Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Simulación y control de un quadrotor

Autor: Rafael Luque Berraquero
Tutor: Francisco Rodríguez Rubio

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Simulación y control de un quadrotor

Autor:
Rafael Luque Berraquero

Tutor:
Francisco Rodríguez Rubio
Catedrático de Universidad

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo de Fin de Grado: Simulación y control de un quadrotor

Autor: Rafael Luque Berraquero

Tutor: Francisco Rodríguez Rubio

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi tutor, Francisco Rodríguez, por su tiempo y dedicación.

A mi familia, por su apoyo incondicional.

A mis compañeros.

Agradecimientos

En primer lugar, dar las gracias a mi familia, ellos me han animado y alentado en los momentos complicados a lo largo de toda mi vida y siempre confiaron en mí. Dar las gracias al profesor Francisco Rodríguez que me dio la oportunidad de realizar este trabajo y que me ha ayudado en todo lo que he necesitado. A mis profesores, porque de una manera u otra todos tienen una parte en este documento. Con todo ello mencionar y dar las gracias a mis compañeros y amigos por formar parte de este camino en el que hemos demostrado que con esfuerzo y dedicación nos podemos superar.

Rafael Luque Berraquero

Sevilla, 2017

Resumen

El objetivo final de este proyecto es obtener un simulador de seguimiento de trayectorias en un entorno con obstáculos para un robot móvil aéreo y autónomo.

Para ello, en primer lugar, se realiza un modelado dinámico de un tipo concreto de quadrotor, de forma que este simule el movimiento real que realizaría el vehículo y devuelva los datos reales que aportarían los sensores.

Tras esto, se desarrollará un generador de trayectorias que resuelva el problema de la evitación de obstáculos, empleando mapas conocidos, para las coordenadas iniciales y finales deseadas por el usuario, utilizando para ello técnicas basadas en exploración mediante grafos.

A continuación, se plantea un sistema de control de la trayectoria que se divide en dos partes: "position controller" y "attitude controller". La primera de ellas se encarga de realizar el control de las coordenadas X, Y y Z, y la segunda, de los ángulos de rotación. Para la implementación de dicho control se utilizará un regulador con doble lazo de realimentación, para el cual se calculará el valor de los parámetros que mejor se ajusten en cada variable.

Por último, se diseñará una interfaz gráfica que permita al usuario realizar pruebas de simulación para diferentes entornos y con distintas trayectorias de modo que se puedan comparar los resultados obtenidos y la validez de los controladores.

Abstract

The main purpose of this project is to design a simulator of an UAV trajectory tracking, in an environment with obstacles.

For this, first, a dynamic model of a particular kind of quadrotor is performed, so that it simulates the real movement that the vehicle would perform and returns the real data that the sensors would provide.

After this, a trajectory generator will be developed so that it solves the obstacle avoidance problem, using known maps, for the initial and final coordinates desired by the user, using techniques based on graphs search.

Then, a trajectory control system is proposed that is divided into two parts: "position controller" and "attitude controller". The first one is responsible for the control of the X, Y and Z coordinates, and the second one, of the rotation angles. For the implementation of this control, a regulator with double feedback loop will be used, for which the value of the parameters that best fit in each variable will be calculated.

Finally, a graphical interface will be designed to allow the user to make simulation tests for different environments and with different trajectories, so that the results obtained and the validity of the controllers can be compared.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

A^T	Matriz traspuesta
$\mathbb{R}$	Todos los números reales
sen	Función seno
cos	Función coseno
$\partial y/\partial x$	Derivada parcial de *y* respecto x
$A \rightarrow B$	Del sistema de referencia A al sistema B

1 INTRODUCCIÓN

Los vehículos aéreos no tripulados (del inglés UAV's – Unmanned Aerial Vehicles) son cada vez más comunes y presentan una enorme gama de tamaños y formas. Las aplicaciones de estos vehículos incluyen desde operaciones militares, vigilancia, espionaje, búsqueda y rescate, hasta investigaciones meteorológicas, filmación cinematográfica e investigación robótica (Corke P., 2011).

Hasta hace poco tiempo, controlar un vehículo aéreo de manera autónoma era una tarea complicada debido a las limitaciones impuestas por el hardware hasta entonces existente. Lo que hizo posible la construcción de robots aéreos autónomos fue los recientes avances tecnológicos en actuadores y sensores en escala reducida, así como en el almacenamiento de energía y en el procesamiento de datos. Además, el diseño de controladores para este tipo de vehículos es complejo, debido principalmente a la dinámica presente en los sistemas aerodinámicos, los cuales son multivariables, subactuados y además presentan diversas características no lineales. Esto significa que las leyes clásicas de control lineales y monovariables pueden ver reducida su utilidad cuando se opera en condiciones no muy lejanas a las de equilibrio, ya que pueden aparecer inestabilidades. Por otra parte, las técnicas desarrolladas para robots totalmente actuados tampoco se aplican directamente al caso de sistemas mecánicos no lineales subactuados (Vianna G., 2007).

Dentro de los UAV se pueden encontrar:

- UAV's de ala fija, que se basan en un principio similar al de los aviones de pasajeros compuestos por un propulsor o "jet" que proporciona el empuje delantero y unas alas, que permiten la sustentación, con una superficie controlable, para maniobrar.
- UAV's de alas rotativas, con una gran variedad de configuraciones entre las que se incluyen el helicóptero convencional con un rotor principal y otro en la cola; el "coax", con rotores coaxiales contrarrotativos; y los "quadrotors" o helicópteros cuatrirrotor.

En este documento se van a tratar los vehículos aéreos quadrotor, los cuales están ampliamente extendidos. Estos vehículos, en comparación con los helicópteros convencionales, presentan una serie de ventajas: son capaces de despegar verticalmente; son más fáciles de pilotar, modelar y maniobrar, pues el movimiento se consigue a través del accionamiento directo de los rotores, variando sus velocidades, mientras que, en un helicóptero convencional, la velocidad de giro de las hélices suele ser constante, controlando el movimiento mediante la variación de los ángulos de ataque de las palas, lo que los hace adecuados para el uso en laboratorio o como hobby; y poseen motores eléctricos en lugar de motores de combustión. Por contrapartida, presentan un aumento de peso y un aumento en el consumo de energía debido a los motores extras.

Desde el punto de vista del control, la construcción de este tipo de helicóptero miniatura está lejos de simplificar el problema: más bien sucede lo contrario. Esto se debe a que los pares y fuerzas necesarios para controlar el sistema son aplicados no solo a través de efectos aerodinámicos, sino también a través del efecto de acoplamiento que aparece entre la dinámica de los rotores y la del cuerpo de la maqueta, como consecuencia del principio de acción-reacción originado en la aceleración y desaceleración de los grupos motor-hélice (efecto que no sucede en el control con velocidad de hélices constantes) (Vianna G., 2007).

2 ALCANCE

El objetivo planteado para este proyecto es obtener las ecuaciones dinámicas que permitan modelar un vehículo quadrotor, basándose en las leyes y principios de Newton y Euler.

Así como también se pretende desarrollar el sistema de control, para el cual se procederá a realizar una estrategia de control de las variables de entrada de forma desacoplada: cada variable se controlará de forma independiente a las demás. Como consecuencia, se realizará su división en dos lazos: "position controller" y "attitude controller".

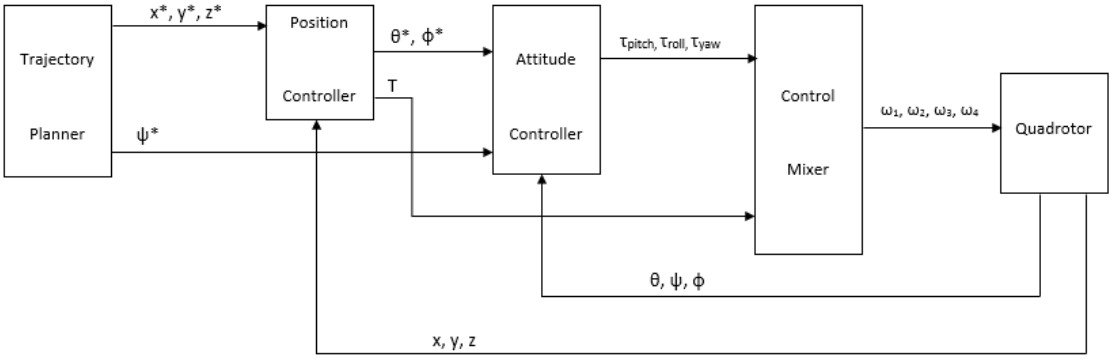


Figura 2-1. Esquema completo del simulador.

Por otro lado, se diseñará una interfaz gráfica que permita tanto la elección de los parámetros iniciales como la visualización de la trayectoria en 3D seguida y las gráficas correspondientes a las evoluciones temporales de las coordenadas y las fuerzas y momentos ejercidos por el quadrotor.

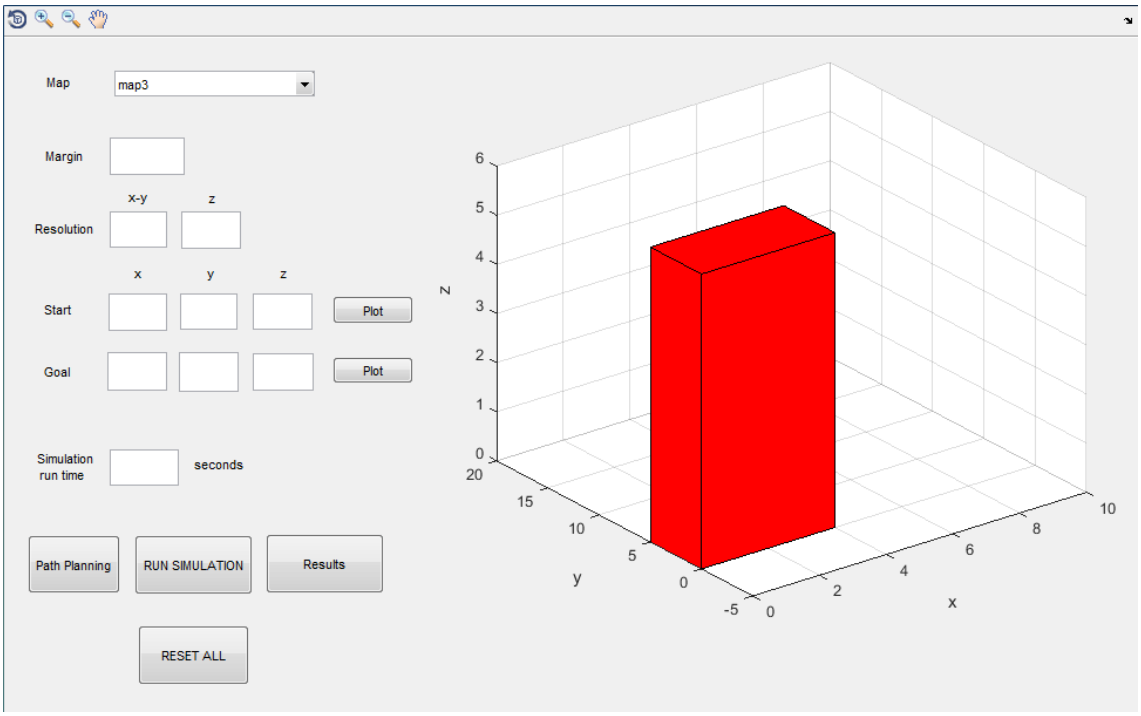


Figura 2-2. Interfaz de usuario.

Finalmente, se implementará un generador de trayectorias que se encargará de calcular la trayectoria deseada a partir de los datos aportados por el usuario a través de la interfaz. Una vez calculados los valores para las variables de posición a lo largo de la trayectoria, serán utilizados como consignas de entrada para la simulación, cuando esta se ejecute.

Todos los objetivos anteriormente mencionados estarán integrados en un archivo simulink.

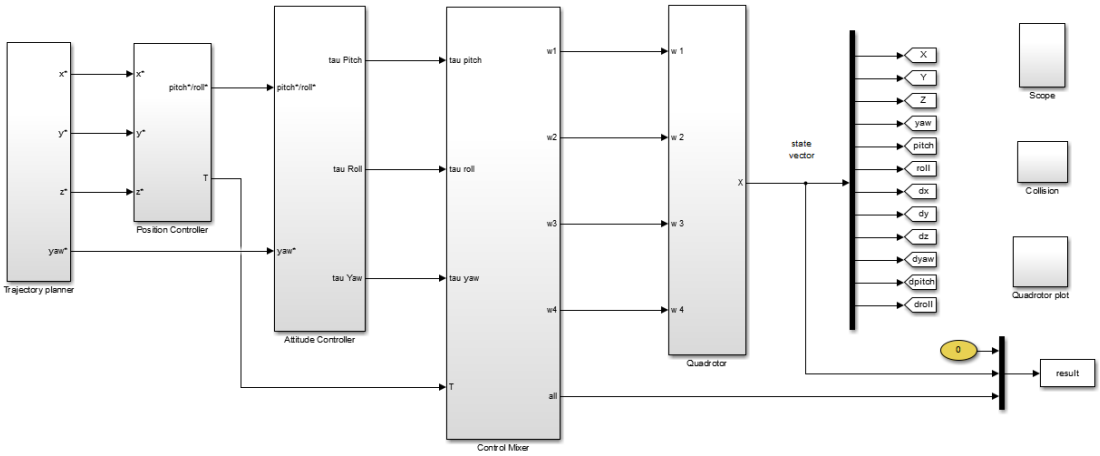


Figura 2-3. Esquema completo del simulador representado en Simulink.

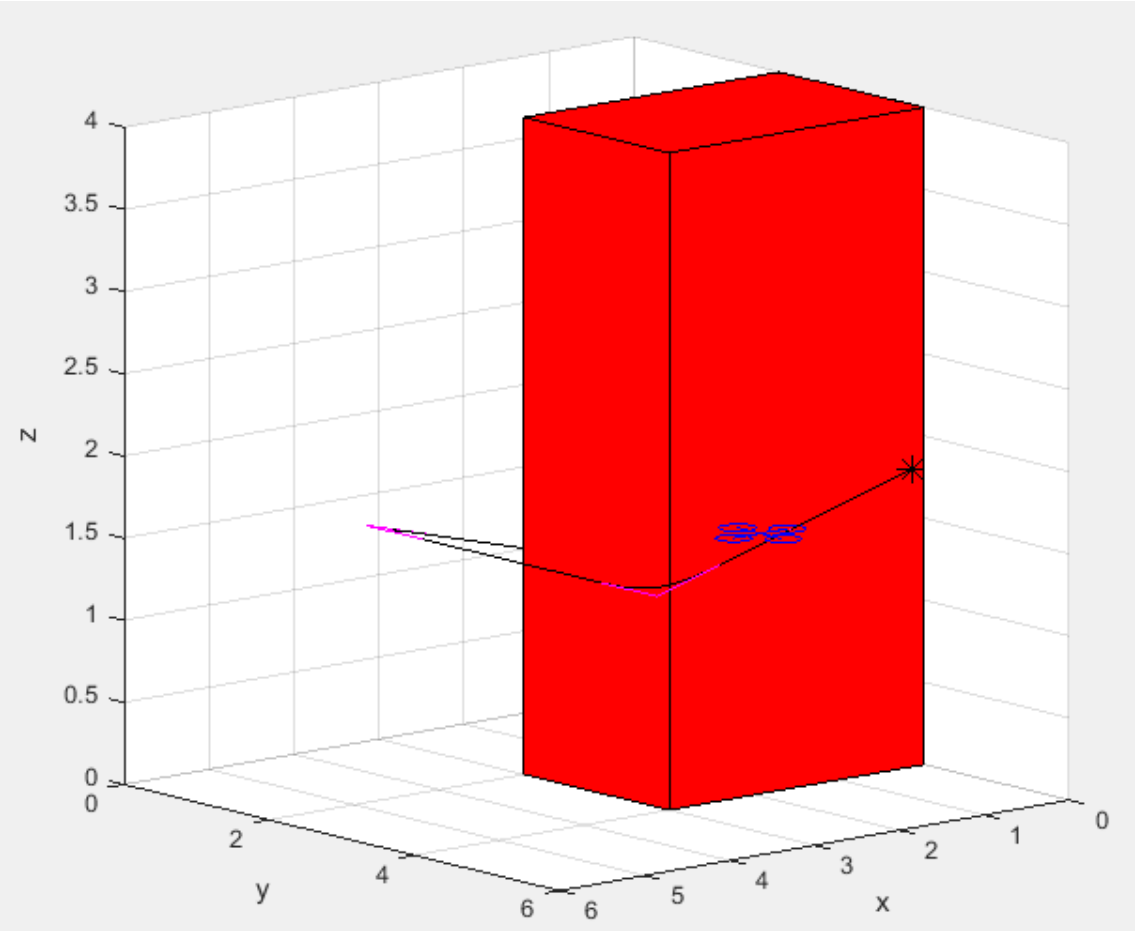


Figura 2-4. Fotograma de simulación.

3 MODELADO

3.1 Quadrotors

Como ya se comentó anteriormente, un quadrotor es un robot aéreo de tipo helicóptero, también denominado VTO (del inglés - Vertical tale-off and landings), formado por cuatro rotores coplanarios compuestos de un motor eléctrico de corriente continua, un mecanismo de engranaje y un rotor de palas, los cuales se encuentran orientados hacia arriba y a la misma distancia del centro de gravedad del fuselaje del aparato. Además, se suelen construir de forma que éstos se encuentren por encima de dicho punto para que se comporte de forma estable.

Figura 3-1. Ejemplo quadrotor real.

El modelo concreto empleado en este documento se basa en las propiedades del quadrotor "X-4 Flyer Mark III" el cual posee una masa de 4 kg y una carga útil máxima de 1 kg. Este quadrotor fue diseñado para desarrollar un alto empuje con un tamaño reducido (20% más pequeño que el quadrotor "RCtoys Draganflyer IV"), de modo que supuso un avance hacia los quadrotors industriales de alta potencia (Corke P., 2011).

Tabla 3–1. Propiedades físicas X4-Flyer.

Símbolo	Descripción	Valor
g	Gravedad	$9.81\ m/s^2$
ρ	Densidad del aire	1.184
μ	Viscosidad del aire	$1.5*10^{-5}$
M	Masa del quadrotor	$4\ kg$
J	Matriz de inercia	$\begin{bmatrix} 0.082 & 0 & 0 \\ 0 & 0.082 & 0 \\ 0 & 0 & 0.149 \end{bmatrix}\ kgm^2$
h	Altura de los rotores respecto al CdG	$0.007\ m$
d	Longitud de los brazos del quadrotor	$0.315\ m$
n_b	Número de aspas en cada rotor	2
r	Radio del rotor	$0.165\ m$
c	Longitud de cuerda del aspa	$0.018\ m$
e	Offset del eje de giro	0
M_b	Masa del aspa	$0.005\ kg$
M_c	Masa estimada de la abrazadera del eje	$0.01\ kg$
e_c	Desplazamiento de la abrazadera del inicio del aspa	$0.004\ m$
I_b	Momento de inercia del aspa	$M_b(r - e_c)^{\frac{2}{4}}$
I_c	Momento de inercia estimado de la abrazadera del eje	$M_c(e_c)^{\frac{2}{4}}$
m_b	Momento estático del aspa	$g\left(M_c\frac{e_c}{2} + M_b\frac{r}{2}\right)$
I_r	Momento de inercia total del rotor	$n_b(I_b + I_c)$
C_t	Coeficiente adimensional de empuje	0.0048
C_q	Coeficiente adimensional del torque	$C_t\sqrt{(C_t/2)}$
σ	Relación de solidez del rotor	0.0694
θ_t	Ángulo de la punta del aspa	$6.8\frac{\pi}{180}$
θ_0	Ángulo del inicio del aspa	$14.6\frac{\pi}{180}$
θ_1	Ángulo de torsión del aspa	$\theta_t - \theta_0$
θ_{75}	¾ del ángulo del aspa	$\theta_0 + 0.75\theta_1$
θ_i	Aproximación ideal el ángulo de inicio del aspa	$\theta_t\left(\frac{r}{e}\right)$
a	Pendiente del gradiente de la fuerza de sustentación	5.5
A	Área del disco del rotor	πr^2
ɣ	Número de bloqueo	$\frac{\rho a c r^4}{(I_b + I_c)}$
b	Constante de elevación	$C_t\rho A r^2$
k	Constante de resistencia aerodinámica	$C_q\rho A r^3$

El movimiento de un quadrotor se origina a partir de los cambios de velocidad de los rotores, los cuales giran en el mismo sentido dos a dos: los motores 1 y 3 giran en sentido horario mientras que el 2 y el 4 lo hacen en sentido contrario. Es decir, los que están en el mismo brazo giran en el mismo sentido. En función de la velocidad que estos posean, se pueden realizar diferentes desplazamientos a lo largo de los ejes, para algunos de los cuales se requerirán unas leves rotaciones respecto del sistema de referencia tomado desde el quadrotor, así como también, mantenerlo en sustentación.

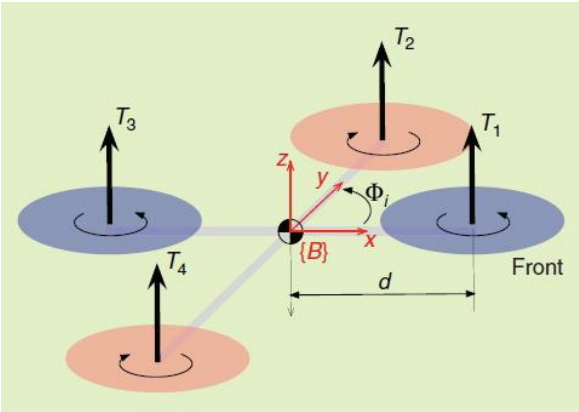


Figura 3-2. Notación rotores.

Si lo que se requiere es hacer un movimiento en el eje Z, es decir, suba o baje, la velocidad de los motores deberá ser mayor o menor que la de sustentación (velocidad mínima necesaria para que el quadrotor se mantenga en equilibrio), pero de igual magnitud para todos ellos. El giro sobre dicho eje se obtiene a partir de la diferencia en el par de torsión entre cada par de rotores, o sea, se acelera los dos rotores con sentido horario mientras se desacelera los rotores con sentido anti-horario, y viceversa.

Para realizar traslaciones respecto a los ejes X e Y, hay que hacer que uno de los motores de la rama correspondiente a la dirección que se pretende seguir gire a menos velocidad que el que esté enfrente: si se quiere mover sobre el eje X, el motor uno deberá girar más despacio que el motor 3. Esto provoca que se dé una rotación sobre el eje Y, modificando el empuje que hay en Z. Si el ángulo es muy pronunciado, éste no será suficiente para mantener en el aire el aparato, de forma que perderá altitud, pudiendo llegar a caer al suelo.

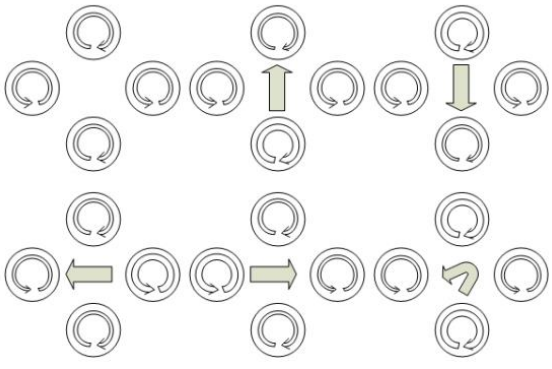


Figura 3-3. Sentido y velocidad de giro de los rotores necesario para realizar movimientos.

Realmente, la causa de esta pérdida son las limitaciones existentes respecto a las baterías que utilizan esta clase de robot como fuente de alimentación. En la situación descrita anteriormente, el sistema de control requiere una acción tan elevada que la alimentación no puede entregar la energía necesaria para que se dé dicha acción, de ahí la perdida de empuje y de altitud. Sumando a esto el hecho de que es muy difícil de implementar un control como el que podría llevar a cabo un piloto de forma manual, hace que sea complicado que el drone pueda realizar

maniobras arriesgadas como pueden ser giros de 360º sobre alguno de los ejes horizontales sin que acabe en el suelo (Martín I., 2012).

A pesar de ello, este tipo de robots ofrecen la ventaja de tener una dinámica más simple que los de un solo rotor, además de mantenerse mejor en situación de sustentación al ejercerse el empuje con los cuatro rotores.

3.2 Descripción

En esta sección se desarrollará el modelado basado en leyes físicas que describan la posición y orientación del helicóptero quadrotor. El modelo dinámico del helicóptero se presenta bajo la formulación matemática de Newton-Euler, siendo otra posible, la de Lagrange-Euler.

Para obtener tal modelo dinámico se supone el vehículo como un cuerpo rígido en el espacio, sujeto a una fuerza principal (empuje) y tres momentos (pares), generados por los rotores, responsables del movimiento del aparato, como el de la siguiente figura.

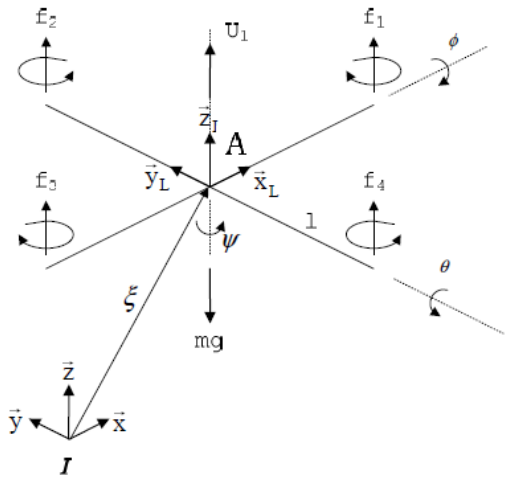


Figura 3-4. Notación sistemas de referencia inerciales y no inerciales.

El par para generar un movimiento de balanceo (eje Y) o "roll" (ángulo ϕ) se realiza mediante un desequilibrio entre las fuerzas f_2 y f_4. Para el movimiento de cabeceo (eje X) o "pitch" (ángulo θ), el desequilibrio se realizará entre las fuerzas f_1 y f_3. El movimiento en el ángulo de guiñada o "yaw" (ángulo ψ) se realizará por el desequilibrio entre los conjuntos de fuerzas (f_1, f_3) y (f_2, f_4). Este movimiento será posible ya que los rotores 1 y 3 giran en sentido contrario a los rotores 2 y 4. Finalmente, el empuje total, que hará que el quadrotor se desplace perpendicularmente al plano de los rotores (eje Z), se obtendrá como suma de las cuatro fuerzas que ejercen los rotores (Vianna G., 2007).

Estos tipos de vehículos son sistemas de vuelo de estructura ligera, por lo que el modelo dinámico deberá incluir los efectos físicos que actúan sobre el aparato: efectos aerodinámicos y pares inerciales opuestos (debidos a la rotación de las hélices), efectos de la gravedad, efectos giroscópicos (debidos a la rotación del cuerpo rígido en el espacio) y fricción del aire.

Sin embargo, debido a las diversas complejidades presentadas, se realizarán algunas simplificaciones para desarrollar el modelo como que se despreciarán el efecto suelo, se asumirá el centro de masa coincidente con el origen del sistema de coordenadas fijo al helicóptero, y que la estructura del helicóptero sea simétrica, lo que resulta en la matriz de inercia diagonal.

Para continuar con el modelado dinámico del quadrotor, es necesario conocer cómo se estima la posición y

orientación del vehículo con respecto a un sistema de coordenadas de referencia inercial. El aparato, como sólido rígido, está caracterizado por un sistema de coordenadas ligado a él y con origen en su centro de masa. Este sistema se define considerando $A = \{E_1^a, E_2^a, E_3^1\}$ como el sistema de coordenadas fijo al helicóptero, y $I = \{E_x, E_y, E_z\}$ como el sistema de coordenadas inercial, que se considerará fijo con respecto a la tierra.

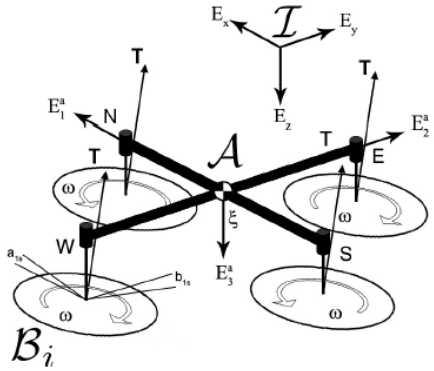


Figura 3-5. Esquema modelo del quadrotor.

Como se puede ver en la imagen, se designará el vector $\vec{\xi} = \{x, y, z\}$ como el vector de posición del centro de masa del helicóptero con respecto al sistema inercial I. Así mismo, la orientación del vehículo se supondrá dada por una matriz de rotación ${}^{I}R_A: A \rightarrow I$, donde ${}^{I}R_A$ es una matriz de rotación ortonormal.

La rotación del UAV puede ser obtenida utilizando diversos métodos, entre ellos, uno muy conocido es la convención-xyz (giro alrededor de x, y, z) para los ángulos de Euler, denominado, ángulos de Tait-Bryan. Este consiste en describir una rotación general en el espacio euclídeo tridimensional a través de tres rotaciones sucesivas en torno a los ejes del sistema móvil en el cual están definidos (Vianna G., 2007).

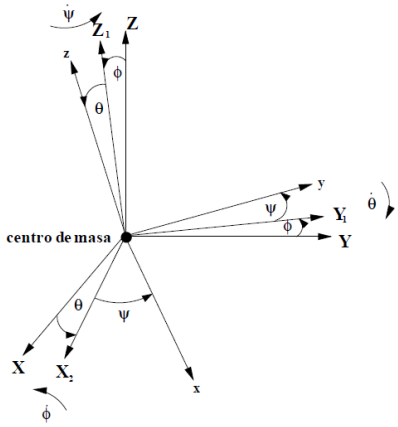


Figura 3-6. Representación de los ángulos de Tait-Brain.

Rotación según $\vec{x}$ de ϕ: $R(x,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\operatorname{sen}\phi \\ 0 & \operatorname{sen}\phi & \cos\phi \end{bmatrix}$

Rotación según $\vec{y}$ de θ: $R(y,\theta) = \begin{bmatrix} \cos\theta & 0 & \operatorname{sen}\theta \\ 0 & 1 & 0 \\ -\operatorname{sen}\theta & 0 & \cos\theta \end{bmatrix}$

Rotación según $\vec{z}$ de ψ: $R(z,\psi) = \begin{bmatrix} \cos\psi & -\operatorname{sen}\psi & 0 \\ \operatorname{sen}\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

La matriz de rotación completa de {A} respecto a {I}, viene dada por:

$$ {}^{I}R_{A} = R(z,\psi)R(y,\theta)R(x,\phi) \tag{3-1}$$

$$ {}^{I}R_{A} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\operatorname{sen}\theta\operatorname{sen}\phi - \operatorname{sen}\psi\cos\phi & \cos\psi\operatorname{sen}\theta\cos\phi + \operatorname{sen}\psi\operatorname{sen}\phi \\ \operatorname{sen}\psi\cos\theta & \operatorname{sen}\psi\operatorname{sen}\theta\operatorname{sen}\phi + \cos\psi\cos\phi & \operatorname{sen}\psi\operatorname{sen}\theta\cos\phi - \cos\psi\operatorname{sen}\phi \\ -\operatorname{sen}\theta & \cos\psi\operatorname{sen}\phi & \cos\theta\cos\phi \end{bmatrix} \tag{3-2}$$

y donde ${}^{A}R_{I} = {}^{I}R_{A}^{T}$.

A partir de la matriz de rotación generada por las tres rotaciones sucesivas y su propiedad de ortonormalidad, relacionando la derivada de la matriz ortonormal con una cierta matriz anti-simétrica, se puede obtener las ecuaciones cinemáticas de rotación del vehículo que establecen las relaciones entre las velocidades angulares (Vianna G., 2007).

3.3 Cinemática y dinámica

Antes de proceder con el modelo, es necesario realizar un balance (recuento) de todas las fuerzas ($\mathrm{F_A} = {}^{\mathrm{A}}\mathrm{R_B F_B}$) y momentos ($\tau_B$), expresadas respecto al sistema de referencia inercial, que actúan sobre el quadrotor: el peso, el empuje, los momentos de inercia, fuerzas y momentos sobre el fuselaje debido a efectos aerodinámicos y fricción del aire, etc. (Pounds P. *et al.*, 2010).

Balance de fuerzas: $F_A = {}^{A}R_B F_B = P + \sum_{i=1}^{4} t_i + A_T$ (3-3)

Balance de momentos: $\tau_B =: \sum_{i=1}^{4} m_i + \sum_{i=1}^{4} q_i + A_R$ (3-4)

- Peso: $\mathrm{P} = -\mathrm{mgE_z}$
- Empujes debidos a movimiento de cada rotor:

  Dado que el sistema de coordenadas fijo {I} tiene su eje z hacia abajo, siguiendo la convención aeroespacial, el empuje de cada rotor es un vector ascendente

$$ t_i = b\omega_i^2, \quad i = 1,2,3,4 \tag{3-5}$$

  en la dirección z negativa del vehículo, donde b, como ya se ha comentado en la tabla de propiedades del quadrotor, es la constante de elevación, que depende de la geometría y del perfil del rotor, y ω_i la velocidad angular de cada rotor.

  En situación de sustentación, el empuje total que se aplica al vehículo es la suma de todos los empujes generados por cada uno de los rotores de forma individual.

$$T = \sum_{i=1}^{4} |t_i| = b \sum_{i=1}^{4} \omega_i^2 = 4b\omega_i^2 \tag{3-6}$$

Estas expresiones son válidas siempre que el empuje generado por cada rotor esté orientado en el sentido del eje Z del quadrotor. Sin embargo, esto no siempre se cumple: cuando se producen desplazamientos en horizontal se produce un efecto aerodinámico denominado "rotor flapping", que, debido a una diferencia entre el empuje que generan las aspas que avanzan y las que retroceden (disimetría en el empuje), como consecuencia de diferentes ángulos de ataque y flujos de aire, provoca una inclinación en el plano en el que se encuentra el área del disco del rotor, y, con ello, una disminución del empuje vertical (Pounds P. *et al.*, 2010).

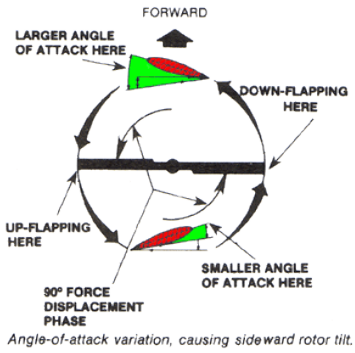


Figura 3-7. Representación de la variación del ángulo de ataque.

De modo que el empuje se debe redefinir

$$t_i = b\omega_i^2 \begin{pmatrix} -\sin a_{1_s i} \\ \cos a_{1_s i} \sin b_{1_s i} \\ -\cos b_{1_s i} \cos a_{1_s i} \end{pmatrix} \tag{3-7}$$

donde $a_{1_s i}$ y $b_{1_s i}$ corresponden a los primeros armónicos de los ángulos de inclinación longitudinal y lateral de cada rotor.

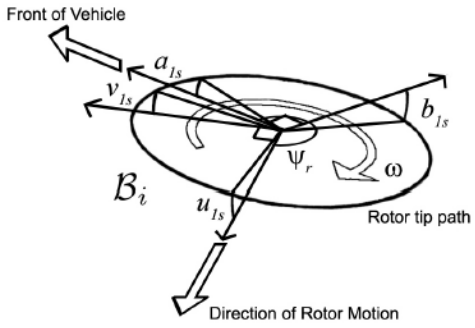


Figura 3-8. Representación de los ángulos de inclinación del plano del rotor debido al efecto del "rotor flapping"

Debido a la complejidad y los conocimientos en aerodinámica que se requieren para desarrollar los conceptos anteriores, no se profundiza más en el tema, permitiendo posibilidad de investigar más en el documento original (Pounds P. *et al.*, 2010).

- Momentos debidos a los empujes de los rotores:

  Las diferencias entre los empujes de cada rotor generan los pares que hacen que el vehículo gire. El par de torsión en el eje x del vehículo, par de alabeo, es la diferencia entre el momento producido por el empuje del rotor 4 y el empuje producido por el rotor 2, calculados como el producto vectorial del vector empuje y la distancia al centro de gravedad del quadrotor.

$$\tau_x = \tau_\theta = m_x = m_4 - m_2 = t_4 \times d - t_2 \times d \tag{3-8}$$

  Al ser vectores perpendiculares se puede reescribir como

$$m_x = dt_4 - dt_2$$

  Escribiéndolo ahora en términos de las velocidades angulares de los rotores

$$m_x = db(\omega_4^2 - \omega_2^2) \tag{3-9}$$

  y de manera similar para el eje y, el par de cabeceo es

$$\tau_y = \tau_\phi = m_y = db(\omega_1^2 - \omega_3^2) \tag{3-10}$$

- Pares de reacción debidos al movimiento de cada rotor:

  El par de reacción que aparece debido a la resistencia aerodinámica como consecuencia del giro del rotor se puede describir de forma semejante al empuje

$$q_i = k\omega_i^2 \tag{3-11}$$

  donde k depende de los mismos factores que b. Este par ejerce un par de reacción en el fuselaje que actúa para hacerlo girar en sentido opuesto a su dirección de rotación.

  El par de reacción total alrededor del eje z es

$$\tau_z = \tau_\psi = q_1 - q_2 + q_3 - q_4 = k(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \tag{3-12}$$

  donde los diferentes signos se deben a las diferentes direcciones de rotación de los rotores.

  Expresando en forma matricial las fuerzas y momentos desarrollados hasta ahora,

$$\begin{pmatrix} T \\ \Gamma \end{pmatrix} = \begin{pmatrix} T \\ \tau_\theta \\ \tau_\phi \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} -b & -b & -b & -b \\ 0 & -db & 0 & db \\ db & 0 & -db & 0 \\ k & -k & k & -k \end{pmatrix} \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} = A \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} \tag{3-13}$$

  donde la matriz A es de rango completo si b, k, d > 0 y por tanto puede ser invertida, se puede deducir la expresión que relaciona las velocidades de los rotores requeridas para aplicar un empuje especificado T y un momento Γ al fuselaje (Corke P., 2011).

$$\begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} = A^{-1} \begin{pmatrix} T \\ \tau_\theta \\ \tau_\phi \\ \tau_\psi \end{pmatrix} \tag{3-14}$$

Esta expresión es la que se usa en el bloque simulink "Control mixer" para

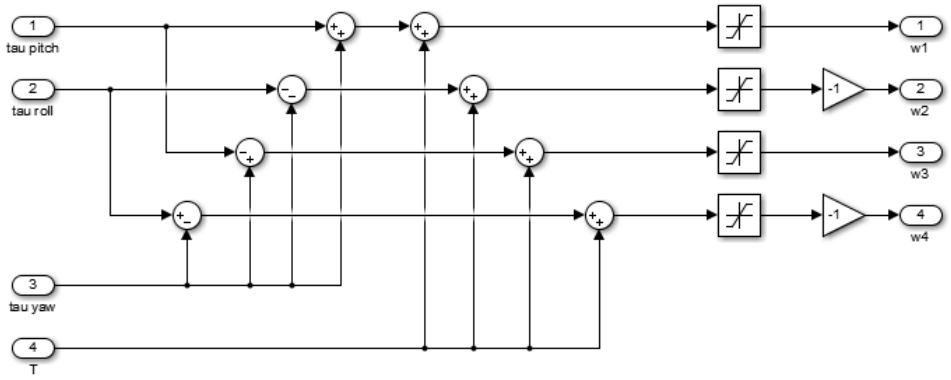


Figura 3-9. Diagrama simulink correspondiente al bloque "Control mixer" donde se realiza la operación matricial.

- Resistencia aerodinámica debida al empuje: $A_T = \frac{1}{2}\rho C_i W^2$ donde ρ es la densidad del aire, C_i son coeficientes aerodinámicos y W es la velocidad del quadrotor con respecto al aire.
- Par de resistencia aerodinámica debida a los pares y momentos: Se calcula de forma similar a A_T pero con otros coeficientes aerodinámicos, $A_R = \frac{1}{2}\rho C_i W^2$.

  Sin embargo tanto A_T como A_R se desprecian a la hora de desarrollar las ecuaciones.

Una vez calculadas las fuerzas, las ecuaciones de movimiento en coordenadas globales de un cuerpo rígido vienen determinadas por las leyes de Newton y Euler (Vianna G., 2007).

$$\begin{cases} \dot{\xi} = v = {}^{A}R_B V & (3\text{-}15) \\ m\dot{v} = {}^{A}R_B F_B \leftarrow 2^{a}\ \text{Ley de Newton, dinámica de traslación} & (3\text{-}16) \\ {}^{A}\dot{R}_B = {}^{A}R_B \Omega_x & (3\text{-}17) \\ J\dot{\Omega} = -\Omega \times J\Omega + \tau_B \leftarrow \text{Ecuación del movimiento de Euler, aceleración rotacional} & (3\text{-}18) \end{cases}$$

donde g es la aceleración gravitatoria, m es la masa total del vehículo, V es el vector velocidad traslacional (en {A}), ξ y ν representan respectivamente la posición y velocidad lineal expresadas en el sistema de referencia inercial {I}, Ω la velocidad angular de {A} respecto {I}, Ω× la matriz anti-simétrica de Ω de forma que se cumpla que $\Omega_x \times \nu = \Omega \times \nu$ para el vector resultante del producto vectorial y cualquier vector ν perteneciente a $\mathbb{R}_3$ y J es la matriz de inercia 3 × 3 del vehículo

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

donde $I_{xx} = I_{yy}$ debido a la simetría del aparato.

El movimiento del quadrotor se obtiene integrando estas ecuaciones dinámicas. Tanto el modelo como la integración de las ecuaciones está englobado en el bloque simulink "Quadrotor".

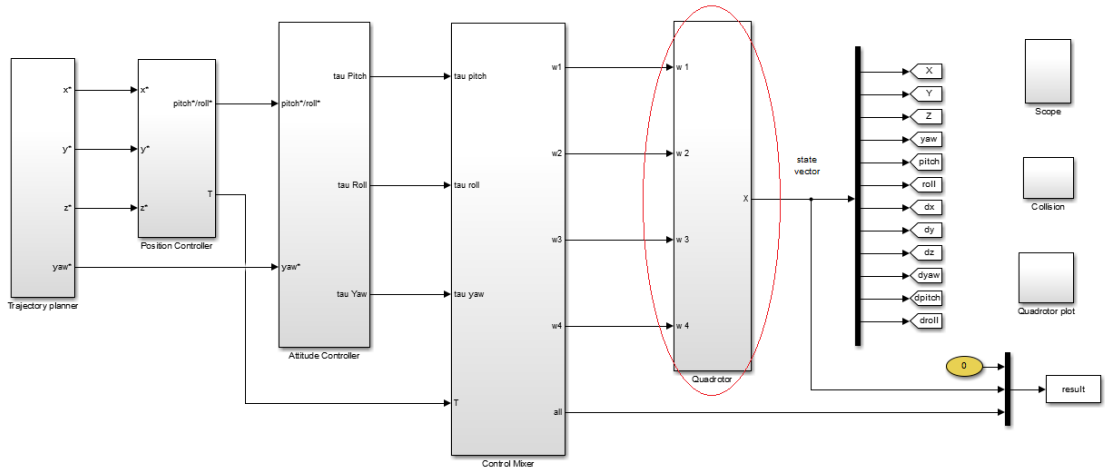


Figura 3-10. El bloque "Quadrotor" engloba las ecuaciones que modelan al quadrotor.

4 GENERACIÓN DE TRAYECTORIAS

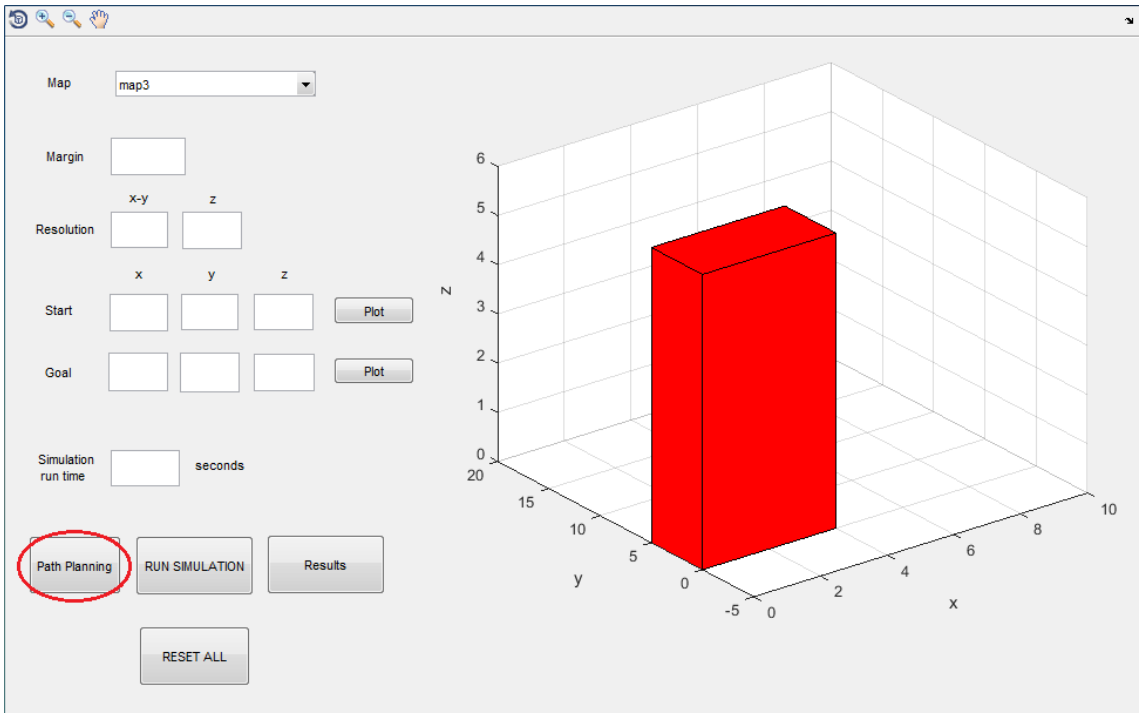


Figura 4-1. Botón de la interfaz de usuario correspondiente a la ejecución del cálculo de trayectoria.

La planificación de trayectorias (bloque del simulink: "Generador de trayectorias") constituye la primera aplicación del simulador, la cual, elegido un entorno con obstáculos y rellenados los argumentos iniciales (margen, resolución, puntos inicial y final, tiempo), consiste en la búsqueda de una secuencia de posiciones para un robot de forma que este se mueva desde el estado inicial al estado final, entendiendo por estado la descripción de la ubicación del robot referida a un marco de referencia absoluto.

La configuración que adquiere una determinada trayectoria depende de la distribución de los obstáculos a lo largo de todo el espacio de trabajo y, por supuesto, de la geometría del robot y sus capacidades de movimiento. De esta manera, la topología del ambiente de trabajo restringirá el espacio libre de obstáculos en el cual se pueden expresar las posibles trayectorias para alcanzar el estado final deseado.

Generalmente, se recurre a una representación realizada a partir de la discretización del espacio del ambiente de trabajo, con lo que se extrae una representación segura, es decir, se tendrá la garantía de que el espacio libre podrá ser ocupado por el robot (sin riesgo de colisión), por lo tanto, es necesario que tal discretización se haga en base a las características geométricas, tanto del robot como de los obstáculos.

Existen dos formas básicas para especificar el movimiento (Ollero A., 2005):

- Suministrando puntos consecutivos e ignorando la trayectoria espacial que describe el robot entre cada dos puntos.
- Especificando el camino que debe unir los puntos mediante una determinada trayectoria, tal como una línea recta o un círculo, que debe describir el robot en el espacio de trabajo.

La primera alternativa, denominada tradicionalmente control punto a punto, sólo tiene interés práctico cuando los puntos están suficientemente separados, ya que, en caso contrario, la especificación sería muy tediosa. Por otra parte, los puntos tampoco pueden estar muy separados pues entonces el riesgo de que se generen movimientos imprevisibles o no controlados, es grande.

En el control punto a punto, el sistema de control automático del robot debe realizar la interpolación entre los puntos especificados, de forma tal que, posteriormente sea posible realizar el control de movimientos para que el robot pase por dichos puntos.

La segunda estrategia se denomina control de trayectoria continua. En este caso, el sistema de control debe hacer que el robot reproduzca lo más fielmente posible la trayectoria especificada.

La importancia de la planificación de trayectorias radica en la búsqueda y obtención de estrategias de control para obtener del robot trayectorias adecuadas, seguras y que posean la mayor calidad en su desplazamiento.

Existen diferentes métodos o algoritmos de planificación basados en diversas técnicas de exploración de entornos: métodos basados en grafos, basados en diagramas de Voronoi, en descomposición en celdas, algoritmos RRT (del inglés – "Rapidly-exploring Random Tree"), etc., presentando unos, ciertas ventajas frente a otros.

En este documento se centrará en los algoritmos basados en grafos más comunes como son: Breadth First Search (búsqueda en anchura), Dijkstra o Greedy Best-First Search (búsqueda voraz) y A* (A asterisco) que se detallarán consecutivamente pues cada uno de ellos es herencia del anterior de modo que se van presentando mejoras en la optimización.

4.1 Simulación del entorno

La primera fase de la aplicación consiste en interpretar y representar el mapa elegido. Para ello se emplea un archivo de texto donde viene descrita la disposición de los obstáculos en el entorno.

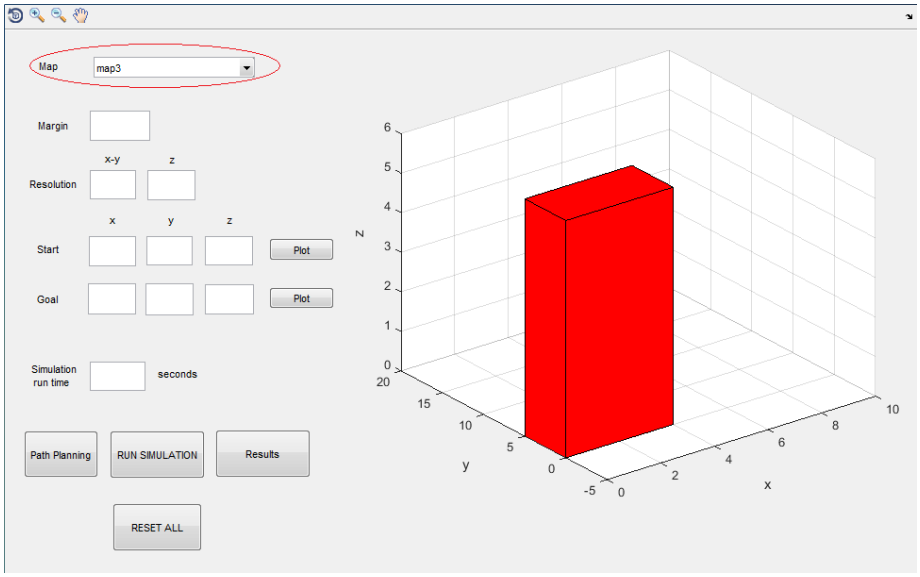


Figura 4-2. Botón de selección del entorno.

En este archivo de texto se pueden encontrar los siguientes tipos de sentencias:

- '# map1': Indica el nombre del mapa. No tiene utilidad.
- 'boundary xmin ymin zmin xmax ymax zmax': Sirve para declarar los bordes del mapa (los límites de los ejes de la figura).

  Ej: boundary 0.0 -5.0 0.0 10.0 20.0 6.0
- 'block xmin ymin zmin xmax ymax zmax rojo verde azul': Indica los límites de cada bloque en todos los ejes (x, y, z) para rellenarlo y representarlo. Los últimos tres números sirven para elegir el color, indicando las cantidades de cada tono en un rango de 0 a 255 (en este caso sería completamente rojo).

  Ej: block 0.0 2.0 0.0 10.0 2.5 1.5 255 0 0

A modo de ejemplo, para el siguiente entorno,

map6

boundary 0.0 0.0 0.0 5.0 5.0 5.0

block 0.0 1.0 0.0 2.0 2.0 5.0 0 0 255

block 3.0 3.0 0.0 5.0 3.5 5.0 0 0 255

se tendrían dos obstáculos en azul y los ejes de la figura serían x=[0 5], y=[0 5] y z=[0 5].

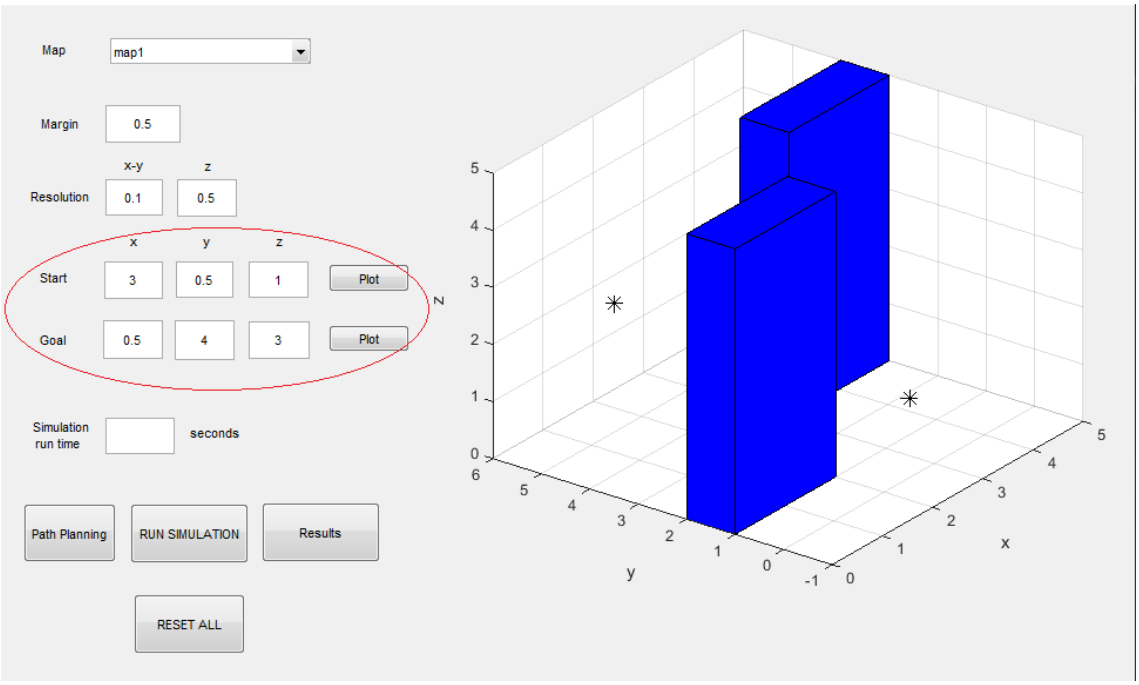


Figura 4-3. Ejemplo de representación de entorno, obstáculos y puntos final e inicial de la trayectoria.

Donde los asteriscos representan el punto inicial (3, 0.5, 1) y el punto final (0.5, 4, 3).

Cabe mencionar también que los únicos obstáculos que se pueden representar con el método aquí empleado son paralelepípedos. Sin embargo, la correcta combinación de estos permite crear entornos tales como:

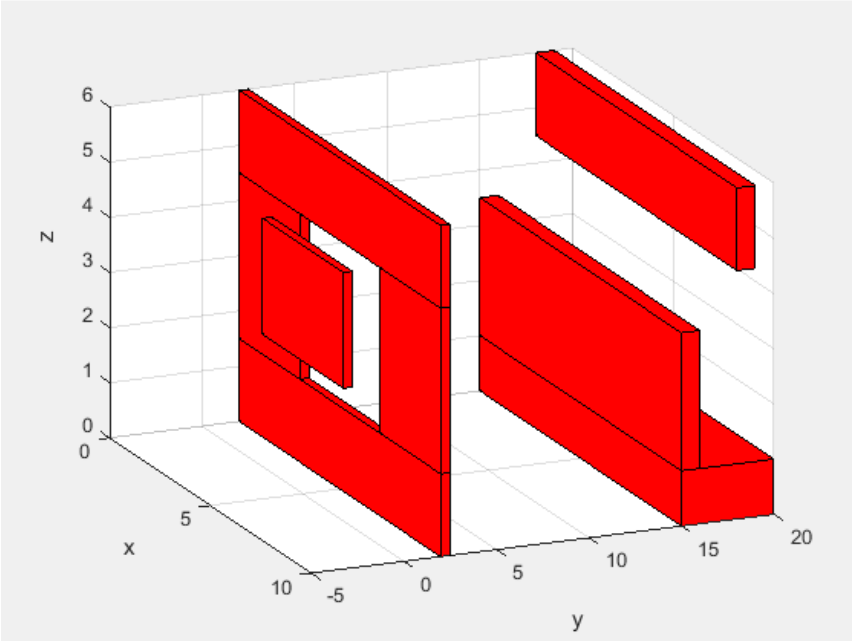


Figura 4-4. Otro ejemplo de entorno con obstáculos, más complejo.

4.2 Cálculo de trayectorias

Una vez que se representa el entorno se procede ahora a calcular la trayectoria más corta para lo que, como ya se comentó anteriormente, existen diferentes técnicas.

Sin embargo, antes de proceder con la explicación de los algoritmos es necesario realizar un paso previo, que consiste en crear una cuadrícula del mapa, es decir, se genera lo que se denominan los nodos sobre los que se realizará la exploración. La separación entre unos nodos y otros dependerá de la resolución, que vendrá definida por el usuario.

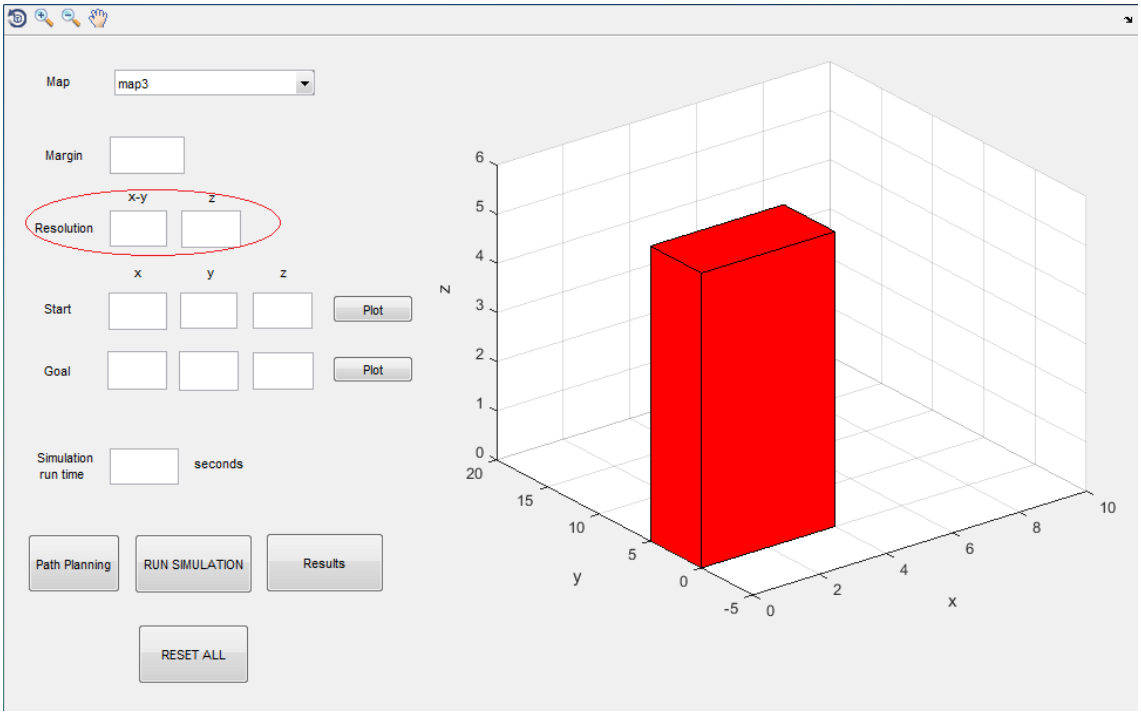


Figura 4-5. Selección de resolución de matriz de nodos en la interfaz de usuario.

Una vez generados los nodos es necesario comprobar que no pertenecen a zonas prohibidas (como es el caso de puntos en el interior de un obstáculo), los cuales, de ser así, deben ser eliminados del conjunto a explorar.

Se demuestra en la siguiente imagen, mediante puntos rojos, cómo quedarían los nodos generados empleando una resolución de 0.5 en el plano x-y y 0.5 en z.

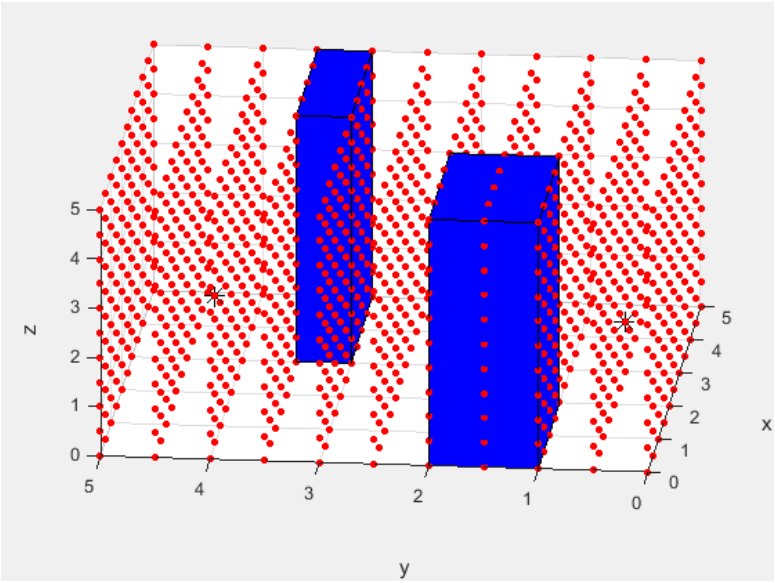


Figura 4-6. Matriz de nodos.

Por último, se utiliza un parámetro adicional a la resolución de la malla, parámetro 'margen' de la interfaz, que se emplea para eliminar los nodos que se encuentren a una cierta distancia de seguridad de los obstáculos. De modo que, si el planificador de trayectorias se utiliza para un vehículo u otro, se pueda variar fácilmente el valor de dicha variable en función del tamaño del robot que se utilice, evitando así posibles choques con los obstáculos a la hora de realizar las maniobras. Y, como resulta obvio, no generaría una trayectoria a través de un espacio por el cual el vehículo no pudiese acceder, aunque este fuese el más corto, teniendo que buscar una alternativa.

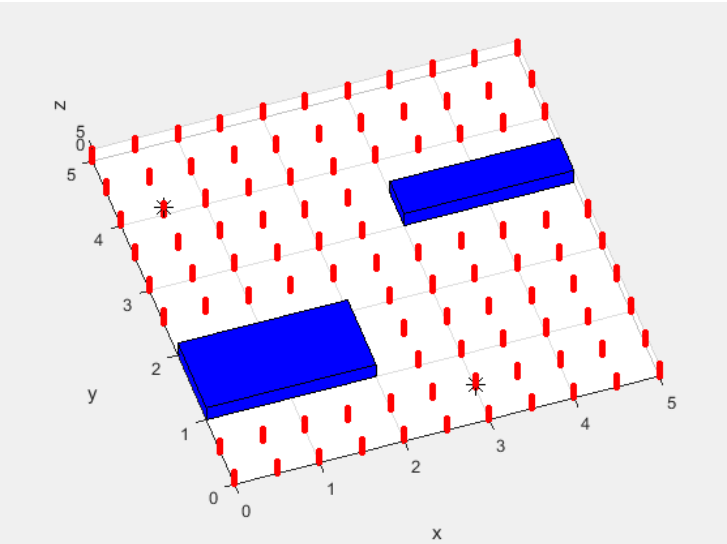


Figura 4-7. Matriz de nodos tras eliminar los que no respetan la distancia de seguridad.

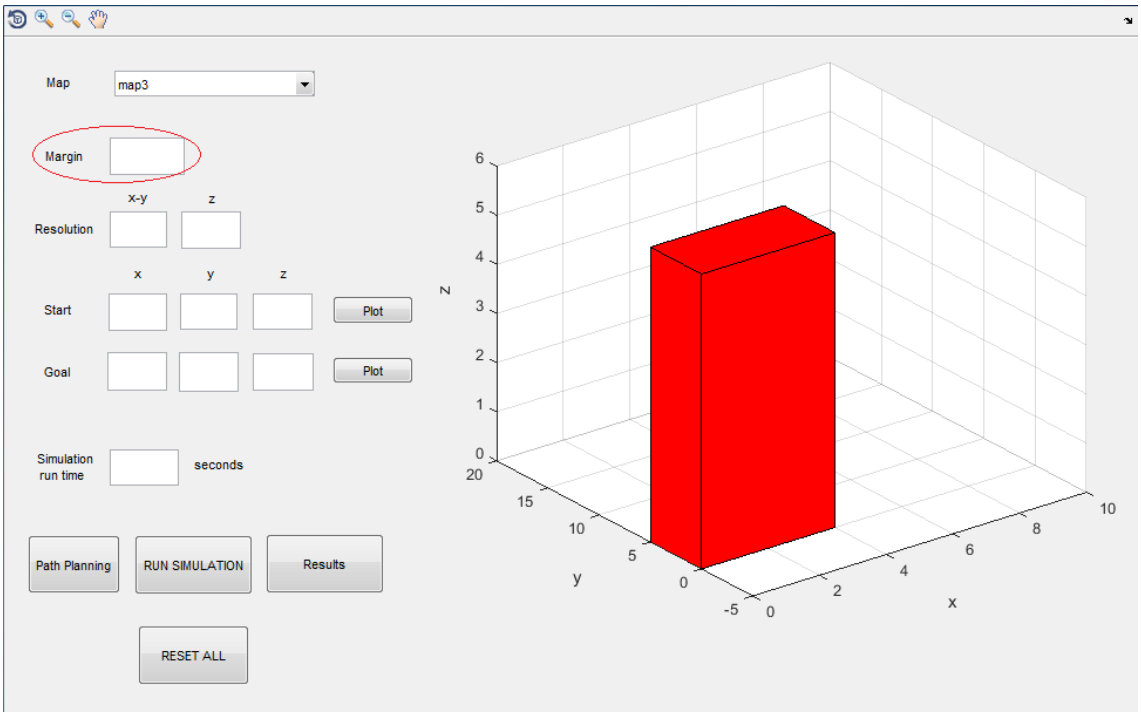


Figura 4-8. Selección de distancia de seguridad en la interfaz de usuario.

4.2.1 Breadth First Search (BFS)

Este primer algoritmo es el "padre" de todos ellos, así como también es el más simple de todos los métodos de búsqueda en grafos.

El fundamento básico de este consiste en recorrer los nodos, comenzando por el nodo inicial, explorando todos los vecinos adyacentes a este. A continuación, para cada uno de los vecinos, se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el árbol.

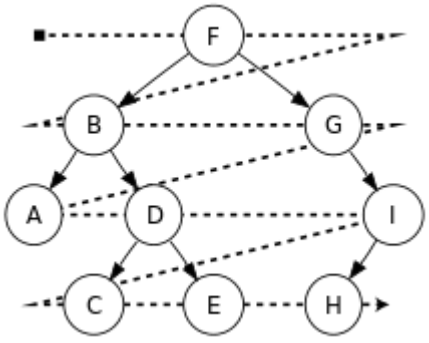


Figura 4-9. Evolución en grafo mediante algoritmo BFS.

Formalmente, BFS es un algoritmo de búsqueda sin información, que se expande y examina todos los nodos de un árbol sistemáticamente para buscar una solución, o lo que es lo mismo, con información, pero con una estrategia que no priorice ningún nodo frente a los demás. Este sería el caso si a cada nodo se le asignase, por ejemplo, el coste real del camino recorrido (distancia Manhattan) para llegar al nodo actual desde el nodo inicial, es decir, la distancia entre estos calculada como la suma de las diferencias (absolutas) de sus coordenadas.

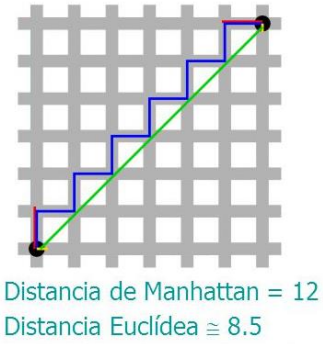


Figura 4-10. Medida de longitud mediante el método Manhattan y el método euclídeo.

De esta forma, se podría decir que el algoritmo no usa ningún tipo de estrategia heurística, es decir, no presenta ninguna estrategia que facilite la búsqueda de la solución, sino que simplemente se exploran todas las opciones.

El pseudocódigo más general que describe el funcionamiento de este algoritmo es:

```
Marcar (v)
Poner (v) en una COLA
Mientras la COLA no sea vacía hacer
        Quitar el primer elemento w de la COLA
        Para cada vértice x adyacente a w hacer
                Si x no está marcado entonces
                        Marcar x
                        Poner x en la COLA
```

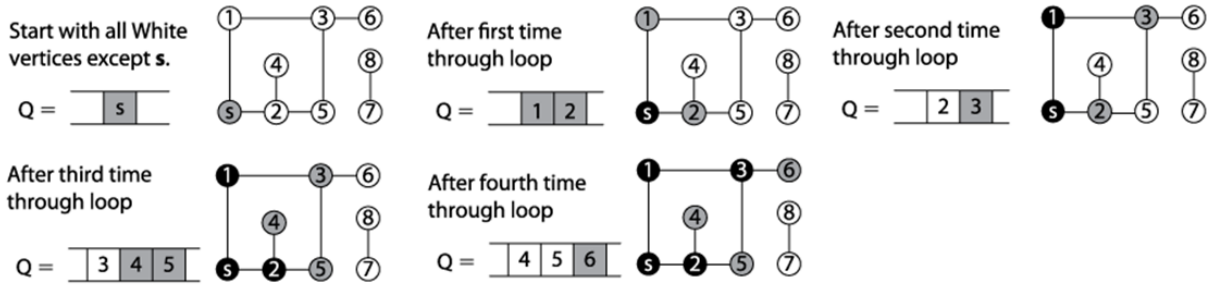


Figura 4-11. Exploración de los nodos mediante algoritmo BFS.

Esta manera de explorar los nodos se asemeja gráficamente a la forma en la que se extiende el agua sobre una superficie en todas direcciones, como una ola.

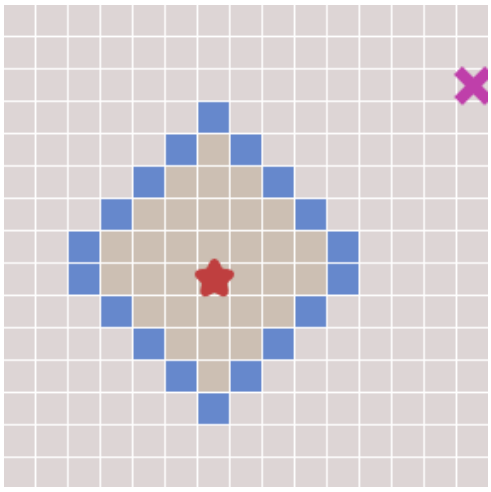
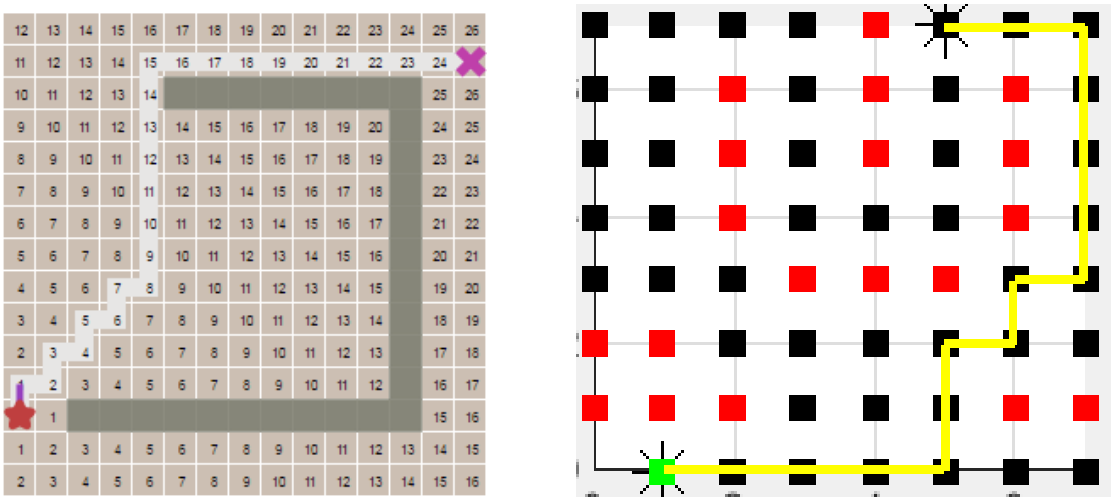


Figura 4-12. El avance en la exploración mediante
el método BFS se asemeja a una ola.

Las siguientes imágenes son un ejemplo del problema que presenta este método. Se puede observar cómo, aunque se haya descubierto el camino más corto, ha sido necesario explorar prácticamente todos los nodos del mapa, lo que supone un gran coste y tiempo de ejecución.



a) b)

Figura 4-13. a) Resultado final exploración con método BFS,
mapa 1. b) Resultado final exploración con método BFS, mapa 2.

4.2.2 Dijkstra

También llamado Greedy Best-First Search (búsqueda voraz), presenta un funcionamiento muy parecido al anterior con la gran diferencia de que, para resolver el problema, sí sigue una heurística consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima. Es decir, en vez de explorar todos los caminos posibles en todas direcciones, se favorecen los caminos de menor coste en base a una estrategia de optimización.

En nuestro caso, la heurística empleada es la distancia euclídea (módulo del vector) entre el nodo que se está explorando y el nodo final, de forma que cuando se va a elegir el siguiente nodo o vecino a explorar, se elige aquel cuya distancia sea menor.

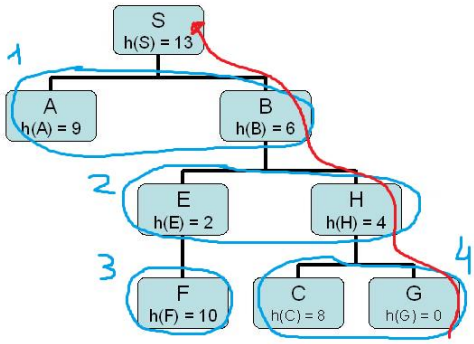
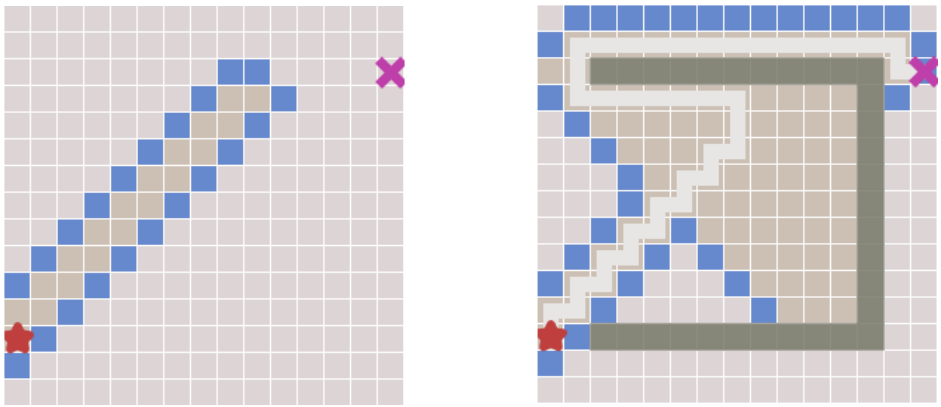


Figura 4-14. Evolución en grafo mediante algoritmo Dijkstra.

Esto supone una focalización de la exploración hacia el punto de destino

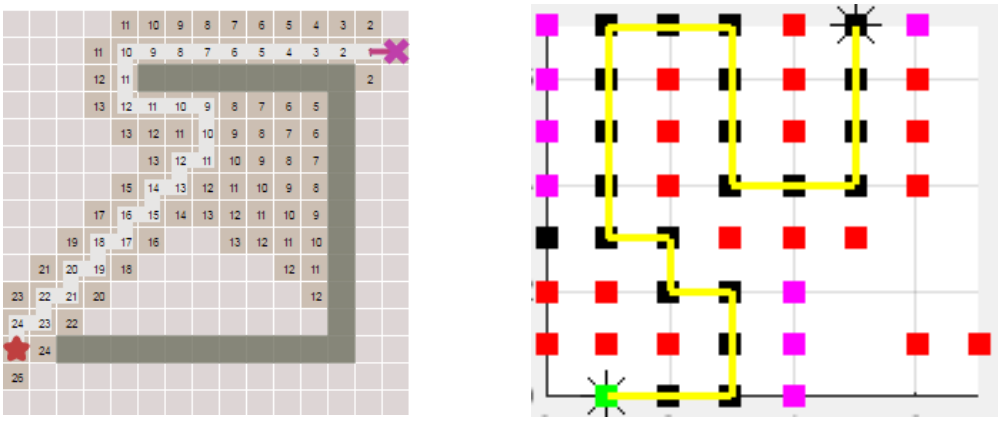


a) b)

Figura 4-15. a) Avance en la exploración mediante el método Dijkstra.

b) Resultado final exploración con método Dijkstra.

que puede provocar que, aunque la búsqueda del punto final sea mucho más rápida, se olvide el objetivo de obtener la trayectoria más corta, pues no se tiene en cuenta el coste real de desplazamiento al elegir esa trayectoria.



a) b)

Figura 4-16. a) Resultado final exploración con método Dijkstra,
mapa 1. b) Resultado final exploración con método Dijkstra, mapa 2.

A diferencia del algoritmo BFS, no es necesario explorar casi todos los nodos del mapa, sin embargo, este sí que obtenía la trayectoria más corta.

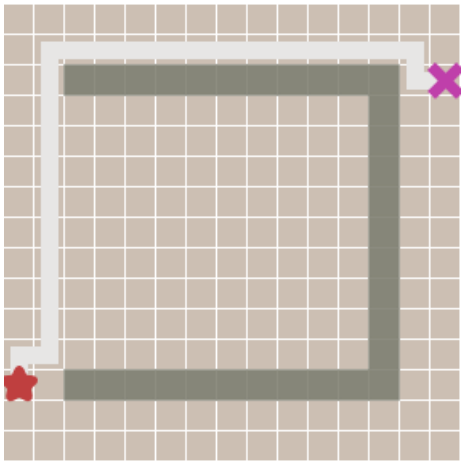


Figura 4-17. Trayectoria más corta obtenida con el método BFS.

4.2.3 A*

El problema de los algoritmos anteriores de búsqueda en grafos informados, es que se guían en exclusiva por la función heurística, como puede ser el algoritmo voraz (dijkstra), la cual puede no indicar la trayectoria más corta, o en exclusiva por el coste real de desplazarse de un nodo a otro (algoritmos de escalada, BFS), pudiéndose dar el caso de que sea necesario realizar un movimiento de coste mayor para alcanzar la solución. Es por ello que un buen algoritmo de búsqueda informada debería tener en cuenta ambos factores, el valor heurístico de los nodos y el coste real del recorrido.

Así, el algoritmo A* utiliza una función de evaluación f(n)=g(n)+h(n), donde h(n) representa el valor heurístico del nodo a evaluar desde el actual, n, hasta el final, y g(n), el coste real del camino recorrido para llegar a dicho nodo, n, desde el nodo inicial.

A* mantiene dos estructuras de datos auxiliares, que podemos denominar abiertos, implementado como una cola de prioridad (ordenada por el valor f(n) de cada nodo), y cerrados, donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté primero en abiertos, y en caso de que no sea un nodo objetivo, calcula la f(n) de todos sus vecinos, los inserta en abiertos, y pasa el nodo evaluado a cerrados (López C. 2014).

El algoritmo es una combinación entre búsquedas del tipo anchura con tipo profundidad: mientras que h(n) tiende primero a profundidad, g(n) tiende primero a anchura. De este modo, se cambia de camino de búsqueda cada vez que existen nodos más prometedores.

Como todo algoritmo de búsqueda en amplitud, A* es un algoritmo completo: en caso de existir una solución, siempre dará con ella.

Para garantizar la optimización del algoritmo, la función h(n) debe ser una heurística admisible, esto es, que no sobrestime el coste real de alcanzar el nodo objetivo.

Tabla 4–1. Esquematización algoritmo A*.

f(n) =	g(n)	+	h(n)
	coste real del recorrido	+	valor heurístico del nodo
	BFS	+	Dijkstra
	algoritmo en anchura	+	algoritmo en profundidad
	algoritmo de escalada	+	algoritmo voraz

Si para todo nodo n del grafo se cumple g(n)=0, nos encontramos ante una búsqueda voraz. Si para todo nodo n del grafo se cumple h(n)=0, A* pasa a ser una búsqueda no informada.

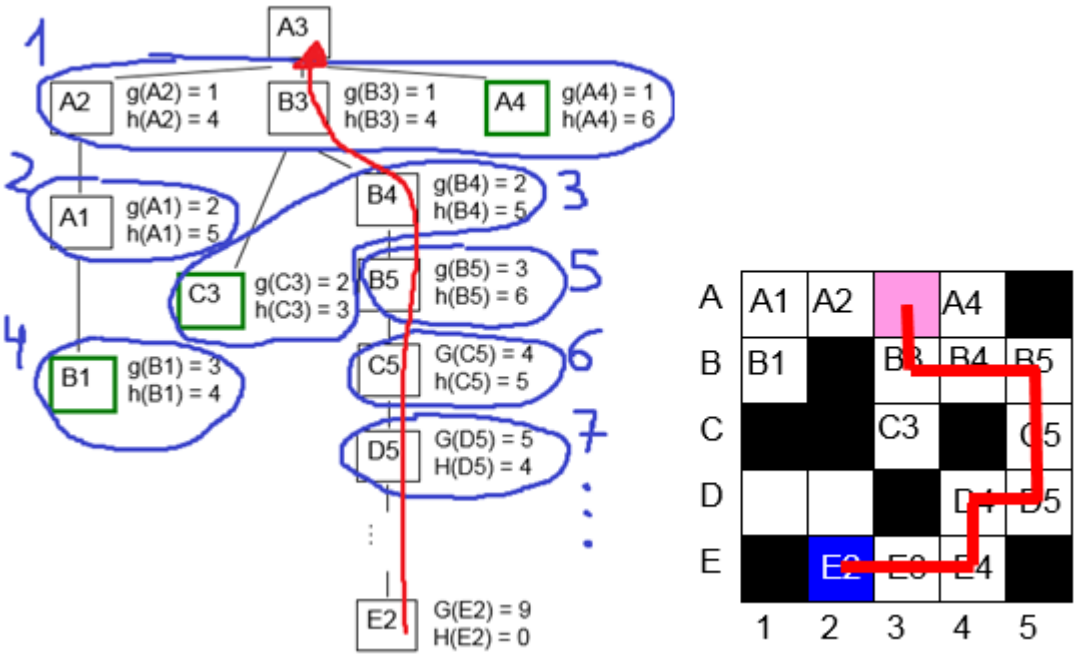


Figura 4-18. Evolución en grafo mediante algoritmo A*.

El siguiente pseudocódigo representa el código empleado para desarrollar este algoritmo:

Crear cuadrícula (malla) de nodos y comprobar si los nodos no se encuentran en un obstáculo y respetan distancia de seguridad

Inicializar estructuras de datos auxiliares necesarias para ejecución del algoritmo

Inicializar Valor de evaluación del NodoInicial = 0 (0 + 0)

Mientras haya nodos a evaluar

- Obtener índice NodoActual (Nodo de la lista con Valor de evaluación menor)

Si NodoActual está explorado, Ir a Mientras
Si NodoActual es el NodoFinal, Ir a Fin Mientras

Actualizar NodoActual a nodo explorado dejando de estar como nodo a explorar

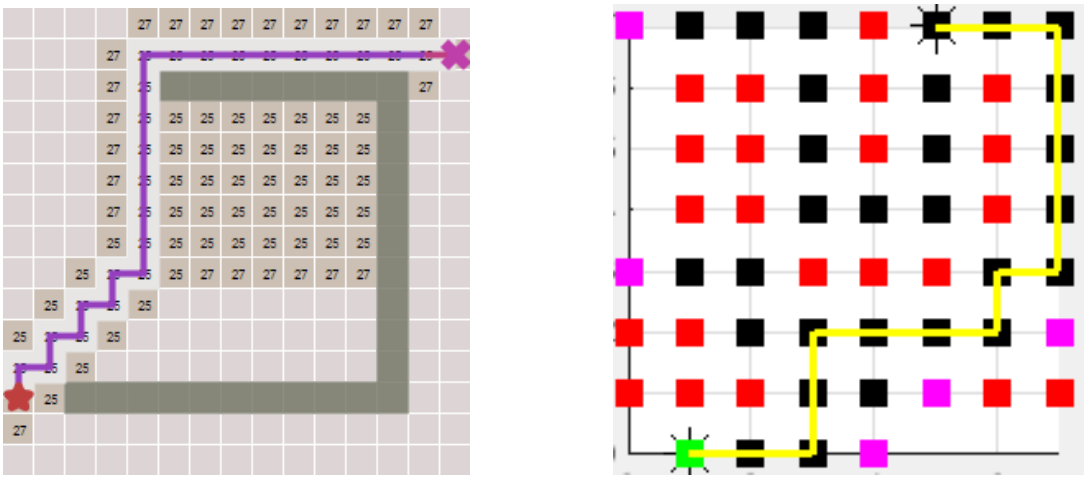
Obtener Vecinos del NodoActual y eliminar los que no sirvan
Si no hay Vecinos posibles, Ir a Mientras
Para cada Vecinos(i)
    Si Vecinos(i) ya está explorado, ir a Para
    Si Vecinos(i) no es nodo a explorar
        Actualizar valor de evaluación del NodoActual = Coste real de desplazamiento + Heurística
        Actualizar vector de nodos anteriores para reconstruir trayectoria
        Actualizar Vecinos(i) a nodo a explorar

Eliminar NodoActual de la lista de nodos (Valor de evaluación = NaN)

Donde se puede comprobar que, en cada iteración, se elige como NodoActual a aquel nodo cuyo valor de evaluación, suma entre el coste del camino recorrido (módulo del vector entre el NodoInicial y el NodoActual) y la heurística del nodo (módulo del vector entre el NodoActual y el NodoFinal), sea menor.

Este mismo código se podría transformar fácilmente en uno válido para el algoritmo BFS, simplemente obviando el término de heurística de forma que, como ya se dijo anteriormente, el valor de evaluación va adquiriendo valores que no aportan ninguna información adicional que permita la optimización. Así como también se podría transformar en uno válido para Dijkstra, eliminando el término del coste real del desplazamiento.

Como se puede observar en las siguientes soluciones, empleando este último algoritmo no ha sido necesario explorar todos los nodos y así como también se ha encontrado la trayectoria más corta.



a) b)

Figura 4-19. a) Resultado final exploración con método A*, mapa 1. b) Resultado final exploración con método A*, mapa 2.

A continuación, una vez explorados los nodos necesarios hasta que se ha alcanzado el nodo final, solo queda reconstruir la trayectoria, la cual se obtiene deshaciendo los pasos seguidos, de modo que esta resulta ser la más corta. Para ello se apoya en uno de los vectores de datos auxiliares en el que, en cada elemento (cada elemento representa un nodo), se ha ido almacenando el nodo anterior desde el que se alcanzó el actual, de forma que saltando de un nodo a otro se llega de nuevo el nodo inicial.

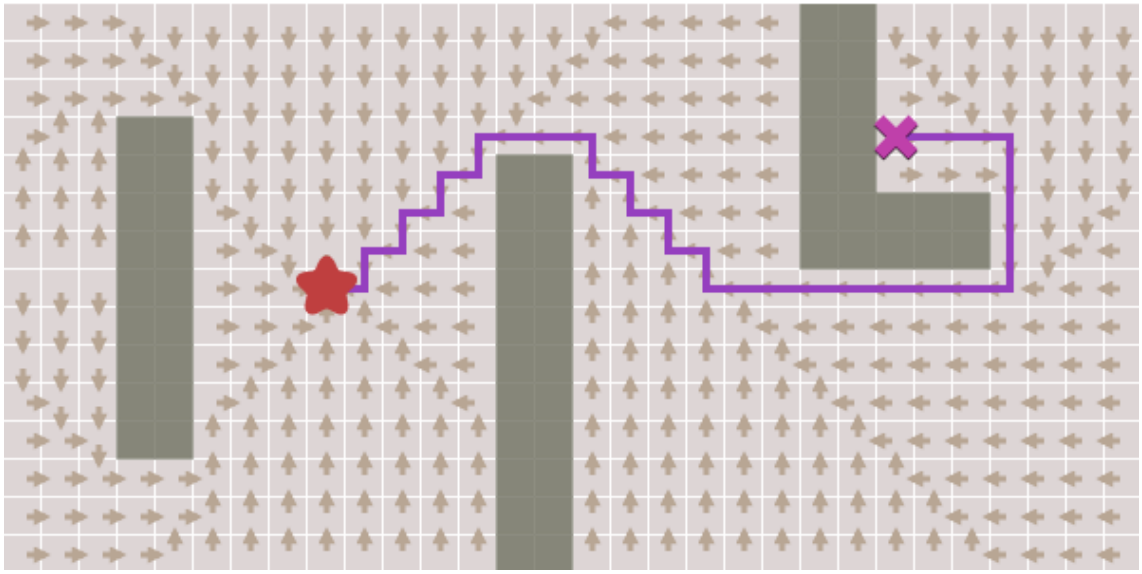


Figura 4-20. Reconstrucción de la trayectoria más corta.

4.3 Suavizado

El último paso e igual de importante es el suavizado de la trayectoria. Tras obtener la solución dada por el algoritmo se puede comprobar que dicha trayectoria presenta escalones, más cuanto mayor sea la resolución que se haya empleado en la matriz de nodos.

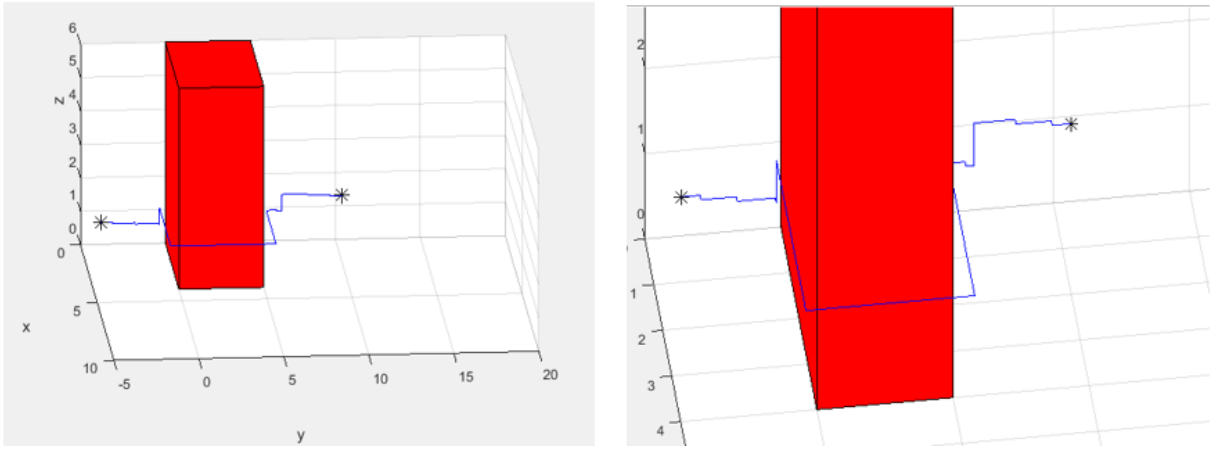


Figura 4-21. Ejemplo de trayectoria final sin suavizar.

Como es normal, estos escalones no interesan, pues sería muy poco eficiente que un vehículo siguiese esta trayectoria, por lo que es necesario un suavizado de esta.

La trayectoria obtenida estará constituida por una cantidad de puntos, cuya magnitud dependerá, entre otras cosas como las dimensiones del entorno, etc., de la resolución. El método de suavizado que aquí se desarrolla se realiza en varias etapas.

La primera fase tiene como objetivo reducir el número de puntos que constituyen el vector que conforma la trayectoria, eliminando los puntos intermedios de los tramos rectos, es decir, solo permanecen el punto inicial y final de cada escalón, excluyendo los puntos intermedios entre estos dos. Se realiza esta primera fase para simplificar significativamente el proceso de cálculo de la segunda. Una vez explicado esto se comenta ahora la técnica que se emplea mediante su pseudocódigo.

```
Errorx_anterior=0
Errory_anterior=0
Errorz_anterior=0

Para cada Punto de la trayectoria inicial
    Errorx=0
    Errory=0
    Errorz=0

    Si la coordenada x del Punto(i+1) sufre un incremento respecto de la del Punto(i)
        Errorx=1

    Si la coordenada y del Punto(i+1) sufre un incremento respecto de la del Punto(i)
        Errory=1

    Si la coordenada z del Punto(i+1) sufre un incremento respecto de la del Punto(i)
        Errorz=1

    Si (Errorx ha variado respecto del Errorx_anterior) ó (Errory ha variado respecto del
    Errory_anterior) ó (Errorz ha variado respecto del errorz_anterior)
        Se guarda el punto(i)

    Errorx_anterior= Errorx
    Errory_anterior= Errory
    Errorz_anterior= Errory
```

En la siguiente imagen se pueden observar los puntos descartados (azules) y los puntos seleccionados (verdes) de la trayectoria ejemplo.

Figura 4-22. Representación de los escalones presentes en una trayectoria sin suavizar.

A continuación, se realiza una segunda fase que consiste en reducir la nueva trayectoria al menor número de nodos posibles que puedan ser unidos por una recta, sin que dicha recta toque a ningún obstáculo. Es decir, se empieza por el primer nodo y el último, y se comprueba si estos pueden ser unidos sin que se atraviese a un obstáculo. Si es así, ya se tiene la trayectoria final, una recta formada por únicamente dos puntos, el inicial y el final, si no, se prueba a unir de nuevo el nodo inicial con el penúltimo y así sucesivamente hasta que se pueda unir el punto inicial con alguno de los otros puntos del vector obtenido en la fase anterior. Una vez unido el primer nodo, se sigue con el que se ha unido, repitiendo el mismo proceso comenzando con el nodo final, hasta que finalmente alguno se pueda unir con el último.

De esta manera queda por detallar cómo se asegura que la recta que une dos nodos no toca ningún obstáculo. Para ello lo que se hace es crear una malla de puntos (en las 3 direcciones) entre los dos nodos, y se comprueba que ninguno de ellos pertenece al obstáculo.

La siguiente batería de imágenes permite observar el proceso que sigue el algoritmo con mayor claridad:

1. Se prueba a unir los nodos inicial y final, comprobando que ningún punto de la malla generada toca el obstáculo.

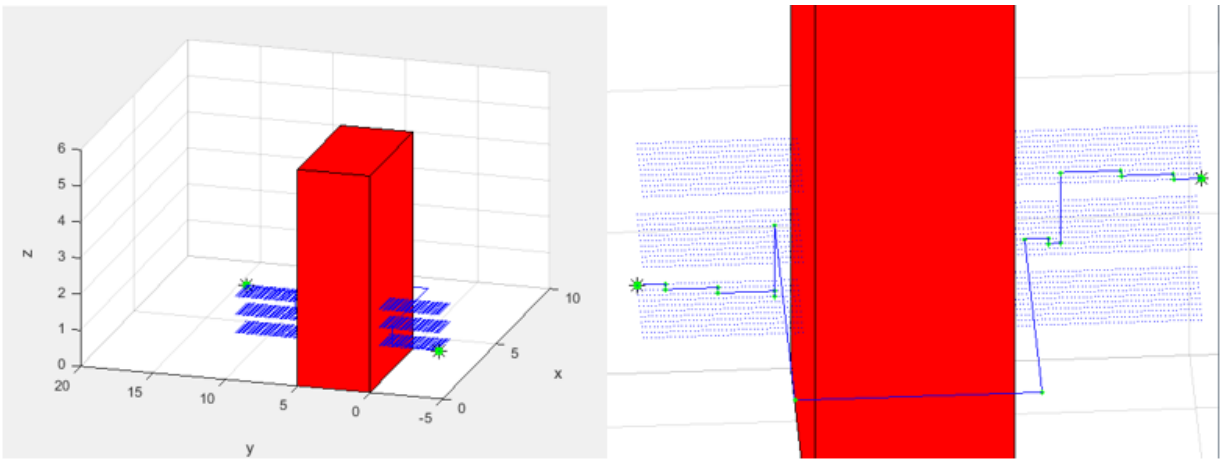


Figura 4-23. Creación de malla y primer intento de unión.

2. Como parte de estos puntos se encuentran en el obstáculo, estos nodos no se pueden unir y se prueba

con el penúltimo punto de la trayectoria obtenida en la fase 1.

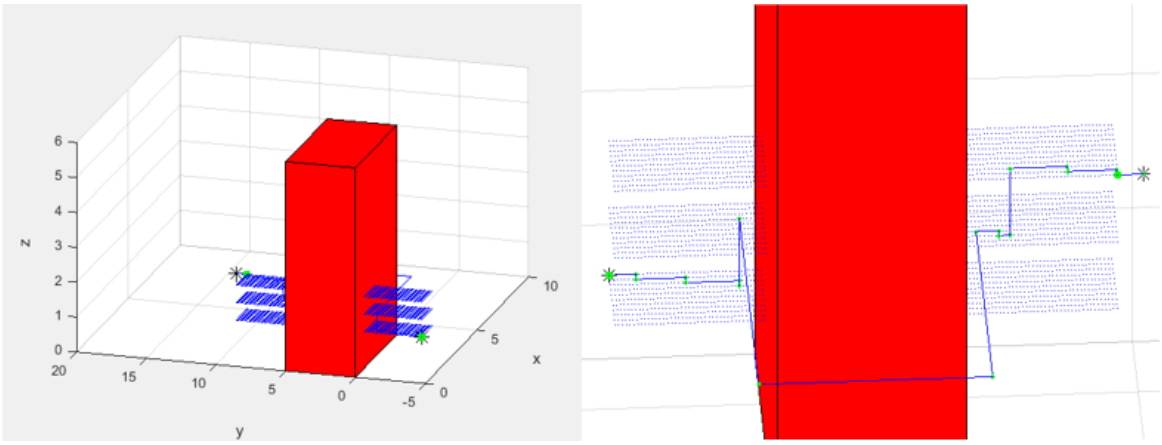


Figura 4-24. Nueva creación de malla y segundo intento de unión.

3. Se sigue probando con los siguientes nodos.

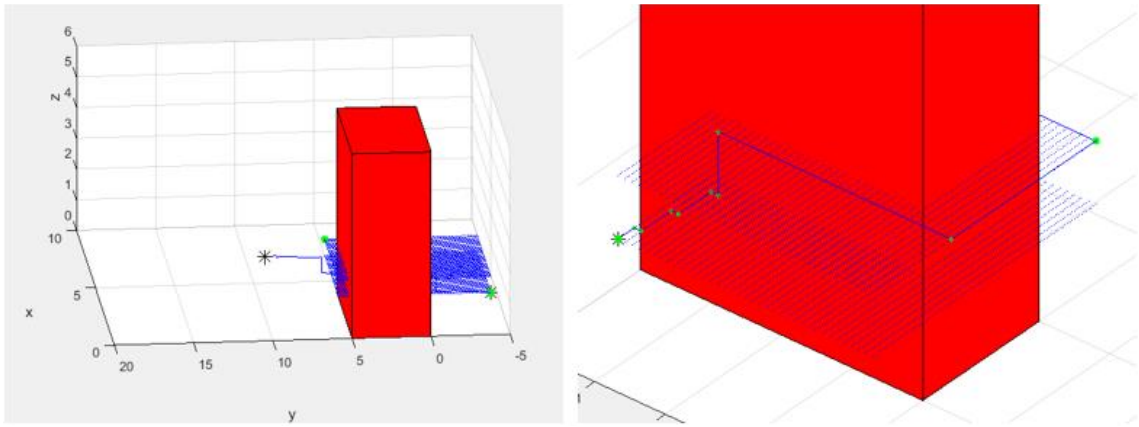


Figura 4-25. Nueva creación de malla e intento de unión.

4. Hasta que se alcanza uno con el que sí se puede unir sin que ningún punto de la malla esté en una zona no permitida.

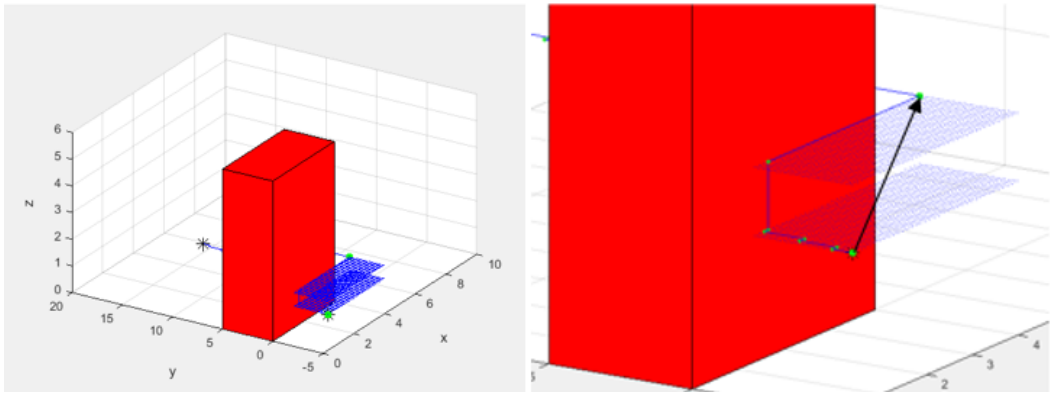


Figura 4-26. Representación del vector de unión entre dos nodos.

5. Se vuelve a repetir el proceso, pero ahora con el nodo recién unido.

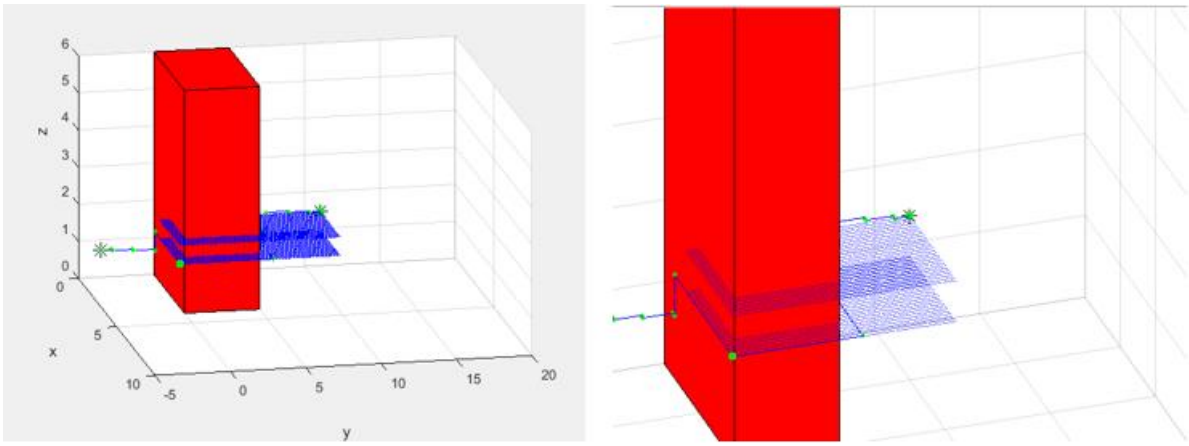


Figura 4-27. Repetición del proceso (creación de malla e intento de unión) desde nuevo nodo.

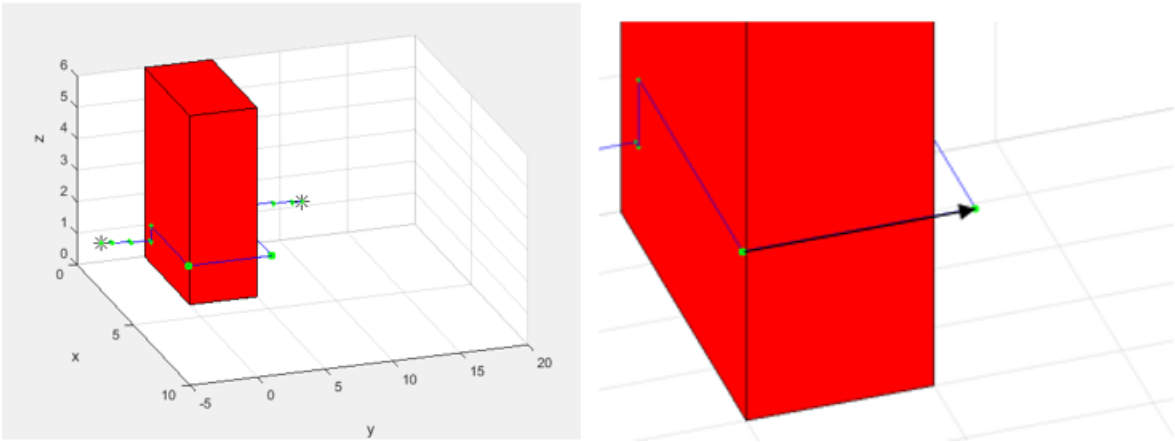


Figura 4-28. Representación del vector de unión desde el nuevo nodo con otro.

6. Hasta que se consigue unir alguno con el final.

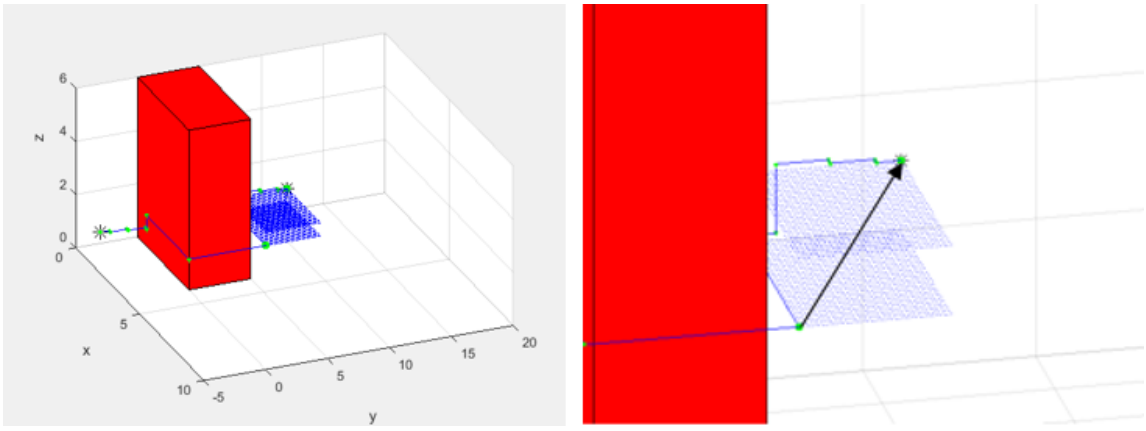


Figura 4-29. Representación del vector de unión con el nodo final.

Resultando ser la trayectoria final la línea en color magenta, la cual queda definida por el mínimo número de puntos que pueden ser unidos por líneas rectas, en este caso por 4 puntos.

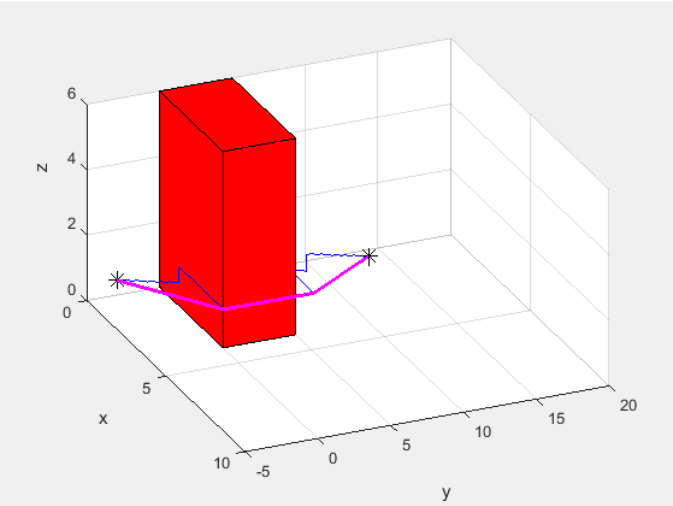


Figura 4-30. Trayectoria final tras el segundo paso del suavizado.

Otros ejemplos de suavizado de trayectorias pueden ser:

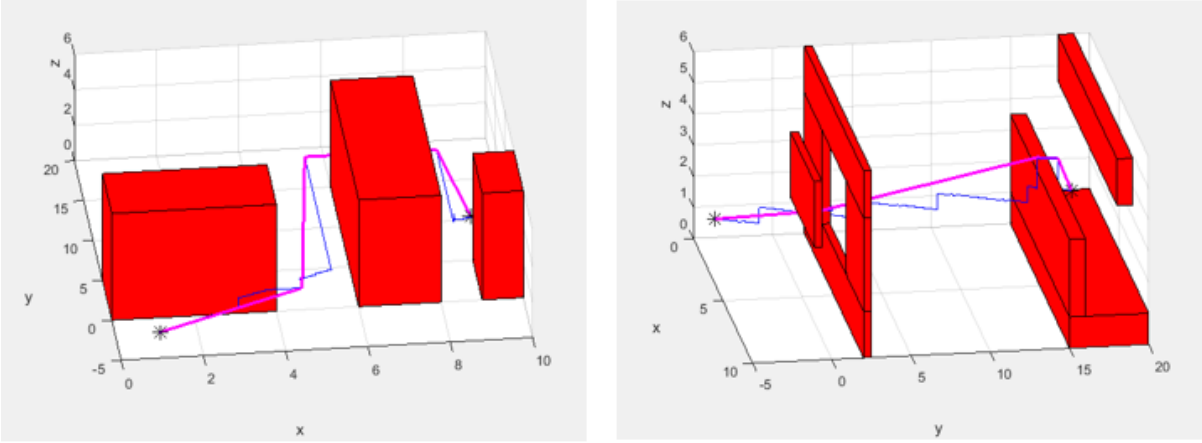


Figura 4-31. Ejemplo de suavizado de trayectorias en otros entornos con obstáculos.

Una vez aquí, lo que se tiene es el menor número de puntos necesarios por los que tiene que pasar el robot para llegar desde el inicial al final. Sin embargo, la referencia de entrada a los controladores no puede estar tan simplificada y distanciada, pues no se aseguraría que el vehículo no interceptase algún obstáculo, sino que debe estar constituida por un vector de puntos que una y recoja todos los mínimos necesarios, de forma que se asegure que esto no ocurre.

La primera técnica que se emplea, consiste simplemente en unir cada par de puntos mínimos necesarios mediante tramos rectos, de forma que estos están constituidos por puntos separados un pequeño intervalo. Para ello, se utiliza la función "jtraj (punto 1, punto 2, vector de tiempos)" del "Robotic Toolbox de Matlab" cuyos parámetros de entrada son las coordenadas del punto 1, las coordenadas del punto 2 y el tiempo de duración para el segmento (las duraciones de tiempo de cada tramo de la trayectoria se elegien ponderando en función de su longitud), subdividido con el intervalo elegido (en el caso de este proyecto el intervalo de subdivisión de tiempo es de 0.05s correspondiente al paso de simulación), que devuelve el vector de puntos del tramo recto de unión, ya subdividido en función de dicho vector de tiempos.

El problema de esta técnica es que no se está trazando una trayectoria realmente suave pues solamente está formada por tramos rectos, no hay curvas, y además no hay continuidad sobre las derivadas, lo que provoca que aparezcan problemas de control en las zonas de cambio de dirección, como se puede observar.

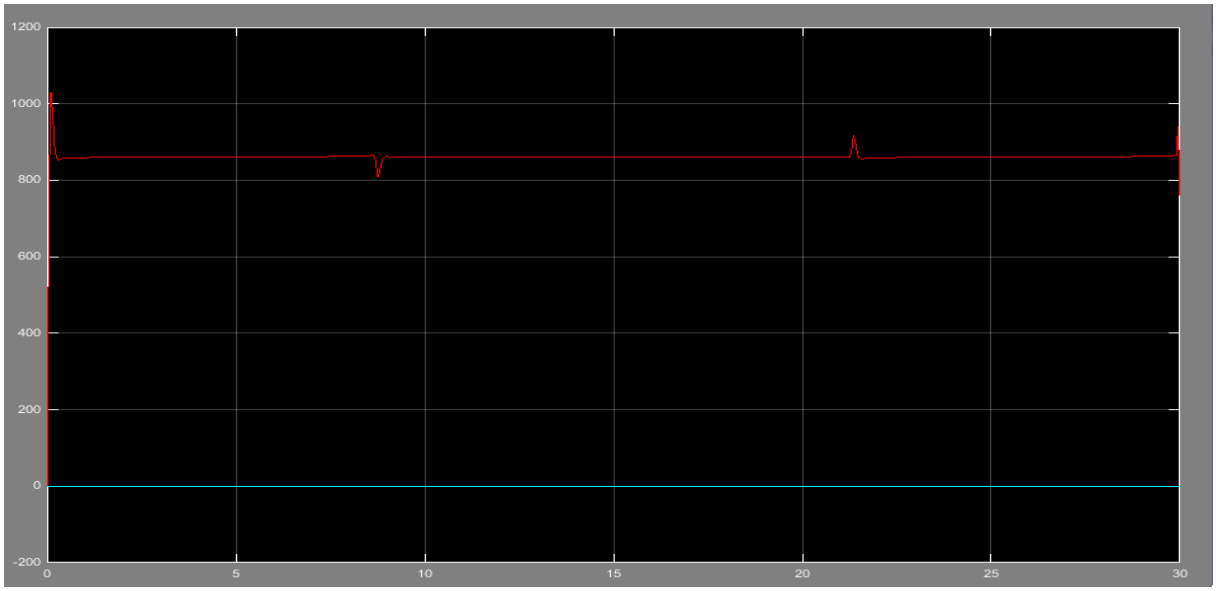


Figura 4-32. Se representa la fuerza de empuje (rojo) y los momentos resultantes del control en una trayectoria formada por 4 puntos (3 tramos rectos) para demostrar el problema que surge debido a la discontinuidad en las derivadas.

Que las trayectorias sean suaves, en general, implica restricciones sobre las derivadas. Normalmente, se exige que al menos la primera derivada sea continua, pudiendo exigirse también la continuidad de derivadas de orden superior.

Para asegurar esta continuidad se emplea la función "mstraj (vector de puntos mínimos necesarios, velocidad máxima para cada tramo, vector de tiempos para cada tramo, punto inicial, intervalo, duración del tiempo de aceleración en las zonas de unión de tramos)" del "Robotic Toolbox de Matlab" cuyos parámetros de entrada son: las coordenadas de los puntos mínimos necesarios menos el inicial; la velocidad máxima permitida en cada tramo de unión (en el caso de este proyecto se ha dejado vacío, por lo que no se indica una velocidad máxima); duración en segundos para cada tramo (igual que en la función anterior); las coordenadas del punto inicial de la trayectoria; el valor del intervalo para subdividir la duración de cada tramo (al igual que en anterior, 0.05s); y, por último (t_{acc}), la duración en segundos del tramo de aceleración en las zonas de unión de segmentos (que se le ha dado el valor de 1s), que devuelve el vector de puntos de la trayectoria completa, ya subdivida en función de las duraciones y el intervalo, formada por tramos rectos y curvas polinómicas (polinomios quínticos).

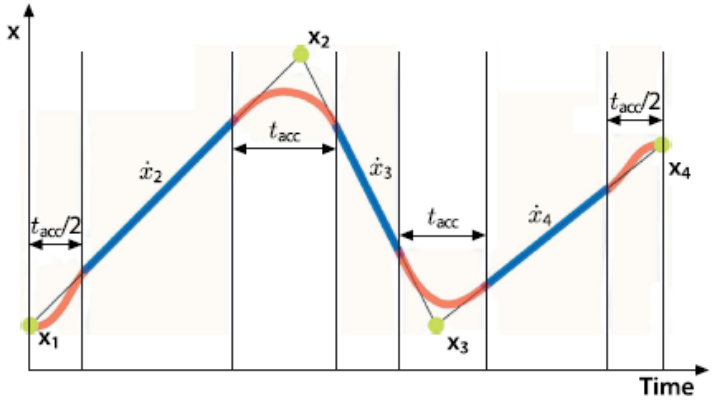


Figura 4-33. Notación para una trayectoria multi-segmentos. Los segmentos de color azul indican tramos de velocidad constante mientras que los rojos indican tramos de aceleración.

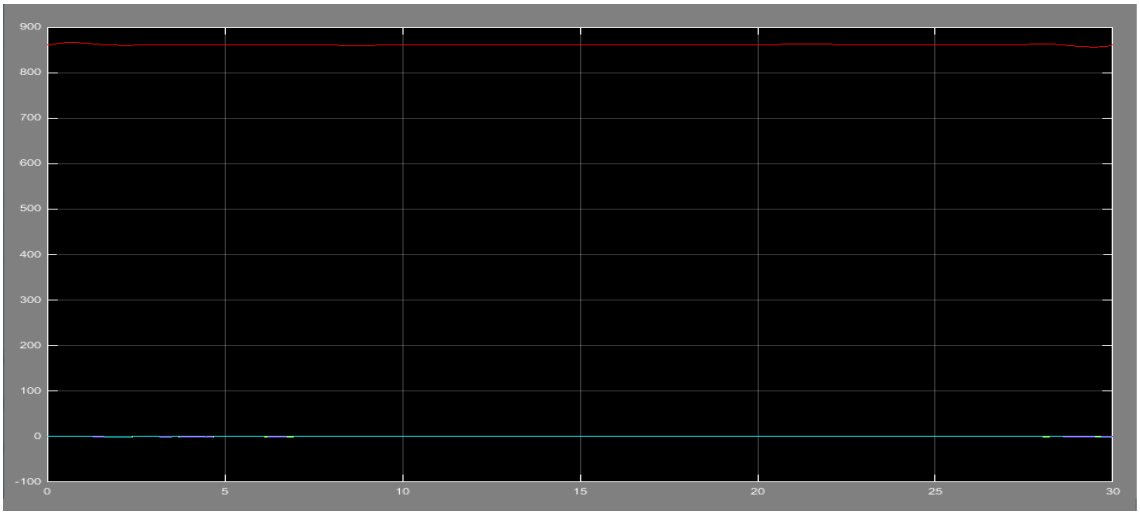


Figura 4-34. Se representa la fuerza de empuje (rojo) y los momentos resultantes del control en una trayectoria formada por 4 puntos (3 tramos rectos) para demostrar la mejora como consecuencia de la continuidad en derivadas.

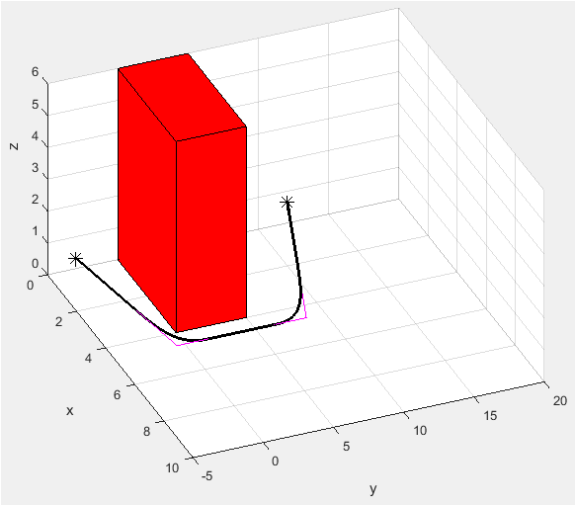


Figura 4-35. Representación de la trayectoria tras suavizado e interpolación.

4.4 Referencia

Para terminar con el generador de trayectorias, queda detallar sobre la forma de dar la referencia de entrada a los controladores. Se van a hablar en este apartado de dos formas diferentes de dar dicha referencia.

La primera forma consistiría en que, teniendo por un lado el vector de la trayectoria de $\text{longitud} = \frac{\text{tiempo de simulación elegido}}{\text{intervalo (0.05s)}}$, y por otro, el simulador, con fin de simulación igual al tiempo de simulación elegido e intervalo también 0.05s, se va dando como referencia, en cada paso de simulación, el siguiente punto de la trayectoria que corresponda, así hasta que se agote el tiempo de simulación fijado, de forma que se ha recorrido todo el vector. En este caso las referencias toman una forma bastante continua.

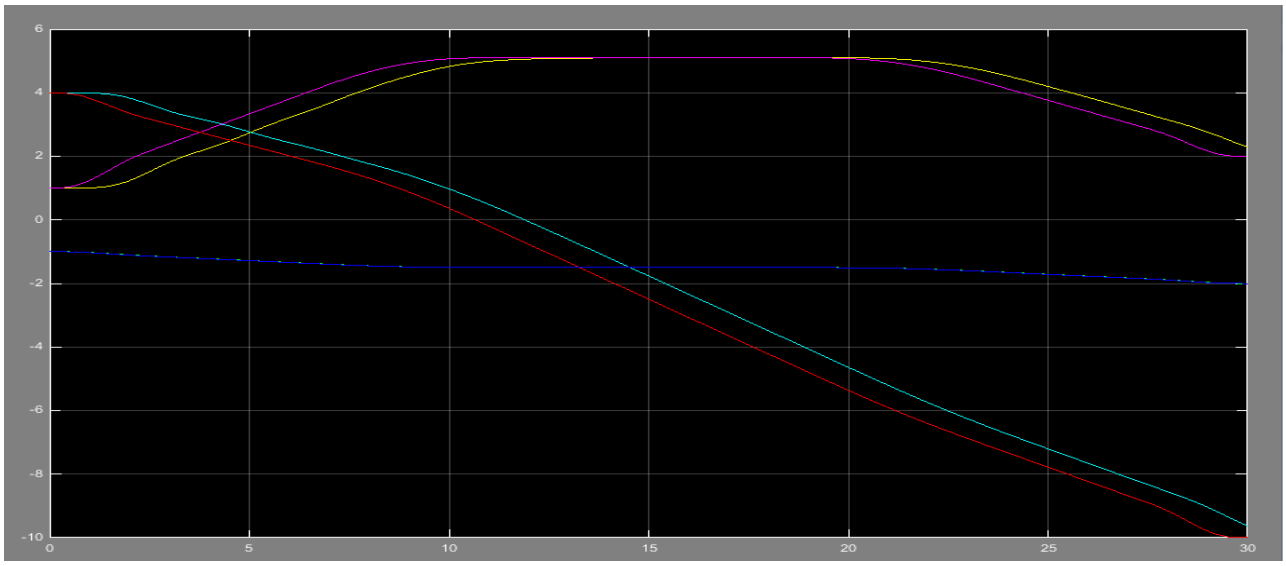


Figura 4-36. Representación de las variables de posición de referencia (x*- magenta, y*- rojo, z*- azul) y las realmente seguidas (x - amarillo, y - cian, z - verde), para una trayectoria con 4 puntos mínimos necesarios.

Para esta técnica, es necesario, como ya se ha dicho, especificar en la interfaz de usuario el tiempo límite de simulación. Cuanto mayor sea el tiempo especificado, mejor será el control y mayor la precisión de seguimiento de la trayectoria, pues los controladores, como se verá más adelante, tienen un tiempo de respuesta.

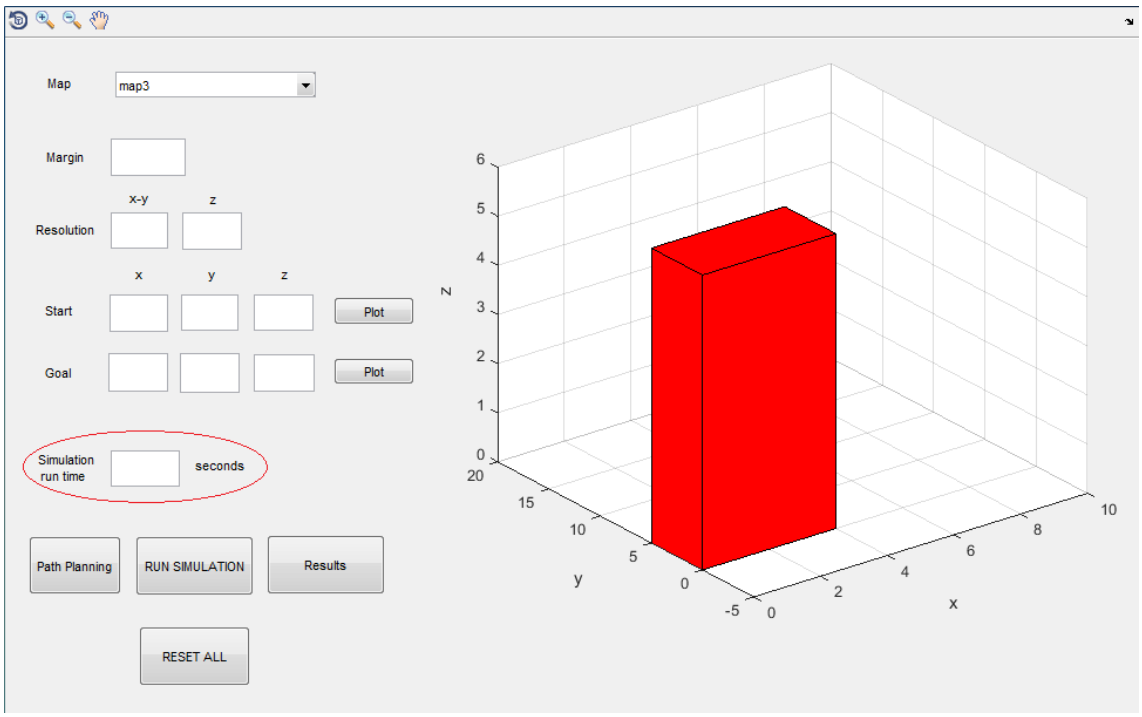


Figura 4-37. Selección del tiempo de simulación en la interfaz de usuario.

La segunda forma de dar la referencia consiste en elegir, de cada tramo de la trayectoria devuelta por la función "mstraj", algunos puntos, que sean equidistantes, descartando los demás, y, posteriormente, dar como referencia cada uno de ellos, cambiando al siguiente cuando se observe una aproximación del quadrotor hasta dicho punto, dentro de un área elegida, por ejemplo, 0.3 m. En este caso no haría falta especificación del tiempo, sino que la simulación (que tiene como límite de tiempo = infinito) acabaría cuando la aproximación al punto final fuese inferior a 0.3 m, después de haber pasado por todos los demás anteriormente. En este caso, la referencia tendría

una forma más escalonada pues la trayectoria no está formada por tantos puntos, como se puede ver en la siguiente figura.

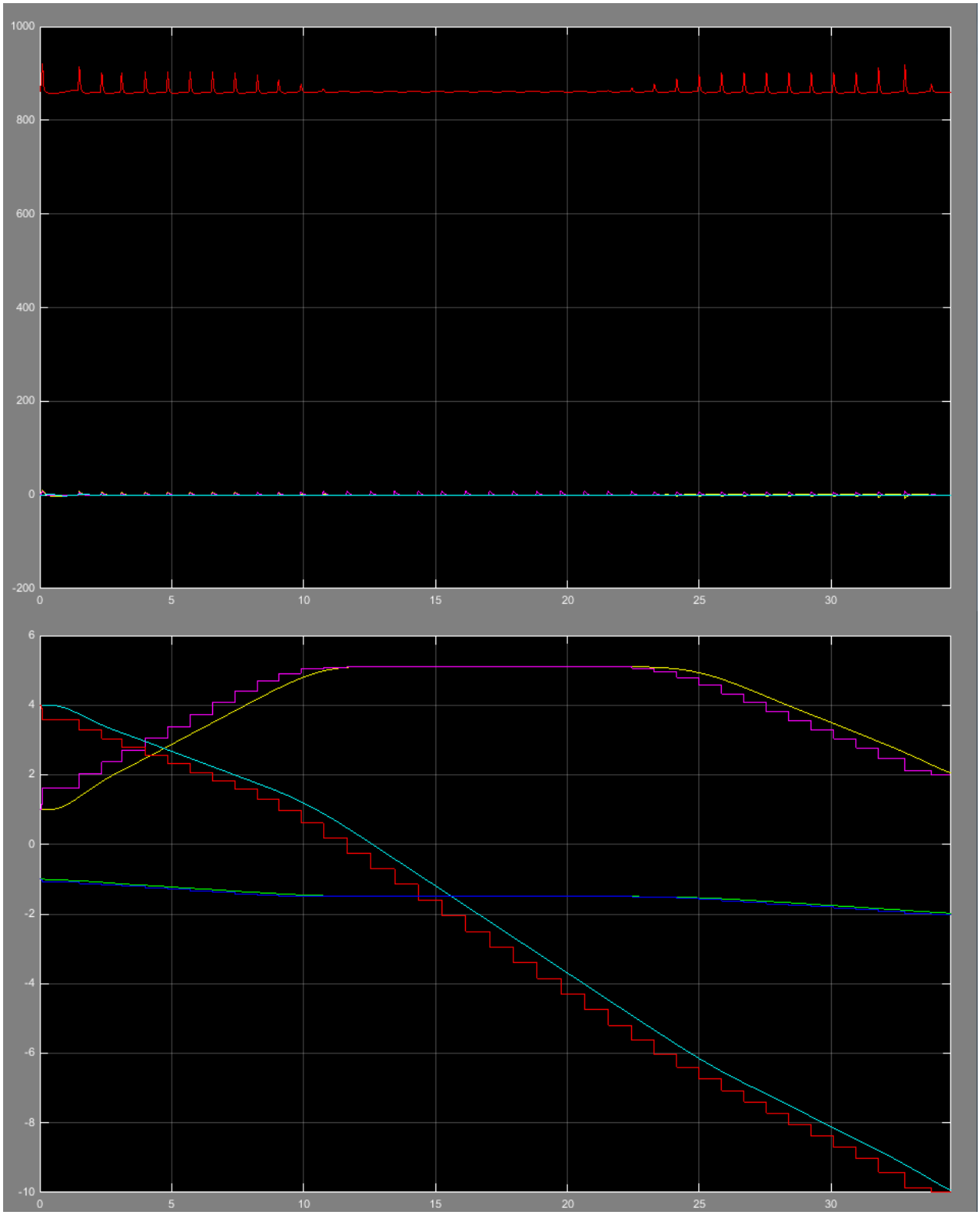


Figura 4-38. a) Se representa la fuerza de empuje (rojo) y los momentos resultantes del control en una trayectoria formada por 4 puntos (3 tramos rectos) para demostrar el problema que surge con la segunda forma de dar la referencia. b) Representación de las variables de posición de referencia (x*- magenta, y*- rojo, z*- azul) y las realmente seguidas (x - amarillo, y - cian, z - verde), para la misma trayectoria.

El problema que surge con esta segunda técnica, es que, con los controladores diseñados en este proyecto y explicados en el siguiente apartado, la respuesta ante cada escalón es demasiado brusca y se observa un pico en el empuje en cada cambio de referencia.

5 CONTROL

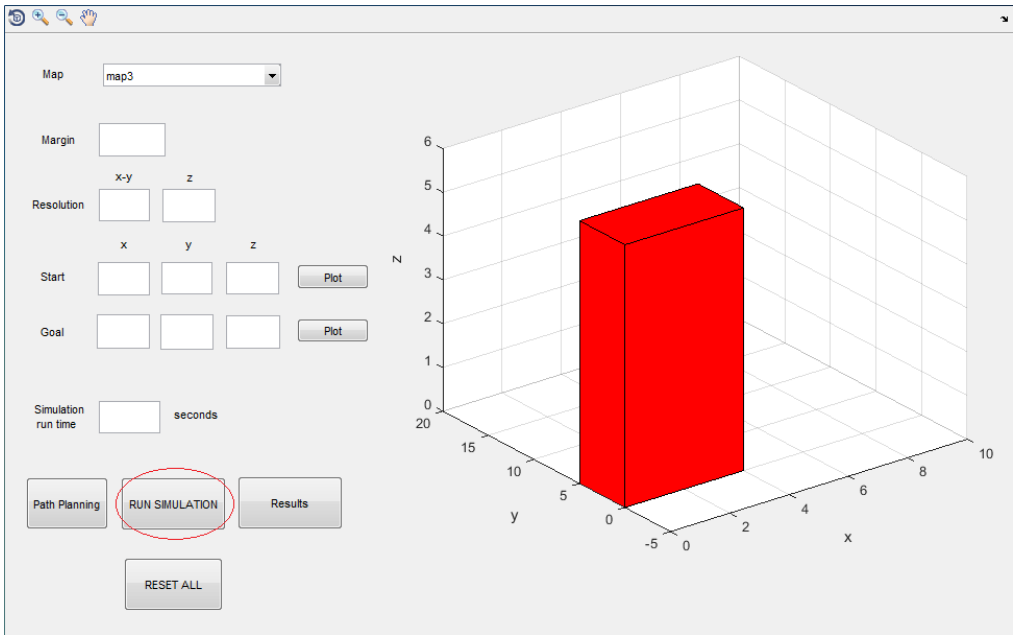


Figura 5-1. Botón de la interfaz de usuario correspondiente a la ejecución de la simulación.

A la vista de lo explicado en el apartado anterior, la trayectoria que debe seguir el robot se especifica en el espacio cartesiano. En este caso existen dos alternativas para su ejecución:

- Definir los lazos de control directamente en el espacio cartesiano y controlar el robot para que se anule el error de seguimiento de trayectoria en este espacio.
- Transformar la trayectoria del espacio cartesiano al espacio de las variables articulares y controlar la evolución de cada una de las variables articulares definiendo los lazos de control en este espacio.

Como es lógico la opción que aquí nos concierne es la primera, caso el cual es comúnmente empleado en robots móviles, como el que aquí se trata, donde la curvatura del camino generado en el espacio cartesiano está directamente relacionada con la variable de control que se emplea para el seguimiento de trayectorias.

El robot móvil, quadrotor, explicado en este documento, presenta seis grados de libertad (seis variables de localización), tres de posición y tres de orientación, que se procederán a controlar independientemente, es decir, de forma desacoplada. La validez de dicha estrategia viene ligada al hecho de que la sustentación no se vea afectada cuando se realicen desplazamientos en el plano horizontal, es decir, que los ángulos de giro sobre los ejes X e Y ("roll" y "pitch") sean muy pequeños, de forma que su coseno sea aproximable a uno. Como consecuencia a dicho control desacoplado se procederá a su división en dos partes: "position controller" y "attitude controller". La ventaja de esto es que permite que dicho sistema sea más simple al no depender la acción de los reguladores de las otras variables.

El primer bloque se encarga del control de las coordenadas (x,y,z) del centro de gravedad del robot en el entorno, a partir de las cuales, una vez controladas, se obtienen:

- Los ángulos de rotación, "roll" (ϕ*) y "pitch" (θ*), de referencia necesarios para realizar las traslaciones en el plano X-Y (mediante un cambio de sistema de coordenadas)
- El empuje necesario para la traslación en el eje Z.

De este modo, las entradas a este bloque serán las variables X*, Y* y Z* de referencia, y las salidas serán los ángulos roll y pitch citados anteriormente y la acción de control resultante, correspondiente al control en altitud.

Las variables "roll" y "pitch" (ángulos de giro sobre X e Y respectivamente), proporcionados por el anterior control de posición, junto a la "yaw" (ángulo de giro sobre eje Z), actuarán como entradas del "attitude controller", en el cual se llevará a cabo el control de las orientaciones respecto a los tres ejes de coordenadas. Las componentes de orientación sobre X e Y son de carácter transitorio: su valor en régimen permanente (en situación de sustentación, en ascenso/descenso o en ausencia de perturbaciones) es nulo.

Estos dos bloques representan los dos lazos que componen el controlador, uno interior, más rápido, y uno exterior, más lento, cuyos parámetros se diseñarán en función de los tiempos de respuesta del lazo interior.

Esta división da lugar a un control más delicado en lo que se refiere a los giros sobre los ejes horizontales, ya que como tanto "pitch" como "roll" influyen a la hora de realizar desplazamientos en el plano X-Y, si se da una fuerte inclinación sobre uno de estos ejes, se reduce el valor del empuje en Z y el quadrotor cae al suelo al no poder compensar dicha pérdida, debido a que la acción del control de altitud sólo depende de la altura (Martín I. 2012).

Según los párrafos anteriores y los parámetros de entrada del simulador, aunque se trata de un robot con seis grados de libertad, se está tratando de controlar un sistema subactuado pues se introducen un número de consignas de entrada menor al realmente necesario: las tres correspondientes a su localización en el espacio cartesiano, que irán variando conforme al cálculo de la trayectoria realizado en el apartado anterior, y la orientación respecto del eje z, "yaw", que se considerará de valor nulo para todo el documento (otro valor de referencia para esta variable sería útil si se quisiese conseguir una orientación final concreta).

Para obtener el sistema de control, se calcularán los parámetros de un controlador con doble lazo de realimentación para cada variable, con el fin de conseguir que no haya sobreoscilaciones y que el tiempo de respuesta sea el más bajo posible sin que se produzcan inestabilidades.

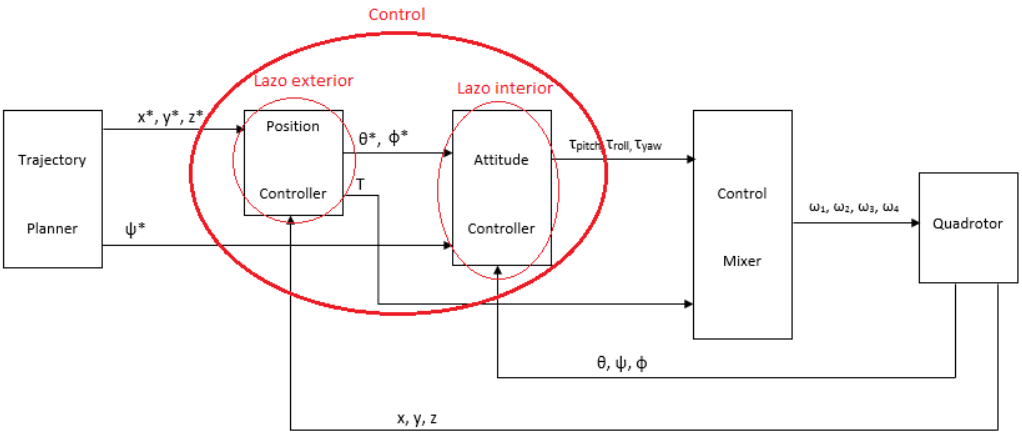


Figura 5-2. Detalle lazos de control interior y exterior.

5.1 Position Controller

5.1.1 Control en Z

Previamente al cálculo del controlador, el primer paso a realizar es determinar el tipo de sistema. El modelado de éste se realiza aplicando la 2ª Ley de Newton, en la que se cuenta la gravedad y la fricción viscosa del aire, en el campo transformado de Laplace.

$$m\ddot{z} = \sum \vec{F} => m\ddot{z} + f\dot{z} = \vec{T} - \vec{P}$$

donde $\vec{T}$ es el empuje generado por los rotores y $\vec{P}$ es el peso del quadrotor.

$$m\ddot{z} + f\dot{z} = 4bw^2 - mg \tag{5-19}$$

Despreciando el término debido a la fricción viscosa

$$m\ddot{z} = 4bw^2 - mg$$

y aplicando la transformada de Laplace

$$ms^2Z(s) = 4bw^2(s) - mg \tag{5-20}$$

se puede observar que aparece un término que no es controlable pues no depende de la velocidad de los rotores. Por ello, se considera que la función de transferencia del sistema es:

$$G(s) = \frac{Z(s)}{w^2(s)} = \frac{4b}{s(ms)} = \frac{4b}{ms^2} = \frac{K}{s^2} \tag{5-3}$$

y que el término será compensando bien sea, por ejemplo, con un controlador anticipativo para un sistema con perturbación constante o con un controlador con prealimentación, que anulen las perturbaciones debidas a la gravedad.

A la vista de los resultados, se puede comprobar que, en el caso del control de Z, se va a realizar un control lineal respecto a una variable la cual no es lineal, sino que tiene forma cuadrática, ω^2. Esto se debe a que el empuje que produce movimientos en dicho eje es dependiente de forma lineal del cuadrado de la velocidad angular de los distintos rotores (Martín I. (2012).

Una vez identificado el sistema, se puede comenzar ahora a diseñar los parámetros del controlador. Según la función de transferencia, el sistema posee dos integradores, lo que quiere decir que los errores de posición y velocidad serán nulos, pero hacen que el sistema sea marginalmente estable. En este caso, al aplicar un proporcional derivativo lo que hacemos es incluir un cero en la función de transferencia para dotar al sistema de cierta estabilidad en bucle cerrado.

Este tipo de control está compuesto dos lazos de realimentación: uno de posición y otro de velocidad.

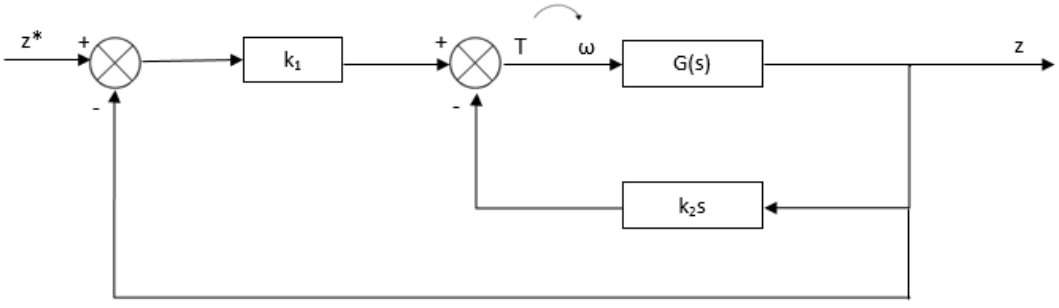


Figura 5-3. Diagrama de bloques del regulador con doble lazo de realimentación, siendo la k_1 la ganancia de la parte proporcional, la k_2 la ganancia de la parte derivativa, G(s) la función de transferencia del sistema para z y z* el valor de consigna.

Simplificando el esquema

$$H(s) = \frac{\frac{G(s)k_1}{1 + k_2 G(s)s}}{1 + \frac{G(s)k_1}{1 + k_2 G(s)s}} = \frac{G(s)k_1}{1 + k_2 G(s)s + G(s)k_1}$$

y sustituyendo la función de transferencia, $G(s)$

$$H(s) = \frac{k_1 \frac{K}{s^2}}{1 + k_2 \frac{K}{s^2} s + k_1 \frac{K}{s^2}} = \frac{k_1 K}{s^2 + k_2 K s + k_1 K} \tag{5-4}$$

resulta un sistema de segundo orden

$$H(s) = \frac{k_1 K}{s^2 + k_2 K s + k_1 K} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{5-5}$$

cuya respuesta depende de los valores de los parámetros, los cuales se obtienen a partir de las especificaciones que elijamos para el sistema. En este caso: para que no haya sobreoscilaciones amortiguadas, ξ=1, y para un tiempo de respuesta (t_e) de 10s, $\omega_n = \frac{4.75}{t_e \xi} = 0.475$.

Igualando ahora los términos, se obtienen los parámetros del controlador.

$$k_1 K = \omega_n^2 => k_1 = \frac{\omega_n^2}{K} = \frac{\omega_n^2 m}{4b} = \frac{0.475^2 * 4}{4 * 1.3234 * 10^{-5}} = 1.7 * 10^4$$

$$2\xi\omega_n = k_2 K => k_2 = \frac{2\xi\omega_n}{K} = \frac{2\xi\omega_n m}{4b} = \frac{2 * 1 * 0.475 * 4}{4 * 1.3234 * 10^{-5}} = 7.2 * 10^4$$

Sin embargo, tras implementar todos los controladores, tanto este como los que se explicarán más adelante, se modifican los valores buscando una mejor respuesta. Estos valores se fijan para un $t_e = 30s$, por lo que $k_1 = 1.9 * 10^3$ y $k_2 = 2.3 * 10^4$, además, también hay que tener en cuenta el signo de estas ganancias pues hay que recordar que la referencia en Z es negativa.

Para complementar el controlador, como ya se dijo anteriormente, se procede a implementar una estructura de control con prealimentación para anular las perturbaciones debidas a la gravedad. Lo que se consigue con esta estructura es que la salida siga lo más rápidamente posible las variaciones de la entrada cuando éstas se produzcan.

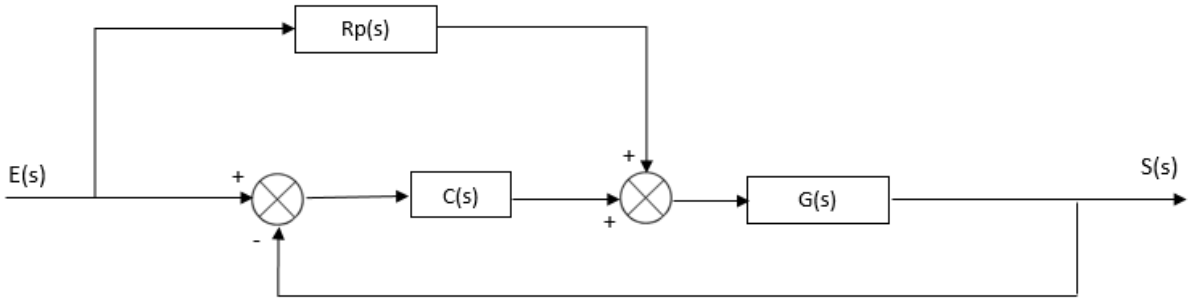


Figura 5-4. Diagrama de bloques de un controlador con prealimentación.

Al introducir la prealimentación, la salida del sistema queda de la forma

$$S(s) = \frac{R_p(s)C(s)G(s)}{1 + C(s)G(s)}E(s) + \frac{R_p(s)G(s)}{1 + C(s)G(s)}E(s) = \frac{R_p(s) + C(s)}{1 + C(s)G(s)}G(s)E(s)$$

Donde se debe cumplir que la función de transferencia sea la unidad

$$\frac{R_p(s) + C(s)}{1 + C(s)G(s)}G(s) = 1 \Leftrightarrow R_p(s) = \frac{1}{G(s)}$$

Y sustituyendo finalmente $G(s)$

$$R_p(s) = \frac{s^2}{K} = \frac{s^2 m}{4b} = 7.55 * 10^4 * s^2 \tag{5-6}$$

Se observa que se trata de una prealimentación doblemente derivativa, por lo que también es necesario calcular las velocidades y aceleraciones. Recordando lo que se dijo en el apartado anterior sobre las dos formas de proporcionar la referencia, en el primer caso se calcularían aceleraciones fuera del simulink (debido a que la función "mstraj" no devuelve el vector de velocidades y aceleraciones, estas se calculan mediante la fórmula derivada a partir del vector de trayectoria), mientras que en el segundo caso se calcularía durante la propia simulación.

Por último, como se dijo anteriormente, la otra posibilidad para eliminar las perturbaciones debidas a la gravedad es utilizando un controlador anticipativo para un sistema con perturbación constante, en el que el término adicional sería la velocidad media del rotor necesaria para generar un empuje igual al peso del vehículo que contrarrestaría dicho efecto.

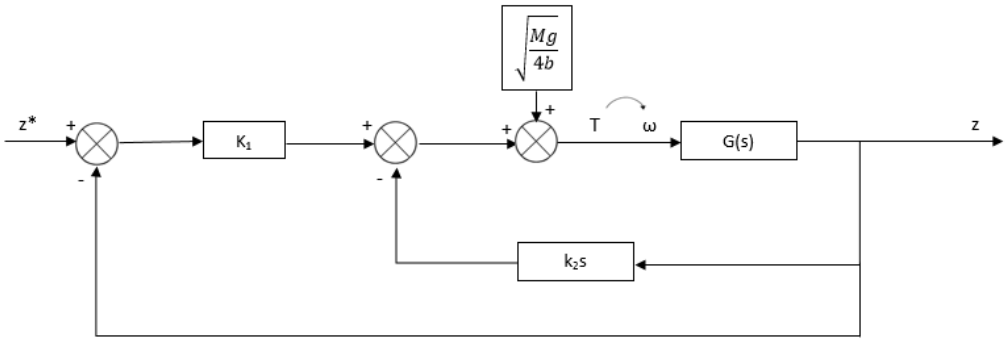


Figura 5-5. Diagrama de bloques de un controlador anticipativo.

Otras alternativas para el control anticipativo podrían ser, aplicar una ganancia muy alta para el bucle de altitud, pero esto podría conducir a saturación e inestabilidad del actuador, o utilizar un controlador proporcional-integral (PI), pero este, por otro lado, podría requerir mucho tiempo hasta que el término integral alcanzase el valor deseado, conduciendo a un sobretiempo.

5.1.2 Control en X/Y

A diferencia del control de Z, en este caso no se realiza un control lineal sobre una variable cuadrática, sino sobre una lineal, es decir, los empujes que provocan cambios en los ejes X e Y no dependen de ω^2, aunque a simple vista pueda parecerlo, por lo que el control de estas variables sí que se tratará de un control lineal propiamente dicho.

Como en el caso anterior, el controlador utilizado será un proporcional derivativo al que opcionalmente se le podría añadir una prealimentación de consigna, la cual sería también, como en el control de altitud, doblemente derivativa, aunque, sin embargo, este control no estrará tan bien ajustado debido al efecto aerodinámico de "rotor flapping", anteriormente comentado

En este caso, se deben distinguir dos sistemas de referencia: el sistema de referencia absoluto y el sistema de referencia ligado al cuerpo del quadrotor. Esto se debe a que las rotaciones que se producen en "yaw" hacen que las coordenadas absolutas no sean las mismas respecto a las del aparato. Sin embargo, los datos que proporciona el modelo (hipotéticos sensores) sí que proporcionan la posición absoluta, por lo que lo que se tendría que hacer es obtener el error en el primer sistema de referencia y, mediante rotación en Z, pasarlo al sistema que maneja el quadrotor.

$$\overrightarrow{\xi_E} = {}^0R_E^T \overrightarrow{\xi_O} => \begin{bmatrix}\theta\\ \phi\\ z\end{bmatrix} = \begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}^{-1} \begin{bmatrix}x\\ y\\ z\end{bmatrix} \tag{5-7}$$

De la relación entre los ángulos "roll" y "pitch" con las coordenadas en el plano horizontal, respecto a la referencia absoluta,

$$\ddot{X} = g(\cos(\psi)\theta + \sin(\psi)\phi) \tag{5-8}$$

$$\ddot{Y} = g(-\cos(\psi)\phi + \sin(\psi)\theta) \tag{5-9}$$

se pueden obtener las funciones de transferencia entre dichas coordenadas y ángulos, suponiendo que "yaw"=0.

$$G(s) = \frac{X(s)}{\theta(s)} = \frac{g}{s^2} \tag{5-10}$$

$$G(s) = \frac{Y(s)}{\phi(s)} = \frac{-g}{s^2} \tag{5-11}$$

De la misma forma que en el control para Z, se calculan ahora los parámetros para cada controlador, con ξ=1 (sobreoscilaciones amortiguadas), y $\omega_n = 1.5833$ (tiempo de respuesta de 3s)

$$H(s) = \frac{G(s)k_1}{1 + k_2 G(s)s + G(s)k_1} = \frac{k_1 \frac{g}{s^2}}{1 + k_2 \frac{g}{s^2} s + k_1 \frac{g}{s^2}} = \frac{k_1 g}{s^2 + k_2 g s + k_1 g} \tag{5-12}$$

$$k_1 g = \omega_n^2 => k_1 = \frac{\omega_n^2}{g} = \frac{1.5833^2}{9.81} = 0.2555$$

$$2\xi\omega_n = k_2 g => k_2 = \frac{2\xi\omega_n}{g} = \frac{2 * 1 * 1.5833}{9.81} = 0.322$$

siendo estos mismos valores válidos tanto como para el controlador en X como en Y, pero con signos diferentes.

De forma similar al caso anterior, se calcula la prealimentación para que la función de transferencia sea de valor unidad.

$$R_p(s) = \frac{1}{G(s)} = \frac{s^2}{g} \tag{5-13}$$

5.2 Attitude Controller

5.2.1 Control en yaw

Repitiendo los pasos de los apartados anteriores, lo primero es obtener la función de transferencia correspondiente a la entrada τ_{yaw}. Para ello, se parte de la condición que se debe dar en las las velocidades de los rotores para que haya un empuje que provoque un giro en el eje Z.

$$\tau_{yaw} = k(\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2) = k(2\omega_2^2 - 2\omega_1^2)$$

Si se aplica dicho empuje en dicha entrada cuando el quadrotor se encuentra en situación de sustentación, entonces

$$\tau_{yaw} = k\left[2\left(\omega_{yaw} - \omega_o\right)^2 - 2\left(\omega_{yaw} + \omega_o\right)^2\right] = 2k\left(-4\omega_{yaw}\omega_o\right)$$

donde $\omega_o = \sqrt{\frac{mg}{4b}} = \sqrt{\frac{4*9.81}{4*1.3234*10^{-5}}} = 860.97$ es la velocidad que deben tener los cuatro rotores para mantener al quadrotor en sustentación. Igualando la expresión anterior junto con la definición de momento angular de un sólido rígido

$$\tau_{yaw} = -I_{zz}\frac{d^2\psi}{dt^2} => \tau_{yaw}(s) = I_{zz}s^2\psi(s)$$

se puede despejar la función de transferencia

$$G(s) = \frac{\psi(s)}{w_{yaw}(s)} = \frac{-8\omega_o k}{I_{zz}s^2} = \frac{K_1}{s^2} = \frac{-8 * 860.97 * 1.0697 * 10^{-7}}{0.149 * s^2} = \frac{-0.004945}{s^2} = \frac{K_1}{s^2} \tag{5-14}$$

En este caso, aunque al principio pueda parecer que se trata de un control sobre una variable cuadrática, como ocurre en los apartados anteriores, en realidad es un control lineal sobre una variable lineal, es decir, sobre ω.

Por último, se calculan los parámetros del controlador, exactamente de la misma forma que en los casos anteriores, pero ahora variando las especificaciones: $\xi = 1$ (sobreoscilaciones amortiguadas), pero ω_n ahora se calcula para un tiempo de respuesta de 1s, pues, como se dijo anteriormente, el lazo interior debe ser más rápido que el lazo exterior, por lo que $\omega_n = 4.75$.

$$H(s) = \frac{G(s)k_1}{1 + k_2 G(s)s + G(s)k_1} = \frac{k_1 \frac{K_1}{s^2}}{1 + k_2 \frac{K_1}{s^2} s + k_1 \frac{K_1}{s^2}} = \frac{k_1 K_1}{s^2 + k_2 K_1 s + k_1 K_1} \quad (5\text{-}15)$$

$$k_1 K_1 = \omega_n^2 => k_1 = \frac{\omega_n^2}{K_1} = \frac{4.75^2}{-0.004945} = -4.56 * 10^3$$

$$2\xi\omega_n = k_2 K_1 => k_2 = \frac{2\xi\omega_n}{K_1} = \frac{2 * 1 * 4.75}{-0.004945} = -1.92 * 10^3$$

Tanto como para este controlador, como para los siguientes, no se añade ninguna prealimentación.

5.2.2 Control en pitch/roll

Finalmente, solo queda diseñar los controladores para las entradas que se encargan de los desplazamientos en horizontal, los ángulos "pitch" y "roll", los cuales se verán modificados durante dichas traslaciones. Dichas modificaciones en los valores de estos provocaran una disminución del empuje en vertical al cambiar el plano horizontal del quadrotor, de forma que este será $T_z = T\cos(\theta)$ si se produce un movimiento en el eje X o $T_z = T\cos(\phi)$ si es en el eje Y. Para que el valor del empuje en Z se mantenga lo más constante posible, estos ángulos deben ser muy pequeños de manera que el coseno de éstos sea prácticamente la unidad (Martín I. 2012).

Las expresiones de los empujes responsables de los movimientos en el plano X-Y, como ya se explicó en apartados anteriores, son:

$$\tau_{roll} = db(\omega_2^2 - \omega_4^2) \quad (5\text{-}16)$$

que corresponde con el necesario para provocar un giro sobre el eje Y, y

$$\tau_{pitch} = db(\omega_3^2 - \omega_1^2) \quad (5\text{-}17)$$

el necesario para provocar un giro sobre el eje X, donde d es la distancia de cada rotor al centro de gravedad. Como también se dijo anteriormente, las coordenadas X e Y no se manejan en el mismo sistema de referencia que en el que se manejan estos ángulos.

Para obtener las funciones de transferencia se procederá de la misma forma que para el ángulo "yaw", igualando las definiciones de los momentos angulares

$$\tau_{roll} = -I_{xx} \frac{d^2 roll}{dt^2} \quad (5\text{-}18)$$

$$\tau_{pitch} = -I_{yy} \frac{d^2 pitch}{dt^2} \quad (5\text{-}19)$$

con las expresiones de estos, en situación de sustentación y suponiendo que no se produce giro alguno sobre el eje Z (variación de "yaw").

$$\tau_{roll} = -4db\omega_o w_r$$

$$\tau_{pitch} = -4db\omega_o w_p$$

De modo que se obtiene

$$G(s) = \frac{\phi(s)}{w_r(s)} = \frac{-4db\omega_o}{I_{xx}s^2} = \frac{-4 * 0.315 * 1.3234 * 10^{-5} * 860.97}{0.082s^2} = \frac{-0.1751}{s^2} = \frac{K_2}{s^2} \quad (5\text{-}20)$$

$$G(s) = \frac{\theta(s)}{w_p(s)} = \frac{-4db\omega_o}{I_{yy}s^2} = \frac{-0.1751}{s^2} = \frac{K_2}{s^2} \quad (5\text{-}21)$$

Como se cumple que $I_{xx} = I_{yy}$, la función de transferencia es la misma para ambos, y, por tanto, solo se realiza el diseño del controlador una vez, pues serán los mismos valores como para uno como para otro.

$$H(s) = \frac{G(s)k_1}{1 + k_2G(s)s + G(s)k_1} = \frac{k_1\frac{K_2}{s^2}}{1 + k_2\frac{K_2}{s^2}s + k_1\frac{K_2}{s^2}} = \frac{k_1K_2}{s^2 + k_2K_2s + k_1K_2} \quad (5\text{-}22)$$

Las especificaciones, como es lógico, serán $\xi = 1$ y $\omega_n = 4.75$ (tiempo de respuesta de 1s).

$$k_1K_2 = \omega_n^2 => k_1 = \frac{\omega_n^2}{K_2} = \frac{4.75^2}{-0.1751} = -128.86$$

$$2\xi\omega_n = k_2K_2 => k_2 = \frac{2\xi\omega_n}{K_2} = \frac{2 * 1 * 4.75}{-0.1751} = -54.26$$

La mayor diferencia que presenta este control respecto de los demás, es que la entrada no viene dada por una referencia externa, sino que es resultado de un bucle anterior. Cabría pensar que en el control de la variable "yaw" ocurre lo mismo pues no se aprecia en ningún sitio que se le pueda indicar un valor deseado, sin embargo, esto no es cierto, lo que ocurre es que en este documento no se han considerado cambios en esta variable, por lo que la referencia, de forma interna, se mantiene constante de valor nulo.

6 INTERFAZ Y SIMULADOR

Aunque ya se han ido comentado a lo largo de todo el documento las distintas funcionalidades de la interfaz de usuario diseñada, se describe brevemente en este apartado el desarrollo de dicha esta, así como también algunas funciones, adicionales al modelo dinámico del quadrotor, al generador de trayectorias y a los controladores, que también se incluyen en el simulador y que ayudan a mejorarlo.

6.1 Interfaz de usuario

6.1.1 Map

Esta lista desplegable permite elegir el entorno con obstáculos en el que se quiere simular. Se pueden añadir más entornos entrando en el código de la interfaz y diseñando dichos entornos tal y como se explica en el apartado correspondiente de generación de trayectorias.

6.1.2 Margin

Permite al usuario elegir la distancia de seguridad que se quiere mantener para asegurar que el vehículo no interfiera con ningún obstáculo.

6.1.3 Resolution

Se elige la resolución que se quiere utilizar para la creación de la matriz nodos de exploración, tanto como para el plano x-y como para el eje z.

6.1.4 Start

Se introducen las coordenadas del punto inicial de la trayectoria y se comprueba que dicho punto es posible, es decir, no se encuentra dentro de ningún bloque y además respeta la distancia de seguridad. Si esto no se cumple, aparece un mensaje de error. Si, por el contrario, el punto es válido, este se puede representar mediante el botón de “plot”.

6.1.5 Goal

Al igual que para el punto de inicio, el punto final de la trayectoria se puede representar si es válido, apareciendo un mensaje de error si no lo es.

6.1.6 Simulation run time

Para la primera forma de dar la referencia a los controladores, de la que se habló en el apartado de control, es necesario especificar el tiempo en el que se quiere que el vehículo recorra la trayectoria. Como ya se dijo, cuanto mayor sea dicho tiempo, más precisa será la trayectoria.

6.1.7 Path planning

Al ejecutar este botón se realizan todos los pasos correspondientes a la generación de trayectorias y se representa la trayectoria final a seguir por el vehículo en la ventana de simulación.

6.1.8 Run simulation

Este botón inicia la simulación.

6.1.9 Ventana de simulación

Una vez ejecutada la simulación, se puede comprobar la posición del vehículo durante todo el trayecto, así como también, tener una primera impresión de la precisión de dicho seguimiento.

6.1.10 Results

Una vez terminada la simulación, cuando el vehículo ha alcanzado el punto final, se puede comprobar mediante tres gráficas:

- La trayectoria tridimensional teórica calculada por el generador y la realmente seguida por el aparato.

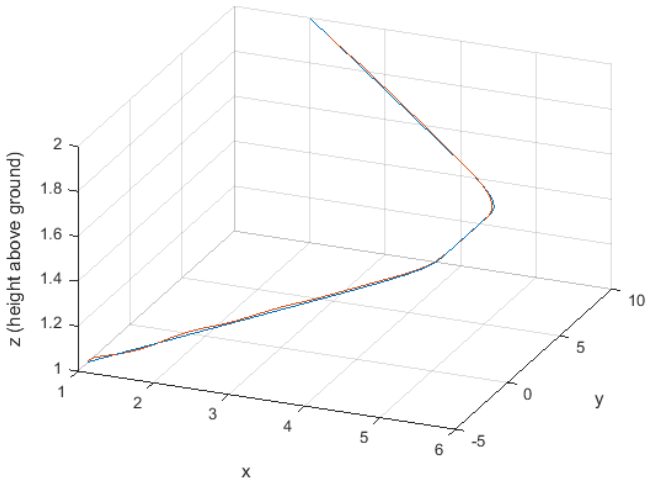


Figura 6-1. Representación trayectoria tridimensional real (roja) y deseada (azul).

- La evolución de las fuerzas ejercidas por el quadrotor (empuje y momentos) durante todo el trayecto.

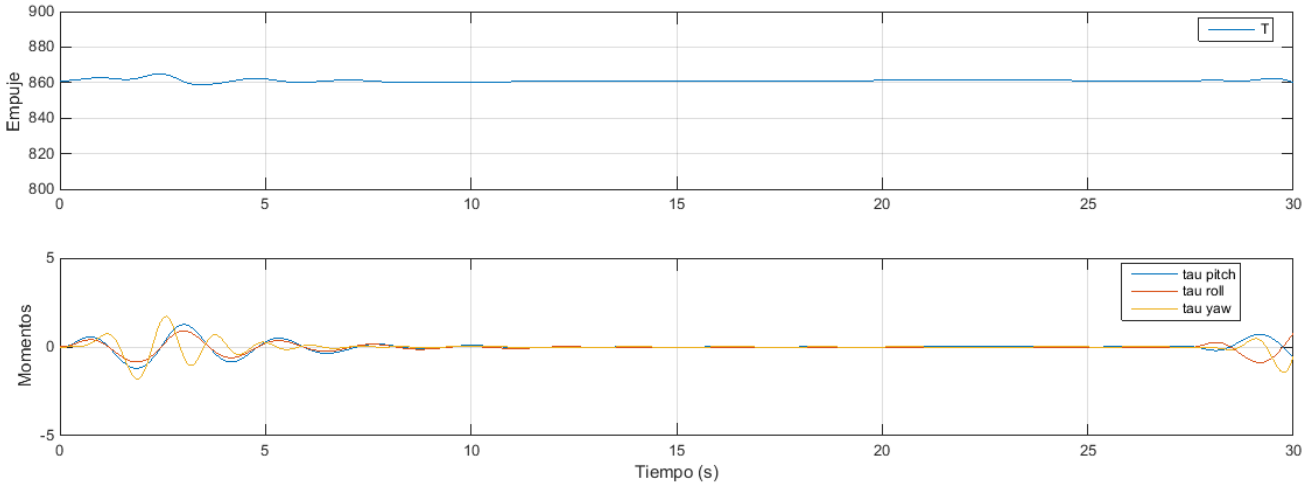


Figura 6-2. Representación las fuerzas (empuje y momentos) ejercidos por el quadrotor.

- La evolución de las tres variables de posición (x, y, z), tanto las de referencia como las realmente seguidas por el vehículo como consecuencia de los controladores.

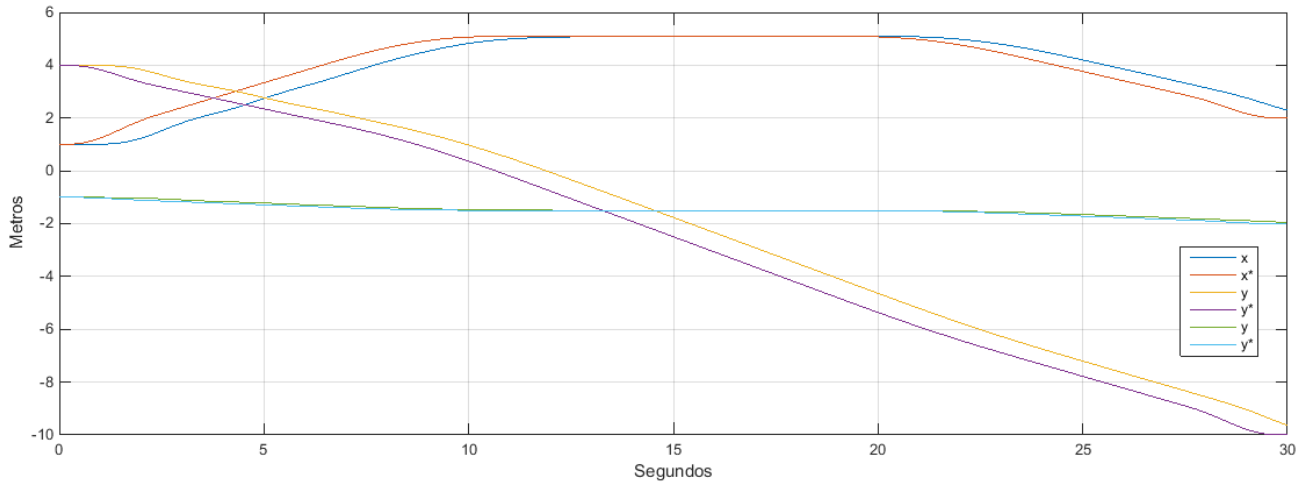


Figura 6-3. Representación por separado de las variables de posición (x, y, z) reales y deseadas (con un *).

6.1.11 Reset

Elimina los datos introducidos en todos los campos de la interfaz y resetea la ventana de simulación.

6.2 Simulador

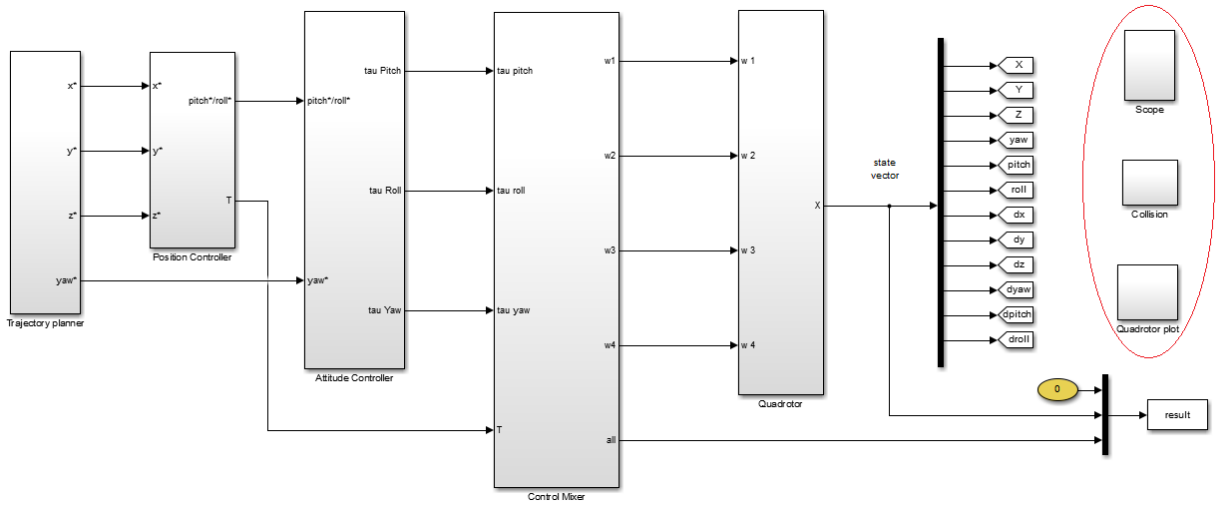


Figura 6-4. Funciones adicionales del simulador.

6.2.1 Collision

Este bloque se encarga de comprobar si en cada momento (en cada paso de simulación) el quadrotor respeta la distancia de seguridad que se ha introducido o si choca contra el suelo, deteniendo la simulación si en algún momento ocurre alguna de estas dos cosas, debido, por ejemplo, a un mal control.

6.2.2 Quadrotor plot

Permite la representación del vehículo durante toda la simulación.

6.2.3 Scope

Recoge los datos resultantes de la simulación para poder representarlos al final de esta, como se vio en el epígrafe anterior.

Finalmente, para ejecutar el simulador, se abre el archivo “GUIDE.m” y se ejecuta. Este, automáticamente, abrirá la interfaz de usuario en la que se procederá como corresponde.

REFERENCIAS

[1] Barragán Guerrero, D. O. (2007). Manual de intefaz gráfica de usuario en Matlab: Parte I. Recuperado de: https://www.dspace.espol.edu.ec/bitstream/123456789/10740/19/%255Bmatlab%255D_MATLAB_GUIDE.pdf

[2] Berrutti J., Falkenstein, L. y Favaro, F. (2015) *Vuelo autónomo de un cuadricóptero* (Proyecto fin de carrera). Recuperado de: https://www.colibri.udelar.edu.uy/handle/123456789/5232

[3] Cherevatsky, B. (2016). *Trajectory generation.* (Powerpoint). Recuperado de: http://slideplayer.com/slide/6149223/

[4] Corke, P. (2011). Time and motion. En Bruno Siciliano y Oussama Khatib (Eds.) *Robotic, Vision and Control*, (pp. 46-48). Berlin: Springer.

[5] Corke, P. (2011). Mobile Robot Vehicles. En Bruno Siciliano y Oussama Khatib (Eds.) *Robotic, Vision and Control*, (pp. 78-83). Berlin: Springer.

[6] Corke, P. (2012). *Robotics Toolbox for Matlab*. Recuperado de: http://opencourses.emu.edu.tr/pluginfile.php/2445/mod_resource/content/0/lecture%20notes/robot.pdf

[7] Corke, P. (2017). Petercorke: Resources for robotics education: code, books and MOOCS. Recuperado el 31 de agosto de 2017, de http://petercorke.com/wordpress/toolboxes/robotics-toolbox

[8] Ferrera, E. & Capitán, J. (2017). *Unmaned middle-size Ground Robots: Collision avoidance and path planning.* (Powerpoint).

[9] López, C. (2014). *El algoritmo A**. Recuperado de: https://es.slideshare.net/CristinaLopez35/el-algoritmo-a-asterisco?from_action=save

[10] Martín Blázquez, I. (2012) *Caracterización empírica y diseño del sistema de control de trayectoria de un quadrotor mediante simulación*. (Trabajo fin de carrera). Recuperado de: https://zaguan.unizar.es/record/11809/files/TAZ-PFC-2013-397.pdf

[11] Ollero Baturone, A. (2005). Generación de trayectorias. *Robótica: manipuladores y robots móviles*, (pp. 304-305). Barcelona: Marcombo y Alfaomega.

[12] Patel, A. (2016). redblobgames: Introduction to A*. Recuperado el 31 de agosto de 2017, de http://www.redblobgames.com/pathfinding/a-star/introduction.html

[13] Pounds, P., Mahony, R. & Corke, P. (2010) Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7), pp. 691-699. Recuperado de: https://eprints.qut.edu.au/33732/1/cep2009_modelling_and_control_paper_sub_final.pdf

[14] Vendittelli, M. (2017). *Elective in Robotics: Quadrotor modeling.* (Powerpoint). Recuperado de: http://www.dis.uniroma1.it/~venditt/didattica/eir/01_Modeling.pdf

[15] Vianna Raffo, G. (2007). *Modelado y control de un helicóptero quadrotor* (Trabajo de Fin de Máster). Recuperado de: http://bibing.us.es/proyectos/abreproy/70017/fichero/Tesis_Master_GuilhermeRaffo.pdf

[16] Vianna Raffo, G., Ortega, M. G. & Rubio, F. (2010). An integral predictive/nonlinear H∞ control structure for a quadrotor helicopter. *Automatica (Journal of IFAC)*, 46(1), pp. 29-39. Recuperado de: http://dl.acm.org/citation.cfm?id=1688046

[17] Vora, A. (2016). Ankit Vora: Motion planning for quadrotors. Recuperado el 31 de agosto de 2017, de http://ankitvora.tk/