para el estudio dinámico del dron, se parte del modelo desarrollado en [9] y posteriormente recogido en [3,8].

4.1. Modelo dinámico de un cuadricóptero

Un dron cuadricóptero no consta de plato oscilante como los helicópteros. Tampoco requiere ningún tipo de control de ángulo de paso de las hélices. Las únicas entradas del propio dron son los empujes de cada una de sus hélices combinados como se muestra en la figura 4.2.

De esta manera, atendiendo a la figura 4.2, se puede deducir que para lograr el movimiento de cabeceo del dron es necesario aumentar f_1 y disminuir f_3. Análogamente, para el movimiento de alabeo es necesario aumentar f_4 y disminuir f_2. Finalmente, para el movimiento de guiñada es necesario aumentar la velocidad de dos rotores opuestos y disminuir la de la otra pareja. Normalmente se aumenta f_2 y f_4

(rotores delantero y trasero) y se disminuye f_1 y f_3 (rotores laterales). Sea cual sea la situación, la suma total de los empujes de todos los rotores ha de mantener constante la componente vertical total para evitar que el dron descienda en altura.

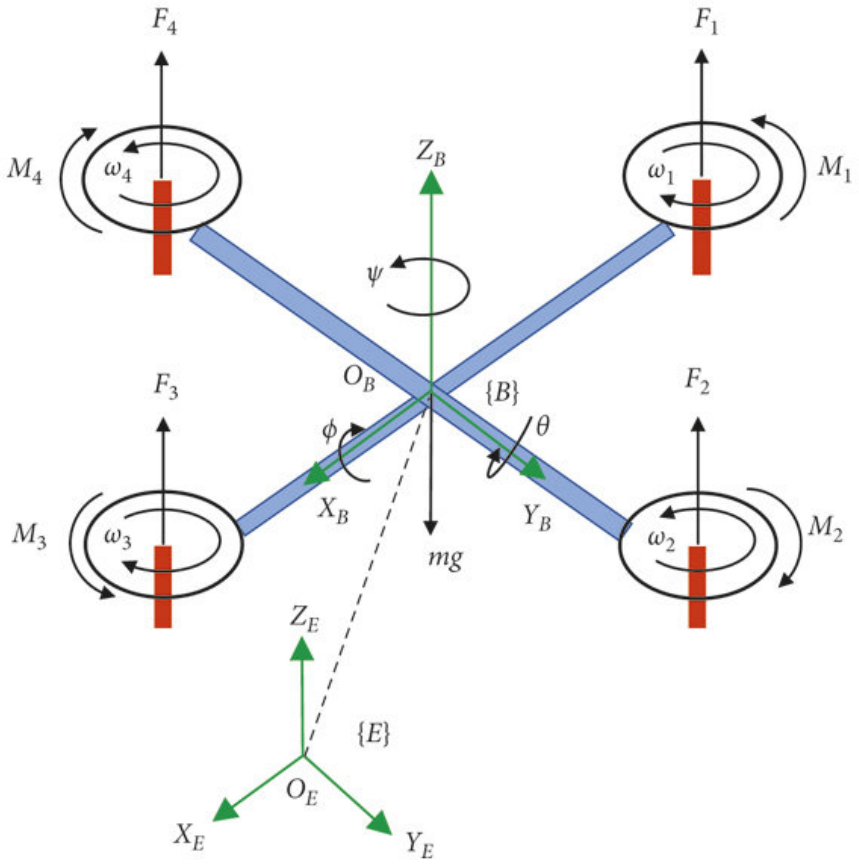

Figura 4.2: Diagrama del cuerpo libre de un cuadricóptero. Fuente: [10].

Las coordenadas generalizadas del cuadricóptero son

$$q = (x, y, z, \phi, \theta, \psi) \in \mathbb{R}^6 \tag{4.1}$$

donde (x, y, z) denota la posición del centro de masas del dron en relación al marco de referencia inercial, y (ϕ, θ, ψ) son los ángulos de Euler (orientación): ϕ es el ángulo de alabeo (*roll*), θ es el ángulo de cabeceo (*pitch*) y ψ el ángulo de guiñada (*yaw*). Teniendo en cuenta estas consideraciónes, el modelo se puede subdividir de manera natural en sus coordenadas traslacionales y rotacionales

$$\xi = (x, y, z) \in \mathbb{R}^3\,, \qquad \eta = (\phi, \theta, \psi) \in \mathbb{S}^3 \tag{4.2}$$

De esta manera, la energía cinética de traslación del vehículo es

$$T_{trans} = \frac{m}{2}\dot{\xi}^T\dot{\xi} \tag{4.3}$$

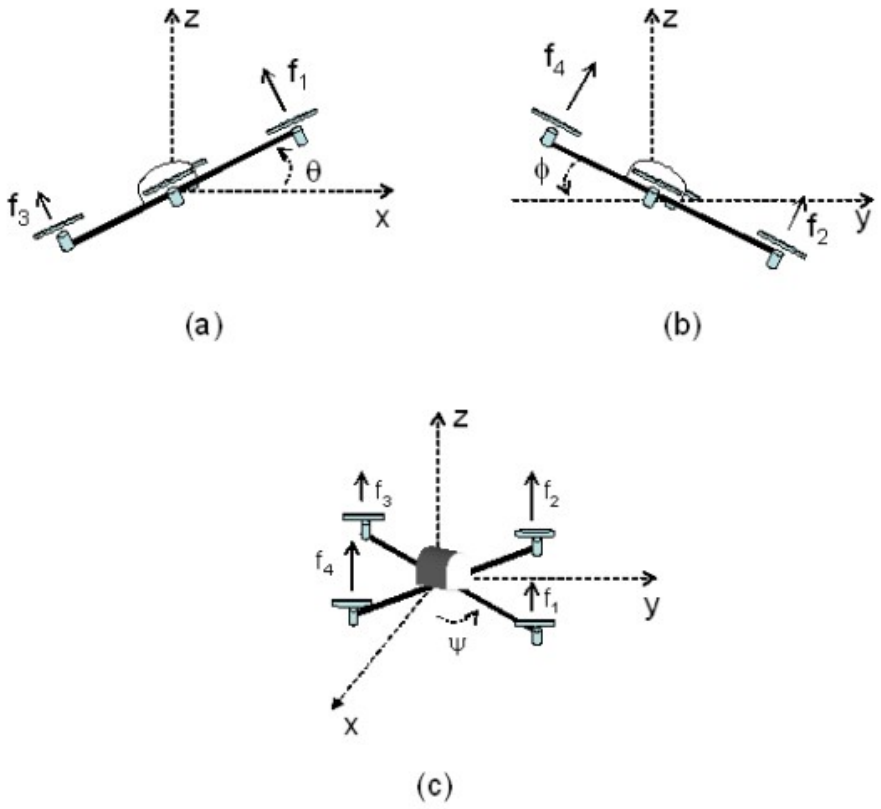

Figura 4.3: (a) Ángulo de cabeceo, (b) ángulo de alabeo y (c) ángulo de guiñada. Fuente: [9]

y la energía cinética de rotación

$$T_{rot} = \frac{m}{2}\dot{\eta}^T \mathbb{J} \dot{\eta} \tag{4.4}$$

La matrix $\mathbb{J}$ funciona como una matriz de inercia para el cálculo de la energía cinética de rotación total del cuadricóptero expresada en el sistema de coordenadas de η. La única energía potencial que se considera es la propia de la gravedad dada por

$$U = mgz \tag{4.5}$$

El Lagrangiano es

$$L(q, \dot{q}) = T_{trans} + T_{rot} - U = \frac{m}{2}\dot{\xi}^T \dot{\xi} + \frac{m}{2}\dot{\eta}^T \mathbb{J} \dot{\eta} - mgz \tag{4.6}$$

El modelo para la dinámica global del dron se obtiene de las ecuaciones de Euler-Lagrange para una fuerza externa generalizada

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = F \tag{4.7}$$

donde $F = (F_{\dot{\xi}}, \tau)$, τ agrupa los momentos generalizados y $F_{\dot{\xi}}$ es la fuerza de traslación aplicada al vehículo por las entradas de control. Se ignoran las fuerzas existentes en las direcciones x e y al ser mucho menores en magnitud que las entradas de control u

y τ. Por lo que

$$\widehat{F} = \begin{pmatrix} 0 \\ 0 \\ u \end{pmatrix} \tag{4.8}$$

donde (véanse las figuras 4.2 y 4.3)

$$u = f_1 + f_2 + f_3 + f_4 \tag{4.9}$$

y

$$f_i = k_i w_i^2 , \qquad i = 1, ..., 4 \tag{4.10}$$

$k_i > 0$ es una constante y w_i es la velocidad angular del motor i. De esta manera

$$F_{\dot{\xi}} = R\widehat{F} \tag{4.11}$$

tomando R es la matriz de orientación del dron (se emplea la notación cθ en lugar de $\cos\theta$ y sθ en lugar de $\sin\theta$ para los tres ángulos por motivos de espacio y no existiendo dudas de interpretación)

$$R = \begin{pmatrix} \mathrm{c}\theta\mathrm{c}\psi & \mathrm{s}\theta\mathrm{s}\psi & -\mathrm{s}\theta \\ \mathrm{s}\phi\mathrm{s}\theta\mathrm{c}\psi - \mathrm{c}\phi\mathrm{s}\psi & \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi + \mathrm{c}\phi\mathrm{c}\psi & \mathrm{s}\phi\mathrm{c}\theta \\ \mathrm{c}\phi\mathrm{s}\theta\mathrm{c}\psi + \mathrm{s}\phi\mathrm{s}\psi & \mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi - \mathrm{s}\phi\mathrm{c}\psi & \mathrm{c}\phi\mathrm{c}\theta \end{pmatrix} \tag{4.12}$$

Los momentos generalizados en las variables de η son

$$\tau = \begin{pmatrix} \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{pmatrix} \tag{4.13}$$

donde

$$\begin{aligned} \tau_\psi &= \sum_{i=1}^{4} \tau_{M_i} \\ \tau_\theta &= (f_2 - f_4)l \\ \tau_\phi &= (f_3 - f_1)l \end{aligned} \tag{4.14}$$

donde l es la distancia entre los rotores y el centro de gravedad del vehículo y τ_{M_i} es el par generado por el motor M_i. Representado de manera matricial y considerando una constante c que relaciona la fuerza y el momento de los motores

$$\begin{bmatrix} u \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -l & 0 & l & 0 \\ 0 & l & 0 & l \\ c & -c & c & -c \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{4.15}$$

Ya que el Lagrangiano no contiene términos cruzados en la energía cinética combinando $\dot{\xi}$ y $\dot{\eta}$ (véase 4.6), la ecuación de Euler-Lagrange se puede dividir entre la dinámica en las coordenadas $\dot{\xi}$ y $\dot{\eta}$, obteniendo

$$m\ddot{\xi} + \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = F_{\dot{\xi}} \tag{4.16}$$

$$\mathbb{J}\ddot{\eta} + \dot{\mathbb{J}}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial\eta}(\dot{\eta})^R\mathbb{J}\dot{\eta} = \tau \tag{4.17}$$

Definiendo el vector de Coriolis

$$\bar{V}(\eta,\dot{\eta}) = \dot{\mathbb{J}}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial\eta}(\dot{\eta}^T\mathbb{J}\dot{\eta}) \tag{4.18}$$

podemos escribir

$$\mathbb{J}\ddot{\eta} + \bar{V}(\eta,\dot{\eta}) = \tau \tag{4.19}$$

Si reescribimos $\bar{V}(\eta,\dot{\eta})$ como

$$\bar{V}(\eta,\dot{\eta}) = (\dot{\mathbb{J}} - \frac{1}{2}\frac{\partial}{\partial\eta}(\dot{\eta}^T\mathbb{J}))\dot{\eta} = C(\eta,\dot{\eta})\dot{\eta} \tag{4.20}$$

donde $C(\eta,\dot{\eta})$ se refiere a los términos de Coriolis y contiene los términos giroscópicos y centrífugos asociados con la dependencia de η con $\mathbb{J}$.

Finalmente se obtiene

$$m\ddot{\xi} = u \cdot \begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \sin\theta\cos\phi \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} \tag{4.21}$$

$$\mathbb{J}\ddot{\eta} = -C(\eta,\dot{\eta})\ddot{\eta} + \tau \tag{4.22}$$

Reordenando 4.22 y teniendo en cuenta 4.13

$$\tau = C(\eta,\dot{\eta})\dot{\eta} + \mathbb{J}\tilde{\tau} \tag{4.23}$$

podemos reescribir la variable de entrada $\ddot{\eta}$ como

$$\ddot{\eta} = \tilde{\tau} = \begin{pmatrix} \tilde{\tau}_\psi \\ \tilde{\tau}_\theta \\ \tilde{\tau}_\phi \end{pmatrix} \tag{4.24}$$

Reescribiendo las ecuaciones 4.21 y 4.22:

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} -u\cdot\sin\theta \\ u\cdot\cos\theta\sin\phi \\ u\cdot\cos\theta\cos\phi - mg \\ \tilde{\tau}_\phi \\ \tilde{\tau}_\theta \\ \tilde{\tau}_\psi \end{bmatrix} \tag{4.25}$$

Debido a la arquitectura del control interno del AR.Drone 2.0, las acciones de control del controlador secundario deberán de actuar sobre el ángulo de alabeo y cabeceo para el movimiento en el plano XY, la velocidad vertical para el desplazamiento en el eje Z y sobre la velocidad de rotación alrededor del eje Z para el seguimiento de la orientación.

Tabla 4.1: Resumen de la estrategia de control.

Control	**Descripción**
Altitud	u se emplea para hacer que el dron alcance la altitud deseada (posición en el eje Z).
Orientación	$\tilde{\tau}_\psi$ se usa para controlar la orientación.
Alabeo	$\tilde{\tau}_\theta$ se usa para controlar el alabeo y la posición en el eje Y.
Cabeceo	$\tilde{\tau}_\phi$ se usa para controlar el cabeceo y la posición en el eje X.

El modelo dinámico del dron recogido en 4.25 se puede desarrollar como un sistema de grado 12 resultando en el sistema 4.26. Este modelo se muestra representado en *Simulink* en la figura 4.4. Cabe mencionar que los integradores localizados inmediatamente antes de las salidas llevan programados las condiciones iniciales de posición y orientación para la primera iteración de las simulaciones.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{u}{m}\sin x_9 \\ x_4 \\ \frac{u}{m}\cos x_9 \sin x_7 \\ x_6 \\ \frac{u}{m}\cos x_9 \cos x_7 - g \\ x_8 \\ \tilde{\tau}_\phi \\ \dot{x}_{10} \\ \tilde{\tau}_\theta \\ x_{12} \\ \tilde{\tau}_\psi \end{bmatrix}; \quad \text{donde} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix} \tag{4.26}$$

4.2. Implementación de la metodología LAB

4.2.1. Implementación en continuo

En este apartado se seguirán los pasos definidos en 3.2.3. En primer lugar, se construyó un modelo del dron con el controlador simulable en continuo como se describe a continuación. Como se ha mencionado, el modelo dinámico 4.25, del dron resulta en un sistema de ecuaciones de grado 12. Para facilitar el desarrollo matemático, se ha simplificado la notación de 4.25 obteniendo el sistema 4.26.

Siguiendo la metodología según la expresión 3.7, podemos construir el sistema de la siguiente manera

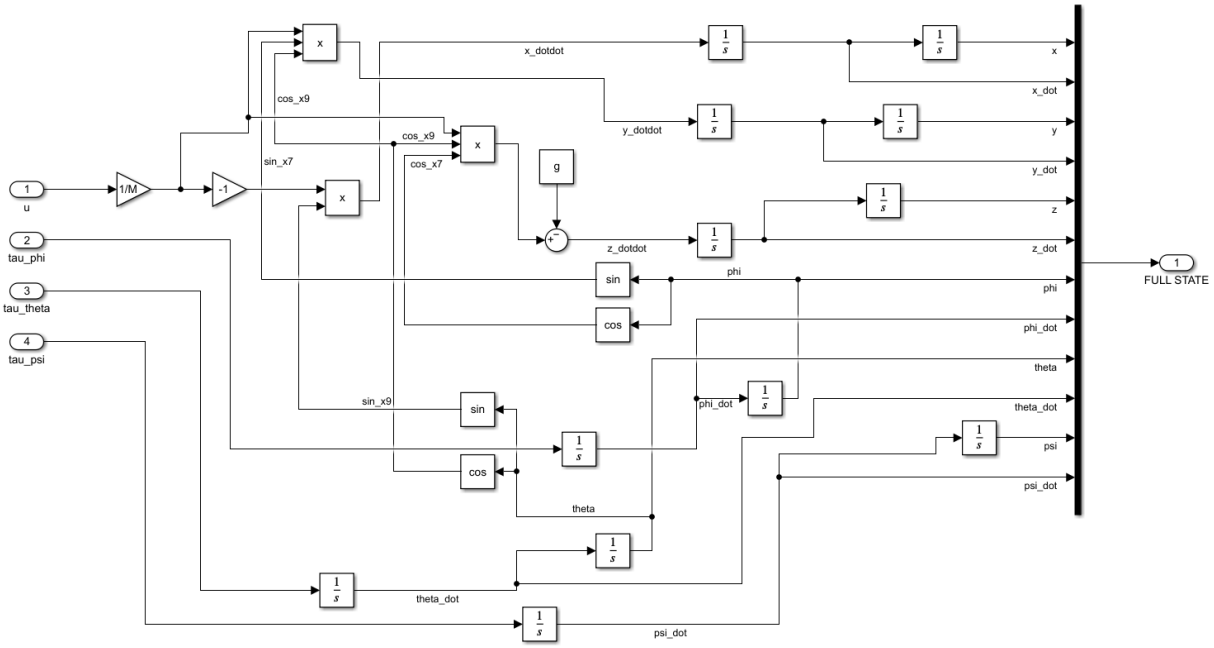

Figura 4.4: Modelo dinámico del dron construido en *Simulink*.

$$\begin{bmatrix} \dot{x}_{1,ref} - k_1(x_{1,ref} - x_1) \\ \dot{x}_{2,ref} - k_2(x_{2,ref} - x_2) \\ \dot{x}_{3,ref} - k_3(x_{3,ref} - x_3) \\ \dot{x}_{4,ref} - k_4(x_{4,ref} - x_4) \\ \dot{x}_{5,ref} - k_5(x_{5,ref} - x_5) \\ \dot{x}_{6,ref} - k_6(x_{6,ref} - x_6) \\ \dot{x}_{7,ref} - k_7(x_{7,ref} - x_7) \\ \dot{x}_{8,ref} - k_8(x_{8,ref} - x_8) \\ \dot{x}_{9,ref} - k_9(x_{9,ref} - x_9) \\ \dot{x}_{10,ref} - k_{10}(x_{10,ref} - x_{10}) \\ \dot{x}_{11,ref} - k_{11}(x_{11,ref} - x_{11}) \\ \dot{x}_{12,ref} - k_{12}(x_{12,ref} - x_{12}) \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{u}{m}\sin x_9 \\ x_4 \\ \frac{u}{m}\cos x_9 \sin x_z \\ x_6 \\ \frac{u}{m}\cos x_9 \cos x_7 - g \\ x_8 \\ \tilde{\tau}_\phi \\ \dot{x}_{10} \\ \tilde{\tau}_\theta \\ x_{12} \\ \tilde{\tau}_\psi \end{bmatrix} \tag{4.27}$$

y a su vez se puede descomponer y expresar como $A(t) \cdot u(t) = b(t)$

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m}\sin x_9 & 0 & 0 & 0 \\ \frac{1}{m}\cos x_9 \sin x_7 & 0 & 0 & 0 \\ \frac{1}{m}\cos x_9 \cos x_7 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u \\ \tilde{\tau}_\phi \\ \tilde{\tau}_\theta \\ \tilde{\tau}_\psi \end{bmatrix}}_{u} = \underbrace{\begin{bmatrix} \dot{x}_{1,ref} - k_1(x_{1,ref} - x_1) - x_2 \\ \dot{x}_{3,ref} - k_3(x_{3,ref} - x_3) - x_4 \\ \dot{x}_{5,ref} - k_5(x_{5,ref} - x_5) - x_6 \\ \dot{x}_{7,ref} - k_7(x_{7,ref} - x_7) - x_8 \\ \dot{x}_{9,ref} - k_9(x_{9,ref} - x_9) - x_{10} \\ \dot{x}_{11,ref} - k_{11}(x_{11,ref} - x_{11}) - x_{12} \\ \dot{x}_{2,ref} - k_2(x_{2,ref} - x_2) \\ \dot{x}_{4,ref} - k_4(x_{4,ref} - x_4) \\ \dot{x}_{6,ref} - k_6(x_{6,ref} - x_6) + g \\ \dot{x}_{8,ref} - k_8(x_{8,ref} - x_8) \\ \dot{x}_{10,ref} - k_{10}(x_{10,ref} - x_{10}) \\ \dot{x}_{12,ref} - k_{12}(x_{12,ref} - x_{12}) \end{bmatrix}}_{b} \tag{4.28}$$

Teniendo en cuenta que se conoce la referencia posicional y de orientación de la trayectoria, y asumiendo una aproximación proporcional, se puede calcular la referencia de las variables x_2, x_4, x_6 y x_8 como

$$x_{2,ref} = \dot{x}_{1,ref} - k_1(x_{1,ref} - x_1) \tag{4.29}$$

$$x_{4,ref} = \dot{x}_{3,ref} - k_3(x_{3,ref} - x_3) \tag{4.30}$$

$$x_{6,ref} = \dot{x}_{5,ref} - k_5(x_{5,ref} - x_5) \tag{4.31}$$

$$x_{12,ref} = \dot{x}_{11,ref} - k_{11}(x_{11,ref} - x_{11}) \tag{4.32}$$

Los valores para la referencia de las variables x_8 y x_{10} se computan de la misma manera

$$x_{8,ref} = \dot{x}_{7,ref} - k_7(x_{7,ref} - x_7) \tag{4.33}$$

$$x_{10,ref} = \dot{x}_{9,ref} - k_9(x_{9,ref} - x_9) \tag{4.34}$$

Para estimar los valores de $x_{8,ref}$ y $x_{10,ref}$, se han de conocer previamente los valores de $x_{7,ref}$ y $x_{9,ref}$ y el de sus derivadas. Estos valores se calculan estableciendo como condición que el sistema tenga solución exacta

$$\begin{bmatrix} -\sin x_9 \\ \cos x_9 \sin x_7 \\ \cos x_9 \cos x_7 \end{bmatrix} \cdot u = m \cdot \begin{bmatrix} \dot{x}_{2,ref} - k_2(x_{2,ref} - x_2) \\ \dot{x}_{4,ref} - k_4(x_{4,ref} - x_4) \\ \dot{x}_{6,ref} - k_6(x_{6,ref} - x_6) + g \end{bmatrix} \tag{4.35}$$

Para que este subsistema tenga solución, es necesario que se satisfaga

$$\tan x_{7,ref} = \frac{\sin x_{7,ref}}{\cos x_{7,ref}} = \frac{\dot{x}_{4,ref} - k_4(x_{4,ref} - x_4)}{\dot{x}_{6,ref} - k_6(x_{6,ref} - x_6) + g} \tag{4.36}$$

$$\tan x_{9,ref} = -\frac{\dot{x}_{2,ref} - k_2(x_{2,ref} - x_2)}{\dot{x}_{4,ref} - k_4(x_{4,ref} - x_4)} \cdot \sin x_{7,ref} \tag{4.37}$$

Finalmente, para resolver el sistema y despejar las acciones de control, se puede utilizar el método de los mínimos cuadrados, resultando en las siguientes expresiones

$$u = m \cdot (-\Delta_{x_2} \sin x_{9,ref} + \Delta_{x_4} \cos x_{9,ref} \sin x_{7,ref} + (\Delta_{x_6} + g) \cos x_{9,ref} \cos x_{7,ref}) \quad (4.38)$$

$$\tau_\phi = \Delta_{x_8} \quad (4.39)$$

$$\tau_\theta = \Delta_{x_{10}} \quad (4.40)$$

$$\tau_\psi = \Delta_{x_{12}} \quad (4.41)$$

donde también aparecen las derivadas de las referencias de las variables sacrificadas que habrá que calcular o estimar

$$\Delta_{x_2} = \dot{x}_{2,ref} - k_2(x_{2,ref} - x_2) \quad (4.42)$$

$$\Delta_{x_4} = \dot{x}_{4,ref} - k_4(x_{4,ref} - x_4) \quad (4.43)$$

$$\Delta_{x_6} = \dot{x}_{6,ref} - k_6(x_{6,ref} - x_6) \quad (4.44)$$

$$\Delta_{x_8} = \dot{x}_{8,ref} - k_8(x_{8,ref} - x_8) \quad (4.45)$$

$$\Delta_{x_{10}} = \dot{x}_{10,ref} - k_{10}(x_{10,ref} - x_{10}) \quad (4.46)$$

$$\Delta_{x_{12}} = \dot{x}_{12,ref} - k_{12}(x_{12,ref} - x_{12}) \quad (4.47)$$

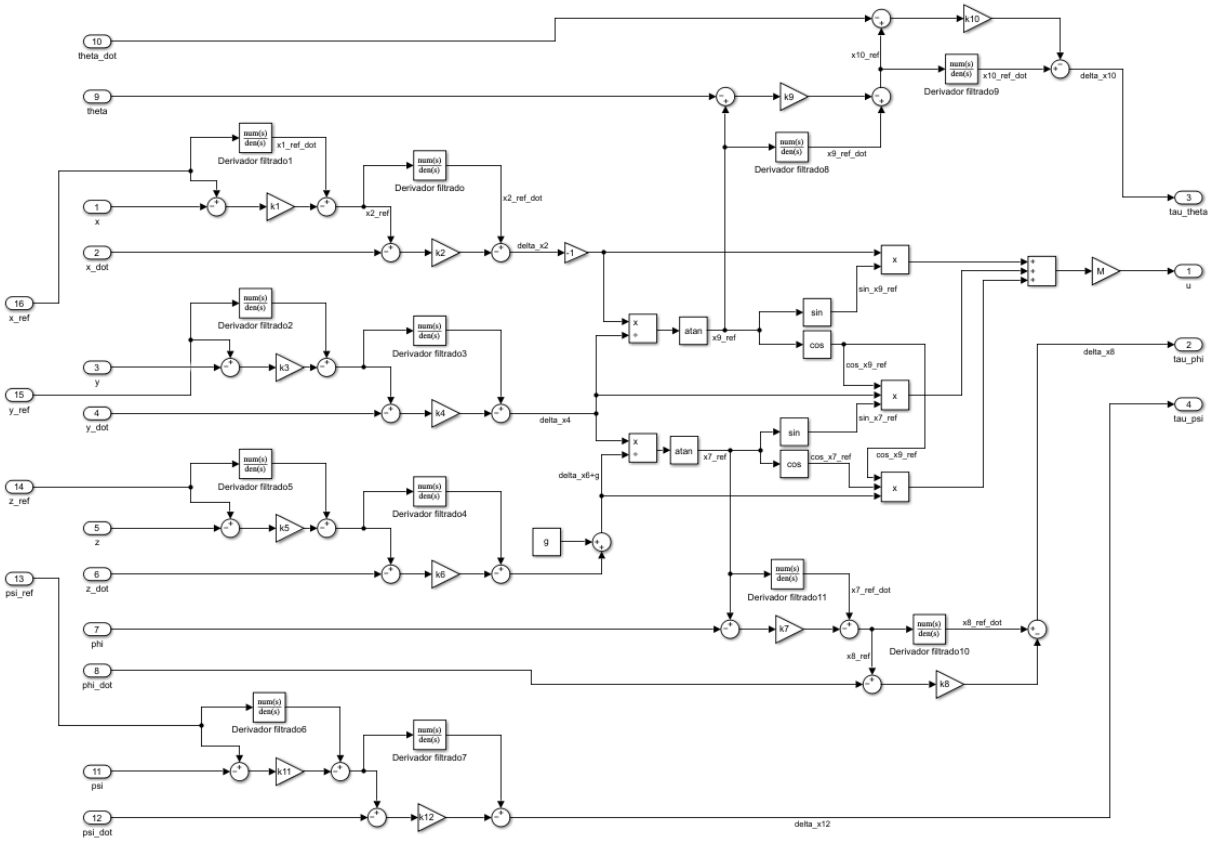

Figura 4.5: Modelo del controlador LAB en continuo representado en *Simulink*.

Utilizando este método de simulación, la estimación de las constantes del controlador para que siguiese una trayectoria simple de referencia resultó insatisfactoria, obteniendo excepciones y errores por la aparición de valores infinitos durante las simulaciones. Estos valores desproporcionados pueden ser fruto de inestabilidades ocurridas debido a la derivación de las variables sacrificadas para la obtención de sus valores futuros, que aquí se han computado mediante los denominados «derivadores filtrados». Seguir con esta investigación requeriría un tiempo y esfuerzo que se plantea para trabajos futuros en el apartado 6.

El modelo del controlador construido en continuo en *Simulink* se muestra en la figura 4.5 y se plantea su corrección para trabajos futuros dentro de esta línea de investigación. Ante esta situación, se probó la implementación de la metodología discretizando el controlador según se expone a continuación

4.2.2. Implementación en discreto

Para el diseño del controlador LAB en este caso se realizó una discretización según la aproximación del Euler del modelo dinámico recogido en 4.26

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{T}{m}\sin x_{9,n} & 0 & 0 & 0 \\ \frac{T}{m}\cos x_{9,n}\sin x_{7,n} & 0 & 0 & 0 \\ \frac{T}{m}\cos x_{9,n}\cos x_{7,n} & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix} \begin{bmatrix} u_n \\ \tilde{\tau}_{\phi,n} \\ \tilde{\tau}_{\theta,n} \\ \tilde{\tau}_{\psi,n} \end{bmatrix} = \begin{bmatrix} x_{1,n+1} - x_{1,n} - Tx_{2,n} \\ x_{3,n+1} - x_{3,n} - Tx_{4,n} \\ x_{5,n+1} - x_{5,n} - Tx_{6,n} \\ x_{7,n+1} - x_{7,n} - Tx_{8,n} \\ x_{9,n+1} - x_{9,n} - Tx_{10,n} \\ x_{11,n+1} - x_{11,n} - Tx_{12,n} \\ x_{2,n+1} - x_{2,n} \\ x_{4,n+1} - x_{4,n} \\ x_{6,n+1} - x_{6,n} + gT \\ x_{8,n+1} - x_{8,n} \\ x_{10,n+1} - x_{10,n} \\ x_{12,n+1} - x_{12,n} \end{bmatrix} \tag{4.48}$$

de donde se puede deducir que

$$\begin{bmatrix} x_{1,n+1} \\ x_{2,n+1} \\ x_{3,n+1} \\ x_{4,n+1} \\ x_{5,n+1} \\ x_{6,n+1} \\ x_{7,n+1} \\ x_{8,n+1} \\ x_{9,n+1} \\ x_{10,n+1} \\ x_{11,n+1} \\ x_{12,n+1} \end{bmatrix} = \begin{bmatrix} x_{1,n} + Tx_{2,n} \\ x_{2,n} - T\frac{u}{m}\sin x_{9,n} \\ x_{3,n} + Tx_{4,n} \\ x_{4,n} + T\frac{u}{m}\cos x_{9,n}\sin x_{7,n} \\ x_{5,n} + Tx_{6,n} \\ x_{6,n} + T\frac{u}{m}\cos x_{9,n}\cos x_{7,n} - g \\ x_{7,n} + Tx_{8,n} \\ x_{8,n} + T\tilde{\tau}_{\psi,n} \\ x_{9,n} + Tx_{10,n} \\ x_{10,n} + T\tilde{\tau}_{\theta,n} \\ x_{11,n} + Tx_{12,n} \\ x_{12,n} + T\tilde{\tau}_{\phi,n} \end{bmatrix} \tag{4.49}$$

El valor siguiente de las variables definidas en 4.49 se ha definido como una aproximación al de sus referencias. Esta aproximación es la etapa de diseño del control. Adicionalmente, para que el sistema 4.48 tenga una solución exacta, las seis primeras filas tienen que ser igual a cero. Por lo tanto, como la trayectoria es conocida y asumiendo una aproximación proporcional del sistema a dicha referencia, podemos calcular la referencia de las variables (x_2, x_4, x_6 y x_{12}) de la siguiente manera

$$x_{2,ref,n} = \frac{x_{1,ref,n+1} - k_1(x_{1,ref,n} - x_{1,n}) - x_{1,n}}{T} \tag{4.50}$$

$$x_{4,ref,n} = \frac{x_{3,ref,n+1} - k_3(x_{3,ref,n} - x_{3,n}) - x_{3,n}}{T} \tag{4.51}$$

$$x_{6,ref,n} = \frac{x_{5,ref,n+1} - k_5(x_{5,ref,n} - x_{5,n}) - x_{5,n}}{T} \tag{4.52}$$

$$x_{12,ref,n} = \frac{x_{11,ref,n+1} - k_{11}(x_{11,ref,n} - x_{11,n}) - x_{11,n}}{T} \tag{4.53}$$

Los valores para la referencia de las variables x_8 y x_{10} se computan de manera similar

$$x_{8,ref,n} = \frac{x_{7,ref,n+1} - k_7(x_{7,ref,n} - x_{7,n}) - x_{7,n}}{T} \tag{4.54}$$

$$x_{10,ref,n} = \frac{x_{9,ref,n+1} - k_9(x_{9,ref,n} - x_{9,n}) - x_{9,n}}{T} \tag{4.55}$$

Para estimar $x_{8,ref}$ y $x_{10,ref}$ de 4.54 y 4.55, se ha de calcular previamente $x_{7,ref}$ y $x_{9,ref}$. Estos valores se pueden obtener buscando las condiciones que hacen que el sistema 4.49 tenga solución exacta

$$\begin{bmatrix} -\sin x_{9,n} \\ \cos x_{9,n} \sin x_{7,n} \\ \cos x_{9,n} \cos x_{7,n} \end{bmatrix} u_n = \frac{m}{T} \begin{bmatrix} x_{2,ref,n+1} - k_2(x_{2,ref,n} - x_{2,n}) - x_{2,n} \\ x_{4,ref,n+1} - k_2(x_{4,ref,n} - x_{4,n}) - x_{4,n} \\ x_{6,ref,n+1} - k_2(x_{6,ref,n} - x_{6,n}) - x_{6,n} + gT \end{bmatrix} \tag{4.56}$$

de donde se pueden extraer las siguientes relaciones

$$\tan x_{7,ref,n} = \frac{x_{4,ref,n+1} - k_2(x_{4,ref,n} - x_{4,n}) - x_{4,n}}{x_{6,ref,n+1} - k_6(x_{6,ref,n} - x_{6,n}) - x_{6,n} + gT} \tag{4.57}$$

$$\tan x_{9,ref,n} = -\frac{x_{2,ref,n+1} - k_2(x_{2,ref,n} - x_{2,n}) - x_{2,n}}{x_{4,ref,n+1} - k_4(x_{4,ref,n} - x_{4,n}) - x_{4,n}} \cdot \frac{1}{\sin x_{7,ref,n}} \tag{4.58}$$

Teniendo en cuenta la siguiente notación

$$\Delta_{x,2} = x_{2,ref,n+1} - k_2(x_{2,ref,n} - x_{2,n}) - x_{2,n} \tag{4.59}$$

$$\Delta_{x,4} = x_{4,ref,n+1} - k_4(x_{4,ref,n} - x_{4,n}) - x_{4,n} \tag{4.60}$$

$$\Delta_{x,6} = x_{6,ref,n+1} - k_6(x_{6,ref,n} - x_{6,n}) - x_{6,n} \tag{4.61}$$

$$\Delta_{x,8} = x_{8,ref,n+1} - k_8(x_{8,ref,n} - x_{8,n}) - x_{8,n} \tag{4.62}$$

$$\Delta_{x,10} = x_{10,ref,n+1} - k_{10}(x_{10,ref,n} - x_{10,n}) - x_{10,n} \tag{4.63}$$

$$\Delta_{x,12} = x_{12,ref,n+1} - k_{12}(x_{12,ref,n} - x_{12,n}) - x_{12,n} \tag{4.64}$$

se ha aplicado nuevamente el método de los mínimos cuadrados para resolver el sistema 4.49

$$\begin{aligned} u_n = \frac{m}{T} \cdot \{ &- \Delta_{x,2} \sin x_{9,ref,n} + \\ &\Delta_{x,4} \cos x_{9,ref,n} \sin x_{7,ref,n} + (\Delta_{x,6} + gT) \cos x_{9,ref,n} \cos x_{7,ref,n} \} \end{aligned} \tag{4.65}$$

$$\tilde{\tau}_{\psi,n} = \frac{\Delta_{x,8}}{T} \tag{4.66}$$

$$\tilde{\tau}_{\theta,n} = \frac{\Delta_{x,10}}{T} \tag{4.67}$$

$$\tilde{\tau}_{\phi,n} = \frac{\Delta_{x,12}}{T} \tag{4.68}$$

Se ha de tener en cuenta que, como se observa en 4.59, es necesario el valor de $x_{2,ref,n+1}$ para poder calcular la acción de control que llevará al sistema a seguir las referencias establecidas. Este valor (y los demás valores equivalentes de las expresiones análogas sobre el resto de variables de estado) se puede extrapolar siguiendo por ejemplo una aproximación de Taylor

$$x_{2,ref,n+1} \approx x_{2,ref,n} + \frac{dx_{2,ref,n}}{dt} T + \frac{d^2 x_{2,ref,n}}{dt^2} \frac{T^2}{2} + ... + C \tag{4.69}$$

donde C es un término complementario de la serie. De esta manera, si el periodo de muestreo es pequeño, $x_{2,ref,n+1}$ puede aproximarse de una de las siguientes maneras

$$x_{2,ref,n+1} \approx x_{2,ref,n} \tag{4.70}$$

$$x_{2,ref,n+1} \approx x_{2,ref,n} + \frac{dx_{2,ref,n}}{dt} T \approx 2 \cdot x_{2,ref,n} - x_{2,ref,n-1} \tag{4.71}$$

$$x_{2,ref,n+1} \approx x_{2,ref,n} + \frac{x_{2,ref,n} - x_{2,ref,n-1}}{T} T + \frac{x_{2,ref,n} - x_{2,ref,n-1} - x_{2,ref,n-2}}{T^2} \frac{T^2}{2} \tag{4.72}$$

En primer lugar se probó a implementar la aproximación definida por 4.71 y se obtuvieron buenos resultados a nivel de seguimiento de la trayectoria, pero esta aproximación incurría en acciones de control exorbitadas sobre todo en los instantes iniciales donde no existe $x_{2,ref,n-1}$. Se probó igualmente a combinar las aproximaciones 4.70 y 4.71 para resolver el problema de las acciones de control en los instantes iniciales, pero se observó que se generaban oscilaciones en el seguimiento cuando se realizaba el cambio entre una y otra. Finalmente, la aproximación implementada fue la 4.70, dando lugar a un buen seguimiento con unas acciones de control razonables, reforzando la hipótesis de que la simulación en continuo presenta problemas debido al cálculo de las derivadas de las referencias de las variables sacrificadas.

Una vez tenido todo en cuenta, se ha construido un modelo de simulación en *Simulink* que se muestra en la figura 4.6

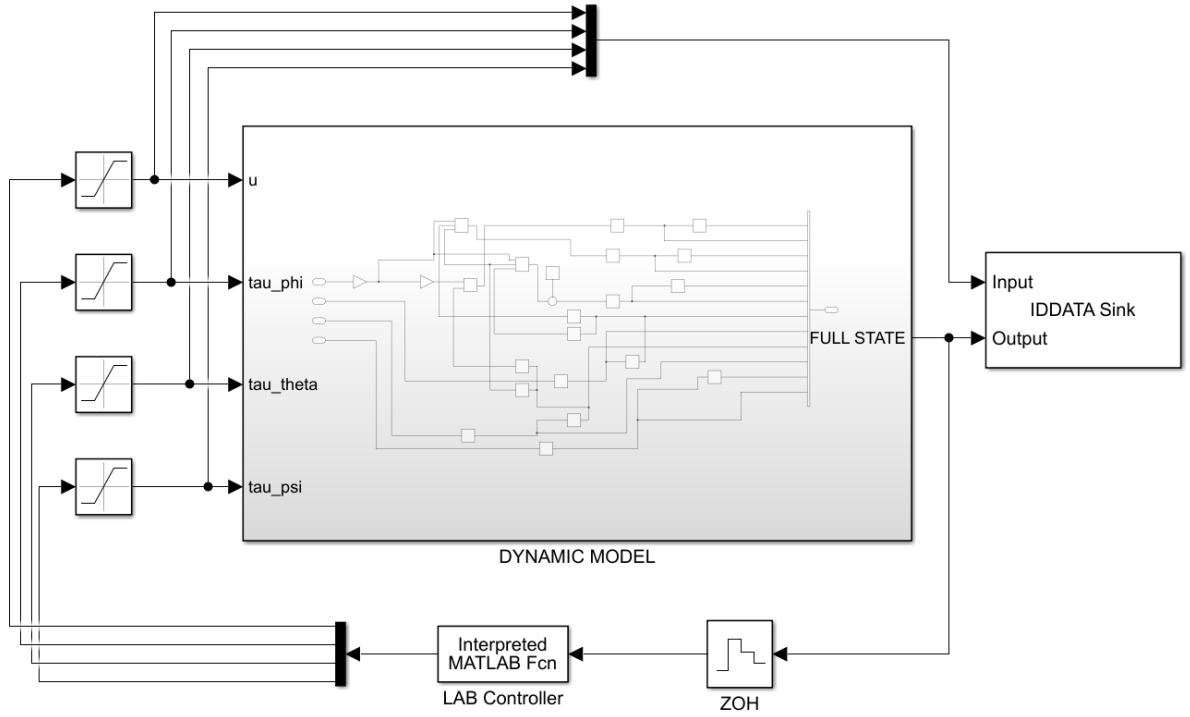

Figura 4.6: Modelo de *Simulink* del dron con el controlador LAB.

4.2.2.1 Desarrollo del algoritmo LAB

El controlador LAB se ha incorporado en una función de *Matlab* en la que se introducen los datos de las 12 variables de estado tras pasar a través de un discretizador de orden cero y devuelve las cuatro acciones de control. A continuación se describe la función que recoge el controlador.

En primer lugar, en las líneas 3 y 4 se declaran ciertas variables globales para poder lanzar las simulaciones desde un código de *Matlab*. Dentro de estas variables globales se encuentran constantes del proceso, constantes del controlador y un conjunto de variables que recogen los valores de las referencias sacrificadas en el instante anterior para poder utilizar la aproximación de Taylor correctamente.

Posteriormente, se actualiza la notación de las referencias y de las variables de estado que se introducen a la función para evitar confusiones futuras.

De la línea 27 a la 36 se computan las referencias de las variables sacrificadas haciendo uso de los errores de posición con respecto a la referencia que se desea seguir y el valor siguiente de la referencia.

Las líneas 39 a 42 recogen la aproximación de Taylor para obtener las referencias en $n+1$ de las variables sacrificadas. A continuación, con estos valores se computan las expresiones desde 4.59 a 4.64.

Finalmente, con estos valores se estiman las referencias de x_7 y x_9 y se sigue un

proceso análogo para calcular $\Delta_{x,8}$ y $\Delta_{x,10}$. De la línea 68 a la 77 se calculan y procesan las acciones de control obtenidas.

El último paso es actualizar los valores de las variables globales de los estados y el contador para hacer uso de los mismos en la siguiente iteración.

```
1  function U = LAB_controller(X)
2  global Ts g M i x_ref y_ref z_ref psi_ref
3  global k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_10 k_11 k_12
4  global x2r0 x4r0 x6r0 x7r0 x8r0 x9r0 x10r0 x12r0 aprox
5
6  % References
7  x1_ref = x_ref;
8  x3_ref = y_ref;
9  x5_ref = z_ref;
10 x11_ref = psi_ref;
11
12 % N instant
13 x1 = X(1);    % x
14 x2 = X(2);    % x_dot
15 x3 = X(3);    % y
16 x4 = X(4);    % y_dot
17 x5 = X(5);    % z
18 x6 = X(6);    % z_dot
19 x7 = X(7);    % phi
20 x8 = X(8);    % phi_dot
21 x9 = X(9);    % theta
22 x10 = X(10); % theta_dot
23 x11 = X(11); % psi
24 x12 = X(12); % psi_dot
25
26
27 % Position and orientation errors
28 e1 = x1_ref(i)-x1;
29 e3 = x3_ref(i)-x3;
30 e5 = x5_ref(i)-x5;
31 e11 = x11_ref(i)-x11;
32
33 % References of the sacrificed variables
34 x2_ref_n = (x1_ref(i+1)-k_1*e1-x1)/Ts;
35 x4_ref_n = (x3_ref(i+1)-k_3*e3-x3)/Ts;
36 x6_ref_n = (x5_ref(i+1)-k_5*e5-x5)/Ts;
37 x12_ref_n = (x11_ref(i+1)-k_11*e11-x11)/Ts;
38
39 % Approximation of the sacrificed references in the next
     instant
```

```
if aprox==1
    x2_ref_n1 = x2_ref_n;
    x4_ref_n1 = x4_ref_n;
    x6_ref_n1 = x6_ref_n;
    x12_ref_n1 = x12_ref_n;
elseif aprox==2
    x2_ref_n1 = 2*x2_ref_n-x2r0;
    x4_ref_n1 = 2*x4_ref_n-x4r0;
    x6_ref_n1 = 2*x6_ref_n-x6r0;
    x12_ref_n1 = 2*x12_ref_n-x12r0;
end

delta_x2 = x2_ref_n1-k_2*(x2_ref_n-x2)-x2;
delta_x4 = x4_ref_n1-k_4*(x4_ref_n-x4)-x4;
delta_x6 = x6_ref_n1-k_6*(x6_ref_n-x6)-x6;
delta_x12 = x12_ref_n1-k_12*(x12_ref_n-x12)-x12;

if delta_x4==0
    delta_x4=0.0000001;
end

x7_ref_n = atan(delta_x4/(delta_x6+g*Ts));
x9_ref_n = atan(-1*delta_x2/delta_x4*sin(x7_ref_n));

if aprox==1
    x7_ref_n1 = x7_ref_n;
    x9_ref_n1 = x9_ref_n;
elseif aprox==2
    x7_ref_n1 = 2*x7_ref_n-x7r0;
    x9_ref_n1 = 2*x9_ref_n-x9r0;
end

x8_ref_n = (x7_ref_n1-k_7*(x7_ref_n-x7)-x7)/Ts;
x10_ref_n = (x9_ref_n1-k_9*(x9_ref_n-x9)-x9)/Ts;

if aprox==1
    x8_ref_n1 = x8_ref_n;
    x10_ref_n1 = x10_ref_n;
elseif aprox==2
    x8_ref_n1 = 2*x8_ref_n-x8r0;
    x10_ref_n1 = 2*x10_ref_n-x10r0;
end

delta_x8 = x8_ref_n1-k_8*(x8_ref_n-x8)-x8;
delta_x10 = x10_ref_n1-k_10*(x10_ref_n-x10)-x10;

```

```
u1 = -delta_x2*sin(x9_ref_n);
u2 = delta_x4*cos(x9_ref_n)*sin(x7_ref_n);
u3 = (delta_x6+g*Ts)*cos(x9_ref_n)*cos(x7_ref_n);
u = M/Ts*(u1+u2+u3);

tau_phi = delta_x8/Ts;
tau_theta = delta_x10/Ts;
tau_psi = delta_x12/Ts;

U = [u tau_phi tau_theta tau_psi];

i=i+1;

% Resetting variables for the next iteration
x2r0 = x2_ref_n;
x4r0 = x4_ref_n;
x6r0 = x6_ref_n;
x7r0 = x7_ref_n;
x8r0 = x8_ref_n;
x9r0 = x9_ref_n;
x10r0 = x10_ref_n;
x12r0 = x12_ref_n;

end
```

Código 4.1: Código que recoge el algoritmo del controlador LAB implementado.

Como se ha visto, la aplicación de la metodología LAB es sencilla y las mayores dificultades se presentan al calcular los valores futuros de las referencias de las variables sacrificadas. Esto también se ha visto reflejado en el tratamiento en continuo, en el que la simulación del cálculo de las derivadas de estas referencias se torna complejo e inestable.

Es conveniente resaltar que el diseño del control se realiza en la sección de los coeficientes k_i en 4.27 en continuo y después de 4.49 en discreto. Si se requiere un control más preciso, es posible definir de otra forma la aproximación entre derivadas de las referencias y de las variables de estado.

4.3. Implementación de la linealización por realimentación

La expresión obtenida 4.25 de la dinámica del dron se puede escribir en forma de una expresión de espacio de estados eliminando los estados que no son tan relevantes para el control como

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\sin\theta & 0 & 0 & 0 \\ \cos\theta\sin\phi & 0 & 0 & 0 \\ \cos\theta\cos\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ \tilde{\tau}_\phi \\ \tilde{\tau}_\theta \\ \tilde{\tau}_\psi \end{bmatrix} \tag{4.73}$$

Nótese que la matriz que multiplica a las variables de control es singular, lo que implica que no hay una realimentación estática que linealice 4.73. En este caso, se puede emplear una técnica denominada inversión dinámica que puede realizarse gracias a una expansión dinámica del sistema o simplemente localizando dos integradores antes de la entrada u. Por lo tanto, derivando dos veces las expresiones dependientes de u se puede garantizar que la matriz no sea singular. Se ha utilizado la notación $(\sin\phi, \cos\phi, \sin\theta, \cos\theta)$ y $(\mathrm{s}\phi, \mathrm{c}\phi, \mathrm{s}\theta, \mathrm{c}\theta)$ indiferentemente atendiendo a cuestiones de espacio y siempre que no existan dudas de interpretación.

$$\begin{cases} \ddot{x} = -\sin\theta \cdot u \\ \dddot{x} = -\cos\theta \cdot \dot{\theta} \cdot u - \sin\theta \cdot \dot{u} \\ x^{(4} = (\sin\theta \cdot \dot{\theta}^2 - \cos\theta \cdot \ddot{\theta}) \cdot u - 2\cos\theta \cdot \dot{\theta} \cdot \dot{u} - \sin\theta \cdot \ddot{u} \end{cases} \tag{4.74}$$

$$\begin{cases} \ddot{y} = & \cos\theta\sin\phi \cdot u \\ \dddot{y} = & (\cos\phi\cos\theta \cdot \dot{\phi} - \sin\theta\sin\phi \cdot \dot{\theta}) \cdot u + \cos\theta\sin\phi \cdot \dot{u} \\ y^{(4} = & \left(-2\mathrm{s}\theta\mathrm{c}\phi \cdot \dot{\theta}\dot{\phi} - \mathrm{s}\phi\mathrm{c}\theta \cdot \dot{\phi}^2 - \mathrm{c}\theta\mathrm{s}\phi \cdot \dot{\theta}^2 + \mathrm{c}\theta\mathrm{c}\phi \cdot \ddot{\phi} - \mathrm{s}\theta\mathrm{s}\phi \cdot \ddot{\theta}\right) \cdot u \\ & +2(\mathrm{c}\phi\mathrm{c}\theta \cdot \dot{\phi} - \mathrm{s}\theta\mathrm{s}\phi \cdot \dot{\theta}) \cdot \dot{u} + \mathrm{c}\theta\mathrm{s}\phi \cdot \ddot{u} \end{cases} \tag{4.75}$$

$$\begin{cases} \ddot{z} = & \cos\theta\cos\phi \cdot u - g \\ \dddot{z} = & (-\sin\theta\cos\phi \cdot \dot{\theta} - \sin\phi\cos\theta \cdot \dot{\phi}) \cdot u + \cos\theta\cos\phi \cdot \dot{u} \\ z^{(4} = & \left(2\mathrm{s}\theta\mathrm{s}\phi \cdot \dot{\theta}\dot{\phi} - \mathrm{c}\phi\mathrm{c}\theta \cdot \dot{\phi}^2 - \mathrm{c}\theta\mathrm{c}\phi \cdot \dot{\theta}^2 - \mathrm{s}\theta\mathrm{c}\phi \cdot \ddot{\theta} - \mathrm{s}\phi\mathrm{c}\theta \cdot \ddot{\phi}\right) \cdot u \\ & -2(\mathrm{s}\theta\mathrm{c}\phi \cdot \dot{\theta} + \mathrm{s}\phi\mathrm{c}\theta \cdot \dot{\phi}) \cdot \dot{u} + \mathrm{c}\theta\mathrm{c}\phi \cdot \ddot{u} \end{cases} \tag{4.76}$$

Teniendo en cuenta la siguiente notación $[u, \tilde{\tau}_\phi, \tilde{\tau}_\theta, \tilde{\tau}_\psi] = [u_1, u_2, u_3, u_4]$ y atendiendo a 4.25, podemos escribir las cuartas derivadas como

$$\begin{cases} x^{(4} & = -\sin\theta \cdot \ddot{u}_1 - u_1\cos\theta \cdot u_3 + f_x \\ f_x & = u_1\sin\theta \cdot \dot{\theta}^2 - 2\dot{u}_1\cos\theta \cdot \dot{\theta} \end{cases} \tag{4.77}$$

$$\begin{cases} y^{(4} & = \mathrm{c}\theta\mathrm{s}\phi \cdot \ddot{u}_1 + u_1\mathrm{c}\theta\mathrm{c}\phi \cdot u_2 - u_1\mathrm{s}\theta\mathrm{s}\phi \cdot u_3 + f_y \\ f_y & = (-2\mathrm{s}\theta\mathrm{c}\phi \cdot \dot{\theta}\dot{\phi} - \mathrm{s}\phi\mathrm{c}\theta \cdot \dot{\phi}^2 - \mathrm{c}\theta\mathrm{s}\phi \cdot \dot{\theta}^2) \cdot u_1 + (\mathrm{c}\phi\mathrm{c}\theta \cdot \dot{\phi} \\ & \quad -\mathrm{s}\theta\mathrm{s}\phi \cdot \dot{\theta}) \cdot 2\dot{u}_1 \end{cases} \tag{4.78}$$

$$\begin{cases} z^{(4} & = \mathrm{c}\theta\mathrm{c}\phi \cdot \ddot{u}_1 - u_1 s\phi\mathrm{c}\theta \cdot u_2 - u_1 s\theta\mathrm{c}\phi \cdot u_3 + f_z \\ f_z & = (2s\theta s\phi \cdot \dot{\theta}\dot{\phi} - \mathrm{c}\theta\mathrm{c}\phi \cdot \dot{\theta}^2 - \mathrm{c}\phi\mathrm{c}\theta \cdot \dot{\phi}^2) \cdot u_1 - (s\theta\mathrm{c}\phi \cdot \dot{\theta} \\ & \quad -s\phi\mathrm{c}\theta \cdot \dot{\phi}) \cdot 2\dot{u}_1 \end{cases} \tag{4.79}$$

Finalmente, considerando la segunda derivada de ψ, podemos reconstruir el sistema de la siguiente manera

$$\begin{bmatrix} x^{(4} \\ y^{(4} \\ z^{(4} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_z \\ 0 \end{bmatrix} + \begin{bmatrix} g_{x,\ddot{u}_1} & g_{x,u_2} & g_{x,u_3} & g_{x,u_4} \\ g_{y,\ddot{u}_1} & g_{y,u_2} & g_{y,u_3} & g_{y,u_4} \\ g_{z,\ddot{u}_1} & g_{z,u_2} & g_{z,u_3} & g_{z,u_4} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \ddot{u}_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{4.80}$$

que resulta de la forma

$$\mathbf{x}^{\rho} = \mathbf{F} + \mathbf{G} \cdot \mathbf{u} \tag{4.81}$$

Tanto el vector columna $\mathbf{F} = [f_x, f_y, f_z, 0]^t$ y como la matriz $\mathbf{G}$ se componen de funciones no lineales de los estados y sus derivadas. Las funciones que conforman el vector $\mathbf{F}$ se pueden observar en las segundas expresiones de 4.77, 4.78 y 4.79, y aquellas que definen la matriz $\mathbf{G}$ son las que se recogen a continuación

$$\begin{cases} g_{x,\ddot{u}_1} = -\sin\theta \\ g_{y,\ddot{u}_1} = \cos\theta \sin\phi \\ g_{z,\ddot{u}_1} = \cos\theta\cos\phi \end{cases} \tag{4.82}$$

$$\begin{cases} g_{x,u_2} = 0 \\ g_{y,u_2} = \cos\theta\cos\phi \cdot u_1 \\ g_{z,u_2} = -\sin\phi\cos\theta \cdot u_1 \end{cases} \tag{4.83}$$

$$\begin{cases} g_{x,u_3} = -\cos\theta \cdot u_1 \\ g_{y,u_3} = -\sin\theta\sin\phi \cdot u_1 \\ g_{z,u_3} = -\sin\theta\cos\phi \cdot u_1 \end{cases} \tag{4.84}$$

y

$$g_{x,u_4} = 0 \;\; ; \;\; g_{y,u_4} = 0 \;\; ; \;\; g_{z,u_4} = 0 \tag{4.85}$$

La ley de realimentación del sistema dinámico que linealiza y desacopla las salidas del dron puede ser calculada como

$$\begin{bmatrix} \ddot{u}_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} g_{x,\ddot{u}_1} & g_{x,u_2} & g_{x,u_3} & g_{x,u_4} \\ g_{y,\ddot{u}_1} & g_{y,u_2} & g_{y,u_3} & g_{y,u_4} \\ g_{z,\ddot{u}_1} & g_{z,u_2} & g_{z,u_3} & g_{z,u_4} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} v_1 - f_x \\ v_2 - f_y \\ v_3 - f_z \\ v_4 \end{bmatrix} \tag{4.86}$$

siempre y cuando la matriz que recoge las funciones $\mathbf{G}$ sea invertible. Esto es fácilmente demostrable siempre y cuando el determinante de la submatriz 3×3 de la parte superior izquierda sea distinto de 0. Las variables v_1, v_2, v_3 y v_4 son las nuevas entradas de control de manera que el lazo cerrado de control resulta

$$\begin{bmatrix} x^{(4} \\ y^{(4} \\ z^{(4} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{4.87}$$

La ley de control 4.86 linealiza completamente la dinámica del dron descrita en 4.25 según lo expuesto en 3.13. La inversa de la matriz de desacoplamiento de 4.86 no es singular siempre y cuando u no sea 0. Este hecho concuerda con la idea de que ningún ángulo de cabeceo o alabeo afectará al movimiento del dron si el empuje de las hélices es nulo [5].

Al sistema obtenido se le puede aplicar entonces la metodología de linealización por realimentación. Haciendo uso de las expresiones 3.16 hasta 3.19, podemos asumir que el sistema resultante es de la forma

$$\begin{cases} x_1 = h_1(x) = x \\ x_2 = L_f h_1(x) = \dot{x} \\ x_3 = L_f^2 h_1(x) = \ddot{x} \\ x_4 = L_f^3 h_1(x) = \dddot{x} \\ x_5 = h_2(x) = y \\ x_6 = L_f h_2(x) = \dot{y} \\ x_7 = L_f^2 h_2(x) = \ddot{y} \\ x_8 = L_f^3 h_2(x) = \dddot{y} \\ x_9 = h_3(x) = z \\ x_{10} = L_f h_3(x) = \dot{z} \\ x_{11} = L_f^2 h_3(x) = \ddot{z} \\ x_{12} = L_f^3 h_3(x) = \dddot{z} \\ x_{13} = h_4(x) = \psi \\ x_{14} = L_f h_4(x) = \dot{\psi} \end{cases} \tag{4.88}$$

Tras esta transformación, se pueden escribir los sistemas desacoplados de la siguiente manera

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dddot{x} \\ x^{(4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ \dddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot v_1 \tag{4.89}$$

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \\ \dddot{y} \\ y^{(4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \\ \dddot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot v_2 \tag{4.90}$$

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dddot{z} \\ z^{(4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} z \\ \dot{z} \\ \ddot{z} \\ \dddot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot v_3 \tag{4.91}$$

$$\begin{bmatrix} \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot v_4 \tag{4.92}$$

dando lugar a sistemas de la forma

$$\begin{cases} \dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{v} \\ \mathbf{y} = C\mathbf{x} \end{cases} \tag{4.93}$$

donde A y B son una pareja de matrices controlables y $C = [1, 0, 0, 0]$. Posteriormente, se han de asignar los polos del sistema en bucle cerrado. En particular, se ha de calcular un vector constante de ganancias K de manera que la ley de realimentación del estado: $\mathbf{v} = -K \cdot \mathbf{x}$ genere la dinámica en bucle cerrado deseada.

Las referencias del sistema también se pueden escribir de manera similar

$$\dot{\mathbf{x}}_{ref} = A\mathbf{x}_{ref} + B\mathbf{v}_{ref} \tag{4.94}$$

donde $\mathbf{v}_{ref}$ es, dependiendo del sistema, la cuarta derivada del estado o la segunda.

El error entre los estados y las referencias se define como

$$\tilde{\mathbf{x}} = \mathbf{x}_{ref} - \mathbf{x} \tag{4.95}$$

y substrayendo 4.95 de 4.94 se obtiene

$$\dot{\tilde{\mathbf{x}}} = A\tilde{\mathbf{x}} + B(\mathbf{v}_{ref} - \mathbf{v}) \tag{4.96}$$

Haciendo uso del vector de ganancias K para asignar los polos, la matriz A en bucle cerrado resulta en la siguiente equivalencia

$$A_{CL} = A - B \cdot K \tag{4.97}$$

donde la matriz $B \cdot K$ resultará invertible. De esta manera, el sistema 4.96 se puede reescribir

$$\dot{\tilde{\mathbf{x}}} = (A - BK)\tilde{\mathbf{x}} - BK\tilde{\mathbf{x}} - B(\mathbf{v}_{ref} - \mathbf{v}) \tag{4.98}$$

y factorizando

$$\dot{\tilde{\mathbf{x}}} = (A - BK)\tilde{\mathbf{x}} + B(K\tilde{\mathbf{x}} + \mathbf{v} - \mathbf{v}_{ref}) \tag{4.99}$$

Si el último término de la expresión 4.99 es cero, los errores de los estados convergerán a cero con la dinámica establecida dada la matriz de bucle cerrado. Forzando este término a cero se define la ley de control

$$\mathbf{v} = K(\mathbf{x}_{ref} - \mathbf{x}) + \mathbf{v}_{ref} \tag{4.100}$$

Debido a la forma de las matrices A y B, el vector K habrá de ser de la forma $K = [k_1, k_2, k_3, k_4]$ para los sistemas de cuarto orden y $K = [k_1, k_2]$ para el sistema de segundo orden [11].

Según la expresión 4.100, las señales de control adquieren la forma

$$v_1 = k_{11}(x_{ref} - x) + k_{12}(\dot{x}_{ref} - \dot{x}) + k_{13}(\ddot{x}_{ref} - \ddot{x}) + k_{14}(\dddot{x}_{ref} - \dddot{x}) + x^{(4}_{ref} \tag{4.101}$$

$$v_2 = k_{21}(y_{ref} - y) + k_{22}(\dot{y}_{ref} - \dot{y}) + k_{23}(\ddot{y}_{ref} - \ddot{y}) + k_{24}(\dddot{y}_{ref} - \dddot{y}) + y^{(4}_{ref} \tag{4.102}$$

$$v_3 = k_{31}(z_{ref} - z) + k_{32}(\dot{z}_{ref} - \dot{z}) + k_{33}(\ddot{z}_{ref} - \ddot{z}) + k_{34}(\dddot{z}_{ref} - \dddot{z}) + z^{(4}_{ref} \tag{4.103}$$

$$v_4 = k_{41}(\psi_{ref} - \psi) + k_{42}(\dot{\psi}_{ref} - \dot{\psi}) + \ddot{\psi}_{ref} \tag{4.104}$$

Se realizaron distintos ensayos introduciendo las derivadas de las referencias en todas las variables hasta su grado máximo y se concluyó de manera pseudo-empírica que el funcionamiento del controlador muestra valores satisfactorios cuando se prealimenta tanto la referencia de posición como de velocidad, obviando el resto de derivadas. De esta manera, las nuevas entradas de control adquieren la siguiente formulación

$$v_1 = k_{11}(x_{ref} - x) + k_{12}(\dot{x}_{ref} - \dot{x}) - k_{13}\ddot{x} - k_{14}\dddot{x} \tag{4.105}$$

$$v_2 = k_{21}(y_{ref} - y) + k_{22}(\dot{y}_{ref} - \dot{y})) - k_{23}\ddot{y} - k_{24}\dddot{y} \tag{4.106}$$

$$v_3 = k_{31}(z_{ref} - z) + k_{32}(\dot{z}_{ref} - \dot{z}) - k_{33}\ddot{z} - k_{34}\dddot{z} \tag{4.107}$$

$$v_4 = k_{41}(\psi_{ref} - \psi) + k_{42}(\dot{\psi}_{ref} - \dot{\psi}) \tag{4.108}$$

Siguiendo la expresión 4.97, los polos de los sistemas vienen dados por la ecuación característica de la matriz en bucle cerrado $A - BK$

$$\det[\lambda I - (A - BK)] = 0 \tag{4.109}$$

Para los sistemas de grado 4

$$\det\left(\begin{bmatrix} -\lambda & 1 & 0 & 0 \\ 0 & -\lambda & 1 & 0 \\ 0 & 0 & -\lambda & 1 \\ -k_1 & -k_2 & -k_3 & -\lambda - k_4 \end{bmatrix}\right) = 0 \tag{4.110}$$

$$\lambda^4 + k_4\lambda^3 + k_3\lambda^2 + k_2\lambda + k_1 = 0 \tag{4.111}$$

y para el sistema de grado 2

$$\det\left(\begin{bmatrix} -\lambda & 1 \\ -k_1 & -\lambda - k_2 \end{bmatrix}\right) = 0 \tag{4.112}$$

$$\lambda^2 + k_2\lambda + k_1 = 0 \tag{4.113}$$

Haciendo uso de las expresiones 4.111 y 4.113 y especificando los valores de λ que generen la dinámica deseada, se calcularán las ganancias con los sistemas de ecuaciones resultantes. Para su implementación se ha construido la función de *Matlab* del código A.3.

El enfoque que se propone requiere que se conozcan todos los estados. Mientras que las posiciones y sus derivadas se conocen y se pueden extraer del modelo del sistema, las terceras y cuartas derivadas no. La derivación numérica suele amplificar el ruido

en el sistema, por lo que para su obtención se han utilizado derivadores filtrados de la forma

$$G_{derivador}(s) = \frac{s}{\tau s + 1} \tag{4.114}$$

e implementado en discreto mediante el bloque de *Simulink* que se muestra en la figura 4.7. A pesar de la nomenclatura en continuo, el bloque derivador con filtro pasa a un estado discreto si entre sus parámetros se introduce el periodo de muestreo.

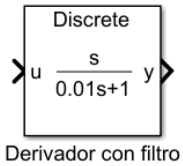

Figura 4.7: Bloque derivador utilizado.

4.3.1. Desarrollo del modelo FL

Para la simulación de esta metodología se ha construido en *Simulink* el diagrama de bloques que se presenta en la figura 4.8. Para dar más claridad a la explicación del modelo de simulación, se han coloreado sus componentes fundamentales.

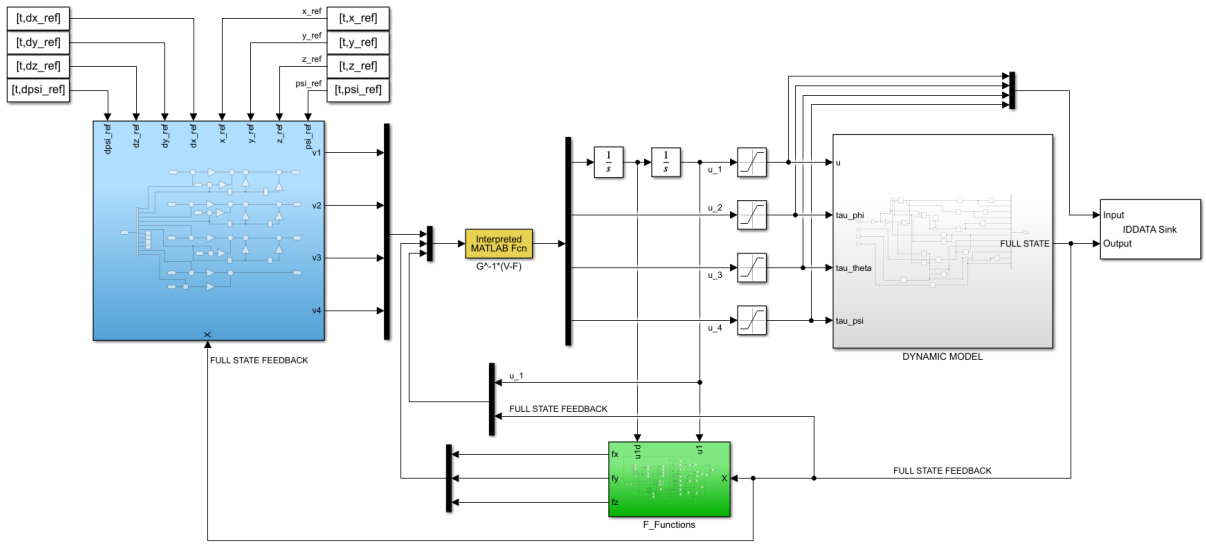

Figura 4.8: Modelo de *Simulink* del dron con el controlador FL.

En azul se recoge el subsistema en el que se implementa la ley de control definida por las expresiones 4.105, 4.106, 4.107 y 4.108. Este bloque se muestra en detalle en la figura 4.9.

El bloque verde de la figura 4.8 aloja las funciones f_x, f_y y f_z obtenidas para realizar la expansión dinámica del sistema. Estas funciones se recogen en las expresiones 4.77,

4.78 y 4.79 y se muestra en la figura 4.10.

El bloque blanco recoge el modelo dinámico del dron presentado en la figura 4.4 y empleado para la simulación de las dos metodologías de control.

Finalmente, la función que se recoge en el bloque amarillo implementa las funciones 4.82, 4.83, 4.84 y 4.85, así como la obtención final de las variables de control según 4.86. El contenido de esta función se muestra en el código 4.2.

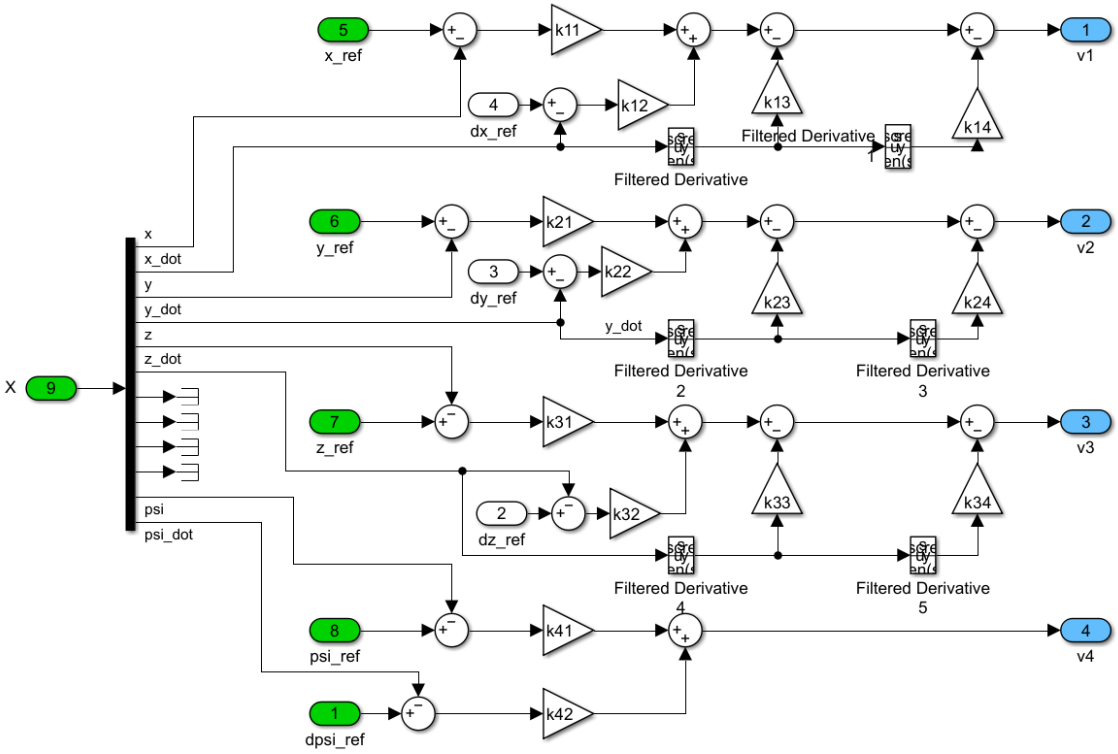

Figura 4.9: Detalle del bloque azul de la figura 4.8.

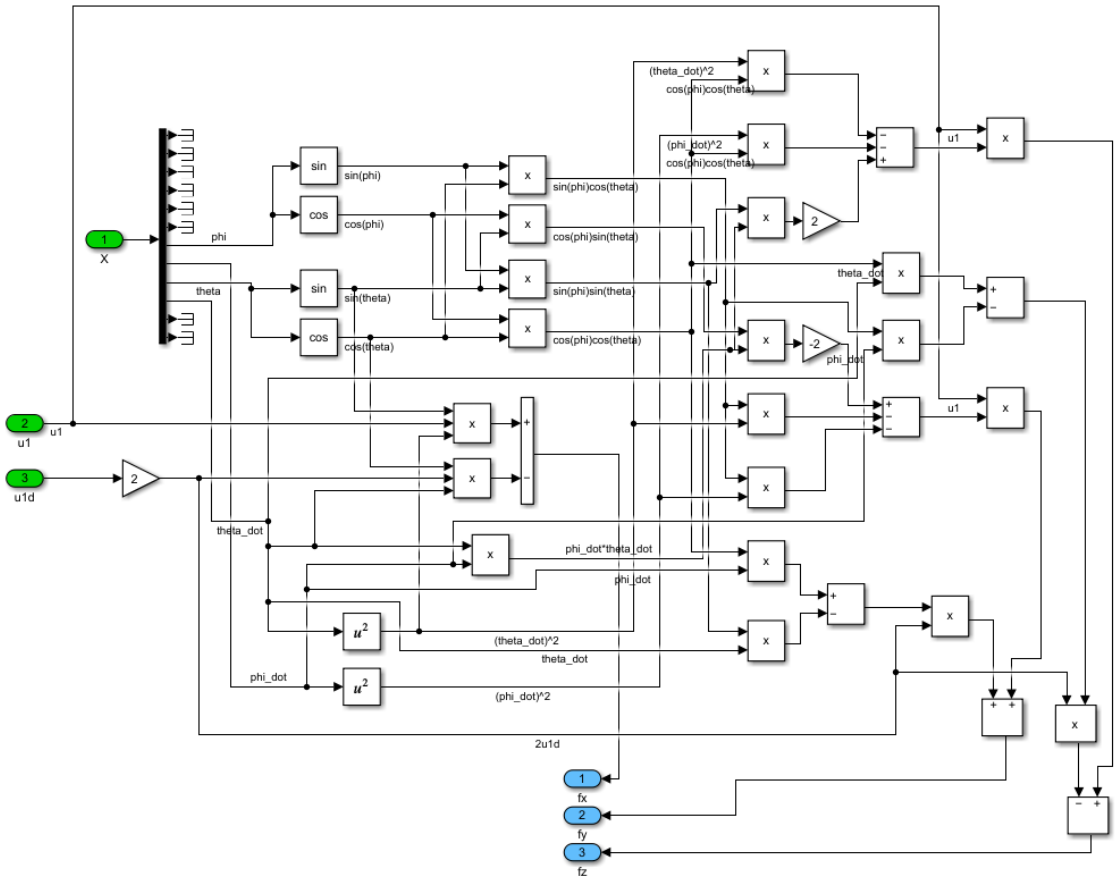

Figura 4.10: Detalle del bloque verde de la figura 4.8.

```
function U = FL_controller(X)
    global i

    % New control imputs
    v1 = X(1);
    v2 = X(2);
    v3 = X(3);
    v4 = X(4);

    % F functions
    fx = X(5);
    fy = X(6);
    fz = X(7);

    % V-f vector
    v_f = [v1-fx v2-fy v3-fz v4];

    % Control input
    u1 = X(8);

    % State variables
    phi   = X(15);
    theta = X(17);

    % g functions
    gxu1 = -sin(theta);
    gxu2 = 0;
    gxu3 = -cos(theta)*u1;
    gxu4 = 0;

    gyu1 = cos(theta)*sin(phi);
    gyu2 = cos(theta)*cos(phi)*u1;
    gyu3 = -sin(theta)*sin(phi)*u1;
    gyu4 = 0;

    gzu1 = cos(theta)*cos(phi);
    gzu2 = -sin(phi)*cos(theta)*u1;
    gzu3 = -sin(theta)*cos(phi)*u1;
    gzu4 = 0;

    % Matrix
    G = [gxu1 gxu2 gxu3 gxu4;
         gyu1 gyu2 gyu3 gyu4;
         gzu1 gzu2 gzu3 gzu4;
         0    0    0      1];
```

```
46
47      U = G\v_f';
48
49      i=i+1;
50  end
```

Código 4.2: Código empleado para obtener las entradas de control del modelo dinámico según 4.86.

5. Simulaciones

5.1. Entorno de simulación 1

Para la comparación del desempeño de las metodologías se ha elaborado una trayectoria que se compone de una primera parte helicoidal de duración 120 s centrada en el punto $(x, y) = (1, 3)$ m con un radio de $R = 2$ m y de una segunda parte en forma de descenso vertical hasta el suelo de duración 50 s. La posición inicial para el dron se encuentra en el punto $(x_0, y_0, z_0) = (2, 0, 0)$ m y la velocidad en el eje Z será de $v_{z,1} = 0.5$ m/s. Para la orientación, el dron girará sobre su propio eje con una velocidad angular de $\omega = 0.2$ rads/s. Esta trayectoria viene definida por las expresiones que se muestran en 5.1 y su representación gráfica se puede observar en las figuras 5.1 y 5.2.

$$\begin{aligned}
&\text{Para} \rightarrow 0\,\text{s} \leqslant t \leqslant 120\,\text{s} \begin{cases} x_1 = 1 + R \cdot \cos\left(\omega t + \frac{\pi}{2}\right) \\ y_1 = 3 + R \cdot \sin\left(\omega t - \frac{\pi}{2}\right) \\ z_1 = 1 + v_{z,1} \cdot t \\ \psi_1 = \omega \cdot t \end{cases} \\
&\text{Para} \rightarrow 120\,\text{s} < t \leqslant 170\,\text{s} \begin{cases} x_2 = x_{1,f}\,[\text{cte}] \\ y_2 = y_{1,f}\,[\text{cte}] \\ z_2 = z_{1,f} + v_{z,2} \cdot t \\ \psi_2 = \psi_{1,f}\,[\text{cte}] \end{cases}
\end{aligned} \tag{5.1}$$

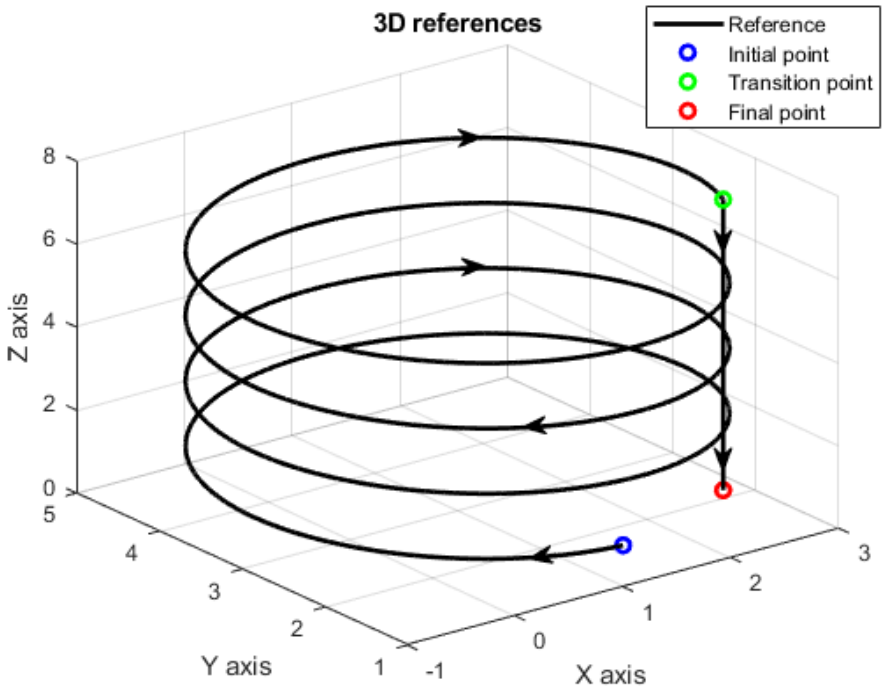

Figura 5.1: Representación tridimensional de la trayectoria creada.

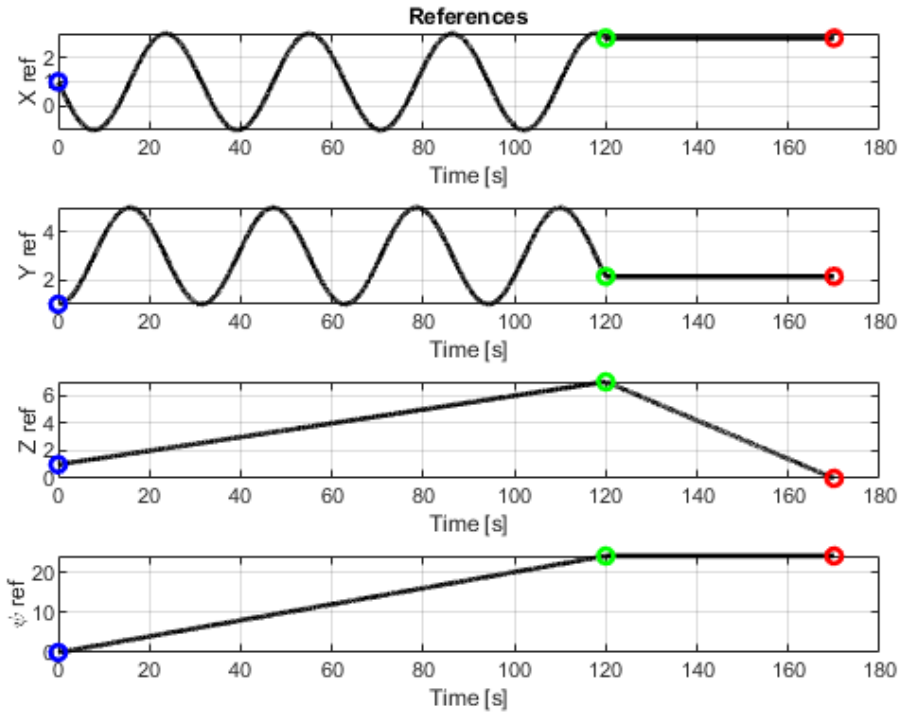

Figura 5.2: Representación en función del tiempo de la trayectoria creada.

Como se menciona en el apartado 4.3.1, en el caso del controlador FL se prealimentan también las derivadas de las referencias

$$\begin{aligned}
&\text{Para} \rightarrow 0\,\text{s} \leqslant t \leqslant 120\,\text{s} \begin{cases} \dot{x}_1 = -\omega \cdot R \cdot \sin\left(\omega t + \frac{\pi}{2}\right) \\ \dot{y}_1 = \omega \cdot R \cdot \cos\left(\omega t - \frac{\pi}{2}\right) \\ \dot{z}_1 = v_{z,1} \\ \dot{\psi}_1 = \omega \end{cases} \\
&\text{Para} \rightarrow 120\,\text{s} < t \leqslant 170\,\text{s} \begin{cases} \dot{x}_2 = 0 \\ \dot{y}_2 = 0 \\ \dot{z}_2 = v_{z,2} \\ \dot{\psi}_2 = 0 \end{cases}
\end{aligned} \tag{5.2}$$

Las constantes del proceso en este caso son la masa y la aceleración de la gravedad. Para ambos casos se han asumido de la siguiente manera

$$\begin{cases} M = 1\,\text{kg} \\ g = 9,81\,\text{m/s}^2 \end{cases} \tag{5.3}$$

Se ha seleccionado un periodo de muestreo de $T_s = 20\,\text{ms}$. La elección de este periodo de muestreo se ha realizado de manera simultánea con la calibración de los parámetros de ambos reguladores. Cabe destacar que periodos de muestreo excesivamente pequeños disparaban las acciones de control en el controlador LAB

debido a que en el modelo discreto construido, todas las derivadas se calculan como una diferencia dividida por este periodo de muestreo. En el controlador FL, según más elevado era dicho periodo, menor tenía que ser la magnitud de los polos y, por ende, el valor final de las ganancias. Finalmente, con el periodo de muestreo mencionado de 20 ms se han obtenido resultados satisfactorios tanto a nivel de acciones de control como de errores de posición y cuadráticos a lo largo de toda la trayectoria.

5.1.1. Simulaciones

Para verificar el funcionamiento de los modelos, en primer lugar se realizaron una serie de simulaciones para el ajuste tanto de las constantes proporcionales del regulador LAB como para el posicionamiento de los polos del controlador FL.

5.1.1.1 Controlador LAB

El desarrollo de este controlador es el justificado en el apartado 4.2.2. Su funcionamiento viene definido por el establecimiento de 12 ganancias, una para cada estado. Tras un ajuste de las mismas basado en un método heurístico y buscando al mismo tiempo reducir las acciones de control y los errores cuadráticos medios para el seguimiento de las cuatro referencias, se han obtenido las siguientes ganancias

$$\begin{aligned} k_1 &= 0.99 \quad k_2 = 0.97 \\ k_3 &= 0.99 \quad k_4 = 0.98 \\ k_5 &= 0.99 \quad k_6 = 0.79 \\ k_7 &= 0.96 \quad k_8 = 0.91 \\ k_9 &= 0.96 \quad k_{10} = 0.92 \\ k_{11} &= 0.99 \quad k_{12} = 0.95 \end{aligned} \tag{5.4}$$

Como se ha mencionado, las simulaciones del controlador se han llevado a cabo para un periodo de muestreo de $T_s = 0.02\,\text{s}$. Los resultados obtenidos para las posiciones se muestran en las figura 5.3 y 5.4.

Los errores cuadráticos obtenidos a lo largo de toda la trayectoria para las variables de seguimiento son los que se muestran en la tabla 5.1.

Tabla 5.1: Errores cuadráticos medios obtenidos para con el controlador LAB.

Error X	$1.77 \cdot 10^{-2}\,\text{m}$
Error Y	$2.13 \cdot 10^{-2}\,\text{m}$
Error Z	$6.18 \cdot 10^{-3}\,\text{m}$
Error ψ	$7.195 \cdot 10^{-5}\,\text{rad}$

Según se puede observar en la tabla 5.1, los errores cuadráticos medios en el seguimiento de la trayectoria no superan un orden de magnitud de 10^{-2}. Esto resulta

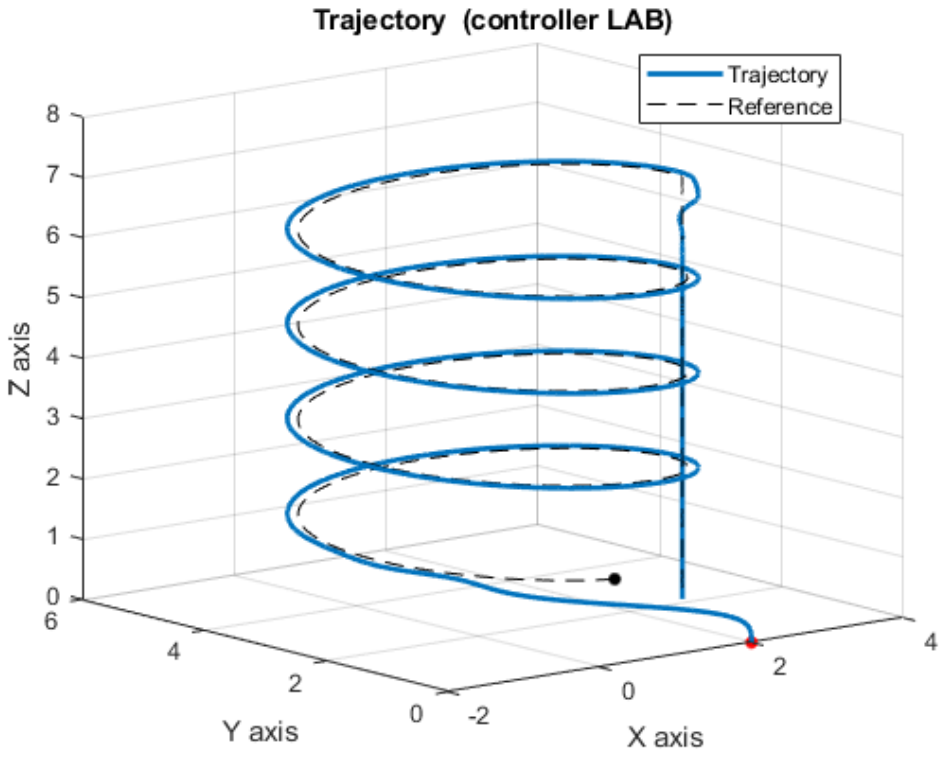

Figura 5.3: Representación tridimensional de la trayectoria seguida por el dron con controlador LAB.

óptimo puesto que se ha de tener en cuenta que se está considerando que el dron es una masa concentrada en un punto coincidente con su centro de masas, de manera que un dron con una longitud máxima entre extremos de 20 cm estará siguiendo la trayectoria casi perfectamente.

Según se observa en la figura 5.5, las acciones de control que se generan con este controlador se encuentran dentro de unos márgenes razonables. El valor máximo de la acción de control u resulta de 15.68 N. Si se tiene en cuenta las constantes establecidas en 5.3, este valor máximo apenas supera 1.5 veces el peso del dron, lo cual explica el ascenso inicial entre el punto de partida y el punto inicial de la trayectoria. Adicionalmente, el valor medio de esta misma acción de control durante el tramo de ascenso en espiral se ha establecido en 9.8115 N, ligeramente superior que el peso del dron (9.81 N). Esta diferencia es suficiente para que el dron ascienda en el eje Z con una velocidad de 0.05 m/s. Asimismo, el valor medio de la acción de control u durante el segundo tramo es ligeramente inferior al peso del dron, lo que explica el descenso.

El resto de acciones de control se miden en Nm y como se observa, muestran picos máximos y un comportamiento oscilatorio en los momentos transitorios. Las entradas $\tilde{\tau}_\phi$ y $\tilde{\tau}_\theta$ se equilibran alrededor de $\pm 10^{-3}$ Nm durante el primer tramo de la trayectoria y disminuyen progresivamente ante la entrada constante del segundo tramo hasta alcanzar un orden de magnitud de $\pm 10^{-10}$ Nm.

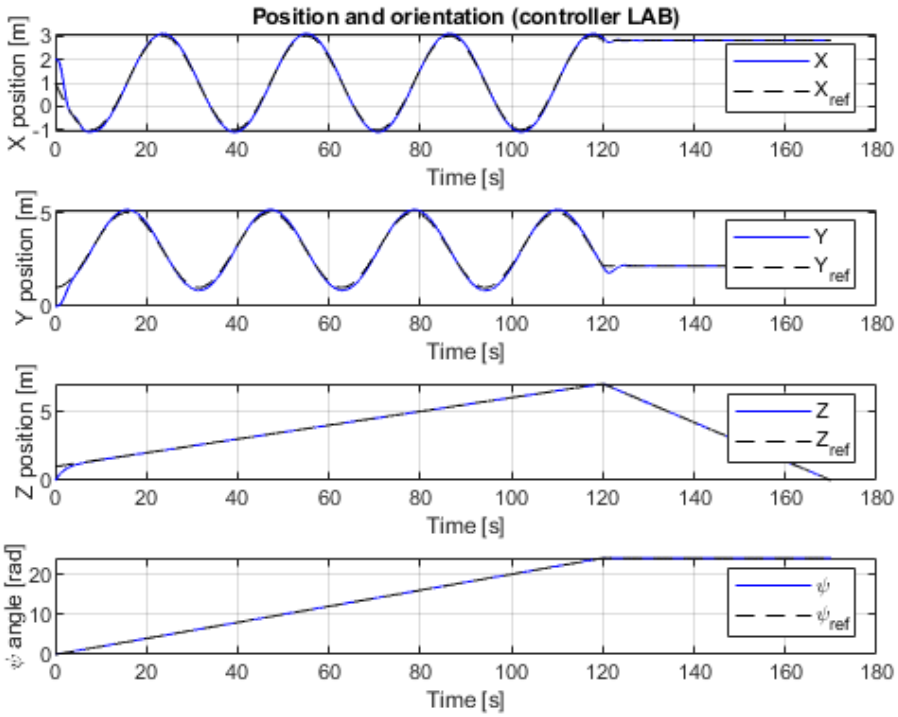

Figura 5.4: Posiciones obtenidas por el dron con el controlador LAB.

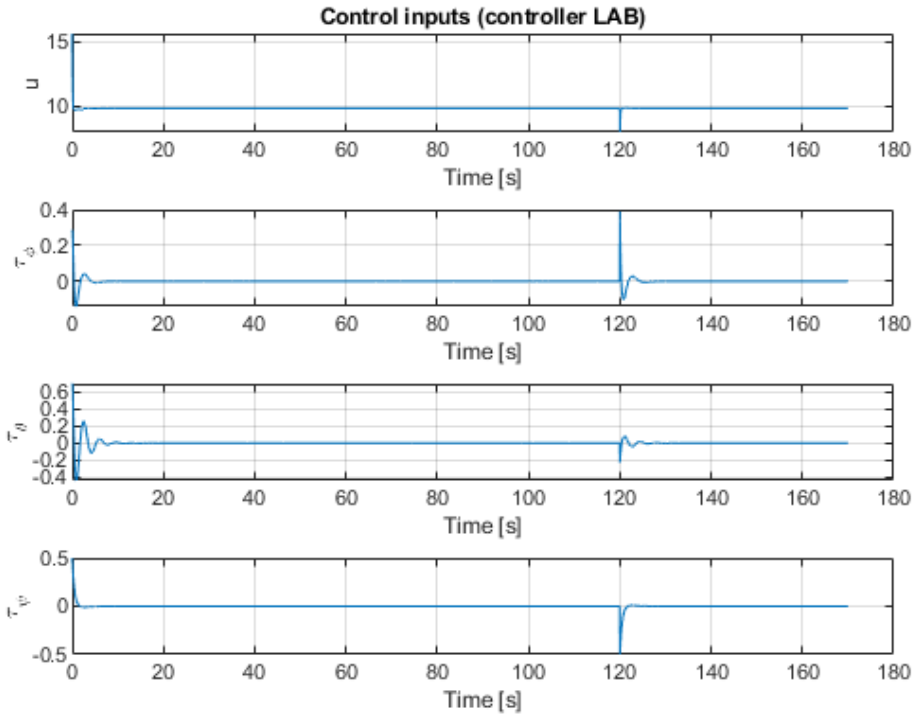

Figura 5.5: Acciones de control aplicadas al dron por el controlador LAB para el seguimiento de la trayectoria.

5.1.1.2 Controlador FL

El desarrollo de este controlador es el presentado en el apartado 4.3.1. Las variables de control sobre las que se actúa en este caso son los polos de los 4 subsistemas que se contemplan en la ley de control. Se han establecido los siguientes polos para obtener las ganancias del modelo y se han ajustado siguiendo un método heurístico

$$
\begin{aligned}
\mathrm{X} \rightarrow & \begin{cases} p_1 = -0.5 \\ p_2 = -2.4 \\ p_3 = 1.1i \\ p_4 = -1.1i \end{cases} \\
\mathrm{Y} \rightarrow & \begin{cases} p_1 = -0.7 \\ p_2 = -1.8 \\ p_3 = -i \\ p_4 = i \end{cases} \\
\mathrm{Z} \rightarrow & \begin{cases} p_1 = -0.6 \\ p_2 = -2.3 \\ p_3 = -1.2i \\ p_4 = 1.2i \end{cases} \\
\Psi \rightarrow & \begin{cases} p_1 = -1 \\ p_2 = -0.5 \end{cases}
\end{aligned}
\tag{5.5}
$$

En este caso, es necesario mencionar que las expresiones definidas de 4.82 a 4.84 se encuentran multiplicadas por la entrada de control u_1. Estas expresiones conforman la matriz **G** que ha de ser invertible en todo momento, por lo que se ha de definir un valor inicial de $u \neq 0$ para que el modelo funcione. Tras diversas pruebas, se ha establecido como valor inicial el propio peso del dron $u_0 = M \cdot g$. Este valor inicial de u se ha introducido en el segundo integrador que precede a la entrada al modelo dinámico que se observa en la figura 4.8.

Los resultados gráficos obtenidos se muestran en las figuras 5.6 y 5.7. Asimismo, los valores de errores cuadráticos medios para todas las referencias seguidas se muestran en la tabla 5.2.

De manera similar al caso anterior, los errores cuadráticos medios que se observan en la tabla 5.2 no exceden el orden de magnitud de 10^{-2}, lo que garantiza un seguimiento de la trayectoria satisfactorio. Por otra parte, cabe destacar que el error en el seguimiento de las referencias Z y ψ sí que es ciertamente peor en comparación con el caso anterior.

De nuevo en este caso, los valores máximos de las acciones de control en los periodos transitorios de la trayectoria no resultan elevados. Cabe destacar que los valores máximos de estas entradas resultan ligeramente menores que en el caso anterior, lo que puede significar que el seguimiento de la referencia cuando suceden cambios

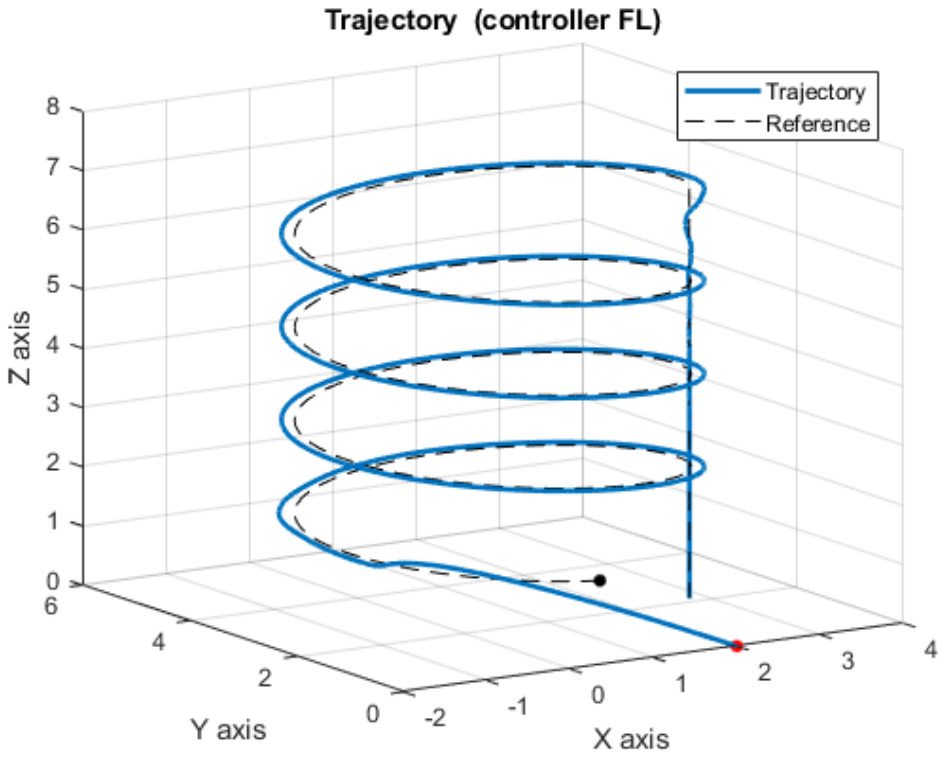

Figura 5.6: Representación tridimensional de la trayectoria seguida por el dron con el controlador FL.

bruscos en la misma es peor. Asimismo, durante el primer tramo de la trayectoria el valor medio de la acción u resulta el mismo que en el caso anterior, y en el segundo tramo es ligeramente inferior, aunque la diferencia no resulta significativa.

Tabla 5.2: Errores cuadráticos medios obtenidos para el controlador FL.

Error X	$2.40 \cdot 10^{-2}$ m
Error Y	$2.12 \cdot 10^{-2}$ m
Error Z	$1.10 \cdot 10^{-2}$ m
Error ψ	$3.25 \cdot 10^{-4}$ rad

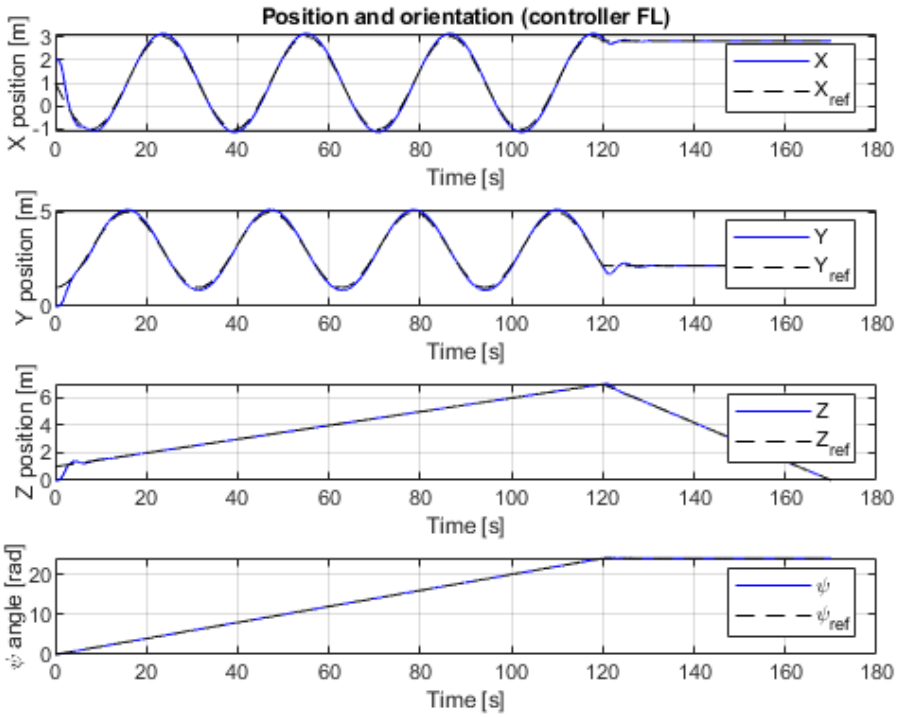

Figura 5.7: Posiciones obtenidas por el dron con el controlador FL.

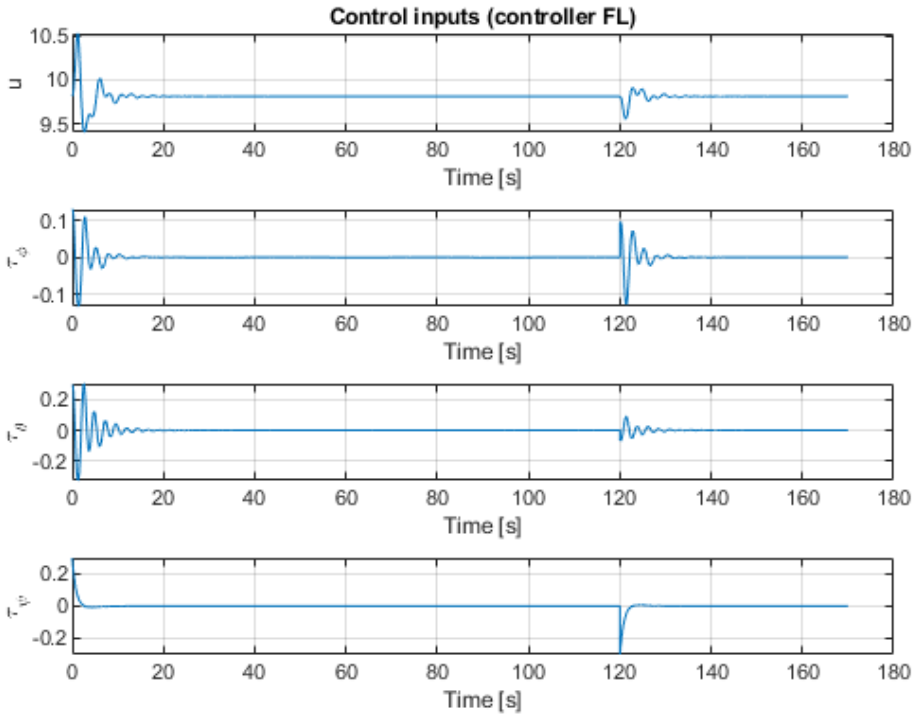

Figura 5.8: Acciones de control aplicadas al dron por el controlador FL para el seguimiento de la trayectoria.

5.1.2. Comparación de ambas metodologías

En este apartado se realizará una comparación cualitativa y cuantitativa de los resultados obtenidos tras aplicar las dos metodologías de control. De manera inicial, se presenta en la figura 5.9 los resultados tridimensionales obtenidos. En azul se muestra el resultado del controlador LAB y en rojo el resultado del controlador FL.

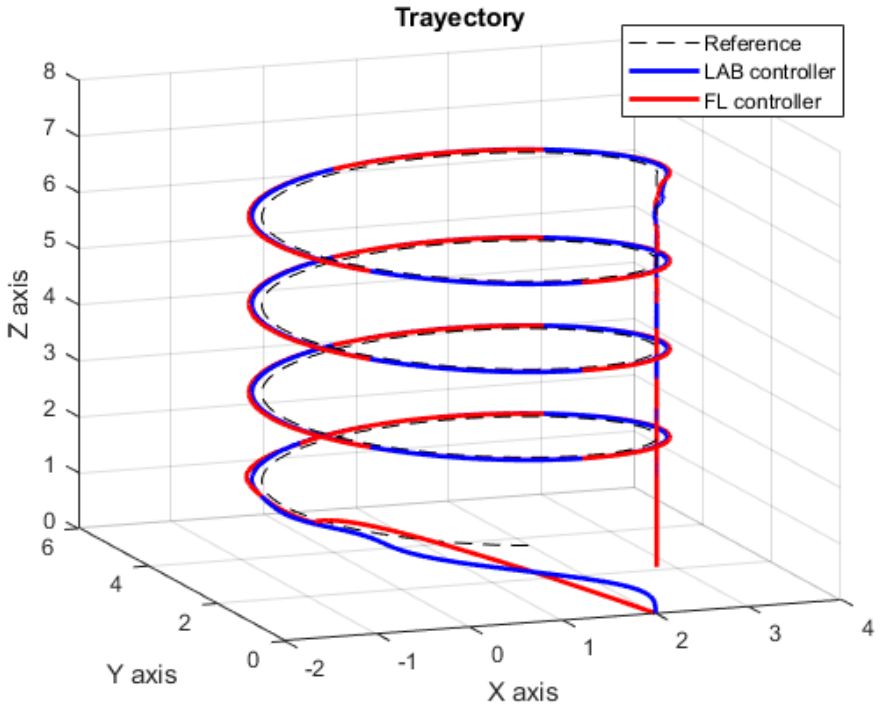

Figura 5.9: Representación tridimensional de los resultados obtenidos con los dos controladores.

A grandes rasgos, se observa en las figuras 5.9 y 5.10 que los resultados alcanzados por ambos controladores resultan similares en términos de posiciones. Si se comparan las tablas de errores cuadráticos 5.1 y 5.2, se puede percibir que el control LAB es superior en el seguimiento de Z y ψ y similar en X e Y.

Por otra parte, según los datos de error de posición representados en la figura 5.11 se ve que el valor de estos resulta notablemente pequeño para las variables que siguen referencias estables como son Z y ψ. Por otra parte, para las variables X e Y, se genera un poco de error de posición durante la fase de la trayectoria helicoidal y dicho error tiende a 0 cuando la referencia se vuelve constante. La magnitud de este error para estas variables en el primer tramo no supera en ninguno de los casos los $\pm 0.15\,\text{m}$.

De la figura 5.12 se puede observar que durante los transitorios, la oscilación en los ángulos de alabeo y cabeceo es mayor para el caso del control FL. Esto se ve reflejado en las figuras 5.13 y 5.14, donde en estos mismos transitorios, las entradas de control muestran mayor oscilación en el caso del control FL.

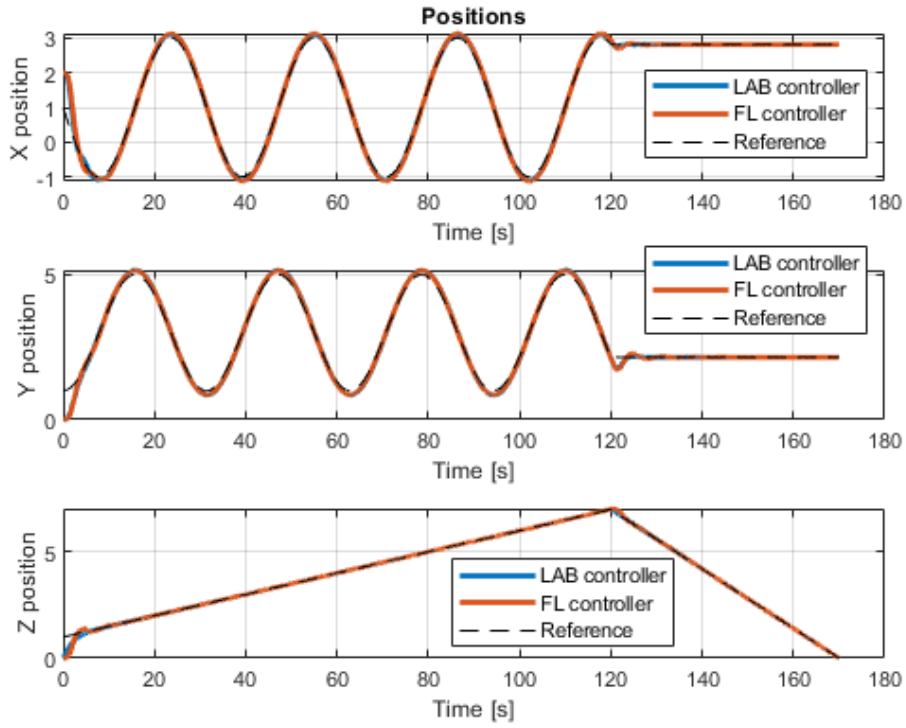

Figura 5.10: Posiciones obtenidas con ambos controladores.

Atendiendo a las entradas de control que se pueden observar en la figura 5.13, es notable que la magnitud máxima para el controlador LAB es superior que para el controlador FL. Salvo el caso de la u, el resto de acciones oscila sobre 0 tomando valores positivos y negativos, con lo cual hacer una lectura de su valor medio no resultaría lógico. En cambio, si se procede a realizar una integral del valor absoluto de cada magnitud para obtener la energía de control se puede ver que la diferencia total entre ambas resulta inferior para el caso del controlador LAB. Esto resulta lógico atendiendo a la figura 5.14, que muestra un *zoom* de dos de las señales de la figura 5.13 ($\tilde{\tau}_\phi$ y $\tilde{\tau}_\theta$). Las oscilaciones que se generan en estas señales con el control FL son considerablemente mayores, así como el tiempo que transcurre hasta que se estabilizan. Estas diferencias entre las energías y acciones de control pueden ser investigada con el fin de justificar un ahorro de baterías y un aumento de autonomía en este tipo de vehículos tal y como se comenta en el apartado 6.

Tabla 5.3: Integral de las acciones de control para la primera trayectoria.

	Control LAB	Control FL
E u	$8.3373 \cdot 10^4$	$8.3376 \cdot 10^4$
E $\tilde{\tau}_\phi$	24.028	35.292
E $\tilde{\tau}_\theta$	63.124	69.408
E $\tilde{\tau}_\psi$	24.351	24.996

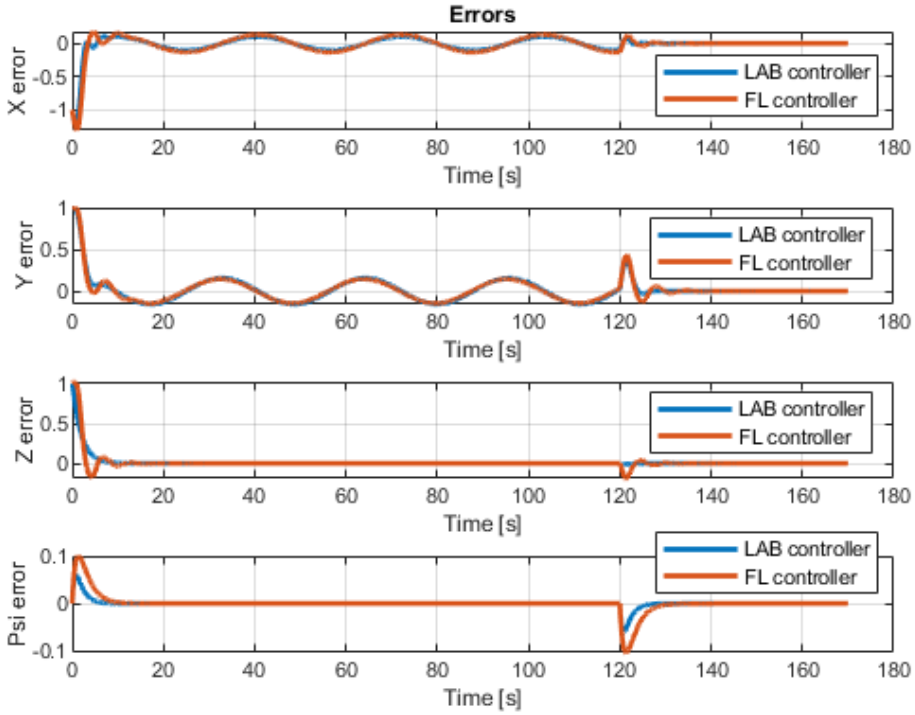

Figura 5.11: Errores obtenidos con ambos controladores.

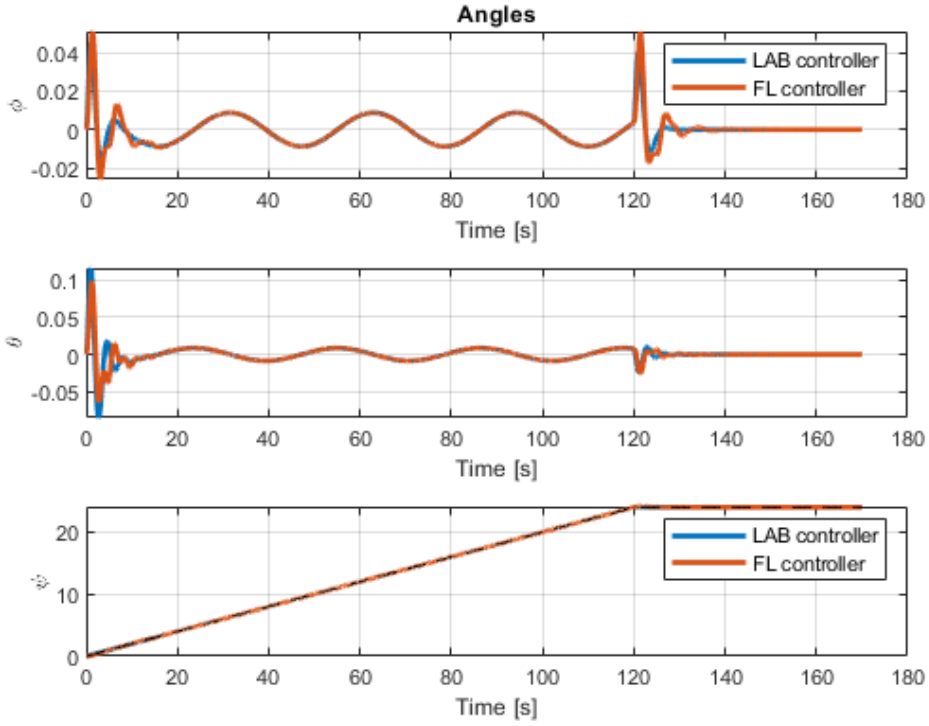

Figura 5.12: Ángulos obtenidos con ambos controladores.

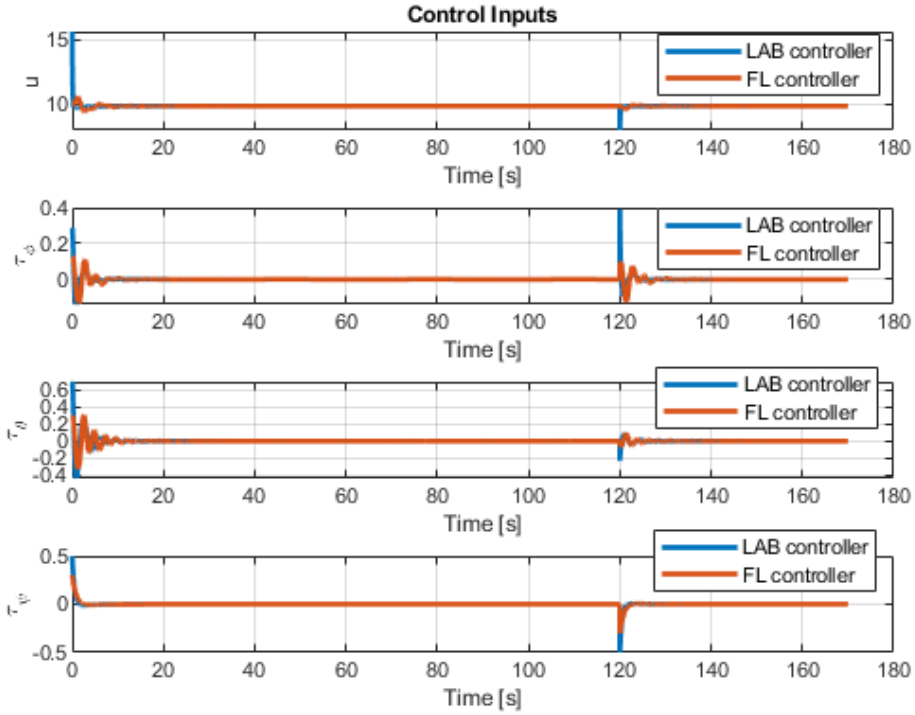

Figura 5.13: Entradas de control generadas con ambos controladores.

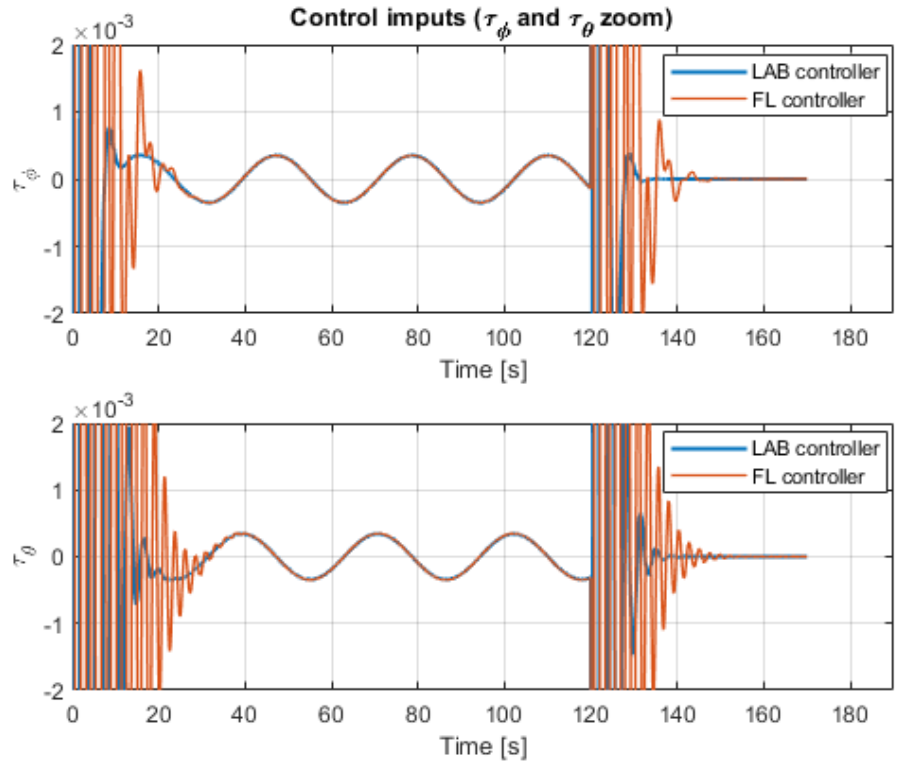

Figura 5.14: Detalle de la figura 5.13.

5.1.2.1 Análisis del primer transitorio de la simulación.

Se presentan en esta sección detalles aislados de la simulación durante el transitorio inicial donde el dron parte de un punto inicial y busca la trayectoria.

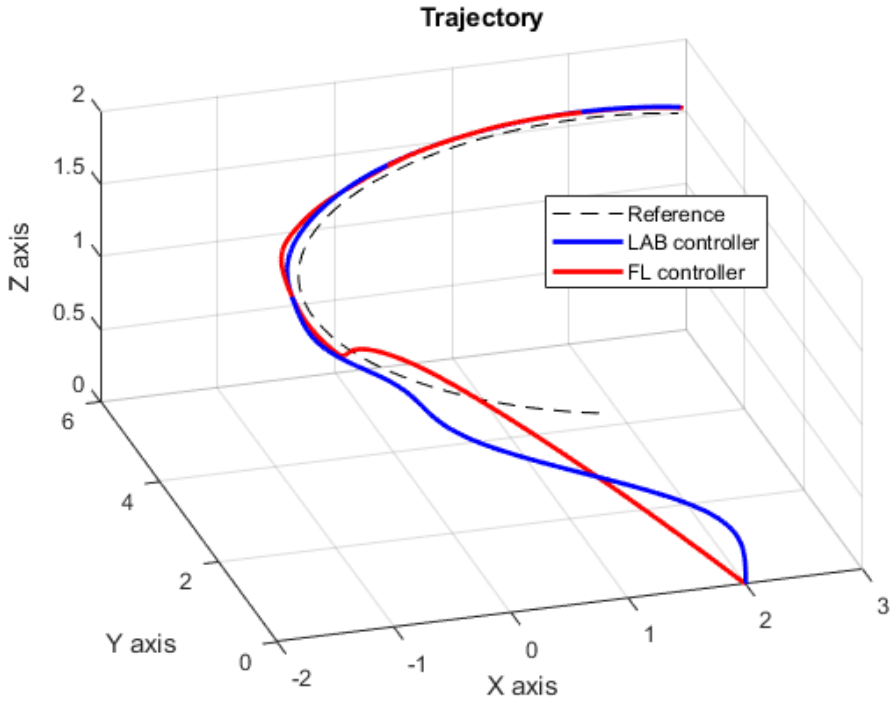

Figura 5.15: Representación tridimensional de la trayectoria durante el primer transitorio.

Lo característico de este transitorio es analizar como se comportan los controladores en los instantes iniciales del seguimiento de la trayectoria. Resulta interesante destacar que el controlador LAB, de la manera en la que está implementado, parte de las posiciones iniciales y genera directamente acciones de control en la primera iteración, mientras que el controlador FL necesita que se introduzca la condición inicial de la acción de control u para empezar a funcionar. Esto se puede traducir en que el valor inicial de esta acción de control difiere notablemente (figura 5.17) y que la evolución del seguimiento de la variable Z es mejor en el caso del LAB ya que no se incurre en ningún tipo de sobreoscilación, como se observa en la figura 5.16. En el Anexo II se recogen unas gráficas que justifican la elección del valor inicial de u para el controlador FL.

Los errores cuadráticos que se presentan también resultan mejores en el caso del controlador LAB. Si se observa la figura 5.17 se puede deducir que las acciones de control en este caso alcanzan valores máximos mayores que con el controlador FL que hacen que la aproximación inicial a la trayectoria sea mejor.

A pesar de que los valores máximos de las entradas de control sean superiores en el caso del controlador LAB, se puede observar en la figura 5.17 que la tendencia

Tabla 5.4: Errores cuadráticos medios durante el primer transitorio.

	Control LAB	Control FL
Error X	$1.23 \cdot 10^{-1}$ m	$1.59 \cdot 10^{-1}$ m
Error Y	$1.09 \cdot 10^{-1}$ m	$1.12 \cdot 10^{-1}$ m
Error Z	$5.24 \cdot 10^{-2}$ m	$9.11 \cdot 10^{-2}$ m
Error ψ	$3.06 \cdot 10^{-5}$ rad	$1.34 \cdot 10^{-3}$ rad

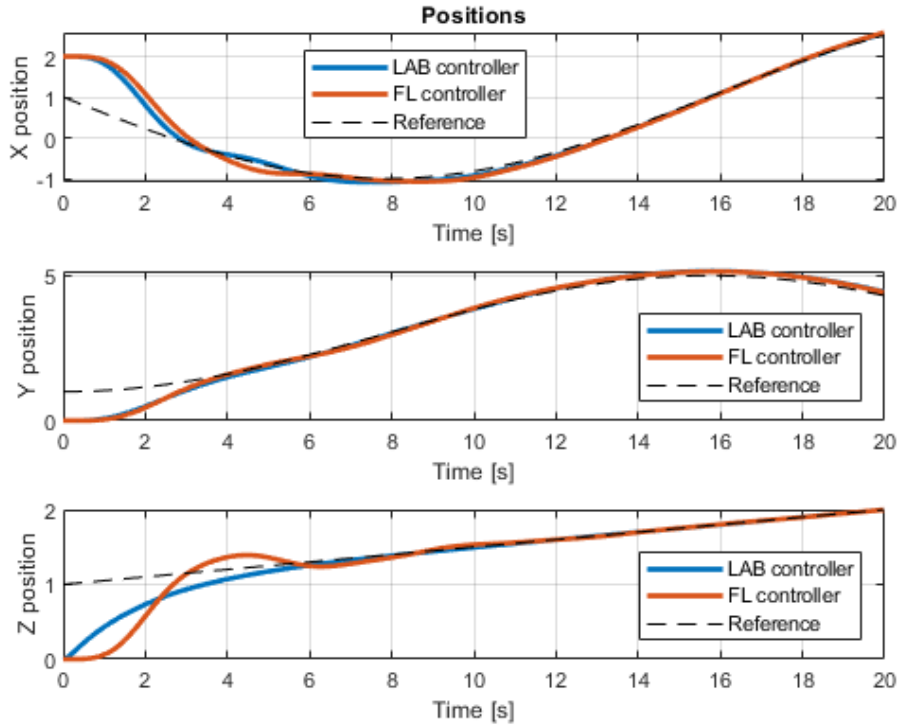

Figura 5.16: Posiciones obtenidas durante el primer transitorio.

de estas señales es menos oscilatoria que en el caso del control FL. Esta diferencia entre la magnitud de las acciones de control para ambos controladores también se traduce en una diferencia entre el consumo de energía de las baterías tal y como se ha mencionado en el apartado anterior.

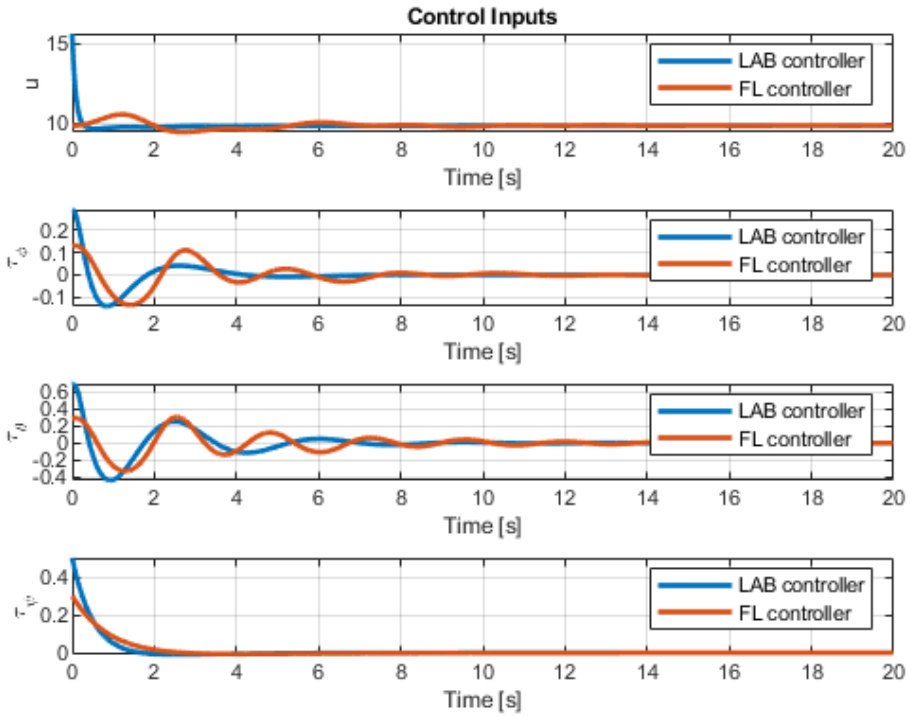

Figura 5.17: Acciones de control generadas durante el primer transitorio.

5.1.2.2 Análisis del segundo transitorio de la simulación.

Este segundo transitorio se corresponde al cambio brusco de la trayectoria entre los dos tramos para evaluar el comportamiento dinámico del dron ante cambios abruptos en las referencias.

Observando la figura 5.18, se observa que ambos controladores muestran una respuesta subamortiguada ante el cambio de referencia. Atendiendo a los datos de errores cuadráticos medios en este transitorio que se muestran en la tabla 5.5, se puede deducir que el controlador LAB es capaz de corregir mejor estos cambios al presentar menores errores cuadráticos medios.

Tabla 5.5: Errores cuadráticos medios durante el segundo transitorio.

	Control LAB	Control FL
Error X	$3.46 \cdot 10^{-3}$ m	$6.47 \cdot 10^{-3}$ m
Error Y	$1.66 \cdot 10^{-2}$ m	$2.32 \cdot 10^{-2}$ m
Error Z	$1.84 \cdot 10^{-5}$ m	$3.52 \cdot 10^{-3}$ m
Error ψ	$4.07 \cdot 10^{-4}$ rad	$1.90 \cdot 10^{-3}$ rad

Los valores máximos en las acciones de control de la figura 5.20 están recogidos en la tabla 5.6 y justifican que la corrección en el seguimiento de la trayectoria frente al

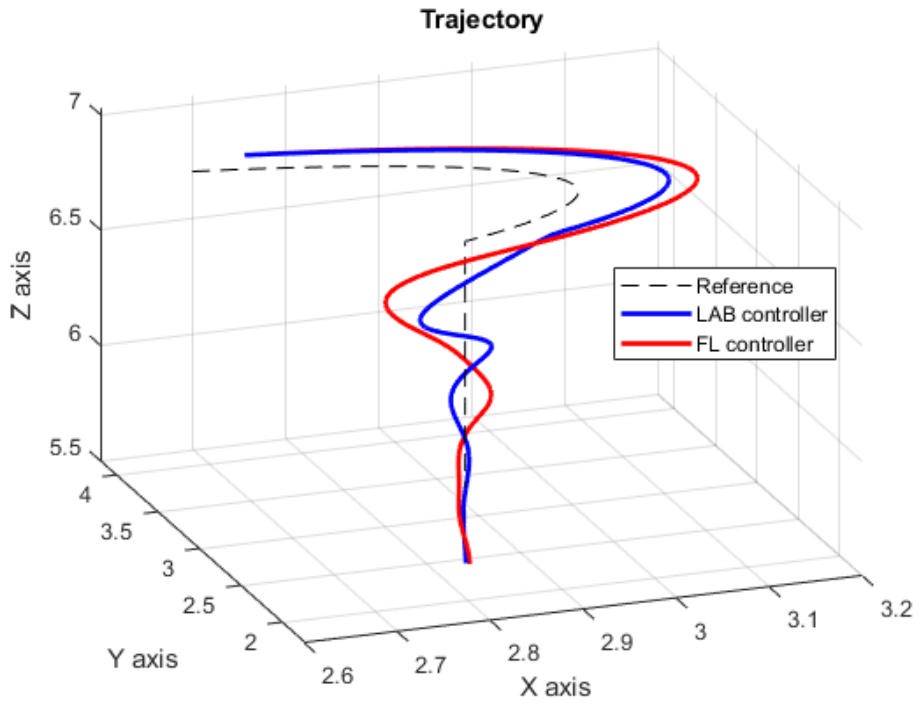

Figura 5.18: Representación tridimensional de la trayectoria durante el segundo transitorio.

cambio se realiza mejor con el control LAB al resultar ligeramente superiores.

Asimismo, de la figura 5.19 se concluye que ante entradas constantes, el error de posición que obtienen ambos controladores tiende a 0. Aún así, se percibe que en ambos casos, el controlador FL tiene transitorios sensiblemente más largos, y por lo tanto, tiempos de establecimiento mayores.

Tabla 5.6: Valores de los picos de las acciones de control durante el segundo transitorio.

	Control LAB	Control FL
u	7.93 N	9.55 N
$\tilde{\tau}_\phi$	$3.91 \cdot 10^{-1}$ Nm	$9.71 \cdot 10^{-2}$ Nm
$\tilde{\tau}_\theta$	$-2.31 \cdot 10^{-1}$ Nm	$-6.85 \cdot 10^{-2}$ Nm
$\tilde{\tau}_\psi$	$-5.00 \cdot 10^{-1}$ Nm	$-3.02 \cdot 10^{-1}$ Nm

Llegados a este punto, se puede plantear la posibilidad de que el hecho de forzar unas condiciones de simulación similares en términos de periodo de muestreo puedan afectar al rendimiento en el control ya que durante el proceso de sintonización y construcción del entorno, se observó que la metodología FL ofrecía mejores resultados con periodos de muestreo pequeños para la trayectoria dada. Por el contrario, según se disminuye el periodo de muestreo, la sintonización de las ganancias del controlador LAB volvía la simulación del controlador más sensible llegando en ocasiones a no converger.

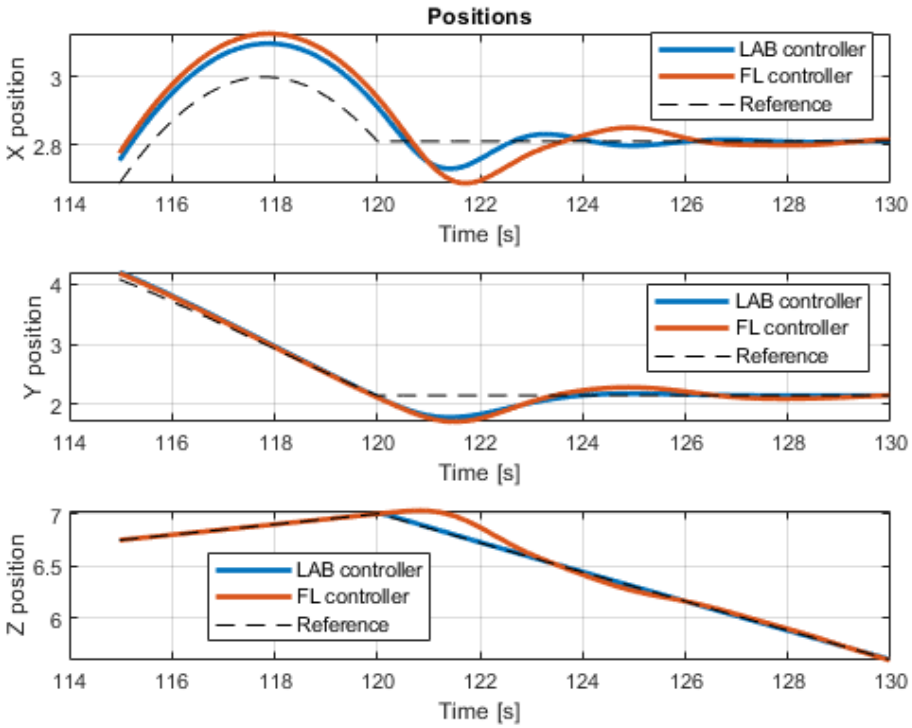

Figura 5.19: Posiciones obtenidas durante el segundo transitorio.

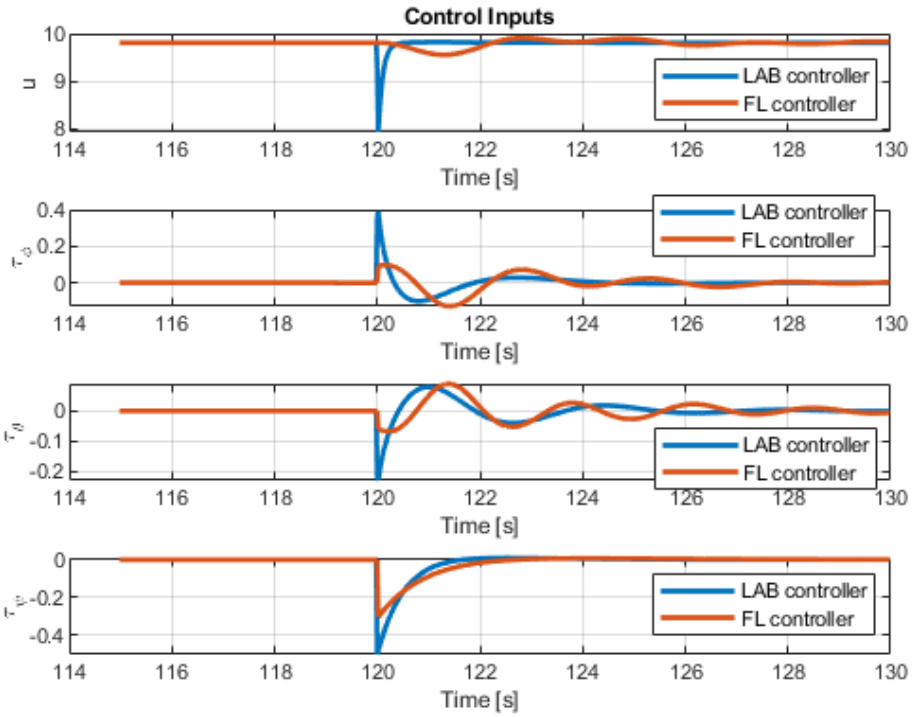

Figura 5.20: Acciones de control generadas durante el segundo transitorio.

5.2. Entorno de simulación 2

Con el fin de continuar la comparación de ambos controladores y siguiendo la metodología para la generación de trayectorias que se describe en el Anexo IV, se ha construido la trayectoria que se muestra en la figura 5.21. Con esta trayectoria se busca principalmente reducir los efectos de los transitorios en el final de cada tramo estableciendo como referencia la evolución de la velocidad, siendo esta nula en el inicio y final de cada tramo. Esto se puede observar de manera clara en la figura D.1 del anexo.

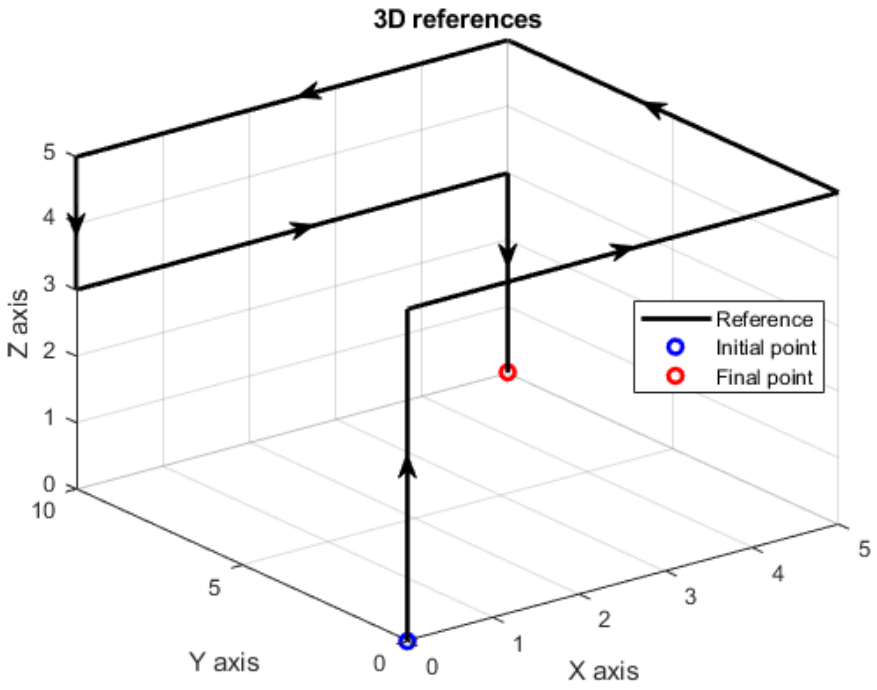

Figura 5.21: Representación tridimensional de la segunda trayectoria.

En este caso, se ha elegido para las simulaciones y la sintonización de ambos reguladores un periodo de muestreo de $T_s = 10\,\text{ms}$. La elección de este periodo de muestreo se justifica en que para periodos mayores, el controlador FL mostraba un seguimiento más lento de la referencia. Esto se podría corregir con una prealimentación de la aceleración de las referencias, pero se observó que para este periodo elegido, el error se reducía notablemente.

El controlador LAB viene definido por las siguientes 12 ganancias

$$\begin{aligned} k_1 &= 0.99 \quad k_2 = 0.89 \\ k_3 &= 0.99 \quad k_4 = 0.89 \\ k_5 &= 0.99 \quad k_6 = 0.79 \\ k_7 &= 0.85 \quad k_8 = 0.81 \\ k_9 &= 0.85 \quad k_{10} = 0.81 \\ k_{11} &= 0.85 \quad k_{12} = 0.95 \end{aligned} \tag{5.6}$$

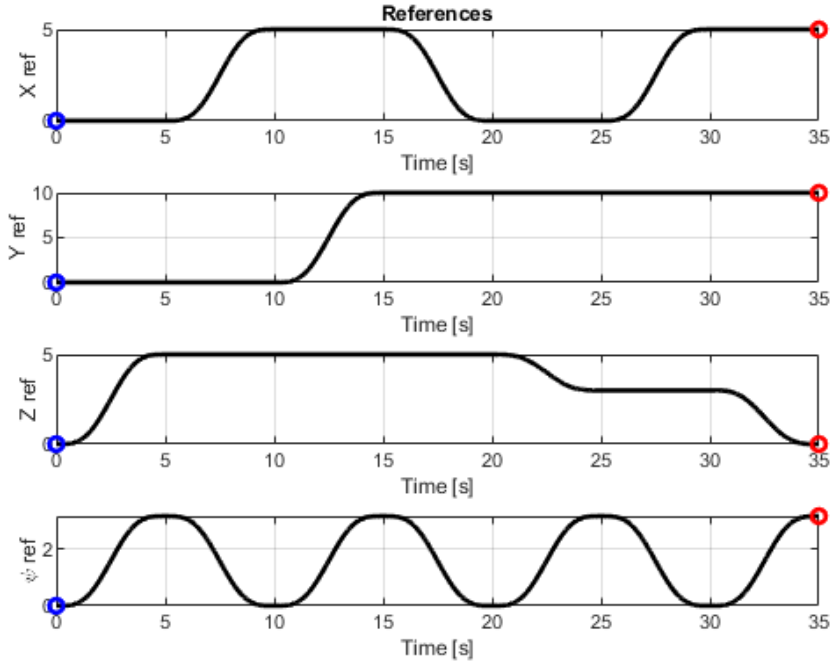

Figura 5.22: Representación en función del tiempo de la segunda trayectoria.

Con respecto a las ganancias presentadas en 5.4, se destaca que se ha reducido la influencia de las velocidades en los tres ejes y de los ángulos de cabeceo y alabeo y sus velocidades angulares. Se observó durante la sintonización que las constantes relacionadas con estas variables de estado tienen una alta dependencia de los cambios bruscos en las referencias, generando acciones de control elevadas en estos puntos e incurriendo en mayores sobreoscilaciones.

$$
\begin{aligned}
\mathrm{X} \rightarrow & \begin{cases} p_1 = 0 \\ p_2 = -7.9 \\ p_3 = 2.5i \\ p_4 = -2.5i \end{cases} \\
\mathrm{Y} \rightarrow & \begin{cases} p_1 = 0 \\ p_2 = -8 \\ p_3 = -2.5i \\ p_4 = 2.5i \end{cases} \\
\mathrm{Z} \rightarrow & \begin{cases} p_1 = 0 \\ p_2 = -8 \\ p_3 = -2.7i \\ p_4 = 2.7i \end{cases} \\
\Psi \rightarrow & \begin{cases} p_1 = -3 + 3i \\ p_2 = -3 - 3i \end{cases}
\end{aligned}
\tag{5.7}
$$

Los polos que definen el funcionamiento del controlador FL para este caso son los mostrados en la expresión 5.7. Como es de suponer, al reducir el periodo de muestreo, la magnitud de los polos se ha incrementado y, por ende, las ganancias obtenidas con dicha asignación de polos.

5.2.1. Resultados obtenidos

En este apartado se muestran los resultados obtenidos para el seguimiento de la nueva trayectoria. De las figuras 5.23 y 5.24 se puede concluir que ambos controladores garantizan un buen seguimiento de las referencias. En la tabla 5.7 se muestran los errores cuadráticos medios para todas las variables de seguimiento. En este caso sí que se observa una clara diferencia siendo el orden de magnitud máximo para los errores de posición no superior a 10^{-3} en el caso del controlador LAB y 10^{-2} en el controlador FL. Con respecto a la orientación, el error cuadrático medio que presenta el controlador LAB es casi dos órdenes de magnitud inferior al obtenido con el controlador FL.

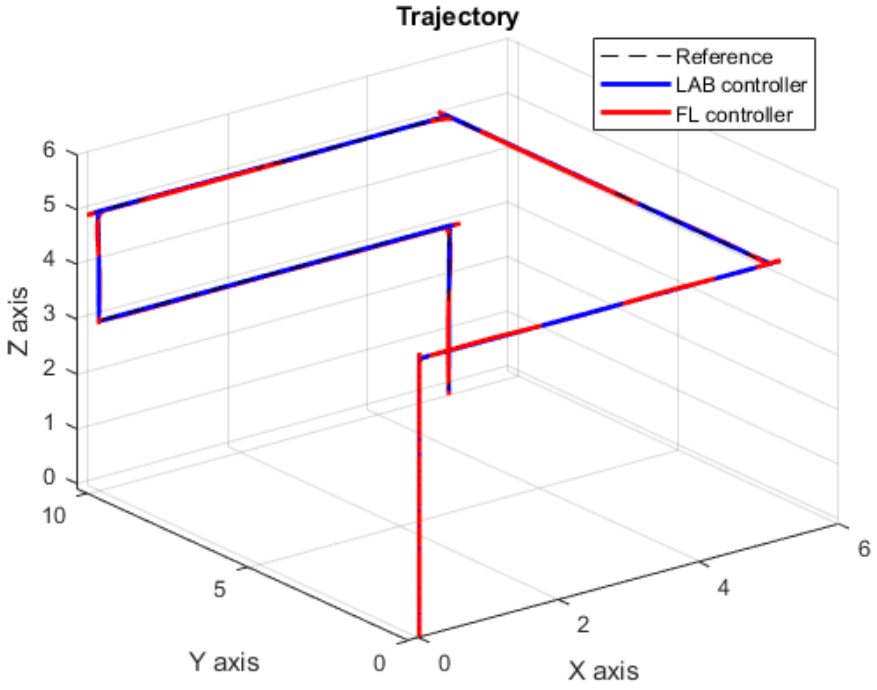

Figura 5.23: Representación tridimensional de los resultados obtenidos para la segunda trayectoria.

Atendiendo a las entradas de control generadas que se muestran en la figura 5.25, se observa una clara superioridad por parte del controlador LAB, siendo este capaz de estabilizar dichas entradas en todos los casos de manera más rápida. Tal y como se hizo para la trayectoria anterior, se han comparado las integrales de las acciones de control para comparar la energía de control, obteniendo los resultados de la tabla

Tabla 5.7: Errores cuadráticos medios para la segunda trayectoria.

	Control LAB	Control FL
Error X	$1.57 \cdot 10^{-3}$ m	$3.10 \cdot 10^{-2}$ m
Error Y	$2.10 \cdot 10^{-3}$ m	$4.10 \cdot 10^{-2}$ m
Error Z	$1.84 \cdot 10^{-4}$ m	$1.20 \cdot 10^{-2}$ m
Error ψ	$7.07 \cdot 10^{-5}$ rad	$1.30 \cdot 10^{-3}$ rad

5.8. De nuevo en este caso, el controlador LAB ofrece resultados bastante inferiores, por lo que su utilización en un caso real supondría un menor consumo de energía por parte del vehículo para realizar un seguimiento de la trayectoria mejor en términos de errores.

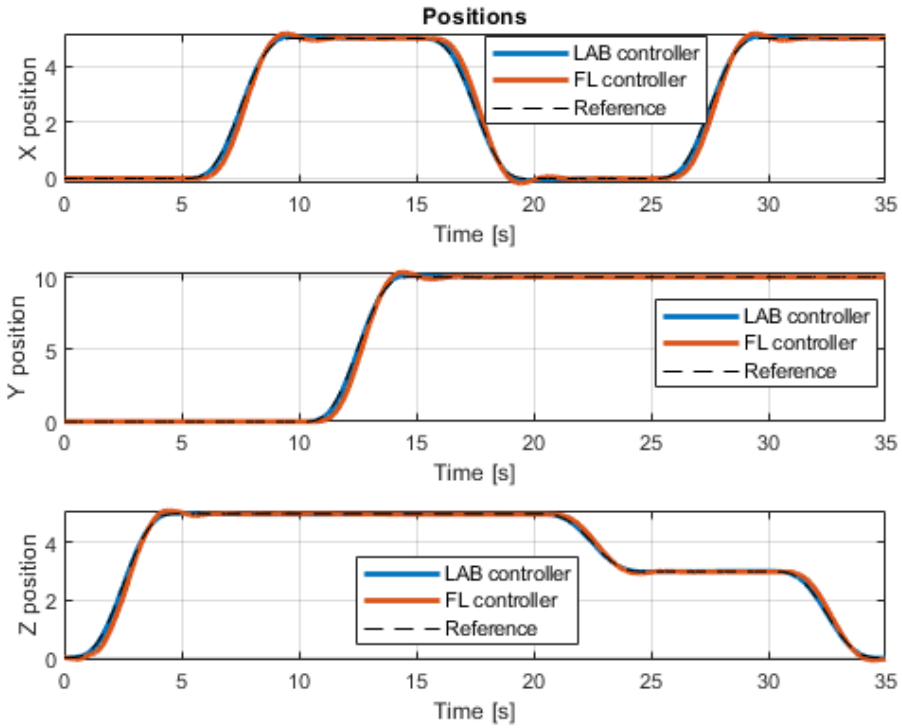

Figura 5.24: Posiciones obtenidas por ambos controladores para el seguimiento de la segunda trayectoria.

Durante el proceso de sintonización del controlador FL con la asignación de polos, se buscó reducir de manera simultánea las oscilaciones en las acciones de control y el error cuadrático medio. Una solución que no se implementa en este trabajo pero que se ha mencionado previamente sería la prealimentación de la dinámica en términos de las derivadas de segundo y tercer orden de las referencias. Si bien es cierto que con la generación polinomial de la trayectoria se puede acceder a ellas, se ha buscado un entorno de simulación con una complejidad similar en ambos casos para realizar la comparación, y puesto que el controlador LAB solo accede de manera externa a las

Tabla 5.8: Integral de las acciones de control para la segunda trayectoria.

	Control LAB	Control FL
E u	$3.4526 \cdot 10^4$	$3.4647 \cdot 10^4$
E $\tilde{\tau}_\phi$	268.999	421.277
E $\tilde{\tau}_\theta$	406.218	623.511
E $\tilde{\tau}_\psi$	$1.961 \cdot 10^3$	$2.063 \cdot 10^3$

referencias de posición y orientación y dentro de su algoritmo estima sus derivadas, se ha visto como suficiente prealimentar únicamente la velocidad en el caso del controlador FL.

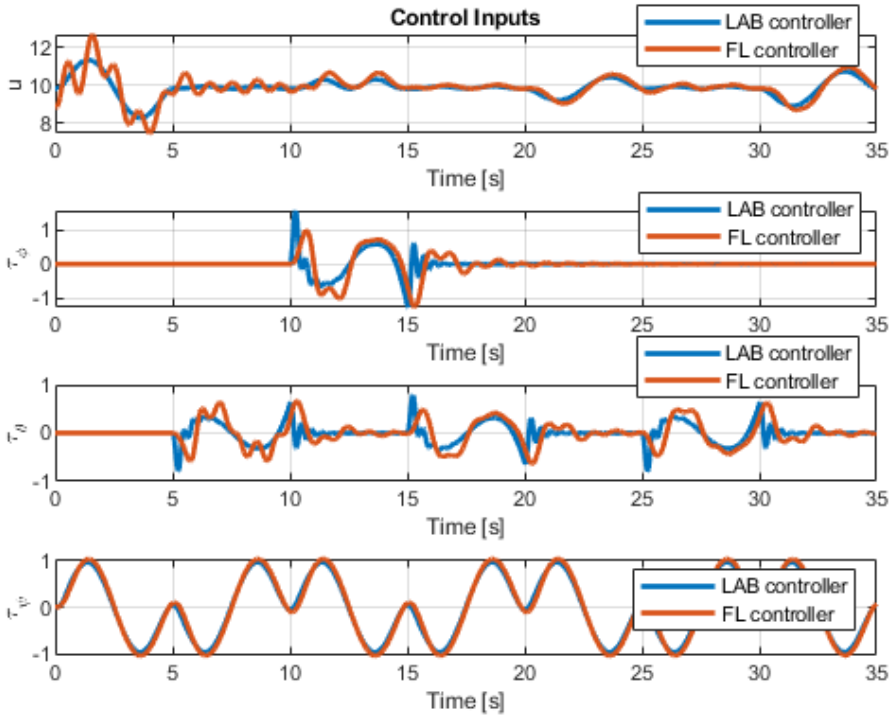

Figura 5.25: Acciones de control generadas por ambos controladores para el seguimiento de la segunda trayectoria.

Asimismo, cabe destacar que la magnitud de las entradas resulta superior al caso anterior. No se ha realizado un estudio de los límites de saturación puesto que sería necesario constar de referencias reales de un dron particular para cuantificar los límites en las velocidades de rotación de los motores y de los rotores. Para mayor precisión, esto debería venir acompañado de un estudio estructural para las vibraciones que se transmitirían a la estructura de la aeronave.

5.3. Conclusiones obtenidas

Los resultados obtenidos con la primera referencia resultaron superiores para el caso del controlador LAB, mostrando menores errores cuadráticos medios y menores acciones de control a lo largo de toda la trayectoria. Finalmente, este último caso en el que se contempla una referencia y una simulación sujeta a distintas condiciones (menor tiempo de muestreo e introducción de referencias de velocidad para reducir oscilaciones en los transitorios) refleja de nuevo una clara superioridad del control LAB con respecto al control basado en linealización por realimentación de estados consiguiendo de nuevo menores errores y menores energías de control.

Más allá de los resultados cuantitativos, durante el propio desarrollo del trabajo, se ha llegado a la conclusión de que los fundamentos teóricos que sustentan la metodología basada en álgebra lineal implementada resultan más sencillos que para el caso del controlador FL, resultando también más sencillo de implementar el controlador LAB discretizado que el controlador FL en términos de modelado con *Matlab*&*Simulink*.

Adicionalmente, el control FL simulado muestra una gran dependencia de los valores iniciales de u para hacer que la matriz $\mathbf{G}$ de la expresión 4.86 sea invertible, mientras que el control LAB muestra dependencia únicamente de las posiciones iniciales.

Por otra parte, el control FL muestra la ventaja de que tras la linealización por realimentación, las nuevas entradas de control v obtenidas para cada referencia hacen que los cuatro subsistemas estén desacoplados. Esto resulta ventajoso a la hora de realizar la sintonización ya que modificar los polos de un subsistema no afecta a la salida del resto.

6. Líneas futuras de investigación

En este apartado se plantean distintas posibilidades que supondrían una continuación de las investigaciones realizadas.

En primer lugar, se propone la realización de una verificación experimental de los controladores propuestos en un dron real, atendiendo a la normativa expuesta en el Anexo III. Esto podría realizarse partiendo de un dron comercial existente o basarse en la construcción propia de uno. El seguimiento de trayectorias resulta sumamente práctico para aplicaciones como la monitorización de terrenos (agricultura, búsqueda y rescate...) [12], por lo que la implementación de técnicas de visión por computador de la mano de un control de trayectorias puede ser una línea de trabajo sumamente interesante.

El planteamiento de estudios experimentales también resultaría interesante en términos de evaluación del impacto de la metodología de control en el ahorro de baterías. Resulta sencillo de intuir que cuanto menor sea la variación en las entradas de control y más estables se puedan mantener las hélices, el consumo global de energía de las baterías debería ser menor. Este dato se ha estimado integrando estas acciones a lo largo de la trayectoria pero está por demostrar de manera empírica. Asimismo, considerar un dron real resultaría interesante de cara a la determinación y el estudio de los límites reales de saturación en los actuadores.

Otra posibilidad que se abre es el estudio de tecnologías de control alternativas como puede ser el control PID o un control predictivo QDMC. Amplificar el campo de estudio ayudaría a determinar las ventajas y desventajas de cada tecnología, pudiendo llegar a determinar cual de todas las numerosas opciones existentes en la actualidad ofrece mejor rendimiento.

El estudio de esta nueva metodología podría complementarse de manera interesante con el control disparado por eventos combinado con técnicas de predicción. De manera introductoria, este tipo de control busca añadir una condición al sensor y al controlador de manera que solo se envíe la información desde el sensor cuando sea necesario. Esto difiere del paradigma tradicional del control por tiempos que envía esta información siempre de manera periódica. Al enviar menos cantidad de información se reduciría el consumo de energía generando un ahorro en las baterías. Para mantener las prestaciones del control, esta técnica debe incluir una estimación o predicción de la información que no se envía.

En todo el proceso de diseño no se ha tenido en cuenta la posible presencia de perturbaciones externas. En el caso de la metodología LAB, estas pueden considerarse y diseñar un control de tipo proporcional integral (no solo proporcional, como se indica en 4.27) con la considerable complejidad adicional en la selección de los parámetros de control. En el caso de la metodología FL, este supuesto no está previsto.

Finalmente, el modelo continuo desarrollado en el apartado 4.2.1 no obtuvo buenos

resultados. El estudio de la metodología LAB y el posterior desarrollo del modelo discretizado refuerza la idea de que las derivadas de las variables sacrificadas pueda afectar a la estabilidad de la simulación. Se deja planteado su desarrollo y estudio para que pueda ser tratado y retomado en investigaciones posteriores.

7. Conclusiones

Las conclusiones a las que se ha llegado durante el desarrollo de esta investigación son las que se recogen a continuación:

- En primer lugar, se ha obtenido una clara imagen de la implicación que tienen los sistemas no-lineales en numerosos y diversos campos de la ingeniería, siendo su control uno de los campos de investigación más importantes dentro de las tecnologías de control modernas.

- Se ha adquirido considerable destreza en términos de modelado y simulación de este tipo de sistemas no-lineales haciendo uso de la herramienta *Matlab*&*Simulink*.

- Tal y como se plantea en el subapartado 5.3, se ha concluido que la metodología LAB ofrece muy buenos resultados dinámicos de cara al seguimiento de trayectorias en drones. Dicha metodología resulta sencilla de entender y durante el desarrollo del modelo del controlador discretizado no se obtuvieron tantos problemas como en el caso del controlador basado en linealización por realimentación de estados. La metodología LAB ofrece ventajas en términos de reducción de errores cuadráticos medios y en términos de reducción de la energía de las acciones de control. Adicionalmente, al ser una metodología ciertamente novedosa, aún está sujeta a nuevas investigaciones e incluso optimizaciones futuras.

- Un tema crucial en el control de trayectorias es la consideración de incertidumbres en el modelo y la presencia de perturbaciones externas, lo que se propone como una clara continuación en esta línea de investigación.

- Finalmente, durante los distintos cursos que tomó este trabajo a lo largo de su completo desarrollo se han implementado los conocimientos adquiridos durante el Máster Universitario en Automática e Informática Industrial impartido por la Universitat Politècnica de València, así como se han adquirido numerosos conocimientos nuevos en el ámbito de la ingeniería de control.

Referencias

[1] Sanjuán, M. A. F. (2016). *Dinámica No Lineal, Teoría del Caos y Sistemas Complejos: una perspectiva histórica.* Rev R Acad Cienc Exact Fís Nat, 109(1–2), 107-126.

[2] Componentes Electrónicas LTDA. 2022. *¿Cómo simular un sistema masa resorte con Simulink? Parte 1 - Componentes Electrónicas LTDA.*. [online] Available at: `https://www.compelect.com.co/2018/01/29/como-simular-un-sistema-masa-resorte-con-simulink-parte-1/` [Accessed 27 June 2022].

[3] Scaglia, G., Serrano, M. and Albertos, P., 2020. *Linear Algebra Based Controllers*. [S.l.]: Springer Nature.

[4] Scaglia, G., Serrano, M. and Albertos, P., 2020. *Control de trayectorias basado en álgebra lineal*. Revista Iberoamericana de Automática e Informática industrial, 17(4), p.344.

[5] Al-Hiddabi, S. A. (2009, March). *Quadrotor control using feedback linearization with dynamic extension*. In 2009 6th International Symposium on Mechatronics and its Applications (pp. 1-3). IEEE.

[6] Khalil, H. K. (2002). *Nonlinear systems* third edition. Patience Hall, 115.

[7] Akhtar, A., Waslander, S. L., and Nielsen, C. (2012, December). *Path following for a quadrotor using dynamic extension and transverse feedback linearization*. In 2012 IEEE 51st IEEE Conference on Decision and Control (CDC) (pp. 3551-3556). IEEE.

[8] Rosales, C., Gandolfo, D., Scaglia, G., Jordan, M. and Carelli, R., 2014. *Trajectory tracking of a mini four-rotor helicopter in dynamic environments - a linear algebra approach*. Robotica, 33(08), pp.1628-1652.

[9] Castillo, P., Lozano, R. and Dzul, A., 2007. *Modelling and control of mini-flying machines*. [Palo Alto, Calif.]: [Ebrary].

[10] Ji, X. and Cai, Z., 2021. *Disturbance Rejection Trajectory Tracking Control for an Unmanned Quadrotor Based on Hybrid Controllers.* Journal of Robotics, 2021, pp.1-12.

[11] Blaabjerg, F. 2018. *Control of Power Electronic Converters and Systems: Volume 2* (Vol. 2). Academic Press.

[12] del Río, P. and González, M. I. (2020, December).*Characterization Of The Electronics And Applications Of Unmanned Air Vehicles*.

Anexos

A. Anexo I: Códigos relacionados con la simulación del modelo y el control

```
clear all
warning off

global Ts aprox

Ts = 0.01;
tau = 0.01;
aprox = 1;
%% SIMULATION PARAMETERS

lim = inf;

global x_ref y_ref z_ref psi_ref

if Ts == 0.02
    tray = 1;
    end_time1 = 120;
    end_time2 = 50;
    end_time = 170;
    t1 = (0:Ts:end_time1)';
    t2 = (0:Ts:end_time2)';
    t = [t1;t2+t1(end)+Ts];
    iters = end_time/Ts+1;
    % Part 1
    R = 2;
    vz1 = 0.05;
    omega = 0.2;
    x_ini = 2;
    y_ini = 0;
    z_ini = 0;

    x1 = 1+R*cos(t1*omega+pi/2);
    dx1 = -omega*R*sin(t1*omega+pi/2);
%     ddx1 = -omega^2*R*cos(t1*omega+pi/2);
%     dddx1 = omega^3*R*sin(t1*omega+pi/2);
%     ddddx1 = omega^4*R*cos(t1*omega+pi/2);

    y1 = 3+R*sin(t1*omega-pi/2);
    dy1 = omega*R*cos(t1*omega-pi/2);
```

```
%     ddy1 = -omega^2*R*sin(t1*omega-pi/2);
%     dddy1 = -omega^3*R*cos(t1*omega-pi/2);
%     ddddy1 = omega^4*R*sin(t1*omega-pi/2);

    z1 = 1+vz1*t1;
    dz1 = vz1*ones(length(t1),1);

    psi1 = omega*t1;
    dpsi1 = omega*ones(length(t1),1);

    % Part 2
    vz2 = -z1(end)/end_time2;

    z2 = z1(end)+vz2*t2;
    dz2 = vz2*ones(length(t2),1);

    x2 = x1(end)*ones(length(t2),1);
    y2 = y1(end)*ones(length(t2),1);
    psi2 = psi1(end)*ones(length(t2),1);

    dx2 = zeros(length(t2),1);
    dy2 = zeros(length(t2),1);
    dpsi2 = zeros(length(t2),1);

    % Final trayectory
    x_ref = [x1;x2];
    y_ref = [y1;y2];
    z_ref = [z1;z2];
    psi_ref = [psi1;psi2];

    dx_ref = [dx1;dx2];
    dy_ref = [dy1;dy2];
    dz_ref = [dz1;dz2];
    dpsi_ref = [dpsi1;dpsi2];

    reference_representation(t,x_ref,y_ref,z_ref,psi_ref,t1)

elseif Ts == 0.01
    tray = 2;
    load referencia_externa.mat
    t = waypoints(:,9);
    iters = length(t);
    end_time = t(end);
    x_ref = waypoints(:,1);
    y_ref = waypoints(:,2);
    z_ref = waypoints(:,3);
```

```
    psi_ref = waypoints(:,4);

    dx_ref = waypoints(:,5);
    dy_ref = waypoints(:,6);
    dz_ref = waypoints(:,7);
    dpsi_ref = waypoints(:,8);

    x_ini = 0;
    y_ini = 0;
    z_ini = 0;

    reference_representation(t,x_ref,y_ref,z_ref,psi_ref,0)
end

%% PROCESS CONSTANTS
global M g
M = 1;     % Mass
g = 9.81;    % Gravity acceleration

%% LAB CONTROLLER CONSTANTS
global k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_10 k_11 k_12

% LAB controller constants
if tray == 1
    k_1 = 0.99;
    k_2 = 0.97;
    k_3 = 0.99;
    k_4 = 0.98;
    k_5 = 0.99;
    k_6 = 0.79;
    k_7 = 0.96;
    k_8 = 0.91;
    k_9 = 0.96;
    k_10 = 0.92;
    k_11 = 0.99;
    k_12 = 0.95;
else
    k_1 = 0.99;
    k_2 = 0.89;
    k_3 = 0.99;
    k_4 = 0.89;
    k_5 = 0.99;
    k_6 = 0.79;
    k_7 = 0.85;
    k_8 = 0.81;
    k_9 = 0.85;
```

```
    k_10 = 0.81;
    k_11 = 0.85;
    k_12 = 0.95;
end

%% Pole placement gains for FL controller
if tray == 1
    p11 = 0.5;
    p12 = 2.4;
    p13 = -1.1i;
    p14 = 1.1i;

    p21 = 0.7;
    p22 = 1.8;
    p23 = -1i;
    p24 = 1i;

    p31 = -1.1i;
    p32 = 1.1i;
    p33 = 0.6;
    p34 = 2.3;

    p41 = 1;
    p42 = 0.5;

else
    p11 = 0;
    p12 = 7.9;
    p13 = -2.5i;
    p14 = 2.5i;

    p21 = 0;
    p22 = 8;
    p23 = -2.5i;
    p24 = 2.5i;

    p31 = -2.7i;
    p32 = 2.7i;
    p33 = 0;
    p34 = 8;

    p41 = 3-3i;
    p42 = 3+3i;
end

K1 = pole_placement(-p11,-p12,-p13,-p14);
```

```
K2 = pole_placement(-p11,-p12,-p13,-p14);
K3 = pole_placement(-p31,-p32,-p33,-p34);
K4 = pole_placement(-p41,-p42);

k11=K1(1);k12=K1(2);k13=K1(3);k14=K1(4);
k21=K2(1);k22=K2(2);k23=K2(3);k24=K2(4);
k31=K3(1);k32=K3(2);k33=K3(3);k34=K3(4);
k41=K4(1);k42=K4(2);

%% SATURATION LIMITS
saturation_limits;

%% Empty vectors
simulations = 2;
empty_vectors;

%% Simulation
global i j;

controller = ["LAB","FL"];
initial_conditions;

for j=1:simulations

    % Reset of counter and initial conditions
    i=1; % Global counter
    global x2r0 x4r0 x6r0 x7r0 x8r0 x9r0 x10r0 x12r0
    global  x1_0 x3_0 x5_0 x7_0 x9_0 x11_0 u1_0 du1_0

    tic % Start stopwatch
    out = sim("modelo_dinamico_"+controller(j)+".slx");
    toc % End stopwatch

    input = out.data.u;
    outputs = out.data.y;
    tsim(:,j) = out.tout();

    % Control inputs
    u(:,j) = input(:,1);
    tau_phi(:,j) = input(:,2);
    tau_theta(:,j) = input(:,3);
    tau_psi(:,j) = input(:,4);

    % System outputs
    x(:,j) = outputs(:,1);
    y(:,j) = outputs(:,3);
```

```
    z(:,j) = outputs(:,5);
    phi(:,j) = outputs(:,7);
    theta(:,j) = outputs(:,9);
    psi(:,j) = outputs(:,11);

    if tray == 1
        diff = 2;
    else
        diff = 1;
    end

    e_x(:,j) = x_ref(1:end-diff)-x(:,j);
    e_y(:,j) = y_ref(1:end-diff)-y(:,j);
    e_z(:,j) = z_ref(1:end-diff)-z(:,j);
    e_psi(:,j) = psi_ref(1:end-diff)-psi(:,j);

    x_dot = outputs(:,2);
    y_dot = outputs(:,4);
    z_dot = outputs(:,6);
    phi_dot = outputs(:,8);
    theta_dot = outputs(:,10);
    psi_dot = outputs(:,12);

end

%% Grahpic representation
tridimensional_representation;
graphic_representation;

%% Errores
for i=1:simulations
    fprintf('\n%s controller',controller(i))
    ecm_x = e_cuad(x_ref(:),x(:,i));
    ecm_y = e_cuad(y_ref(:),y(:,i));
    ecm_z = e_cuad(z_ref(:),z(:,i));
    ecm_psi = e_cuad(psi_ref(:),psi(:,i));
    disp(' ')
    % fprintf('ERROR MEDIO = %d\n',mean([e_x,e_y,e_z]))
    fprintf('   Error X = %d\n',ecm_x)
    fprintf('   Error Y = %d\n',ecm_y)
    fprintf('   Error Z = %d\n',ecm_z)
    fprintf('   Error Psi = %d\n',ecm_psi)
end
```

Código A.1: Código para correr la simulación del modelo controlado según ambas metodologías.

A continuación, se comenta brevemente la estructura del código presentado:

- En la primera sección (líneas 1 a 9) se establecen parámetros globales de la simulación como el periodo de muestreo, el valor de la constante de tiempo empleado para los derivadores filtrados y la aproximación que se empleará para la aproximación de las referencias en $n + 1$ del controlador LAB.

- En la siguiente sección (líneas 9 a 113) se define la trayectoria a seguir. Se han construido dos trayectorias que son intercambiables cambiando el valor del periodo de muestrel en la línea 6. Estas trayectorias son las empleadas para las simulaciones. En la primera trayectoria están comentadas las líneas que se refieren a las derivadas de x e y de grado 2, 3 y 4 ya que se evaluó su influencia en la dinámica global y no afectaba de manera significativa al resultado. Igualmente, se hace una llamada al código *reference_representation* (código A.2) para dibujar la trayetoria. La segunda trayectoria se introduce de manera externa ya que los puntos se generan con la función generadora de trayectorias descrita en el Anexo IV.

- La siguiente sección (líneas 100 a 104) se definen las variables globales de masa y aceleración gravitatoria.

- A continuación, en la sección que abarca de la línea 105 a la 135 se establecen las ganancias del controlador LAB para las dos trayectorias.

- De la línea 137 a 185 se definen los polos del controlador FL y se hace una llamada a la función *pole_placement* (A.3) para estimar las ganancias que garantizan esos polos.

- La siguiente sección (líneas 187 a 198) se emplea para definir límites de saturación en las entradas de control. Estos se utilizaron en etapas tempranas del diseño y el criterio de su selección no sigue un fundamento científico. Asimismo, se incluyen por si fuese necesario definirlos en el futuro.

- La sección definida de la línea 190 a 192 se emplea para definir unos vectores vacíos con unas dimensiones dependientes de la trayectoria para comparar los resultados de ambos controladores llamando al código *empty_vectors.m* (código A.4). Alterando el valor de la variable *simulations* se define el número de controladores que se van a simular.

- A continuación, se encuentra la sección empleada para realizar las simulaciones *per se* (línea 194 a 247). La cadena que se define en la línea 207 se emplea para cambiar el controlador que se simula en la línea 218 (aquí se introduce el nombre del archivo de *Simulink* donde se encuentra el modelo del controlador). Se hace una llamada al código *initial_conditions* (código A.6) en la línea 208 para establecer las condiciones iniciales. Se guardan los resultados obtenidos en los vectores creados en la sección anterior.

- La sección siguiente (líneas 249 a 251) se emplea para la representación gráfica. En primer lugar, se llama al código *tridimensional_representation* (código A.7) para visualizar una animación tridimensional de la trayectoria obtenida. Posteriormente se muestran los resultados con el código *graphic_representation* (código A.8).
- A continuación, (líneas 253 a 266) se imprime información relevante sobre los errores cuadráticos obtenidos gracias a la función construida *e_cuad* (código A.9) y los valores máximos de las acciones de control. Esta información resulta relevante de cara a la sintonización de los controladores.

```
figure
subplot(411)
plot(t,x_ref,'k',linewidth=2);grid on;hold on
plot(t(1),x_ref(1),'ob',linewidth=2)
if tray == 1
    plot(t(length(t1)),x_ref(length(t1)),'og',linewidth=2)
end
plot(t(end),x_ref(end),'or',linewidth=2)
title('References')
xlabel('Time [s]')
ylabel('X ref')

subplot(412)
plot(t,y_ref,'k',linewidth=2);grid on;hold on
plot(t(1),y_ref(1),'ob',linewidth=2)
if tray == 1
    plot(t(length(t1)),y_ref(length(t1)),'og',linewidth=2)
end
plot(t(end),y_ref(end),'or',linewidth=2)
xlabel('Time [s]')
ylabel('Y ref')

subplot(413)
plot(t,z_ref,'k',linewidth=2);grid on;hold on
plot(t(1),z_ref(1),'ob',linewidth=2)
if tray == 1
    plot(t(length(t1)),z_ref(length(t1)),'og',linewidth=2)
end
plot(t(end),z_ref(end),'or',linewidth=2)
xlabel('Time [s]')
ylabel('Z ref')

subplot(414)
plot(t,psi_ref,'k',linewidth=2);grid on;hold on
plot(t(1),psi_ref(1),'ob',linewidth=2)
if tray == 1
    plot(t(length(t1)),psi_ref(length(t1)),'og',linewidth=2)
end
plot(t(end),psi_ref(end),'or',linewidth=2)
xlabel('Time [s]')
ylabel('\psi ref')

figure
plot3(x_ref,y_ref,z_ref,'k',linewidth=2);grid on;hold on
plot3(x_ref(1),y_ref(1),z_ref(1),'ob',linewidth=2)
```

```
plot3(x_ref(end),y_ref(end),z_ref(end),'or',linewidth=2)
xlabel('X axis')
ylabel('Y axis')
zlabel('Z axis')
title('3D references')
if tray == 1
    plot3(x_ref(length(t1)),y_ref(length(t1)),z_ref(length(t1)),'og',linewidth=2)
    legend('Reference', 'Initial point','Transition point','Final point')
else
    legend('Reference', 'Initial point','Final point')
end
```

Código A.2: Código empleado para la representación gráfica de la trayectoria.

```
function K = pole_placement(n1,n2,n3,n4)
    syms n k1 k2 k3 k4
    if nargin == 4 % Grade 4
        A = [-n      1       0       0;
              0     -n       1       0;
              0      0      -n       1;
             -k1    -k2     -k3    -n-k4];
        B = [1;0;0;0];
        d = det(A);
        eq1 = subs(d,n,n1)==0;
        eq2 = subs(d,n,n2)==0;
        eq3 = subs(d,n,n3)==0;
        eq4 = subs(d,n,n4)==0;
        [A,B] = equationsToMatrix([eq1, eq2, eq3, eq4], [k1, k2, k3, k4]);
    elseif nargin == 2 % Grade 2
        A = [-n     1;
             -k1 -n-k2];
        d = det(A);
        eq1 = subs(d,n,n1)==0;
        eq2 = subs(d,n,n2)==0;
        [A,B] = equationsToMatrix([eq1, eq2], [k1, k2]);
    end
    K = double(linsolve(A,B)');
end
```

Código A.3: Función empleada para obtener las ganancias con la asignación de polos.

```
tsim = zeros(iters-1,simulations);
x = zeros(iters-1,simulations);
y = zeros(iters-1,simulations);
z = zeros(iters-1,simulations);
```

```
phi = zeros(iters-1,simulations);
theta = zeros(iters-1,simulations);
psi = zeros(iters-1,simulations);
u = zeros(iters-1,simulations);
tau_phi = zeros(iters-1,simulations);
tau_theta = zeros(iters-1,simulations);
tau_psi = zeros(iters-1,simulations);
e_x=zeros(iters-1,simulations);
e_y=zeros(iters-1,simulations);
e_z=zeros(iters-1,simulations);
e_psi=zeros(iters-1,simulations);
```

Código A.4: Código empleado para crear vectores vacíos para almacenar los resultados.

```
u_max = 20;
u_min = 0;

tau_phi_max = 10;
tau_phi_min = -10;

tau_theta_max = 10;
tau_theta_min = -10;

tau_psi_max = 1;
tau_psi_min = -1;
```

Código A.5: Código empleado para establecer los límites de saturación de las entradas de control.

```
x1_0  = x_ini; % x
x3_0  = y_ini; % y
x5_0  = z_ini; % z
x7_0  = 0;
x9_0  = 0;
x11_0 = 0; % psi
u1_0 = M*g;
du1_0 = 0;
```

Código A.6: Código empleado para establecer las condiciones iniciales.

```
close all

for j=1:2
    figure(j)
    ax = axes();
    ax.DataAspectRatio = [1 1 1];

    view(ax, 3)
```

```
    hold(ax, 'on')
    xlabel('X axis')
    ylabel('Y axis')
    zlabel('Z axis')
    title(["Controller "+controller(j)])

    if tray == 1
        xlim([-2 4])
        ylim([0 6])
        zlim([0 8])
    else
        xlim([-0.5 6.5])
        ylim([-0.5 10.5])
        zlim([0 6])
    end

    plot3(x_ref,y_ref,z_ref,'--k');grid on;hold on

    range = size(tsim,1);
    step = 5;
    speed = 10;
    ref0 = [0.5 0 0];

    pause
    for i=1:step:range
        if i>1
            angle = psi(i,j)-psi(i-step,j);
        else
            angle = psi(i,j);
        end
        ref = vector_rotation(ref0,angle);
        ov = [x(i,j) y(i,j) z(i,j)]+ref;
        figure(j)
        plot3([ov(1) x(i,j)], ...
              [ov(2) y(i,j)], ...
              [ov(3) z(i,j)],'.-r',linewidth=0.1)
        if i > 1
            plot3(x(i-step:i,j),y(i-step:i,j),z(i-step:i,j),'
               -b',linewidth=4)
        else
            plot3(x(i,j),y(i,j),z(i,j),'-b')
        end
        ref0 = ref;
        pause(Ts*step/speed)
    end
    grid on; hold on
```

```
end
```

Código A.7: Código empleado para la representación tridimensional de la trayectoria obtenida.

```
if length(tsim(:,j))~=length(t)
    diff = abs(size(tsim,1)-length(x_ref));
    if length(x_ref)>length(tsim)
        x_ref = x_ref(1:end-diff);
        y_ref = y_ref(1:end-diff);
        z_ref = z_ref(1:end-diff);
        psi_ref = psi_ref(1:end-diff);
    end
end

if simulations == 1
    figure
    subplot(411);plot(tsim,u);grid on
    title("Control inputs (controller "+controller(1)+")")
    xlabel('Time [s]')
    ylabel('u')
    subplot(412);plot(tsim,tau_phi);grid on
    xlabel('Time [s]')
    ylabel('\tau_{\phi}')
    subplot(413);plot(tsim,tau_theta);grid on
    xlabel('Time [s]')
    ylabel('\tau_{\theta}')
    subplot(414);plot(tsim,tau_psi);grid on
    xlabel('Time [s]')
    ylabel('\tau_{\psi}')

    figure
    subplot(311)
    plot(tsim,phi);grid on
    title("Angles (controller "+controller(1)+")")
    xlabel('Time [s]')
    ylabel('\phi')
    subplot(312)
    plot(tsim,theta);grid on
    xlabel('Time [s]')
    ylabel('\theta')
    subplot(313)
    plot(tsim,psi);grid on;hold on
    plot(tsim,psi_ref,'--k')
    xlabel('Time [s]')
    ylabel('\psi')
    legend('\psi','\psi_{ref}')
```

```
43
44      figure
45      subplot(411)
46      plot(tsim,x,'b');hold on;grid on
47      plot(tsim,x_ref,'--k')
48      ylabel('X position [m]')
49      xlabel('Time [s]')
50      title("Position and orientation (controller "+controller
           (1)+")")
51      legend('X','X_{ref}')
52      subplot(412)
53      plot(tsim,y,'b');hold on;grid on
54      plot(tsim,y_ref,'--k')
55      xlabel('Time [s]')
56      ylabel('Y position [m]')
57      legend('Y','Y_{ref}')
58      subplot(413)
59      plot(tsim,z,'b');hold on;grid on
60      plot(tsim,z_ref,'--k')
61      xlabel('Time [s]')
62      ylabel('Z position [m]')
63      legend('Z','Z_{ref}')
64      subplot(414)
65      plot(tsim,psi,'b');hold on;grid on
66      plot(tsim,psi_ref,'--k')
67      xlabel('Time [s]')
68      ylabel('\psi angle [rad]')
69      legend('\psi','\psi_{ref}')
70
71      figure
72      subplot(411)
73      plot(tsim,e_x,'b');grid on;hold on
74      title("Errors (controller "+controller(1)+")")
75      xlabel('Time [s]')
76      ylabel('X error')
77      subplot(412)
78      plot(tsim,e_y,'b');grid on;hold on
79      xlabel('Time [s]')
80      ylabel('Y error')
81      subplot(413)
82      plot(tsim,e_z,'b');grid on;hold on
83      xlabel('Time [s]')
84      ylabel('Z error')
85      subplot(414)
86      plot(tsim,e_psi,'b');grid on;hold on
87      xlabel('Time [s]')
```

```
88      ylabel('Psi error')
89
90      figure
91      plot3(x,y,z,linewidth=2);hold on;grid on
92      plot3(x_ref,y_ref,z_ref,'--k')
93      plot3(x(1),y(1),z(1),'.r',markersize=15,linewidth=2)
94      plot3(x_ref(1),y_ref(1),z_ref(1),'.k',markersize=15,
           linewidth=2)
95      legend('Trajectory','Reference')
96      xlabel('X axis')
97      ylabel('Y axis')
98      zlabel('Z axis')
99      title("Trajectory  (controller "+controller(1)+")")
100
101 elseif simulations == 2
102
103     figure
104     subplot(411)
105     plot(tsim(:,1),u(:,1),linewidth=2);grid on;hold on
106     plot(tsim(:,2),u(:,2),linewidth=2)
107     title("Control Inputs")
108     xlabel('Time [s]')
109     ylabel('u')
110     legend(controller(1)+" controller",controller(2)+"
           controller")
111
112     subplot(412)
113     plot(tsim(:,1),tau_phi(:,1),linewidth=2);grid on;hold on
114     plot(tsim(:,2),tau_phi(:,2),linewidth=2)
115     xlabel('Time [s]')
116     ylabel('\tau_{\phi}')
117     legend(controller(1)+" controller",controller(2)+"
           controller")
118
119     subplot(413)
120     plot(tsim(:,1),tau_theta(:,1),linewidth=2);grid on; hold
           on
121     plot(tsim(:,2),tau_theta(:,2),linewidth=2)
122     xlabel('Time [s]')
123     ylabel('\tau_{\theta}')
124     legend(controller(1)+" controller",controller(2)+"
           controller")
125
126     subplot(414)
127     plot(tsim(:,1),tau_psi(:,1),linewidth=2);grid on; hold on
128     plot(tsim(:,2),tau_psi(:,2),linewidth=2)
```

```
129     xlabel('Time [s]')
130     ylabel('\tau_{\psi}')
131     legend(controller(1)+" controller",controller(2)+"
            controller")
132
133     figure
134     subplot(311)
135     plot(tsim(:,1),phi(:,1),linewidth=2);grid on;hold on
136     plot(tsim(:,2),phi(:,2),linewidth=2)
137     title("Angles")
138     xlabel('Time [s]')
139     ylabel('\phi')
140     legend(controller(1)+" controller",controller(2)+"
            controller")
141
142     subplot(312)
143     plot(tsim(:,1),theta(:,1),linewidth=2);grid on;hold on
144     plot(tsim(:,2),theta(:,2),linewidth=2)
145     xlabel('Time [s]')
146     ylabel('\theta')
147     subplot(313)
148     plot(tsim(:,1),psi(:,1),linewidth=2);grid on;hold on
149     plot(tsim(:,2),psi(:,2),linewidth=2)
150     plot(tsim(:,1),psi_ref,'--k')
151     xlabel('Time [s]')
152     ylabel('\psi')
153     legend(controller(1)+" controller",controller(2)+"
            controller")
154
155     figure
156     subplot(311)
157     plot(tsim(:,1),x(:,1),linewidth=2);hold on;grid on
158     plot(tsim(:,2),x(:,2),linewidth=2)
159     plot(tsim(:,1),x_ref,'--k')
160     ylabel('X position')
161     xlabel('Time [s]')
162     title("Positions")
163     legend(controller(1)+" controller",controller(2)+"
            controller",'Reference')
164
165     subplot(312)
166     plot(tsim(:,1),y(:,1),linewidth=2);hold on;grid on
167     plot(tsim(:,2),y(:,2),linewidth=2)
168     plot(tsim(:,1),y_ref,'--k')
169     xlabel('Time [s]')
170     ylabel('Y position')
```

```
171     legend(controller(1)+" controller",controller(2)+"
            controller",'Reference')
172
173     subplot(313)
174     plot(tsim(:,1),z(:,1),linewidth=2);hold on;grid on
175     plot(tsim(:,2),z(:,2),linewidth=2);hold on;grid on
176     plot(tsim(:,1),z_ref,'--k')
177     xlabel('Time [s]')
178     ylabel('Z position')
179     legend('Z','Z_{ref}')
180     legend(controller(1)+" controller",controller(2)+"
            controller",'Reference')
181
182     figure
183     subplot(411)
184     plot(tsim(:,1),e_x(:,1),linewidth=2);grid on;hold on
185     plot(tsim(:,2),e_x(:,2),linewidth=2)
186     title("Errores (controller "+controller(1)+")")
187     xlabel('Time [s]')
188     ylabel('X error')
189     legend(controller(1)+" controller",controller(2)+"
            controller")
190
191     subplot(412)
192     plot(tsim(:,1),e_y(:,1),linewidth=2);grid on;hold on
193     plot(tsim(:,2),e_y(:,2),linewidth=2)
194     xlabel('Time [s]')
195     ylabel('Y error')
196     legend(controller(1)+" controller",controller(2)+"
            controller")
197
198     subplot(413)
199     plot(tsim(:,1),e_z(:,1),linewidth=2);grid on;hold on
200     plot(tsim(:,2),e_z(:,2),linewidth=2)
201     xlabel('Time [s]')
202     ylabel('Z error')
203     legend(controller(1)+" controller",controller(2)+"
            controller")
204
205     subplot(414)
206     plot(tsim(:,1),e_psi(:,1),linewidth=2);grid on;hold on
207     plot(tsim(:,2),e_psi(:,2),linewidth=2)
208     xlabel('Time [s]')
209     ylabel('Psi error')
210     legend(controller(1)+" controller",controller(2)+"
            controller")
```

```
211
212     figure
213     plot3(x_ref,y_ref,z_ref,'--k');hold on;grid on
214     plot3(x(:,1),y(:,1),z(:,1),'b',linewidth=2)
215     plot3(x(:,2),y(:,2),z(:,2),'r',linewidth=2)
216     xlabel('X axis')
217     ylabel('Y axis')
218     zlabel('Z axis')
219     title("Trajectory")
220     legend("Reference",controller(1)+" controller",controller
           (2)+" controller")
221 end
```

Código A.8: Código emplado para la representación gráfica de los resultados obtenidos.

```
1 function e2=e_cuad(q1,q2)
2     N=length(q1);
3         e2=diag((q1-q2)'*(q1-q2))/N;
4 end
```

Código A.9: Función empleada para calcular el error cuadrático medio entre dos vectores.

B. Anexo II: Efecto del valor inicial de U para el controlador FL

El parámetro U necesita un valor inicial distinto de 0 para que la matriz **G** de la expresión 4.86 sea invertible. Se probaron distintos parámetros para evaluar la influencia de este valor, para posteriormente concluir a la vista de los resultados que se muestran en las figuras B.1 y B.2 que utilizar el propio peso del dron generaba una respuesta aceptable.

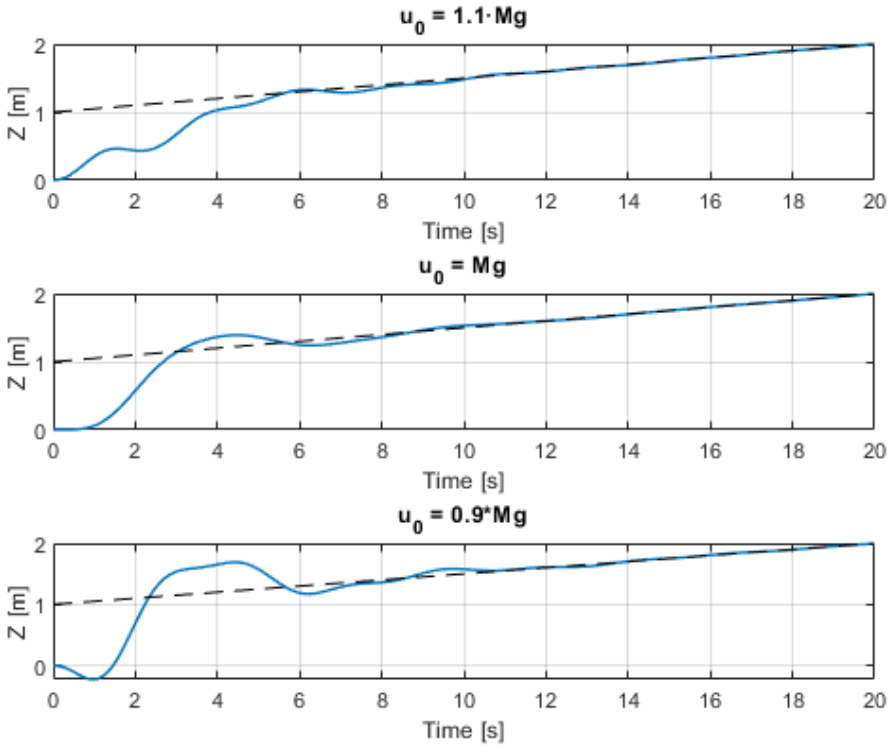

Figura B.1: Variación de la trayectoria en la coordenada Z para distintos valores iniciales de u.

De manera cualitativa, si el valor de u es igual al peso del dron, este estaría en suspensión sin ascender ni descender. Si se tiene en cuenta que el ascenso del dron va a realizarse con una velocidad no demasiado elevada, se puede suponer que el valor de la acción de control u se estabilizará al rededor de este parámetro inicial. Por otra parte, si este valor es menor que el peso en los instantes iniciales, el dron descendería por el efecto de su propio peso hasta conseguir acelerar lo suficiente sus rotores para compensarlo. Esto se puede observar en el caso que se plantea para $u_0 = 0.9 \cdot Mg$. Finalmente, si el valor es mayor al peso, el dron tenderá a oscilar para corregirlo. En este caso, el valor de la oscilación máxima será igual a la diferencia entre $\pm u_0$ y el valor donde se estabiliza la acción de control. Teniendo esto último en cuenta, se puede asumir que emplear como valor inicial de u_0 el propio peso del dron devuelve

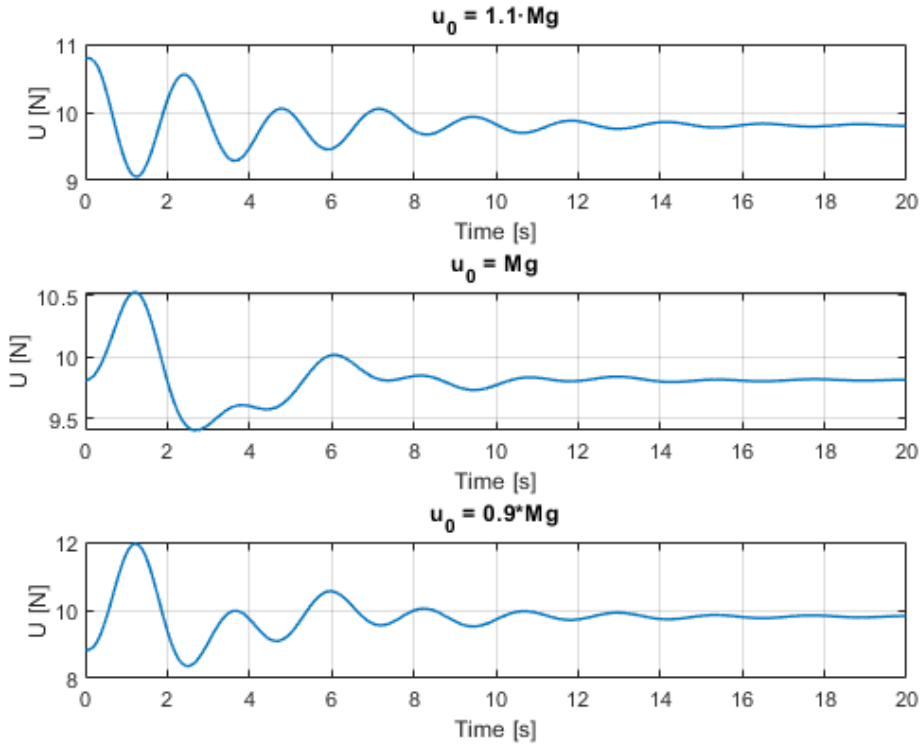

Figura B.2: Variación en la acción de control *u* para distintos valores iniciales.

los mejores resultados ya que las sobreoscilaciones son menores, así como el tiempo que tarda la señal de control en estabilizarse.

C. Anexo III: Legislación vigente alrededor del vuelo y ensayos experimentales con drones

La legislación que concierne a este tipo de vehículos ha estado históricamente relacionada con la legislación concerniente a las aeronaves tripuladas. La Organización de Aviación Civil Internacional (OACI) delega en los estados miembros las competencias de regulación del vuelo de drones a lo largo de la extensión de su territorio. Con el aumento y abaratamiento de las tecnologías que envuelven a estos vehículos se ha producido un aumento notable de su uso que puede poner en situación de riesgo la seguridad aérea.

En España, la regulación de alrededor de los drones viene recogida en el Real Decreto 1036/2017, donde se establece un marco jurídico que aplica a aeronaves no pilotadas de uso civil de pequeño tamaño (peso máximo al despegue inferior a 150 kg), no respaldadas por la legislación de la OACI ni de la Unión Europea.

El Real Decreto estipula que se ha de tener una licencia de drones para todo tipo de operaciones de carácter profesional. Igualmente, todo operario de drones profesional debe estar dado de alta en la Agencia Estatal de Seguridad Aérea (AESA) y constar de los certificados médicos necesarios. Por otra parte, para la realización de vuelos con drones de carácter recreativo (según el Real Decreto: inferiores a 2kg) hay que tener en cuenta las siguientes consideraciones.

Para drones de masa entre 250 g y 2 kg:

- El vuelo debe ser *Visual Line Of Sight* (VLOS).
- Se aconseja constar de seguro de responsabilidad civil.
- Está prohibido volar a menos de 8km de cualquier aeropuerto.
- Es obligatorio que el dron cuente de una placa ignífuga identificativa con los datos del piloto.

Para drones de masa inferior a 250 g

- Está permitido volar sobre grupos de personas siempre y cuando no se supere una altura de vuelo de 20 m.
- Está prohibido volar en Reservas de la Biosfera, Parques Naturales y demás espacios protegidos.
- Está prohibido volar a menos de 8 km de cualquier aeropuerto.

Por otra parte, los drones con sensores fotográficos, estén dentro de la categoría que sea, están igualmente sujetos al cumplimiento de la Ley de Protección de Datos y

el Derecho al Honor e Intimidad de las personas.

Este Decreto contiene también consideraciones para los vuelos nocturnos o en áreas urbanas. Para la realización de vuelos nocturnos es necesario constar con un permiso de la AESA y el dron ha de tener una masa inferior a 10 kg. Para los vuelos en áreas urbanas se aplica también esta última condición de peso pero a mayores el vuelo debe de ser VLOS y es recomendable mantener una distancia mínima lateral de separación con los edificios de 50 m. También se requiere aprobación de la AESA.

Finalmente, cabe destacar que este Real Decreto no es aplicable para los siguientes casos:

- Globos libres no tripulados (globos cautivos).
- Vuelos llevados a cabo en el interior de un espacio cerrado.
- Drones de masa superior a 150 kg.
- Drones de carácter militar, policial, de búsqueda y rescate, lucha contra incendios, exhibición, actividades deportivas, etc.

Estas últimas consideraciones dejan abierta la posibilidad de realizar ensayos con drones sin atenerse a la legislación vigente siempre y cuando estos se realicen en espacios cerrados.

D. Anexo IV: Generador de trayectorias

Resulta de interesante en el ámbito de los drones tratar con precisión la cuestión de las trayectorias que se han de seguir, ya sea por economizar recursos y acortar tiempos de vuelo como para evitar obstáculos. Las trayectorias pueden estar pre-programadas dentro del propio dron y ser fijas o bien modificarse sobre la marcha. Este último aspecto resulta bastante complejo, por lo que en el presente proyecto se ha hecho uso de trayectorias pre-programadas.

D.1. Generación de trayectorias

La generación de una trayectoria se puede resumir en el cálculo de puntos intermedios denominados *waypoints* entre un punto inicial y otro final. Dentro del mundo de los drones estos puntos intermedios tienen asociada igualmente una orientación del dron o un tiempo de espera.

En este anexo se plantean tres tipos de trayectorias: **lineal** , **rotación sobre el eje central** y **circular**, que se han programado en una función de *Matlab*.

Dentro de la trayectoria circular, se han programado dos opciones dependiendo de las entradas: cálculo de la trayectoria circular dándole a la función punto inicial, punto intermedio y punto final y cálculo de la trayectoria circular introduciendo punto inicial, radio, centro, ángulo del arco y plano.

En los siguientes subapartados se explicará con más detalle el funcionamiento de cada sección dependiendo de la trayectoria. De manera general, los *waypoints* que se obtienen forman parte de una matriz de 5 columnas en la que los tres primeros se corresponden con las coordenadas en el espacio cartesiano, el cuarto un ángulo de la orientación del sistema con respecto a una referencia fija y el quinto es el tiempo asociado a dicho punto. Se obtiene un *waypoint* para cada instante de muestreo, considerando tiempos para la realización del tramo y tiempos de espera tras llegar al punto final. La función generadora de trayectorias está construida de manera que el primer argumento que admite es siempre el *waypoint* anterior, y la trayectoria se escoge dependiendo del resto de argumentos.

D.1.1. Trayectoria lineal

Es la más simple de las tres, ya que toma únicamente como entradas la posición del punto inicial y final en el espacio cartesiano, así como el tiempo total para llevar a cabo la trayectoria y el tiempo de espera en el punto final. La llamada a esta sección de la función se realiza cuando el argumento posterior al punto inicial es otro punto. Su cálculo es simple, manteniendo los parámetros previos de orientación, se divide la distancia entre el punto inicial y final en un determinado número de posiciones intermedias. Dividiendo el tiempo total para completar el tramo entre un periodo de muestreo se obtienen los puntos temporales y con un bucle se rellena el vector

waypoints con los datos de posición y orientación.

```
    if nargin == 4 && numel(arg1)~=1
%         disp('Linear trayectory')
        P2 = arg1;
        t_t = arg2;
        t_w = arg3;
        time = wp_ant(end);
        orient = wp_ant(4);
        distance = norm(P2-wp_ant(1:3));
        npoints = ceil(t_t/ts);
        nwpoints = ceil(t_w/ts);
        waypoints = zeros(npoints+nwpoints,5);
        j=1;
        dmin = 0.1;
        waypoints(1,1:4) = wp_ant(1:4);
        while(distance>dmin)
            distance = norm(P2-waypoints(j,1:3));
            direction = (P2-waypoints(j,1:3))/distance;
            delta_d = distance/npoints;
            waypoints(j+1,1:4) = [waypoints(j,1:3)+direction*
               delta_d orient];

            waypoints(j,5) = time + ts;
            time = waypoints(j,5);
            wp_ant(1:3) = waypoints(j,1:3);
            j=j+1;
        end
        for i=0:nwpoints-1
            waypoints(i+j,1:4) = waypoints(j-1,1:4);
            waypoints(i+j,5) = waypoints(i+j-1,5) + ts;
        end
```

Código D.1: Sección del generador de trayectorias encargada de la trayectoria lineal.

D.1.2. Rotación sobre el eje central

La rotación sobre el propio eje central del dron es la trayectoria más sencilla de las tres. Tomando el punto de origen con su orientación inicial, se calcula el ángulo final de la orientación y se divide el arco total de rotación en un número entero de secciones dependientes del tiempo de realización y del periodo de muestreo. A pesar de no ser exactamente *waypoints*, se guardan en la matriz ya que la orientación en el espacio cambia con respecto al punto inicial en función del tiempo.

```
    elseif nargin == 4 && numel(arg1)==1
%         disp('Rotation')
```

```
        t_t = arg2;
        t_w = arg3;
        time = wp_ant(end);
        P = wp_ant(1:3);
        arc = arg1;
        oi = wp_ant(4);
        of = oi+arc;
        npoints = ceil(t_t/ts);
        nwpoints = ceil(t_w/ts);
        delta = arc/npoints;
        waypoints = zeros(npoints,5);
        j = 1;
        for i=oi:delta:of
            waypoints(j,1:4) = [P i];
            waypoints(j,5) = time + ts;
            time = waypoints(j,5);
            j=j+1;
        end
        for i=0:nwpoints-1
            waypoints(i+j,1:4) = waypoints(j-1,1:4);
            waypoints(i+j,5) = waypoints(i+j-1,5) + ts;
        end
```

Código D.2: Sección del generador de trayectorias encargada de la rotación sobre el eje central.

D.1.3. Trayectoria circular con tres puntos

La trayectoria circular es la más compleja de todas. Esta funcionalidad toma como argumentos tres puntos. En primer lugar, detecta el plano en el que se produce la rotación haciendo una comparación entre las coordenadas de los puntos y las reordena para introducir estos datos en la función «circle_calculations» ya que los cálculos se realizan en dos dimensiones. Se guarda en la variable «plane» el plano para reordenar finalmente los puntos de la trayectoria obtenidos. Finalmente, teniendo los ángulos inicial y final de la trayectoria, se calculan los puntos intermedios equiespaciados la distancia determinada por «angular_wp_distance». Con la variable «plane» se reordenan estos puntos en el plano correspondiente.

```
    elseif nargin == 5 && numel(arg1)~=1
%         disp('Circular trayectory')
        orient = wp_ant(4);
        P1 = wp_ant(1:3);
        P2 = arg1;
        P3 = arg2;
        t_t = arg3;
        t_w = arg4;
        time = wp_ant(end);
```

```
10          if P1(1)==P2(1) && P1(1)==P3(1)
11              % Rotation in the plane YZ (X axis)
12              p1 = [P1(2) P1(3)];
13              p2 = [P2(2) P2(3)];
14              p3 = [P3(2) P3(3)];
15              plane = 'yz';
16          elseif P1(2)==P2(2) && P1(2)==P3(2)
17              % Rotation in the plane XZ (Y axis)
18              p1 = [P1(1) P1(3)];
19              p2 = [P2(1) P2(3)];
20              p3 = [P3(1) P3(3)];
21              plane = 'xz';
22          else
23              % Rotation in the plane XY (Z axis)
24              p1 = P1(1:2);
25              p2 = P2(1:2);
26              p3 = P3(1:2);
27              plane = 'xy';
28          end
29          [r,h,k,a1,a2,n] = circle_calculations(p1,p2,p3,plane)
               ;
30          atotal = abs(a1-a2);
31          npoints = ceil(t_t/ts);
32          nwpoints = ceil(t_w/ts);
33          delta_a = abs(atotal)/npoints*n;
34          waypoints = zeros(npoints,5);
35          j=1;
36          for i=a1:delta_a:a2
37              delta_1 = r*cos(i)+h;
38              delta_2 = r*sin(i)+k;
39
40              if strcmp(plane,'yz')
41                  waypoints(j,1:4) = [P1(1) delta_1 delta_2
                       orient];
42              elseif strcmp(plane,'xz')
43                  waypoints(j,1:4) = [delta_1 P2(2) delta_2
                       orient];
44              elseif strcmp(plane,'xy')
45                  waypoints(j,1:4) = [delta_1 delta_2 P3(3)
                       orient];
46              end
47              waypoints(j,5) = time + ts;
48              time = waypoints(j,5);
49              j=j+1;
50          end
51          for i=0:nwpoints-1
```

```
52              waypoints(i+j,1:4) = waypoints(j-1,1:4);
53              waypoints(i+j,5) = waypoints(i+j-1,5) + ts;
54          end
```

Código D.3: Sección del generador de trayectorias encargada de la trayectoria circular con tres puntos.

D.1.4. Trayectoria circular con punto, centro, arco y plano

Esta última trayectoria sigue un proceso de cálculo más sencillo que la anterior. Igualmente se subdivide el ángulo de rotación «arc» en puntos equidistantes. Teniendo un punto y el centro se determina el radio del arco y conociendo el plano, se realiza un cálculo de las coordenadas en dos dimensiones y se ordenan en el vector *waypoints*.

```
1       elseif nargin == 6 && numel(arg1)==1
2   %         disp('Arc trayectory')
3           p1 = wp_ant(1:3);
4           orient = wp_ant(4);
5           arc = arg1;
6           O = arg2;
7           plane = arg3;
8           t_t = arg5;
9           t_w = arg4;
10          time = wp_ant(end);
11          r = norm(p1-O);
12          npoints = ceil(t_t/ts);
13          nwpoints = ceil(t_w/ts);
14          delta_a = abs(arc)/npoints;
15          waypoints = zeros(npoints,5);
16          ai = angle_selector(O,p1,plane);
17          af = ai+arc;
18          j = 1;
19          for i = ai:delta_a*sign(arc):af
20              if strcmp(plane,'yz')
21                  delta_1 = r*cos(i)+O(2);
22                  delta_2 = r*sin(i)+O(3);
23                  waypoints(j,1:4) = [p1(1) delta_1 delta_2
                        orient];
24              elseif strcmp(plane,'xz')
25                  delta_1 = r*cos(i)+O(1);
26                  delta_2 = r*sin(i)+O(3);
27                  waypoints(j,1:4) = [delta_1 p1(2) delta_2
                        orient];
28              elseif strcmp(plane,'xy')
29                  delta_1 = r*cos(i)+O(1);
30                  delta_2 = r*sin(i)+O(2);
```

```
31                  waypoints(j,1:4) = [delta_1 delta_2 p1(3)
                        orient];
32              end
33              waypoints(j,5) = time + ts;
34              time = waypoints(j,5);
35              j=j+1;
36          end
37          for i=0:nwpoints-1
38              waypoints(i+j,1:4) = waypoints(j-1,1:4);
39              waypoints(i+j,5) = waypoints(i+j-1,5) + ts;
40          end
```

Código D.4: Sección del generador de trayectorias encargada de la trayectoria circular con punto centro arco y plano.

D.2. Mejora de la trayectoria lineal

Durante el proceso de diseño del generador de trayectorias se observó que se podía incurrir en casos de aceleración infinita con lo que se planteó la posibilidad de suavizar las trayectorias lineales y de rotación introduciendo una función para estimar la trayectoria con un polinomio interpolador de grado 7.

$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 + c_6 t^6 + c_7 t^7 \tag{D.1}$$

$$v(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4 + 6c_6 t^5 + 7c_7 t^6 \tag{D.2}$$

$$a(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3 + 30c_6 t^4 + 42c_7 t^5 \tag{D.3}$$

Esta función implementa las expresiones anteriores de manera matricial para unas condiciones de contorno de posición determinadas por la posición anterior y la posición final. Las condiciones de contorno para la velocidad y la aceleración son 0 en ambos casos.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & t & t^2 & t^3 & t^4 & t^5 & t^6 & t^7 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t & 3t^2 & 4t^3 & 5t^4 & 6t^5 & 7t^6 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6t & 12t^2 & 20t^3 & 30t^4 & 42t^5 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 24t & 60t^2 & 120t^3 & 210t^4 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} x_i \\ x_f \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{D.4}$$

Despejando el sistema anterior para el tiempo final se obtiene el valor de las constantes del polinomio que se emplean posteriormente para cada instante de la trayectoria para obtener los valores de la posición. En la figura D.1 se muestra un ejemplo del uso de esta función para el trazado de trayectorias lineales.

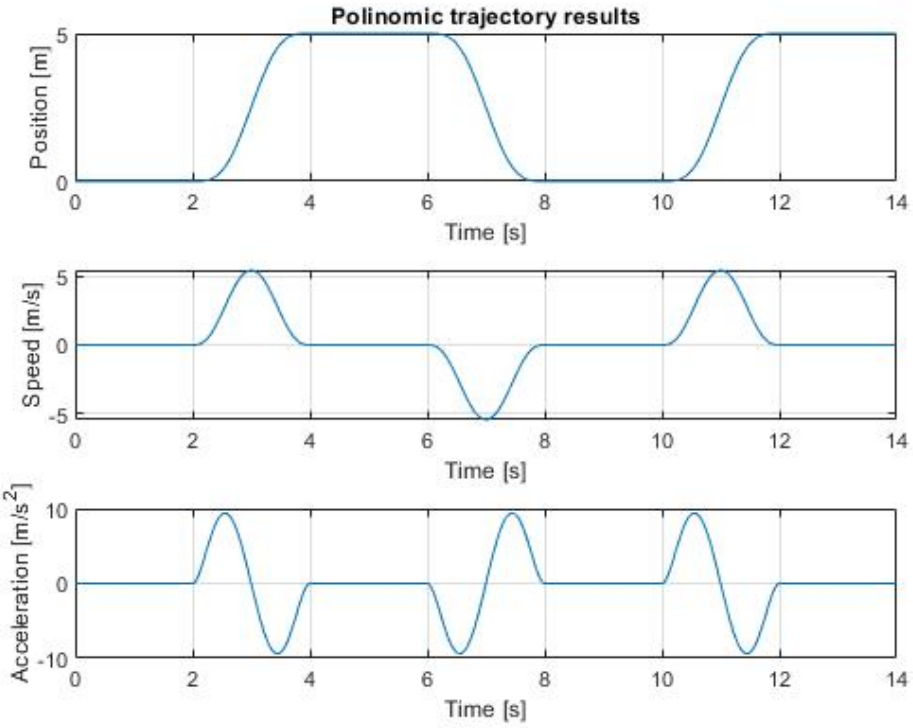

Figura D.1: Resultados del trazado de trayectorias polinomiales de grado 7.

D.3. Resultados

De manera preliminar se han trazado las siguientes trayectorias para comprobar el funcionamiento del generador inicial. Posteriormente, se ha incorporado el generador de trayectorias polinomiales para la trayectoria de la figura D.2 y se ha modificado la orientación obteniendo la trayectoria de la imagen D.5.

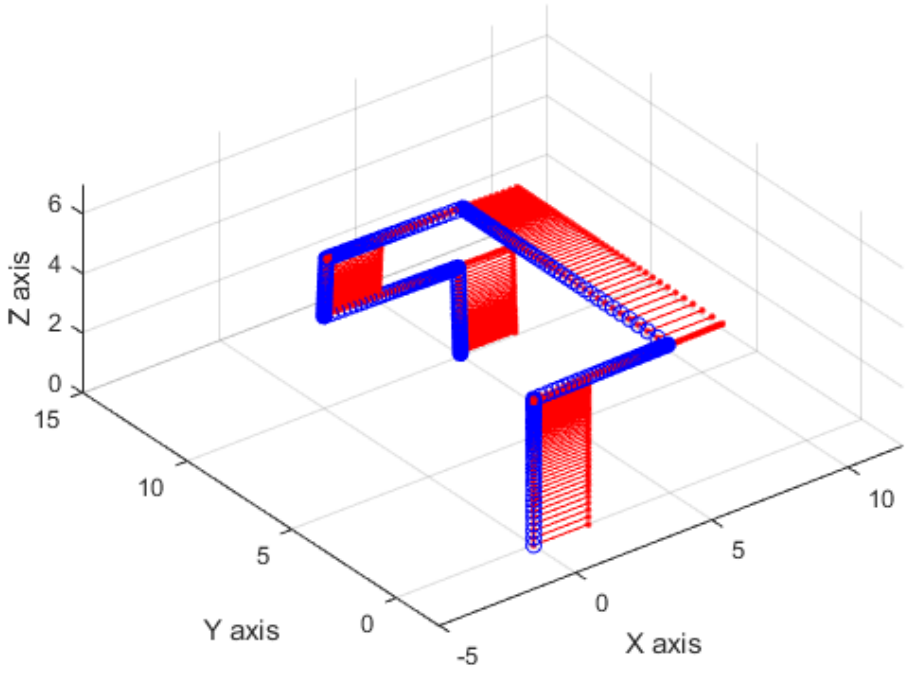

Figura D.2: Trayectoria de prueba 1.

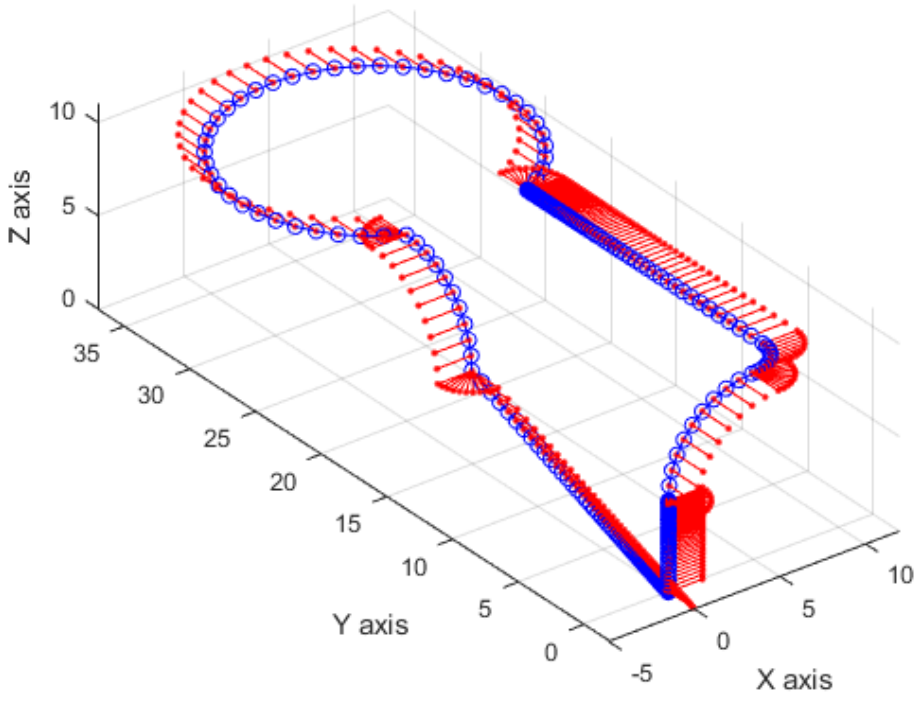

Figura D.3: Trayectoria de prueba 2.

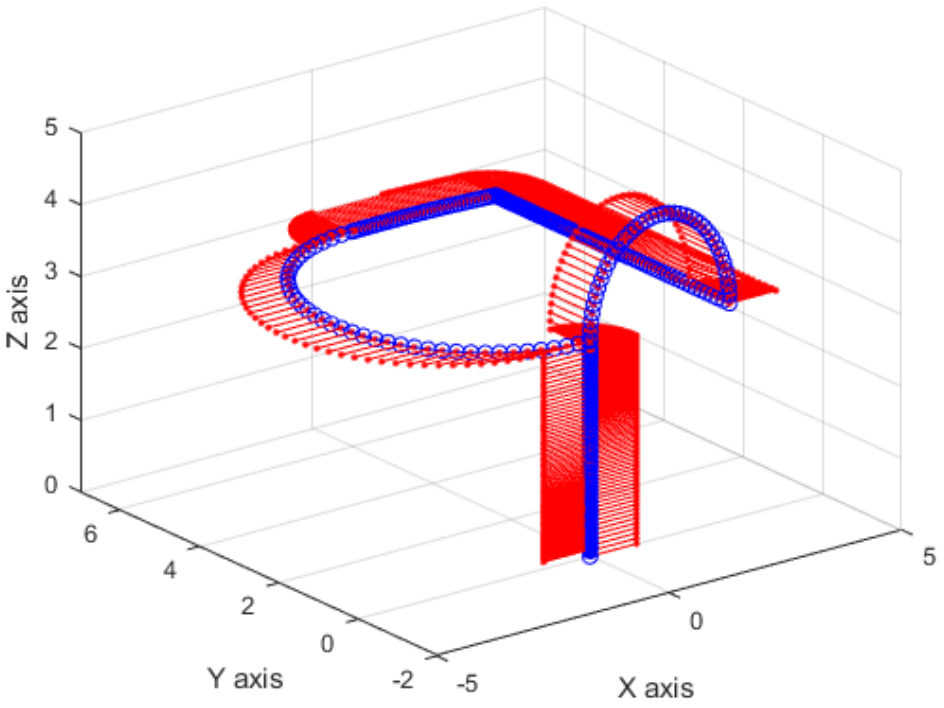

Figura D.4: Trayectoria de prueba 3.

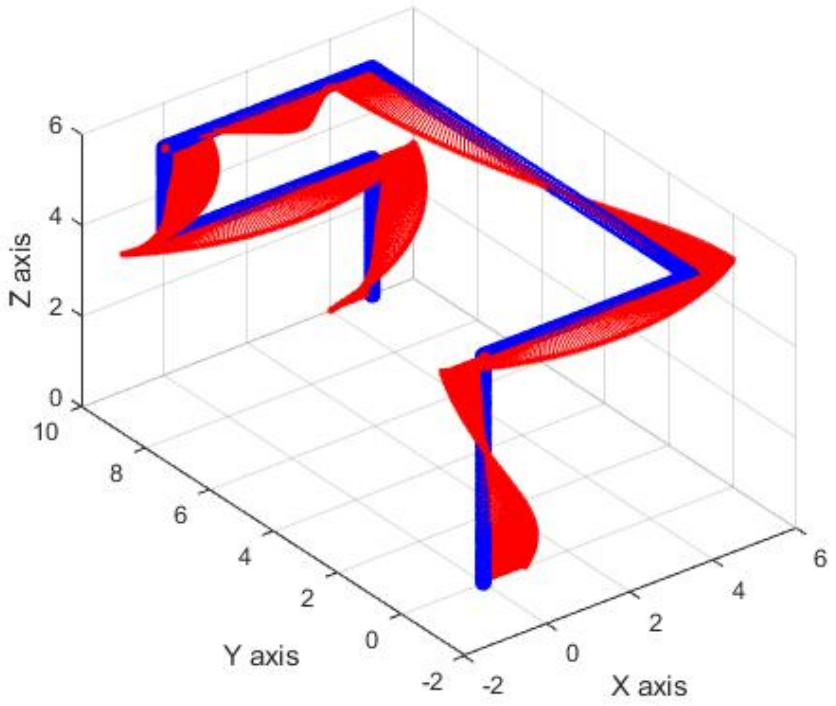

Figura D.5: Trayectoria de prueba 1 mejorada.

Códigos relacionados con el generador de trayectorias.

```
function waypoints = trayectory_calculator(wp_ant,arg1,arg2,
   arg3,arg4,arg5)
    global ts

    if nargin == 5 && numel(arg1)~=1
%         disp('Circular trayectory')
        orient = wp_ant(4);
        P1 = wp_ant(1:3);
        P2 = arg1;
        P3 = arg2;
        t_t = arg3;
        t_w = arg4;
        time = wp_ant(end);
        if P1(1)==P2(1) && P1(1)==P3(1)
            % Rotation in the plane YZ (X axis)
            p1 = [P1(2) P1(3)];
            p2 = [P2(2) P2(3)];
            p3 = [P3(2) P3(3)];
            plane = 'yz';
        elseif P1(2)==P2(2) && P1(2)==P3(2)
            % Rotation in the plane XZ (Y axis)
            p1 = [P1(1) P1(3)];
            p2 = [P2(1) P2(3)];
            p3 = [P3(1) P3(3)];
            plane = 'xz';
        else
            % Rotation in the plane XY (Z axis)
            p1 = P1(1:2);
            p2 = P2(1:2);
            p3 = P3(1:2);
            plane = 'xy';
        end
        [r,h,k,a1,a2,n] = circle_calculations(p1,p2,p3,plane)
           ;
        atotal = abs(a1-a2);
        npoints = ceil(t_t/ts);
        nwpoints = ceil(t_w/ts);
        delta_a = abs(atotal)/npoints*n;
        waypoints = zeros(npoints,5);
        j=1;
        for i=a1:delta_a:a2
            delta_1 = r*cos(i)+h;
            delta_2 = r*sin(i)+k;

```

```
43              if strcmp(plane,'yz')
44                  waypoints(j,1:4) = [P1(1) delta_1 delta_2
                        orient];
45              elseif strcmp(plane,'xz')
46                  waypoints(j,1:4) = [delta_1 P2(2) delta_2
                        orient];
47              elseif strcmp(plane,'xy')
48                  waypoints(j,1:4) = [delta_1 delta_2 P3(3)
                        orient];
49              end
50              waypoints(j,5) = time + ts;
51              time = waypoints(j,5);
52              j=j+1;
53          end
54          for i=0:nwpoints-1
55              waypoints(i+j,1:4) = waypoints(j-1,1:4);
56              waypoints(i+j,5) = waypoints(i+j-1,5) + ts;
57          end
58
59      elseif nargin == 6 && numel(arg1)==1
60  %         disp('Arc trayectory')
61          p1 = wp_ant(1:3);
62          orient = wp_ant(4);
63          arc = arg1;
64          O = arg2;
65          plane = arg3;
66          t_t = arg5;
67          t_w = arg4;
68          time = wp_ant(end);
69          r = norm(p1-O);
70          npoints = ceil(t_t/ts);
71          nwpoints = ceil(t_w/ts);
72          delta_a = abs(arc)/npoints;
73          waypoints = zeros(npoints,5);
74          ai = angle_selector(O,p1,plane);
75          af = ai+arc;
76          j = 1;
77          for i = ai:delta_a*sign(arc):af
78              if strcmp(plane,'yz')
79                  delta_1 = r*cos(i)+O(2);
80                  delta_2 = r*sin(i)+O(3);
81                  waypoints(j,1:4) = [p1(1) delta_1 delta_2
                        orient];
82              elseif strcmp(plane,'xz')
83                  delta_1 = r*cos(i)+O(1);
84                  delta_2 = r*sin(i)+O(3);
```

```
85                  waypoints(j,1:4) = [delta_1 p1(2) delta_2
                        orient];
86              elseif strcmp(plane,'xy')
87                  delta_1 = r*cos(i)+O(1);
88                  delta_2 = r*sin(i)+O(2);
89                  waypoints(j,1:4) = [delta_1 delta_2 p1(3)
                        orient];
90              end
91              waypoints(j,5) = time + ts;
92              time = waypoints(j,5);
93              j=j+1;
94          end
95          for i=0:nwpoints-1
96              waypoints(i+j,1:4) = waypoints(j-1,1:4);
97              waypoints(i+j,5) = waypoints(i+j-1,5) + ts;
98          end
99
100     elseif nargin == 4 && numel(arg1)~=1
101 %           disp('Linear trayectory')
102         P2 = arg1;
103         t_t = arg2;
104         t_w = arg3;
105         time = wp_ant(end);
106         orient = wp_ant(4);
107         distance = norm(P2-wp_ant(1:3));
108         npoints = ceil(t_t/ts);
109         nwpoints = ceil(t_w/ts);
110         waypoints = zeros(npoints+nwpoints,5);
111         j=1;
112         dmin = 0.1;
113         waypoints(1,1:4) = wp_ant(1:4);
114         while(distance>dmin)
115             distance = norm(P2-waypoints(j,1:3));
116             direction = (P2-waypoints(j,1:3))/distance;
117             delta_d = distance/npoints;
118             waypoints(j+1,1:4) = [waypoints(j,1:3)+direction*
                    delta_d orient];
119
120             waypoints(j,5) = time + ts;
121             time = waypoints(j,5);
122             wp_ant(1:3) = waypoints(j,1:3);
123             j=j+1;
124         end
125         for i=0:nwpoints-1
126             waypoints(i+j,1:4) = waypoints(j-1,1:4);
127             waypoints(i+j,5) = waypoints(i+j-1,5) + ts;
```

```
        end

    elseif nargin == 4 && numel(arg1)==1
%         disp('Rotation')
        t_t = arg2;
        t_w = arg3;
        time = wp_ant(end);
        P = wp_ant(1:3);
        arc = arg1;
        oi = wp_ant(4);
        of = oi+arc;
        npoints = ceil(t_t/ts);
        nwpoints = ceil(t_w/ts);
        delta = arc/npoints;
        waypoints = zeros(npoints,5);
        j = 1;
        for i=oi:delta:of
            waypoints(j,1:4) = [P i];
            waypoints(j,5) = time + ts;
            time = waypoints(j,5);
            j=j+1;
        end
        for i=0:nwpoints-1
            waypoints(i+j,1:4) = waypoints(j-1,1:4);
            waypoints(i+j,5) = waypoints(i+j-1,5) + ts;
        end
    end
end
```

Código D.5: Función para la generación de trayectorias

```
function [r,h,k,a1,a3,n]=circle_calculations(p1,p2,p3,plane)

    A = [p1(1) p1(2) 1;
         p2(1) p2(2) 1;
         p3(1) p3(2) 1];
    B = [-(p1(1)^2+p1(2)^2);
         -(p2(1)^2+p2(2)^2);
         -(p3(1)^2+p3(2)^2)];

    X = linsolve(A,B);
    D = X(1);
    E = X(2);
    F = X(3);

    h = D/(-2);             % Coordenada x del centro
    k = E/(-2);             % Coordenada y del centro
```

```
    r = sqrt(h^2+k^2-F);     % Radio

    O2d  = [h k];

    a1 = angle_selector(O2d,p1,plane);
    a2 = angle_selector(O2d,p2,plane);
    a3 = angle_selector(O2d,p3,plane);

    if      a1>a2 && a1>a3 && a2>a3
        n = -1;
        caso = 1;
    elseif a1>a2 && a1>a3 && a2<a3
        n = 1;
        a2 = a2+2*pi;
        a3 = a3+2*pi;
        caso = 2;
    elseif a1>a2 && a1<a3 && a2>a3
        caso = 3;
    elseif a1>a2 && a1<a3 && a2<a3
        n = -1;
        a3 = a3-2*pi;
        caso = 4;
    elseif a1<a2 && a1>a3 && a2>a3
        a3 = 2*pi+a3;
        n = 1;
        caso = 5;
    elseif a1<a2 && a1>a3 && a2<a3
        caso = 6;
    elseif a1<a2 && a1<a3 && a2>a3
        a1 = a1+2*pi;
        n = -1;
        caso = 7;
    elseif a1<a2 && a1<a3 && a2<a3
        n = 1;
        caso = 8;
    else
        n = 0;
    end
end
```

Código D.6: Función para el cálculo de los parámetros de la trayectoria circular.

```
function theta=angle_calculator(v1,v2)
    if numel(v1) == 2
        prod = v1(1)*v2(1)+v1(2)*v2(2);
    else
        prod = v1(1)*v2(1)+v1(2)*v2(2)+v1(3)*v2(3);
```

```
    end
    theta=acos(prod/(norm(v1)*norm(v2)));
end
```

Código D.7: Función para el cálculo del ángulo entre dos vectores mediante el producto escalar.

```
function theta = angle_selector(O,P,plane)
    if numel(O)==3 && numel(P)==3
        if strcmp(plane,'yz')
            h = O(2);
            k = O(3);
            P = [P(2) P(3)];
        elseif strcmp(plane,'xz')
            h = O(1);
            k = O(3);
            P = [P(1) P(3)];
        elseif strcmp(plane,'xy')
            h = O(1);
            k = O(2);
            P = [P(1) P(2)];
        end
    else
        h = O(1);
        k = O(2);
    end
    Oh = [h+1 k]-[h k];
    Ov = [h k+1]-[h k];
    OP = P-[h k];
    Ah = angle_calculator(Oh,OP);
    Av = angle_calculator(Ov,OP);
    if Av<pi/2
        theta = Ah;
    else
        theta = 2*pi-Ah;
    end
end
```

Código D.8: Función para seleccionar el ángulo correcto entre las dos posibilidades del producto escalar.

```
function v_fin = vector_rotation(v_ini,alpha)
    M = [cos(alpha) -sin(alpha) 0;
         sin(alpha)  cos(alpha) 0;
         0           0          1];
    if numel(v_ini)<3
        v = [v_ini 0]*M;
        v_fin = v(1:2);
```

```
    else
        v_fin = v_ini*M;
    end
end
```

Código D.9: Función para rotar un vector inicial un ángulo de α radianes.

```
function [waypoints]=pol_7(P1,P2,tf,ti,ts)
    tf_rel=tf-ti;

    N_tot=round(tf_rel/ts);
    waypoints = zeros(N_tot,13);
    factores=[1     0       0       0       0       0       0
                0;
              1     tf_rel      tf_rel^2    tf_rel^3
                tf_rel^4    tf_rel^5    tf_rel^6    tf_rel
                ^7;
              0     1       0       0       0       0       0
                          0;
              0     1       2*tf_rel    3*tf_rel^2  4*tf_rel
                ^3  5*tf_rel^4  6*tf_rel^5  7*tf_rel^6;
              0     0       2       0       0       0
                0         0;
              0     0       2       6*tf_rel    12*tf_rel^2
                20*tf_rel^3 30*tf_rel^4  42*tf_rel^5;
              0     0       0       6       0       0
                0         0;
              0     0       0       6       24*tf_rel   60*
                tf_rel^2 120*tf_rel^3 210*tf_rel^4];

    for i=1:length(P1)
        qi=P1(i);
        qf=P2(i);

        equalto=[qi;qf;0;0;0;0;0;0];

        solucion=factores^-1*equalto;

        c0=solucion(1);
        c1=solucion(2);
        c2=solucion(3);
        c3=solucion(4);
        c4=solucion(5);
        c5=solucion(6);
        c6=solucion(7);
        c7=solucion(8);

```

```
32          for fila=1:N_tot
33
34          tiempo=(fila-1)*ts;
35
36              if tiempo > tf_rel
37                  qref(fila)=qf;
38                  qdref(fila)=0;
39                  qddref(fila)=0;
40                  qdddref(fila)=0;
41                  q3dref(fila)=0;
42                  q4dref(fila)=0;
43              else
44                  %pos
45                  qref(fila)=c7*tiempo^7+c6*tiempo^6+c5*tiempo
                        ^5+c4*tiempo^4+c3*tiempo^3+c2*tiempo^2+c1*
                        tiempo+c0;
46                  %vel
47                  qdref(fila)=7*c7*tiempo^6+6*c6*tiempo^5+5*c5*
                        tiempo^4+4*c4*tiempo^3+3*c3*tiempo^2+2*c2*
                        tiempo+c1;
48                  %acc
49                  qddref(fila)=42*c7*tiempo^5+30*c6*tiempo
                        ^4+20*c5*tiempo^3+12*c4*tiempo^2+6*c3*
                        tiempo+2*c2;
50  %                 %jerk
51  %                 q3dref(fila)=210*c7*tiempo^4+120*c6*tiempo
    ^3+60*c5*tiempo^2+24*c4*tiempo+6*c3;
52  %                 %jerk derivative
53  %                 q4dref(fila)=840*c7*tiempo^3+360*c6*tiempo
    ^2+120*c5*tiempo+24*c4;
54              end
55          end
56
57          [qref]=[qref]';
58          [qdref]=[qdref]';
59          [qddref]=[qddref]';
60
61          waypoints(:,i)=qref;
62          waypoints(:,i+4)=qdref;
63          waypoints(:,i+8)=qddref;
64
65      end
66
67          t=[ti:ts:tf]';
68
69          if size(t,1)~=size(waypoints,1)
```

```
70              diff = abs(size(t,1)-size(waypoints,1));
71              t = t(1:end-diff,:);
72          end
73
74          waypoints(:,13) = t;
75  end
```

Código D.10: Código que recoge el generador de trayectorias lineales con interpolación de grado 7.

E. Anexo V: Presupuesto

Tabla E.1: Costes materiales.

Descripción	**Uds**	**Precio por ud**	**Uso**	**Total**
Licencia de *Matlab*&*Simulink*	1	840 €/Year	40 %	336 €
Ordenador portátil MSI Modern 14	1	1399€	10 %	140 €
TOTAL	1	-	-	476 €

Tabla E.2: Coste de mano de obra.

Descripción	**Uds (h)**	**Precio por ud**	**Total**
Estudio inicial del proyecto	8	20 €/h	160 €
Investigación inicial	10	20 €/h	200 €
Elaboración del generador de trayectorias	15	20 €/h	300 €
Desarrollo matemático del modelo dinámico del dron	10	20 €/h	400 €
Construcción del modelo dinámico del dron	15	20 €/h	300 €
Desarrollo matemático de la metodología LAB	20	20 €/h	200 €
Desarrollo de los modelos LAB en continuo	40	20 €/h	800 €
Desarrollo del algoritmo LAB en discreto	20	20 €/h	400 €
Estudio tras el cambio de curso del proyecto	10	20 €/h	200 €
Desarrollo matemático de la metodología FL	20	20 €/h	400 €
Implementación de la metodología FL	40	20 €/h	800 €
Construcción del entorno de simulación 1	5	20 €/h	100 €
Construcción del entorno de simulación 2	15	20 €/h	300 €
Simulaciones y calibración del controlador LAB	50	20 €/h	1000 €
Simulaciones y calibración del controlador FL	50	20 €/h	1000 €
Obtención de resultados y análisis	30	20 €/h	600 €
Reuniones con el tutor	15	20 €/h	300 €
Documentación extra	20	20 €/h	400 €
TOTAL	393	-	7860 €

Tabla E.3: Costes totales.

Descripción	**Total**
Costes materiales	476 €
Costes de mano de obra	7860€
TOTAL	8336 €