UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas
Centro de Investigación de Métodos Computacionales

DISEÑO DE SISTEMAS DE CONTROL Y NAVEGACIÓN PARA VEHÍCULOS AÉREOS NO TRIPULADOS MEDIANTE SIMULACIÓN VIRTUAL

Marina Hebe Murillo

Tesis remitida al Comité Académico del Doctorado
como parte de los requisitos para la obtención
del grado de
DOCTOR EN INGENIERIA
Mención Inteligencia Computacional, Señales y Sistemas
de la
UNIVERSIDAD NACIONAL DEL LITORAL

2015

Comisión de Posgrado, Facultad de Ingeniería y Ciencias Hídricas, Ciudad Universitaria, Paraje "El Pozo", S3000, Santa Fe, Argentina.

Índice general

Índice de figuras

Resumen Extendido

El objetivo principal de la presente tesis involucra primordialmente el diseño de sistemas de control para el vuelo y navegación de vehículos aéreos no tripulados (UAVs) autónomos, como así también la utilización y desarrollo de una plataforma de simulación, visualización y control de vuelo que permite la validación y evaluación de los mismos.

Los vehículos aéreos no tripulados son vehículos capacitados para volar sin tripulación a bordo de la aeronave. Se los denomina comúnmente como UAVs por sus siglas en inglés (*Unmanned Aerial Vehicles*). Los mismos poseen diversas formas, configuraciones y son de reducido tamaño. Resultan ser una herramienta de mucha utilidad cuando las condiciones de operación son peligrosas ya que no ponen en riesgo vidas humanas. Debido a estas características, en la actualidad, el interés por el desarrollo y utilización de UAVs ha experimentado un gran crecimiento. Dichos vehículos aéreos tienen un amplio espectro de usos posibles, destacándose su utilización en aplicaciones tales como: agrimensura y agricultura, realizando vuelos sobre campos; seguridad e ingeniería civil, equipados con cámaras de monitoreo; servicios forestales, realizando relevamientos de áreas boscosas y control de incendios; búsqueda y rescate de personas, entre otras. Asimismo, este tipo de aeronaves se han popularizado dentro del ámbito académico y científico. Se clasifican en dos grandes grupos: 1) aquellos que cuentan con la capacidad de realizar misiones sin intervención humana (autónomos), y 2) aquellos controlados por un operador humano de manera remota. En particular, en el marco de la presente tesis, los UAVs autónomos son los de principal interés.

Para que los UAVs puedan volar autónomamente, es indispensable que los mismos cuenten con un adecuado sistema de control de actitud y de navega-

ción. El control de actitud se encarga de computar los valores de deflexiones que deben tomar las superficies aerodinámicas (elevador, alerones y timón vertical) y la columna de propulsión para controlar la orientación y velocidad del vehículo. Por su parte, el control de navegación es el encargado de determinar el camino o recorrido, es decir, la trayectoria de navegación, que el UAV autónomo debe seguir. En esta tesis, el diseño de los sistemas de control de actitud y de navegación se planteó desde dos perspectivas diferentes, una unificada y otra desacoplada. La primera se refiere a la utilización de un único controlador para controlar tanto la dinámica como la navegación. El sistema de control propuesto tiene en cuenta la dinámica completa, de 6-grados de libertad (6-DOF), del mismo. Esto es muy ventajoso ya que no es necesario desacoplar el modelo en sus modos dinámicos, como por ejemplo el longitudinal y el transversal, y en consecuencia los acoples existentes entre modos son tenidos en cuenta. Para diseñar este sistema de control unificado, se desarrolló el método INL-MPC (*Iterative non-Linear Model Predictive Control*), que es una técnica de control predictivo no-lineal iterativo basado en modelos. El mismo extiende las capacidades del control predictivo clásico (MPC clásico) permitiendo el tratamiento de sistemas con dinámicas no-lineales genéricos. La técnica INL-MPC transforma el problema de control óptimo no-lineal de un sistema no-lineal en una serie de sub-problemas de optimización cuadráticos, iterativos, más sencillos de resolver. La segunda perspectiva, se refiere al desacople del sistema de control en dos controladores independientes, uno para controlar la actitud del UAV y otro para obtener las trayectorias de navegación. Para el diseño del controlador de actitud, se utilizó un método basado en la técnica INL-MPC aún más genérico que permite la inclusión, en el problema de optimización, de una función de costo no-lineal. Dicho método se nota como INL-MPC-J. El diseño del sistema de control de navegación se realizó utilizando modelos de vehículos reducidos tipo partícula e INL-MPC, permitiendo la generación de diferentes trayectorias de navegación óptimas, de forma dinámica. El sistema de control de actitud y navegación desacoplado también permite la utilización del modelo completo, de 6-DOF, del UAV. En este caso, a diferencia del sistema de control unificado, los dos controladores se ejecutan de forma independiente en paralelo, reduciéndose, en consecuencia, el costo

computacional del sistema de control global.

Como parte de esta tesis también se desarrolló una plataforma de simulación, visualización y control de vuelo, denominada *Excalibur*, la cual extiende las capacidades de un simulador de vuelo convencional. Los simuladores de vuelo convencionales permiten crear una réplica del comportamiento de un determinado vehículo aéreo. En general, los mismos funcionan en modo manual, es decir, mediante la utilización de un *joystick* o un teclado estándar se comandan las deflexiones de las superficies aerodinámicas y de la columna de propulsión. La plataforma mencionada, además de poseer un simulador de vuelo convencional, posee un módulo de control de actitud y de navegación, tanto unificado como desacoplado. En principio el módulo de control presente en dicha plataforma puede ser utilizado tanto para controlar modelos virtuales del simulador de vuelo como vehículos aéreos reales. El desempeño de los métodos de control desarrollados, tanto unificados como desacoplados, se evaluaron con la ayuda de la plataforma *Excalibur*. Para modelar el UAV de orden completo, de 6-DOF, se usó el modelo matemático del avión Cessna 172. Utilizando el sistema de control de actitud y de navegación unificado, se realizaron diferentes maniobras de vuelo autónomo, como ser ascensos, cambios en la dirección de vuelo y giros coordinados. Asimismo, empleando el sistema de control desacoplado, se computaron diversas trayectorias de navegación, tanto en el plano como en el espacio, y se evaluó el seguimiento de las mismas con el modelo del UAV anteriormente mencionado.

Capítulo 1

Introducción

Una de las áreas de la aviación que ha experimentado un gran crecimiento es aquella que involucra a los vehículos aéreos no tripulados (UAVs). Esto se debe, principalmente, a que los mismos pueden llevar a cabo una amplia gama de aplicaciones a un costo más bajo y sin arriesgar vidas humanas. Los vehículos aéreos no tripulados, como su nombre lo indica, cuentan con la capacidad de volar sin un piloto a bordo, pudiendo ser operados de manera remota o bien de forma totalmente autónoma [1, 2]. Los mismos tienen un extenso campo de aplicaciones. Mayormente se utilizan en misiones como ser, por ejemplo, búsqueda y rescate de personas [3], agricultura de precisión [4], recolección de imágenes [5], seguridad [6], lucha contra incendios forestales [7], entre otras.

Para que los UAVs puedan volar de forma totalmente autónoma, es indispensable que los mismos cuenten con un adecuado sistema de control de actitud y de navegación. El primero computa los valores que deben tomar las entradas de control del UAV para modificar su orientación y velocidad a los valores deseados, mientras que el segundo determina la trayectoria de navegación que el UAV autónomo debe seguir. Numerosas son las técnicas de control que han sido evaluadas para el diseño de este tipo de sistemas de control. El *feedback control* [8, 9] es la técnica de control por excelencia pues permite estabilizar los modos dinámicos inestables, aumentar el amortiguamiento, hacer el comportamiento dinámico menos sensible a variaciones internas de la configuración del avión, etc. Una de las técnicas clásicas de control más conocida es

el control Proporcional-Integral-Derivativo (PID). La misma ha sido utilizada en numerosas ocasiones para el control de vehículos aéreos, por ejemplo, en [10] se desarrolla un sistema de control de actitud y navegación basado en la utilización de tres controladores PID que permiten llevar a un vehículo aéreo tipo cuadricóptero a un estado de vuelo deseado. Generalmente, cuando el control PID se utiliza con sistemas con múltiples entradas y múltiples salidas (sistemas MIMO), como son los vehículos aéreos, se desacopla el modelo de los mismos en sus distintos modos dinámicos. Además, cuando se utilizan sistemas no-lineales con este tipo de controladores, el ajuste de los parámetros del controlador PID se debe realizar dinámicamente. La obtención de la ley de control basada en PID requiere un amplio conocimiento acerca de la física del sistema que se desea controlar.

Desde hace algunas décadas, las técnicas de control clásico han dejado su lugar a técnicas de control óptimo, empezando por el trabajo pionero de Bryson [11, 12, 13]. En la actualidad, son las técnicas más frecuentemente utilizadas en aeronáutica. Sin embargo, desde comienzos de la presente década han cobrado interés el uso de técnicas de control basadas en horizontes móviles, las cuales extienden el rango de aplicación del control óptimo a sistemas con restricciones y con observación temporal discreta [14, 15, 16]. Estas técnicas de control discreto forman parte de una familia más amplia conocida como control predictivo basado en modelos (en inglés, *Model Predictive Control* - MPC) [17, 18].

MPC es un método de control en el cual se utilizan modelos lineales de un determinado sistema para predecir su comportamiento futuro a lo largo de un horizonte de predicción. El método MPC se formula mediante la resolución *online* de un problema de optimización, paramétrico en las acciones de control, que genera como resultado una secuencia de entradas de control óptima a lo largo de un horizonte de control. Solamente el primer elemento de la solución del problema de optimización se aplica al sistema de acuerdo a una estrategia de horizonte deslizante. La técnica MPC ha sido satisfactoriamente aplicada en una amplia variedad de casos. Esto se debe a que tanto los modelos como las restricciones en estados y entradas pueden incorporarse explícitamente en el cálculo de la secuencia de control óptimo [17, 19]. Una revisión acerca de

MPC puede encontrarse en [20, 21, 22].

Como la mayoría de los sistemas físicos son sistemas con dinámicas no-lineales, la técnica MPC clásica ha sido extendida a esta clase de sistemas, dando origen al control predictivo no-lineal basado en modelos (en inglés, *Non-linear Model Predictive Control* - NLMPC) [23, 24, 25, 26]. No obstante, esta técnica de control no-lineal no ha sido ampliamente utilizada para el control y operación de UAVs. Esto se debe a que los UAVs son sistemas no-lineales con dinámicas rápidas y el problema de optimización no-lineal debe resolverse teniendo en cuenta las restricciones temporales impuestas por las aplicaciones de tiempo real. En la actualidad, el desafío por desarrollar nuevos métodos de control que demanden un menor costo computacional continúa. Recientemente, ha surgido un nuevo método de control predictivo no-lineal genérico que extiende las capacidades del MPC clásico y permite el control de sistemas con dinámicas no-lineales. El mismo se denomina control predictivo no-lineal iterativo basado en modelos (en inglés, denominado *Iterative non-Linear Model Predictive Control* - INL-MPC), y fue desarrollado por la autora de la presente tesis y colaboradores [27]. El método INL-MPC permite transformar el problema de control óptimo no-lineal de un sistema no-lineal en una serie de sub-problemas de optimización cuadráticos, iterativos y más sencillos de resolver. Esto se logra utilizando un proceso de linealización generalizado alrededor de trayectorias en espacio de estado iterativas. El método INL-MPC se puede ver como una sucesión iterativa de métodos MPC lineales variantes en el tiempo. En INL-MPC se asume que la función de costo a utilizar en el problema de optimización es una función cuadrática tanto del vector de estados del sistema como del vector velocidad de cambio de entradas de control. Sin embargo, existen situaciones en las cuales se desea incluir en el problema de optimización otras variables que no forman parte de dichos vectores, pero que sí son funciones no-lineales de los mismos. Es por ello que se ha extendido el método INL-MPC a su utilización con funciones de costo no-lineales dependientes del vector de estado y de las velocidades de cambio de las entradas de control. Esta nueva metodología se nota aquí por sus siglas como INL-MPC-J.

Para el cómputo de trayectorias de navegación, los sistemas de control de navegación utilizan diversas metodologías. Por ejemplo, en [28] se generan las

trayectorias de navegación mediante la obtención de *splines* suaves utilizando técnicas de control óptimo y de programación matemática. En primer lugar, el método se centra en la generación de curvas que pasen cerca de los *waypoints* (puntos de paso deseados) y luego se agregan las restricciones de interpolación a la formulación del problema. Es importante destacar que el trabajo anteriormente mencionado computa las trayectorias de navegación *offline* sin considerar la dinámica del vehículo que debe seguirla. En el trabajo [29] se presenta una técnica para la generación de una trayectoria a partir de una serie de *waypoints*. Mediante la utilización de *B-Splines*, la trayectoria puede modificarse en presencia de ambientes dinámicos. En el artículo [30], se presenta un algoritmo para generación de trayectorias de navegación en dos dimensiones. El mismo propone que las transiciones de un segmento de trayectoria a otro se realicen en el menor tiempo posible. El cómputo de las trayectorias de navegación, en ambos casos, se realiza *online* sin considerar la dinámica del vehículo que debe seguirla. En el marco de la presente tesis, se desarrolló una nueva metodología que permite tanto la generación como el seguimiento de trayectorias de navegación en el plano (2D) y en el espacio (3D). La técnica propuesta se basa en el cómputo de una trayectoria de navegación *online* a partir de una colección de puntos de paso, en inglés comúnmente llamados *waypoints*, por los cuales debe pasar la trayectoria de navegación que debe seguir el UAV autónomo. Dichas trayectorias se computan utilizando un modelo de vehículo reducido tipo partícula y la técnica de control INL-MPC. En consecuencia, las trayectorias de navegación obtenidas son aquellas que minimizan la distancia entre el origen y el destino, y que además consideran las restricciones impuestas por la dinámica propia de los modelos utilizados. Tanto el problema de generación de trayectorias de navegación como el seguimiento de las mismas, se plantean como un problema de control óptimo no-lineal con restricciones, utilizándose las técnicas de control INL-MPC e INL-MPC-J, respectivamente.

Como parte de la presente tesis también se desarrolló una plataforma de Simulación, Visualización y Control de vuelo (plataforma SVC), denominada *Excalibur*, la cual extiende las capacidades de los simuladores de vuelo convencionales, ya que la misma, además del funcionamiento en modo manual, puede ser utilizada en modo automático. Esto se logró mediante la implementación

de un módulo de control automático en la plataforma mencionada. Además, la misma puede ser utilizada tanto con modelos virtuales de UAVs como con vehículos aéreos reales. En el marco de la presente tesis, la plataforma SVC se usó con modelos virtuales de UAVs, permitiendo recrear por computadora el comportamiento de un avión real no-tripulado y evaluar el desempeño de los UAVs en vuelo autónomo utilizando la misma como banco de pruebas para diseñar, validar y evaluar los sistemas de control de actitud y de navegación.

Generalmente, lo anteriormente mencionado constituye un paso previo a la implementación de los diferentes sistemas de control de actitud y de navegación en aviones o en aviones autónomos no-tripulados reales [31, 32].

La presente tesis se organiza de la siguiente manera: en el capítulo 2 se presenta una revisión acerca del modelado de sistemas con dinámicas lineales y no-lineales. En el capítulo 3 se introduce la técnica de control MPC clásica. En el capítulo 4 se presenta el desarrollo del método de control INL-MPC. En el capítulo 5 se generaliza la técnica INL-MPC mediante la inclusión de funciones de costo no-lineales que permiten considerar, en el proceso de optimización, variables que no forman parte del vector de estados del sistema, dando origen al método INL-MPC-J. En el capítulo 6 se propone una metodología para la generación de trayectorias de navegación utilizando modelos de vehículos reducidos tipo partícula y la técnica INL-MPC. En el capítulo 7 se diseña un sistema de control unificado utilizando la técnica INL-MPC. El mismo permite controlar tanto la dinámica como la navegación de vehículos aéreos no-tripulados. Para modelar el UAV de orden completo, se usa el modelo matemático del avión Cessna 172. Mediante la realización de diferentes maniobras de vuelo típicas, se evalúa la *performance* del sistema de control propuesto. En el capítulo 8, para evaluar la metodología propuesta para el cómputo y seguimiento de trayectorias de navegación, se diseña, utilizando INL-MPC e INL-MPC-J, un sistema de control desacoplado formado por dos controladores independientes, uno para controlar la actitud del UAV y otro para obtener las trayectorias de navegación del mismo. En el capítulo 9 se detallan los aspectos de diseño de la plataforma SVC *Excalibur*. Finalmente, en el capítulo 10 se presentan las conclusiones y trabajos futuros propuestos.

Capítulo 2

Modelado Matemático de Sistemas Dinámicos

Antes de comenzar a analizar los sistemas de control automático, resulta conveniente realizar una revisión acerca del modelado de sistemas dinámicos.

Un sistema dinámico, lineal o no-lineal, puede representarse mediante un modelo matemático, el cual consiste en un conjunto de ecuaciones diferenciales que son representativas de la dinámica del mismo. La obtención de un buen modelo matemático resulta ser apropiada a la hora de desarrollar un buen algoritmo de control. Como se verá más adelante, en el marco de esta tesis, se utilizará como modelo matemático aquél basado en la representación en espacio de estados.

Este capítulo se organiza como sigue: en la sección 2.1 se describe la representación en espacio de estados de un sistema lineal genérico. En la sección 2.2 se describen los procesos de discretización e integración de un sistema lineal que permiten obtener la versión discreta del modelo en espacio de estados del mismo. En la sección 2.3 se presenta la forma genérica representativa de la dinámica de un sistema no-lineal. En la sección 2.4 se muestra la metodología convencionalmente empleada para aproximar la dinámica de los sistemas no-lineales mediante la utilización de modelos linealizados alrededor de algún punto de operación. En la sección 2.5 se presenta el algoritmo de integración que permite obtener los valores de los estados del sistema en los sucesivos

instantes de tiempo. Finalmente, en la sección 2.6 se presenta el modelo matemático no-lineal que describe la dinámica de vehículos aéreos.

2.1. Dinámica de Sistemas Lineales

Un sistema es lineal si el mismo satisface el principio de superposición. Este principio establece que la respuesta producida por la aplicación de dos causas superpuestas (por ejemplo distintas entradas y/o condiciones iniciales) originan la superposición de los correspondientes efectos. En consecuencia, para los sistemas lineales, la respuesta a varias entradas puede calcularse considerando los efectos de a una entrada por vez y luego sumando las respuestas individuales.

La representación general de un sistema dinámico lineal arbitrario está dada por la siguiente ecuación diferencial de primer orden:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{2.1}$$

donde $\mathbf{x}(t)$ es el vector de estados del sistema, $\mathbf{u}(t)$ es el vector de entradas de control del sistema, $\dot{\mathbf{x}}(t)$ es el vector de velocidad de cambio de estados, $\mathbf{A}$ es la matriz de estados y $\mathbf{B}$ es la matriz de entradas.

La solución general de la Ec. (2.1) es

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t^e)}\mathbf{x}(t^e) + \int_{t^e}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{2.2}$$

2.2. Discretización e Integración de Sistemas Lineales

Se asume que en el instante de tiempo $t = t^0$ el sistema se encuentra en el estado arbitrario $(\mathbf{x}^0, \mathbf{u}^0)$. Los tiempos discretos se definen como $t^k = t^0 + k\Delta T_s$, donde ΔT_s es la longitud del intervalo de tiempo $\left[t^k, t^{k+1}\right]$ durante el cual se aplica la entrada de control $\mathbf{u}(t^{k+1})$, que se mantiene constante a lo largo de dicho intervalo.

Notación: De ahora en adelante se utilizarán superíndices para denotar los valores discretos de las variables en los diferentes instantes de tiempo discretos t^k. Por ejemplo, el valor de la variable $y(t)$ en el instante de tiempo t^k se denotará como y^k.

Evaluando la integral de Ec. (2.2) en el intervalo $t^e = t^k$ y $t = t^{k+1} = t^k + \Delta T_s$, se obtiene que:

$$\mathbf{x}^{k+1} = e^{\mathbf{A}\Delta T_s}\mathbf{x}^k + \int_{t^k}^{t^{k+1}} e^{\mathbf{A}(t^{k+1}-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{2.3}$$

Los elementos de las matrices $\mathbf{A}$ y $\mathbf{B}$ son valores constantes. La entrada de control $\mathbf{u}(\tau)$ también se mantiene constante en el intervalo de integración, tomando el valor $\mathbf{u}^{k+1}$ en el intervalo de tiempo $\left[t^k, t^{k+1}\right]$. Haciendo el cambio de variables $\nu = t^{k+1} - \tau$, la Ec. (2.3) se puede escribir como:

$$\mathbf{x}^{k+1} = e^{\mathbf{A}\Delta T_s}\mathbf{x}^k + \left[\int_0^{\Delta T_s} e^{\mathbf{A}\nu}d\nu\right]\mathbf{B}\mathbf{u}^{k+1} \tag{2.4}$$

Llamando

$$\tilde{\mathbf{A}} = e^{\mathbf{A}\Delta T_s},\ \tilde{\mathbf{B}} = \left[\int_0^{\Delta T_s} e^{\mathbf{A}\nu}d\nu\right]\mathbf{B} \tag{2.5}$$

se obtiene la versión discreta de la ecuación en espacio de estados definida en Ec. (2.1), la cual se puede escribir finalmente como:

$$\mathbf{x}^{k+1} = \tilde{\mathbf{A}}\mathbf{x}^k + \tilde{\mathbf{B}}\mathbf{u}^{k+1} \tag{2.6}$$

2.3. Dinámica de Sistemas no-Lineales

Para el caso de los sistemas no-lineales el principio de superposición no es válido, es decir, la respuesta del sistema a varias entradas no puede calcularse mediante la superposición de las respuestas a entradas individuales.

La representación general de un sistema dinámico no-lineal arbitrario está dada por la siguiente ecuación diferencial de primer orden:

$$\dot{\mathbf{x}}(t) = \bar{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.7}$$

donde $\mathbf{x}(t)$ es el vector de estados del sistema, $\mathbf{u}(t)$ es el vector de entradas de control del sistema, $\dot{\mathbf{x}}(t)$ es el vector de velocidad de cambio de estados y $\bar{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t))$ es una función vectorial que depende del sistema modelado.

2.4. Linealización de Sistemas no-Lineales

La ecuación de estados no-lineal definida en Ec. (2.7) puede linealizarse expandiendo $\bar{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t))$ en serie de Taylor de primer orden alrededor de un punto de operación fijo $(\mathbf{x}_e, \mathbf{u}_e)$ ([8, 17]):

$$\dot{\mathbf{x}}(t) \approx \bar{\mathbf{f}}(\mathbf{x}_e, \mathbf{u}_e) + \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}}|_{(\mathbf{x}_e, \mathbf{u}_e)}(\mathbf{x}(t) - \mathbf{x}_e) + \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}}|_{(\mathbf{x}_e, \mathbf{u}_e)}(\mathbf{u}(t) - \mathbf{u}_e) \tag{2.8}$$

donde $\frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}}$ y $\frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}}$ son las matrices jacobianas del sistema evaluadas en $(\mathbf{x}_e, \mathbf{u}_e)$ y se definen como:

$$\frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_{12}} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_{12}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{12}}{\partial x_1} & \frac{\partial f_{12}}{\partial x_2} & \cdots & \frac{\partial f_{12}}{\partial x_{12}} \end{bmatrix} \quad \text{y} \quad \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_4} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_4} \\ \vdots & \vdots & & \\ \frac{\partial f_{12}}{\partial u_1} & \frac{\partial f_{12}}{\partial u_2} & \cdots & \frac{\partial f_{12}}{\partial u_4} \end{bmatrix} \tag{2.9}$$

Como se mencionó anteriormente, $(\mathbf{x}_e, \mathbf{u}_e)$ es un punto de linealización fijo, arbitrario, por lo tanto, $\bar{\mathbf{f}}(\mathbf{x}_e, \mathbf{u}_e)$ puede ser calculada para cada $(\mathbf{x}_e, \mathbf{u}_e)$ usando la Ec. (2.7) y puede tomar valores no nulos. Luego, si se definen

$$\mathbf{A} = \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}}|_{(\mathbf{x}_e, \mathbf{u}_e)}, \ \mathbf{B} = \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}}|_{(\mathbf{x}_e, \mathbf{u}_e)}, \ \mathbf{d} = \bar{\mathbf{f}}(\mathbf{x}_e, \mathbf{u}_e) - \mathbf{A}\mathbf{x}_e - \mathbf{B}\mathbf{u}_e \tag{2.10}$$

entonces, la Ec. (2.8) puede escribirse en forma compacta como

$$\dot{\mathbf{x}}(t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{d} \tag{2.11}$$

La solución general de la Ec. (2.11) es

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t^e)}\mathbf{x}(t^e) + \int_{t^e}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau + \int_{t^e}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{d}d\tau \tag{2.12}$$

2.5. Integración de Sistemas no-Lineales

Notación: Para simplificar la notación, de aquí en adelante se omitirá la dependencia temporal de las variables, es decir: $\mathbf{x} = \mathbf{x}(t)$, $\mathbf{u} = \mathbf{u}(t)$ y $\dot{\mathbf{x}} = \dot{\mathbf{x}}(t)$.

Para hallar los valores de los estados de un sistema no-lineal en los sucesivos instantes de tiempo discreto t^k, la Ec. (2.7) se puede integrar utilizando algún método de integración numérica. En particular, en esta tesis, se utiliza el método *Runge-Kutta* de cuarto orden, el cual se describe, a modo de revisión, a continuación:

Dado el sistema no-lineal de Ec. (2.7)

$$\dot{\mathbf{x}} = \bar{\mathbf{f}}(\mathbf{x}, \mathbf{u}) \tag{2.13}$$

cuya condición inicial está dada por los vectores $\mathbf{x}^0$ y $\mathbf{u}^0$. El método *Runge-Kutta* de cuarto orden establece que:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \frac{1}{6}h(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{2.14}$$

donde

$$\begin{aligned} \mathbf{k}_1 &= \bar{\mathbf{f}}(\mathbf{x}^k, \mathbf{u}^k) \text{ en } t = t^k \\ \mathbf{k}_2 &= \bar{\mathbf{f}}(\mathbf{x}^k + 0{,}5\mathbf{k}_1 h, \mathbf{u}^k) \text{ en } t = t^k + 0{,}5h \\ \mathbf{k}_3 &= \bar{\mathbf{f}}(\mathbf{x}^k + 0{,}5\mathbf{k}_2 h, \mathbf{u}^k) \text{ en } t = t^k + 0{,}5h \\ \mathbf{k}_3 &= \bar{\mathbf{f}}(\mathbf{x}^k + \mathbf{k}_3 h, \mathbf{u}^k) \text{ en } t = t^k + h \end{aligned} \tag{2.15}$$

donde $k = 0, 1, 2, 3, \cdots$ y h es el tamaño del intervalo de integración.

$\mathbf{x}^{k+1}$ resulta ser la aproximación del vector de estados $\mathbf{x}$ evaluado en el instante de tiempo $t = t^{k+1}$. $\mathbf{x}^{k+1}$ se determina utilizando el valor del estado actual $\mathbf{x}^k$ más el producto del tamaño del intervalo de integración h con el promedio ponderado de los cuatro incrementos $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{k}_3$ y $\mathbf{k}_4$.

2.6. Dinámica no-Lineal de Vehículos Aéreos

Un vehículo aéreo o avión es una aeronave capaz de volar debido a las fuerzas de sustentación que genera su diseño aerodinámico. El vehículo aéreo como sistema dinámico se modela como un sólido rígido de orden completo, es decir

con 6-grados de libertad (6-DOF). Su estado dinámico puede representarse por el siguiente vector de estados:

$$\begin{aligned} \mathbf{x} &= [x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5 \;\; x_6 \;\; x_7 \;\; x_8 \;\; x_9 \;\; x_{10} \;\; x_{11} \;\; x_{12}]^T \\ &= [v_t \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; p \;\; q \;\; r \;\; x_{\mathrm{N}} \;\; y_{\mathrm{E}} \;\; h]^T \end{aligned} \tag{2.16}$$

de dimensión $N_s = 12$ y donde:

- v_t, α y β son la velocidad, el ángulo de ataque y el ángulo de deslizamiento, respectivamente.

- ϕ, θ y ψ son los ángulos de *roll*, *pitch* y *yaw*, respectivamente, que definen la orientación del sistema coordenado cuerpo (B-*Frame*) $[\hat{x}_{\mathrm{Body}}, \hat{y}_{\mathrm{Body}}, \hat{z}_{\mathrm{Body}}]$ con respecto al sistema coordenado de referencia *North-East-Down* (NED-*Frame*) $[\hat{x}_{\mathrm{NED}}, \hat{y}_{\mathrm{NED}}, \hat{z}_{\mathrm{NED}}]$ (ver Fig. 2.1). La secuencia de rotaciones convencionalmente utilizada para describir la actitud instantánea del avión en el B-*Frame* con respecto al NED-*Frame* es la siguiente:

  1. Rotación de un ángulo ψ positivo (nariz hacia la derecha) alrededor del eje $\hat{\mathbf{z}}_{\mathrm{NED}}$.
  2. Rotación de un ángulo θ positivo (nariz hacia arriba) alrededor del eje resultante $\hat{\mathbf{y}}_{\psi}$.
  3. Rotación de un ángulo ϕ positivo (ala derecha hacia abajo) alrededor del eje resultante $\hat{\mathbf{x}}_{\theta}$.

- p, q y r son las componentes del vector velocidad angular del avión expresadas en el B-*Frame*.

- x_{N}, y_{E} y $z_{\mathrm{D}} = -h$ son las componentes de la posición del CG del avión con respecto al NED-*Frame*.

El vector $\mathbf{u}$, cuya dimensión es $N_i = 4$, se define mediante las siguientes variables de control del avión:

$$\mathbf{u} = [u_1 \;\; u_2 \;\; u_3 \;\; u_4]^T = [thtl \;\; \delta_e \;\; \delta_a \;\; \delta_r]^T \tag{2.17}$$

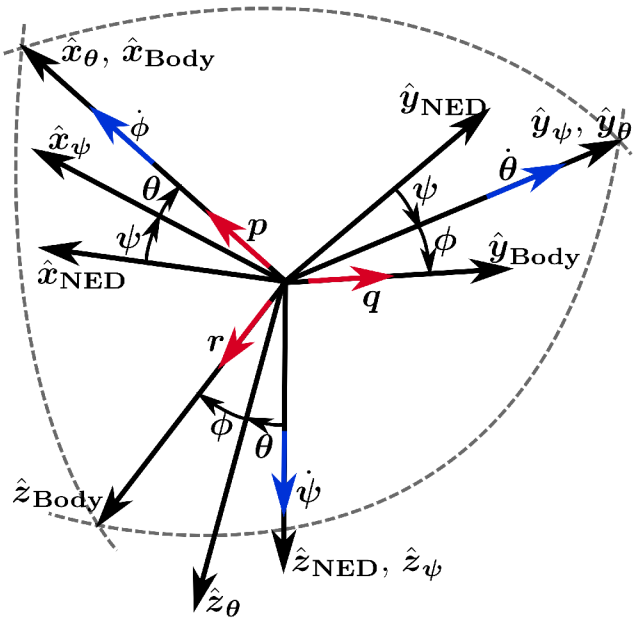


Figura 2.1: Marcos de referencias NED-*Frame* y B-*Frame* con sus correspondientes ángulos de Euler

donde δ_e es la deflexión del elevador en grados, δ_a es la deflexión del alerón en grados, δ_r es la deflexión del timón vertical en grados y *thtl* es la posición de la columna de propulsión cuyo valor se encuentra normalizado entre cero y uno.

En la Fig. 2.2 se puede observar un esquema del avión modelado junto con los marcos de referencia utilizados. El B-*Frame* es una terna que se mueve conjuntamente con el avión, su eje $\hat{x}_{\text{Body}}$ se alinea con la nariz del avión, su eje $\hat{y}_{\text{Body}}$ se alinea con el ala derecha y el eje $\hat{z}_{\text{Body}}$ se selecciona de forma que cumpla con la regla de la mano derecha. El NED-*Frame* es el marco de referencia inercial. El eje $\hat{x}_{\text{Wind}}$ del sistema coordenado viento (W-*Frame*) se alinea con la dirección relativa del viento.

La orientación del NED-*Frame* con respecto al B-*Frame* puede ser siempre descripta mediante una secuencia de tres rotaciones en el plano, es decir mediante rotaciones alrededor de un único eje a la vez (ángulos de Euler). Luego, la rotación tridimensional propiamente dicha puede obtenerse concatenando las rotaciones individuales, es decir, en forma matricial se tiene que:

$$\mathbf{B}_{\text{NED}}^{\text{B}} = \mathbf{B}_{\hat{\mathbf{x}}_\theta}(\phi)\mathbf{B}_{\hat{\mathbf{y}}_\psi}(\theta)\mathbf{B}_{\hat{\mathbf{z}}_{\text{ned}}}(\psi) \tag{2.18}$$

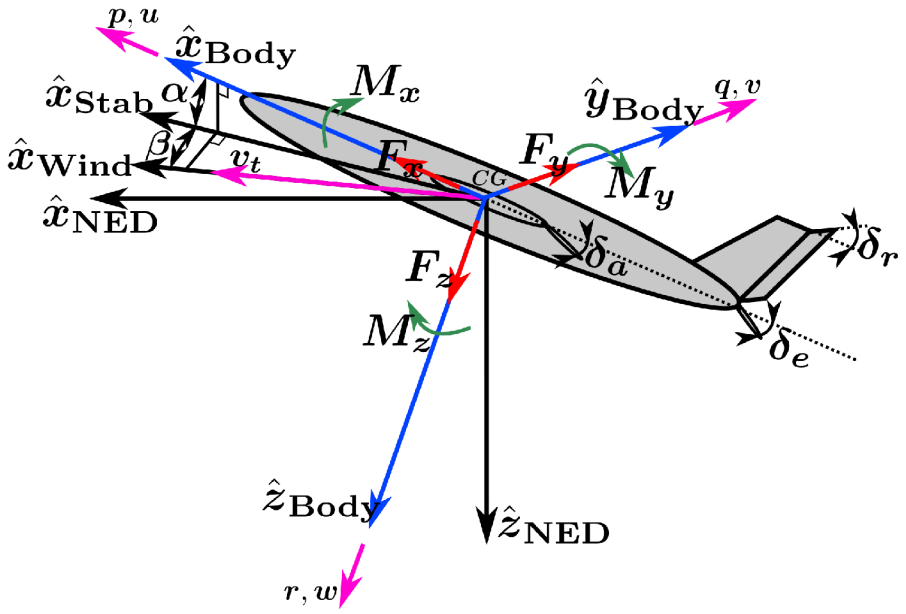


Figura 2.2: Avión modelado y sistemas coordenados de referencia

donde $\mathbf{B}^{\mathrm{B}}_{\mathrm{NED}}$ es la matriz que transforma desde el NED-*Frame* al B-*Frame* y viceversa, tomando su transpuesta. Las matrices $\mathbf{B}_{\hat{\mathbf{z}}_{\mathrm{ned}}}(\psi)$, $\mathbf{B}_{\hat{\mathbf{y}}_{\psi}}(\theta)$ y $\mathbf{B}_{\hat{\mathbf{x}}_{\theta}}(\phi)$ definen las secuencias de rotaciones individuales convencionalmente utilizadas para describir la actitud instantánea del avión, descriptas en los ítems 1, 2 y 3 de la sección 2.6, respectivamente.

Definiendo c_γ, s_γ y t_γ como la notación para $\cos(\gamma)$, $\mathrm{sen}(\gamma)$ y $\tan(\gamma)$, respectivamente para un ángulo genérico γ, las matrices de rotaciones individuales pueden escribirse como:

$$\mathbf{B}_{\hat{\mathbf{x}}_{\theta}}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_\phi & \mathrm{s}_\phi \\ 0 & -\mathrm{s}_\phi & \mathrm{c}_\phi \end{bmatrix}, \mathbf{B}_{\hat{\mathbf{y}}_{\psi}}(\theta) = \begin{bmatrix} \mathrm{c}_\theta & 0 & -\mathrm{s}_\theta \\ 0 & 1 & \mathrm{s}_\phi \\ \mathrm{s}_\theta & 0 & \mathrm{c}_\theta \end{bmatrix} \text{ y}$$
$$\mathbf{B}_{\hat{\mathbf{z}}_{\mathrm{ned}}}(\psi) = \begin{bmatrix} \mathrm{c}_\psi & \mathrm{s}_\psi & 0 \\ -\mathrm{s}_\psi & \mathrm{c}_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

Resolviendo el producto matricial definido en la Ec. (2.18), la matriz que define la transformación completa del NED-*Frame* al B-*Frame* queda definida

como

$$\mathbf{B}_{\text{NED}}^{\text{B}} = \begin{bmatrix} \text{c}_\theta\text{c}_\psi & \text{c}_\theta\text{s}_\psi & -\text{s}_\theta \\ \text{s}_\phi\text{s}_\theta\text{c}_\psi - \text{c}_\phi\text{s}_\psi & \text{s}_\phi\text{s}_\theta\text{s}_\psi + \text{c}_\phi\text{c}_\psi & \text{s}_\phi\text{c}_\theta \\ \text{c}_\phi\text{s}_\theta\text{c}_\psi + \text{s}_\phi\text{s}_\psi & \text{c}_\phi\text{s}_\theta\text{s}_\psi - \text{s}_\phi\text{c}_\psi & \text{c}_\phi\text{c}_\theta \end{bmatrix} \tag{2.20}$$

Puede demostrarse que las columnas de $\mathbf{B}_{\text{NED}}^{\text{B}} = [\mathbf{b_1}\ \mathbf{b_2}\ \mathbf{b_3}]$ cumplen que

$$\dot{\mathbf{b}}_i = -\mathbf{w} \times \mathbf{b}_i \tag{2.21}$$

para $i = 1, 2, 3$ y donde $\mathbf{w} = [p\ q\ r]^T$ son las componentes del vector velocidad angular $\mathbf{w}$ expresado en el B-*Frame*.

Otra matriz de rotación que resulta de importancia, y que será utilizada más adelante, es aquella que permite transformar desde el marco de referencia W-*Frame* al marco de referencia B-*Frame*, la cual se define en [8] como:

$$\mathbf{B}_{\text{W}}^{\text{B}} = \begin{bmatrix} \text{c}_\alpha\text{c}_\beta & -\text{c}_\alpha\text{s}_\beta & -\text{s}_\alpha \\ \text{s}_\beta & \text{c}_\beta & 0 \\ \text{s}_\alpha\text{c}_\beta & -\text{s}_\alpha\text{s}_\beta & \text{c}_\alpha \end{bmatrix} \tag{2.22}$$

donde α y β son los ángulos de ataque y deslizamiento, respectivamente.

Utilizando la Ec. (2.20) y evaluando las Ecs. (2.21) para el primer y segundo elemento de $\dot{\mathbf{b}}_3$ y el primer elemento de $\dot{\mathbf{b}}_2$, se pueden calcular las derivadas temporales de los ángulos de Euler, obteniéndose así las ecuaciones que definen la actitud del avión:

$$\dot{\phi} = p + \text{t}_\theta\,(q\text{s}_\phi + r\text{c}_\phi) \tag{2.23}$$

$$\dot{\theta} = q\text{c}_\phi - r\text{s}_\phi \tag{2.24}$$

$$\dot{\psi} = (q\text{s}_\phi + r\text{c}_\phi)/\text{c}_\theta \tag{2.25}$$

Para obtener las ecuaciones de fuerzas y de momentos que rigen el movimiento traslacional y rotacional, respectivamente, se aplica la segunda ley de Newton, resultando:

Ecuaciones de Fuerzas

$$\dot{u} = rv - qw - g\text{s}_\theta + F_x/m \tag{2.26}$$

$$\dot{v} = -ru + pw + g\mathrm{s}_\phi \mathrm{c}_\theta + F_y/m \tag{2.27}$$

$$\dot{w} = qu - pv + g\mathrm{c}_\phi \mathrm{c}_\theta + F_z/m \tag{2.28}$$

donde u, v, w son las componentes del vector velocidad del centro de masa del avión expresado en el B-*Frame*, g es la aceleración de la gravedad y F_x, F_y y F_z son las fuerzas totales (aerodinámicas más propulsión) actuantes en el avión, expresadas en el B-*Frame*. Estas fuerzas dependen explícitamente de las variables de estado y de control.

Como las fuerzas y los momentos aerodinámicos dependen de los ángulos aerodinámicos, es decir del ángulo de ataque α y del ángulo de deslizamiento β, y de la velocidad verdadera v_t, en general es conveniente utilizar como variables de estado a v_t, α y β en lugar de u, v y w, por lo tanto, en vez de utilizar las Ecs. (2.26 -2.28) generalmente se utilizan las siguientes ecuaciones:

$$\dot{v}_t = (u\dot{u} + v\dot{v} + w\dot{w})/v_t \tag{2.29}$$

$$\dot{\alpha} = (u\dot{w} - w\dot{u})/(u^2 + w^2) \tag{2.30}$$

$$\dot{\beta} = (\dot{v}v_t - v\dot{v}_t)/(v_t^2 \cos\beta) \tag{2.31}$$

Ecuaciones de Momentos

$$\dot{p} = (c_1 r + c_2 p)\, q + c_3 M_x + c_4 M_z \tag{2.32}$$

$$\dot{q} = c_5 pr - c_6 \left(p^2 - r^2\right) + c_7 M_y \tag{2.33}$$

$$\dot{r} = (c_8 p - c_2 r)\, q + c_4 M_x + c_9 M_z \tag{2.34}$$

donde c_i son los coeficientes relacionados con los momentos de inercia del avión y se definen como $c_1 = \frac{1}{\Gamma}\left(J_{zz}J_{yy} - J_{xz}^2 - J_{zz}^2\right)$, $c_2 = \frac{1}{\Gamma}J_{xz}\left(J_{zz} + J_{xx} - J_{yy}\right)$, $c_3 = \frac{1}{\Gamma}J_{zz}$, $c_4 = \frac{1}{\Gamma}J_{xz}$, $c_5 = \frac{J_{zz}-J_{xx}}{J_{yy}}$, $c_6 = \frac{J_{xz}}{J_{yy}}$, $c_7 = \frac{1}{J_{yy}}$, $c_8 = \frac{1}{\Gamma}\left[\left(J_{xx} - J_{yy}\right)J_{xx} + J_{xz}^2\right]$, $c_9 = \frac{J_{xx}}{\Gamma}$, $\Gamma = J_{xx} \cdot J_{zz} - J_{xz}^2$, donde J_{ij} son las componentes del tensor de inercia. M_x, M_y y M_z son los momentos totales (aerodinámicos más propulsión) con respecto a la posición del CG, actuantes en el avión, expresados en el B-*Frame*. Estos momentos también dependen explícitamente de las variables de estado y de control.

Las ecuaciones de movimiento traslacional se completan con las ecuaciones de navegación que relacionan los cambios de posición del avión con las componentes de velocidad. En términos de las velocidades del avión en ejes cuerpo las mismas son:

$$\dot{x}_N = u\mathrm{c}_\theta\mathrm{c}_\psi + v(-\mathrm{c}_\phi\mathrm{s}_\psi + \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{c}_\psi) + w(\mathrm{s}_\phi\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{c}_\psi) \tag{2.35}$$

$$\dot{y}_E = u\mathrm{c}_\theta\mathrm{s}_\psi + v(\mathrm{c}_\phi\mathrm{c}_\psi + \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{s}_\psi) + w(-\mathrm{s}_\phi\mathrm{c}_\psi + \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi) \tag{2.36}$$

$$\dot{h} = u\mathrm{s}_\theta - v\mathrm{s}_\phi\mathrm{c}_\theta - w\mathrm{c}_\phi\mathrm{c}_\theta \tag{2.37}$$

Agrupando las Ecs. (2.29 - 2.31), (2.23 - 2.25), (2.32 - 2.34), (2.35 - 2.37), la función vectorial $\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ queda determinada por:

$$\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} (u\dot{u} + v\dot{v} + w\dot{w})/v_t \\ (u\dot{w} - w\dot{u})/(u^2 + w^2) \\ (\dot{v}v_t - v\dot{v}_t)/(v_t^2 \cos\beta) \\ p + \tan\theta(q\sin\phi + r\cos\phi) \\ q\cos\phi - r\sin\phi \\ (q\sin\phi + r\cos\phi)/\cos\theta \\ (c_1 r + c_2 p)q + c_3 M_x + c_4 M_z \\ c_5 pr - c_6(p^2 - r^2) + c_7 M_y \\ (c_8 p - c_2 r)q + c_4 M_x + c_9 M_z \\ ub_{11} + vb_{21} + wb_{31} \\ ub_{12} + vb_{22} + wb_{32} \\ -ub_{13} - vb_{23} - wb_{33} \end{bmatrix} \tag{2.38}$$

donde b_{ij} representa el elemento de la fila i y la columna j de la matriz de rotación definida en la Ec. (2.20).

2.7. Conclusiones

En este capítulo se presentó la dinámica de sistemas lineales y no-lineales. Se introdujo, además, el concepto de linealización alrededor de puntos de operación arbitrarios y se presentó el método de integración numérica *Runge-Kutta*,

el cual puede ser utilizado para hallar los sucesivos estados temporales del sistema no-lineal. Por último, se describió el modelo matemático no-lineal para el caso de vehículos aéreos, el cual será luego utilizado como modelo de avión no-tripulado (UAV) de los distintos algoritmos de control diseñados.

Capítulo 3

Control Predictivo (MPC)

En este capítulo se presenta el método de control predictivo, comúnmente conocido por su nombre en inglés como *Model Predictive Control* (MPC).

La organización de este capítulo es la siguiente: en la sección 3.1 se introduce el método y además se exponen los conceptos básicos de dicha estrategia de control. En la sección 3.2 se presenta la metodología de predicción de estados futuros del sistema a partir de una secuencia de entradas de control y del estado inicial. En la sección 3.3 se expresan las restricciones en estados, entradas de control y velocidad de cambio de entradas de control como desigualdad matricial. En la sección 3.4 se plantea el problema de minimización, sujeto a restricciones, mediante el cual es posible obtener las acciones de control óptimas. En la sección 3.5 se presenta el funcionamiento de la unidad de control basada en el método MPC. Finalmente, en la sección 3.6 se formaliza el algoritmo MPC.

3.1. Introducción a la Técnica MPC

Desde hace algunas décadas, las técnicas de control clásicas como las utilizadas en [33, 8, 9] han dejado lugar a métodos de control más modernos, como por ejemplo control óptimo [34], lógica difusa [35], MPC [36, 37, 38, 39] y otras [40, 41].

Desde sus inicios, en la década de 1970, el control MPC ha sido objetivo de

estudio de distintas comunidades académicas y científicas. Existen numerosas formulaciones de control predictivo que utilizan las mismas ideas comunes: 1) utilización de un modelo explícito para predecir el comportamiento futuro de un sistema determinado, 2) minimización de una función de costo sujeta a ciertas restricciones en estados y entradas de control, y 3) utilización de una estrategia de horizonte deslizante.

En su versión más clásica, MPC utiliza modelos de sistemas lineales, discretos e invariantes en el tiempo, para predecir el comportamiento futuro de un sistema mediante la resolución *online* de un problema de control óptimo [17, 42]. La secuencia de control óptimo, válida para un determinado horizonte de control t^{h_c}, se calcula mediante la minimización de una función objetivo sujeta a restricciones en estados y entradas. Además, se asume que la secuencia de control se mantiene constante para $t^{h_p} \geq t \geq t^{h_c}$. (ver Fig. 3.1). La entrada de control óptima obtenida correspondiente al primer instante de muestreo es la que se aplica al sistema y el cálculo se reinicia desplazando el horizonte de predicción hacia adelante al próximo instante de muestreo. La técnica MPC

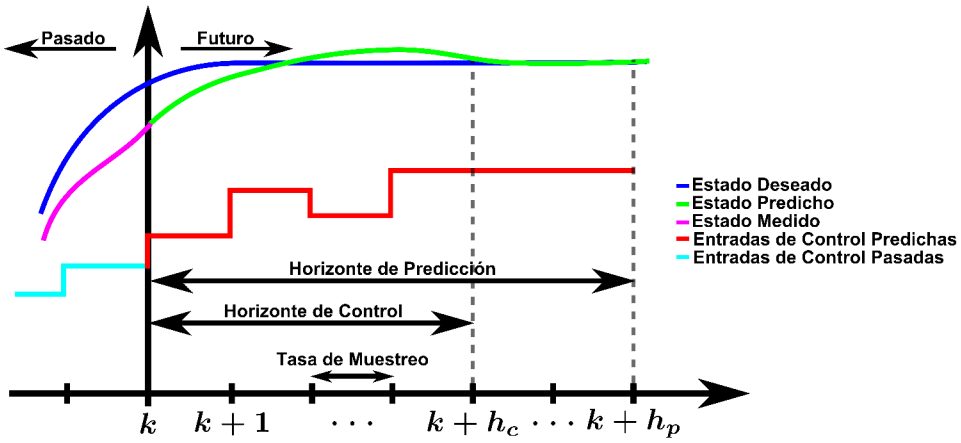


Figura 3.1: Conceptos básicos del control MPC

permite incorporar al problema de control modelos discretos en espacio de estados multivariables [43] y la inclusión de las limitaciones de los sistemas físicos se puede realizar de forma sencilla.

En la actualidad, el control predictivo tiene un amplio espectro de aplicaciones, como en la industria química [44] por citar alguna. Recientemente, ha

comenzado a utilizarse en sistemas de control automático para UAVs autónomos [45, 46, 47, 48].

3.2. Predicción de Estados

Para resolver el problema de control predictivo es necesario primero computar los valores predichos de los estados del sistema. Para ello, se asume que en el instante de tiempo $t = t^0$ el sistema se encuentra en un estado arbitrario $(\mathbf{x}^0, \mathbf{u}^0)$ y que, además, el mismo va a ser controlado por una serie de h_p controles consecutivos constantes por tramos (Ver Fig. 3.2)

$$\mathbf{U}_p = \begin{bmatrix} \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^k \cdots & \mathbf{u}^{h_p} \end{bmatrix}^T \tag{3.1}$$

durante un período de tiempo $\left[t^0, t^{h_p}\right]$. Cada control $\mathbf{u}^k$ en la Ec. (3.1) se aplica durante un intervalo de tiempo $\left[t^{k-1}, t^k\right]$ de longitud ΔT_s, con $k = 1, 2, \cdots, h_p$.

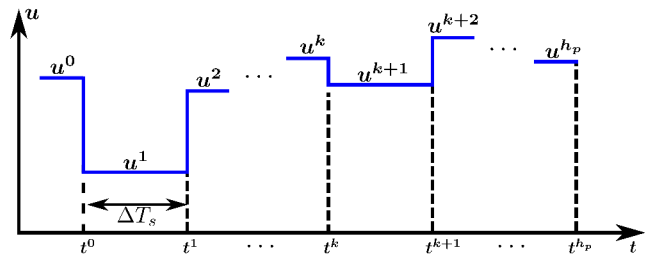


Figura 3.2: Esquema de discretización temporal

Los estados predichos $\mathbf{x}^k$ a lo largo de los instantes de tiempo consecutivos t^k con $k = 1, \ldots, h_p$ se pueden hallar iterando la Ec. (2.6). Teniendo en cuenta que para $k = h_c + 1, \cdots, h_p$ los valores de la secuencia de entradas de control $\mathbf{U}_p$ toman el valor $\mathbf{U}_p(k) = \mathbf{u}^{h_c}$, se puede demostrar que el vector de estados predichos

$$\hat{\mathbf{X}} = \begin{bmatrix} \mathbf{x}^1 & \mathbf{x}^2 & \cdots & \mathbf{x}^k & \cdots & \mathbf{x}^{h_p} \end{bmatrix}^T \tag{3.2}$$

puede calcularse en términos de la secuencia efectiva de control

$$\mathbf{U}_c = \begin{bmatrix} \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^k \cdots & \mathbf{u}^{h_c} \end{bmatrix}^T \tag{3.3}$$

como

$$\hat{\mathbf{X}} = \mathbf{P}\mathbf{x}^0 + \mathbf{H}_u\mathbf{U}_c \tag{3.4}$$

donde

$$\mathbf{P} = \begin{bmatrix} \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}}^2 \\ \tilde{\mathbf{A}}^3 \\ \vdots \\ \tilde{\mathbf{A}}^{h_c} \\ \vdots \\ \tilde{\mathbf{A}}^{h_p} \end{bmatrix} \text{ y } \mathbf{H}_u = \begin{bmatrix} \tilde{\mathbf{B}} & \mathbf{0} & \cdots & \mathbf{0} \\ \tilde{\mathbf{A}}\tilde{\mathbf{B}} & \tilde{\mathbf{B}} & \cdots & \mathbf{0} \\ \tilde{\mathbf{A}}^2\tilde{\mathbf{B}} & \tilde{\mathbf{A}}\tilde{\mathbf{B}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{A}}^{h_c-1}\tilde{\mathbf{B}} & \tilde{\mathbf{A}}^{h_c-2}\tilde{\mathbf{B}} & \cdots & \tilde{\mathbf{B}} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{A}}^{h_p-1}\tilde{\mathbf{B}} & \tilde{\mathbf{A}}^{h_p-2}\tilde{\mathbf{B}} & \cdots & \sum_{j=h_c}^{h_p} \tilde{\mathbf{A}}^{h_p-j}\tilde{\mathbf{B}} \end{bmatrix} \tag{3.5}$$

donde $\tilde{\mathbf{A}}^n$ representa a la potencia n-*ésima* de la matriz $\tilde{\mathbf{A}}$. $\mathbf{P}$ es una matriz de dimensión $(h_p \times N_s) \times N_s$, $\mathbf{H}_u$ es una matriz de dimensión $(h_p \times N_s) \times (h_c \times N_i)$.

Notar que mientras $\mathbf{U}_p$ es un vector de dimensión $h_p \times N_i$ que contiene la secuencia completa de controles, $\mathbf{U}_c$ es un vector de dimensión $h_c \times N_i$ que contiene los controles computados durante el horizonte de control efectivo $(t = [t^0, t^{h_c}])$.

3.3. Restricciones

En la práctica, todos los sistemas físicos presentan restricciones. Por ejemplo, los actuadores tienen un campo limitado de acción debido a sus límites físicos. Existen también límites de seguridad como pueden ser velocidades y temperaturas máximas. MPC provee una metodología que permite incorporar las restricciones de forma sistemática en la fase de diseño del controlador.

Se pueden plantear restricciones que involucren a: entradas de control, sus velocidades de cambio y estados del sistema. En tiempo continuo, dichas restricciones pueden ser expresadas como:

$$\mathbf{u}_m(t) \leq \mathbf{u}(t) \leq \mathbf{u}_M(t) \tag{3.6}$$

$$\dot{\mathbf{u}}_m(t) \leq \dot{\mathbf{u}}(t) \leq \dot{\mathbf{u}}_M(t) \tag{3.7}$$

$$\mathbf{x}_m(t) \leq \mathbf{x}(t) \leq \mathbf{x}_M(t) \tag{3.8}$$

donde $(\mathbf{u}_m(t), \mathbf{u}_M(t))$ representan las restricciones min-max en los controles, $(\dot{\mathbf{u}}_m(t), \dot{\mathbf{u}}_M(t))$ representan las restricciones min-max en la velocidad de cambio de las acciones de control y $(\mathbf{x}_m(t), \mathbf{x}_M(t))$ representan las restricciones min-max en los estados.

Si las Ecs. (3.6)-(3.8) se evalúan en los instantes de muestreo $t = t^k$ con $k = 1, \cdots, h_c$, las restricciones en entradas de control, velocidad de cambio de entradas de control y estados, respectivamente, pueden expresarse como:

$$\mathbf{u}_m^k \leq \mathbf{u}^k \leq \mathbf{u}_M^k \quad k = 1, \cdots, h_c \tag{3.9}$$

$$\dot{\mathbf{u}}_m^k \leq \dot{\mathbf{u}}^k \leq \dot{\mathbf{u}}_M^k \quad k = 1, \cdots, h_c \tag{3.10}$$

y

$$\mathbf{x}_m^k \leq \mathbf{x}^k \leq \mathbf{x}_M^k \quad k = 1, \cdots, h_c \tag{3.11}$$

La Ec. (3.9) puede escribirse en forma matricial como:

$$\begin{bmatrix} \mathbf{I}_c \\ -\mathbf{I}_c \end{bmatrix} \mathbf{U}_c \leq \begin{bmatrix} \mathbf{U}_M \\ -\mathbf{U}_m \end{bmatrix} \tag{3.12}$$

donde $\mathbf{U}_M$ y $\mathbf{U}_m$ son vectores que contienen los límites superior e inferior de las entradas de control, respectivamente, e $\mathbf{I}_c$ es una matriz identidad de dimensión $(N_i \times h_c) \times (N_i \times h_c)$.

De forma similar, evaluando para $k = 1, \cdots, h_c$, la Ec. (3.10) se puede escribir como sigue:

$$\dot{\mathbf{U}}_m \leq \frac{\Delta \mathbf{U}_c}{\Delta T_s} \leq \dot{\mathbf{U}}_M \tag{3.13}$$

donde $\dot{\mathbf{U}}_M$ y $\dot{\mathbf{U}}_m$ son vectores que contienen los límites superior e inferior de la velocidad de cambio de las entradas de control, respectivamente, y donde:

$$\Delta \mathbf{U}_c = \mathbf{E}\mathbf{U}_c + \mathbf{U}_0 \tag{3.14}$$

con

$$\mathbf{E} = \begin{bmatrix} \mathbf{I}_{N_i} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ -\mathbf{I}_{N_i} & \mathbf{I}_{N_i} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N_i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{I}_{N_i} & \mathbf{I}_{N_i} \end{bmatrix} \text{ y } \mathbf{U}_0 = \begin{bmatrix} -\mathbf{u}^0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \tag{3.15}$$

e $\mathbf{I}_{N_i}$ es una matriz identidad de dimensión $N_i \times N_i$. Luego, las restricciones definidas en la Ec. (3.13) pueden expresarse en forma matricial como sigue:

$$\begin{bmatrix} \mathbf{E} \\ -\mathbf{E} \end{bmatrix} \mathbf{U}_c \leq \begin{bmatrix} \dot{\mathbf{U}}_M \\ -\dot{\mathbf{U}}_m \end{bmatrix} \Delta T_s + \begin{bmatrix} -\mathbf{U}_0 \\ \mathbf{U}_0 \end{bmatrix} \tag{3.16}$$

Utilizando la Ec. (3.4), las restricciones en los estados definidas en la Ec. (3.11) se pueden escribir en forma matricial como:

$$\begin{bmatrix} \mathbf{H}_u \\ -\mathbf{H}_u \end{bmatrix} \mathbf{U}_c \leq \begin{bmatrix} \mathbf{X}_M \\ -\mathbf{X}_m \end{bmatrix} + \begin{bmatrix} -\mathbf{P} \\ \mathbf{P} \end{bmatrix} \mathbf{x}^0 \tag{3.17}$$

donde $\mathbf{X}_M$ y $\mathbf{X}_m$ son vectores que contienen los límites superior e inferior de los estados, respectivamente.

Finalmente, juntando las Ecs. (3.12), (3.16) y (3.17), las restricciones se pueden agrupar como

$$\mathbf{A}_{\text{ineq}} \mathbf{U}_c \leq \mathbf{b}_{\text{ineq}} \tag{3.18}$$

donde

$$\mathbf{A}_{\text{ineq}} = \begin{bmatrix} \mathbf{I}_c \\ -\mathbf{I}_c \\ \mathbf{E} \\ -\mathbf{E} \\ \mathbf{H}_u \\ -\mathbf{H}_u \end{bmatrix}, \ \mathbf{b}_{\text{ineq}} = \begin{bmatrix} \mathbf{U}_M \\ -\mathbf{U}_m \\ \dot{\mathbf{U}}_M \Delta T_s - \mathbf{U}_0 \\ -\dot{\mathbf{U}}_m \Delta T_s + \mathbf{U}_0 \\ \mathbf{X}_M - \mathbf{P}\mathbf{x}^0 \\ -\mathbf{X}_m + \mathbf{P}\mathbf{x}^0 \end{bmatrix} \tag{3.19}$$

3.4. Función Objetivo Cuadrática

Como se ha mencionado anteriormente, MPC minimiza una función objetivo o función de costo $\mathscr{J}(\mathbf{x}, \mathbf{u})$. Dicha función a su vez penaliza las desviaciones de los estados predichos $\mathbf{x}$ respecto de los valores de estados deseados $\mathbf{x}_d$ y también puede penalizar variaciones en las entradas de control $\mathbf{u}$. La función de costo que se utiliza generalmente es:

$$\mathscr{J} = \sum_{k=1}^{h_p} (\mathbf{x}_d^k - \mathbf{x}^k)^T \tilde{\mathbf{Q}}_x^k (\mathbf{x}_d^k - \mathbf{x}^k) + \sum_{k=1}^{h_c} \Delta \mathbf{u}^{k^T} \tilde{\mathbf{R}}_{\dot{u}}^k \Delta \mathbf{u}^k \tag{3.20}$$

donde $\tilde{\mathbf{Q}}_x$ y $\tilde{\mathbf{R}}_{\dot{u}}$ son matrices definidas semi-positivas y positivas, respectivamente y $\Delta \mathbf{u}^k = \mathbf{u}^k - \mathbf{u}^{k-1}$. El vector $\mathbf{x}_d^k$ especifica los valores deseados de las variables de estado en el instante de tiempo $t = t^k$. Entonces, planteado como un problema de minimización, el problema de control óptimo del método MPC puede escribirse como:

$$\begin{aligned} \min_{\mathbf{U}_c} \ \mathscr{J} &= \textstyle\sum_{k=1}^{h_p} (\mathbf{x}_d^k - \mathbf{x}^k)^T \tilde{\mathbf{Q}}_x^k (\mathbf{x}_d^k - \mathbf{x}^k) + \sum_{k=1}^{h_c} \Delta \mathbf{u}^{k^T} \tilde{\mathbf{R}}_{\dot{u}}^k \Delta \mathbf{u}^k \\ & st. \quad \mathbf{A}_{\text{ineq}} \mathbf{U}_c \leq \mathbf{b}_{\text{ineq}} \end{aligned} \tag{3.21}$$

La solución del problema de minimización (3.21) da como resultado la secuencia de entradas de control óptima $\mathbf{U}_c^* = [\mathbf{u}^1 \quad \mathbf{u}^2 \quad \cdots \quad \mathbf{u}^{h_c}]^T$ que minimiza tanto las desviaciones de los estados respecto de sus valores deseados, como así también la velocidad de cambio de las entradas de control.

3.5. Metodología de Control

El método MPC puede implementarse en una unidad de procesamiento para formar el sistema de control de un sistema físico arbitrario como se muestra en la Fig. 3.3. Brevemente, la operación *online* de la misma es como se indica a continuación. Primero, se miden el vector de estado y de entradas de control actuales del sistema físico real $\mathbf{x}^0$ y $\mathbf{u}^0$, respectivamente. Luego, utilizando un modelo matemático del sistema a controlar, el control MPC computa la secuencia efectiva de control óptima $\mathbf{U}_c^*$ resolviendo la Ec. (3.21). Como la

función objetivo $\mathscr{J}$ penaliza las desviaciones del vector de estados del modelo del sistema lineal respecto de su vector de estados deseado $\mathbf{x}_d$, la secuencia de control $\mathbf{U}_c^*$ hallada es aquella que minimiza dichas desviaciones. Solamente la entrada de control $\mathbf{u}^1 = \mathbf{U}_c^*(1)$ correspondiente al primer intervalo de muestreo se aplica al sistema físico real. El proceso continúa desplazando el horizonte de tiempo un paso hacia adelante al próximo instante de muestreo $t^0 \leftarrow t^0 + \Delta T_s$.

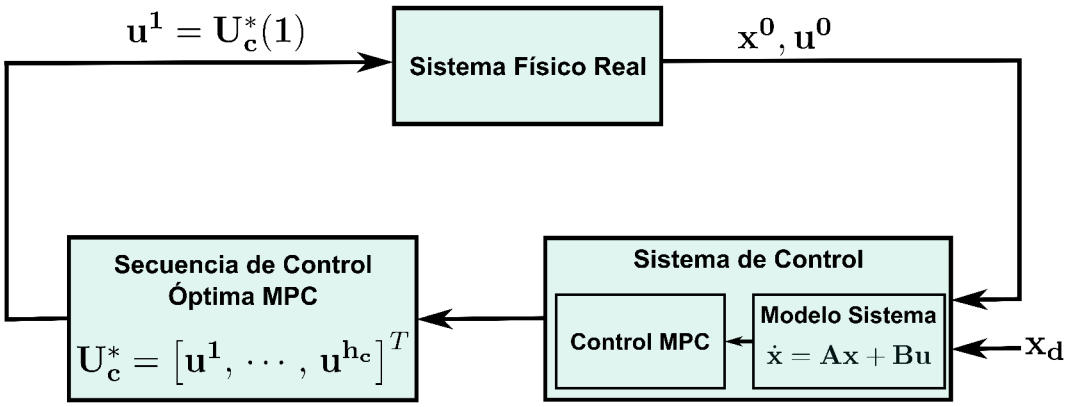


Figura 3.3: Esquema del lazo de control MPC

3.6. Algoritmo

En el Cuadro 3.1 se formaliza el algoritmo MPC detallándose los pasos que involucra el mismo.

3.7. Conclusiones

En este capítulo se realizó una revisión del método MPC, ya que sobre la base de esta técnica de control se trabajará en los capítulos siguientes. Se presentó el mecanismo que permite predecir el comportamiento futuro del sistema mediante la obtención del vector de estados predicho. Además, se planteó el problema de optimización como un problema de minimización cuadrático sujeto a restricciones en estados, entradas de control y velocidad de cambio de

Cuadro 3.1: Algoritmo MPC

Algoritmo
Paso 1: Inicializar toda la secuencia de entradas de control $\mathbf{U}_c$ definida en la Ec. (3.3) con el valor de la entrada de control actual $\mathbf{u}^0$.
Paso 2: Dados el estado actual del sistema físico real $\mathbf{x}^0$ y la secuencia de entradas de control $\mathbf{U}_c$, obtener el vector de estados predichos $\hat{\mathbf{X}}$ como se muestra en la Ec. (3.4)
Paso 3: Computar las matrices de predicción definidas en la Ec. (3.5).
Paso 4: Calcular las matrices de restricciones de Ec. (3.19) y resolver el problema de minimización definido en la Ec. (3.21).
Paso 5: Computar la entrada de control óptima $\mathbf{U}_c^*$ con un algoritmo de programación cuadrática adecuado.
Paso 6: Aplicar la primer entrada de control $\mathbf{u}^1 = \mathbf{U}_c^*(1)$ al sistema físico real.
Paso 7: Actualizar la secuencia de entradas de control $\mathbf{U}_c = \mathbf{U}_c^*$, mover el horizonte de tiempo un paso adelante hasta el próximo instante de muestreo $t^0 \leftarrow t^0 + \Delta T_s$ y regresar al **Paso 2**

entradas de control. La función objetivo utilizada penaliza tanto las desviaciones de los estados respecto de sus valores deseados como la velocidad de cambio de las entradas de control. Finalmente, se formalizó el algoritmo MPC detallando cada uno de los pasos involucrados por el mismo.

Capítulo 4

Un Nuevo Método de Control Predictivo para Sistemas no-Lineales (INL-MPC)

En este capítulo se presenta un nuevo método de control predictivo óptimo e iterativo para sistemas no-lineales, desarrollado en el marco de la presente tesis y denominado INL-MPC (sigla derivada de su nombre en inglés *Iterative non-Linear Model Predictive Control*).

Este capítulo se organiza como sigue: en la sección 4.1 se introduce el método INL-MPC. En la sección 4.2 se presenta el concepto de trayectoria en espacio de estados predicha. La sección 4.3 presenta el mecanismo de linealización generalizado a lo largo de las trayectorias en espacio de estados predichas, obteniéndose además el modelo en espacio de estados discreto del sistema linealizado. Asimismo, se describe la metodología utilizada para expresar las restricciones del sistema como una desigualdad matricial. En la sección 4.4 se presenta la función de costo utilizada en INL-MPC y se halla la relación existente entre la secuencia completa de control y la secuencia de control efectiva. En la sección 4.5 se presenta el funcionamiento de la unidad de control basada en el método INL-MPC. Finalmente, en la sección 4.6 se formaliza el algoritmo INL-MPC desarrollado, detallándose los pasos involucrados por el mismo.

4.1. Introducción a la Técnica INL-MPC

INL-MPC es un método general que extiende las capacidades del control MPC con el objetivo de controlar sistemas no-lineales [27]. El método propuesto transforma el problema no-lineal de control óptimo de un sistema no-lineal en una serie de sub-problemas de optimización cuadráticos, iterativos y más sencillos de resolver. El mismo utiliza un proceso de linealización generalizado que se realiza alrededor de trayectorias candidatas definidas iterativamente en el espacio de estados. La metodología planteada no requiere la definición de estados de equilibrio. La técnica INL-MPC puede verse como una sucesión iterativa de métodos MPC variantes en el tiempo.

4.2. Trayectorias en Espacio de Estados Predichas

Dado un sistema físico real cualquiera, se asume que el mismo en un instante de tiempo t^0 se encuentra en un estado arbitrario $\mathbf{x}^0$ definido por las acciones de control $\mathbf{u}^0$. Se asume ahora que dicho sistema va a ser controlado por una secuencia temporal de h_p controles consecutivos constantes por tramos (Ver Fig. 3.2)

$$\mathbf{U}_p = \begin{bmatrix} \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^k \cdots & \mathbf{u}^{h_p} \end{bmatrix}^T \tag{4.1}$$

durante un período de tiempo $\left[t^0, t^{h_p}\right]$.

Con el sistema físico en el estado inicial $\mathbf{x}^0$ y dada la primer entrada de control $\mathbf{U}_p(1) = \mathbf{u}^1$, la Ec. (2.7) puede integrarse utilizando, por ejemplo, el algoritmo de integración Runge-Kutta presentado en la sección 2.5, para encontrar el nuevo estado del sistema $\mathbf{x}^1$ en el instante de tiempo t^1. Iterativamente, los estados subsiguientes $\mathbf{x}^k$ asociados a la secuencia de control $\mathbf{U}_p$, definida en Ec. (4.1), pueden ser determinados para encontrar el vector total de estados predichos:

$$\hat{\mathbf{X}} = \begin{bmatrix} \mathbf{x}^1 & \mathbf{x}^2 & \cdots & \mathbf{x}^k & \cdots & \mathbf{x}^{h_p} \end{bmatrix}^T \tag{4.2}$$

Esto define una trayectoria en espacio de estados $T(\mathbf{U}_p)$, como se muestra en

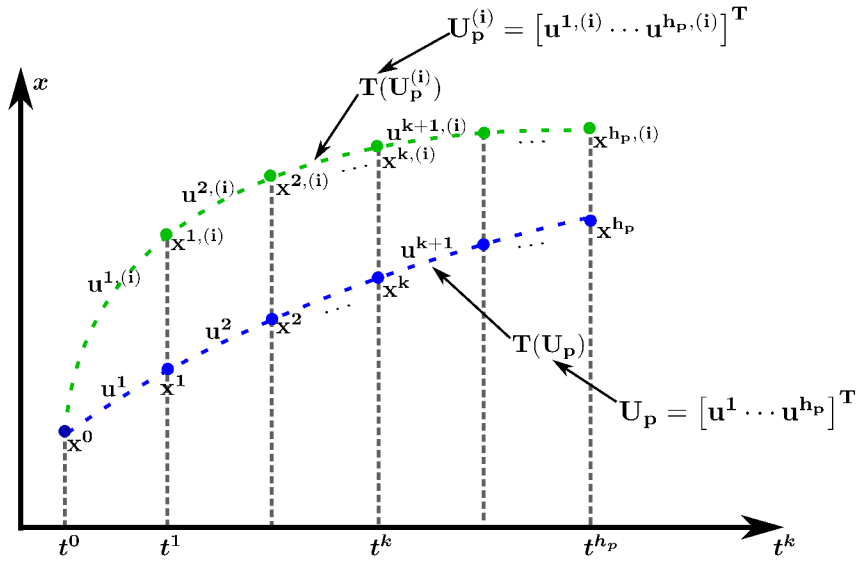


Figura 4.1: Predicción de trayectorias en espacio de estados

la Fig. 4.1. Por supuesto, una secuencia de control $\mathbf{U}_p^{(i)}$ diferente:

$$\mathbf{U}_p^{(i)} = \begin{bmatrix}\mathbf{u}^{1,(i)} & \mathbf{u}^{2,(i)} & \cdots & \mathbf{u}^{k,(i)} & \cdots & \mathbf{u}^{h_p,(i)}\end{bmatrix}^T \tag{4.3}$$

generará una trayectoria en espacio de estados diferente $T(\mathbf{U}_p^{(i)})$ con su correspondiente vector de estados predichos $\hat{\mathbf{X}}^{(i)}$:

$$\hat{\mathbf{X}}^{(i)} = \begin{bmatrix}\mathbf{x}^{1,(i)} & \mathbf{x}^{2,(i)} & \cdots & \mathbf{x}^{k,(i)} & \cdots & \mathbf{x}^{h_p,(i)}\end{bmatrix}^T \tag{4.4}$$

Esto también se muestra en la Fig. 4.1.

4.3. Linealización Generalizada, Restricciones y Discretización Temporal

4.3.1. Linealización

La ecuación de estados no-lineal definida en Ec. (2.7) puede linealizarse expandiendo $\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ en serie de Taylor de primer orden (o de mayor orden [49]) alrededor de un punto de operación arbitrario, variable $(\mathbf{x}_e, \mathbf{u}_e)$ ([8, 17]),

donde $\mathbf{x}_e = \mathbf{x}_e(t)$ y $\mathbf{u}_e = \mathbf{u}_e(t)$:

$$\dot{\mathbf{x}} \approx \bar{\mathbf{f}}(\mathbf{x}_e, \mathbf{u}_e) + \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}}|_{(\mathbf{x}_e,\mathbf{u}_e)}(\mathbf{x} - \mathbf{x}_e) + \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}}|_{(\mathbf{x}_e,\mathbf{u}_e)}(\mathbf{u} - \mathbf{u}_e) \tag{4.5}$$

donde $\frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}}$ y $\frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}}$ son las matrices jacobianas del sistema evaluadas en $(\mathbf{x}_e, \mathbf{u}_e)$ y se definen como:

$$\frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_{12}} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_{12}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{12}}{\partial x_1} & \frac{\partial f_{12}}{\partial x_2} & \dots & \frac{\partial f_{12}}{\partial x_{12}} \end{bmatrix} \quad \text{y} \quad \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \dots & \frac{\partial f_1}{\partial u_4} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \dots & \frac{\partial f_2}{\partial u_4} \\ \vdots & \vdots & & \\ \frac{\partial f_{12}}{\partial u_1} & \frac{\partial f_{12}}{\partial u_2} & \dots & \frac{\partial f_{12}}{\partial u_4} \end{bmatrix} \tag{4.6}$$

Como se mencionó anteriormente, el punto de linealización $(\mathbf{x}_e, \mathbf{u}_e)$ es un punto de operación variable, arbitrario. Notar que $\bar{\mathbf{f}}(\mathbf{x}_e, \mathbf{u}_e)$ puede ser calculada para cada $(\mathbf{x}_e, \mathbf{u}_e)$ utilizando la Ec. (2.7) y puede tomar valores no nulos. Luego, si se definen

$$\mathbf{A} = \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}}|_{(\mathbf{x}_e,\mathbf{u}_e)},\ \mathbf{B} = \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}}|_{(\mathbf{x}_e,\mathbf{u}_e)},\ \mathbf{d} = \bar{\mathbf{f}}(\mathbf{x}_e, \mathbf{u}_e) - \mathbf{A}\mathbf{x}_e - \mathbf{B}\mathbf{u}_e \tag{4.7}$$

entonces, la Ec. (4.5) puede escribirse en forma compacta como

$$\dot{\mathbf{x}}(t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{d} \tag{4.8}$$

La solución general de la Ec. (4.8) es

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t^e)}\mathbf{x}(t^e) + \int_{t^e}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau + \int_{t^e}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{d}d\tau \tag{4.9}$$

Asumiendo un control constante $\mathbf{u}(\tau) = \mathbf{u}^{k+1}$ y dado el estado $\mathbf{x}^k$ al tiempo $t^e = t^k$, por integración de la Ec. (4.9), se puede determinar el estado $\mathbf{x}^{k+1}$ en el instante de tiempo $t = t^{k+1} = t^k + \Delta T_s$ como:

$$\mathbf{x}^{k+1} = \tilde{\mathbf{A}}\mathbf{x}^k + \tilde{\mathbf{B}}\mathbf{u}^{k+1} + \tilde{\mathbf{G}}\mathbf{d}^k \tag{4.10}$$

donde

$$\tilde{\mathbf{A}} = e^{\mathbf{A}\Delta T_s},\ \tilde{\mathbf{G}} = \int_0^{\Delta T_s} e^{\mathbf{A}\tau} d\tau,\ \tilde{\mathbf{B}} = \tilde{\mathbf{G}}\mathbf{B} \tag{4.11}$$

4.3.2. Linealización a lo Largo de Trayectorias en Espacio de Estados

A continuación se procede a mostrar el concepto de linealización generalizada alrededor de trayectorias candidatas en espacio de estados. Para ello, se asume que en el instante de tiempo t^0 el sistema físico se encuentra en un estado conocido $(\mathbf{x}^0, \mathbf{u}^0)$ y que durante el horizonte de predicción $[t^0, t^{h_p}]$ se moverá próximo a una dada trayectoria $T(\mathbf{U}_p^{(i)})$ definida por un vector de entradas de control $\mathbf{U}_p^{(i)}$ y su correspondiente vector de estados predichos $\hat{\mathbf{X}}^{(i)}$ (Ecs. (4.3) y (4.4)). Bajo estas hipótesis, se puede entonces linealizar la Ec. (2.7) en cada intervalo de tiempo $\left[t^k, t^{k+1}\right]$, usando como punto de linealización el set $(\mathbf{x}_e, \mathbf{u}_e, t^e) = (\mathbf{x}^{k,(i)}, \mathbf{u}^{k,(i)}, t^k)$ formado por los estados, entradas de control y el tiempo asociados a la trayectoria en espacio de estados predicha $T(\mathbf{U}_p^{(i)})$.

Este proceso permite formar para cada intervalo de tiempo $\left[t^k, t^{k+1}\right]$ un sistema de la forma de Ecs. (4.7)-(4.11) donde las matrices $\mathbf{A}$, $\mathbf{B}$ y $\mathbf{d}$ definidas en la Ec. (4.7) están dadas por:

$$\begin{aligned} &\mathbf{A}^k = \tfrac{\partial \bar{\mathbf{f}}}{\partial \mathbf{x}}|_{\mathbf{x}^{k,(i)},\mathbf{u}^{k,(i)}},\ \mathbf{B}^k = \tfrac{\partial \bar{\mathbf{f}}}{\partial \mathbf{u}}|_{\mathbf{x}^{k,(i)},\mathbf{u}^{k,(i)}} \\ &\mathbf{d}^k = \bar{\mathbf{f}}(\mathbf{x}^{k,(i)}, \mathbf{u}^{k,(i)}) - \mathbf{A}^k\mathbf{x}^{k,(i)} - \mathbf{B}^k\mathbf{u}^{k,(i)} \end{aligned} \tag{4.12}$$

y donde el sistema de Ec. (4.10) queda expresado como:

$$\mathbf{x}^{k+1} = \tilde{\mathbf{A}}^k\mathbf{x}^k + \tilde{\mathbf{B}}^k\mathbf{u}^{k+1} + \tilde{\mathbf{G}}^k\mathbf{d}^k \tag{4.13}$$

con

$$\tilde{\mathbf{A}}^k = e^{\mathbf{A}^k\Delta T_s},\ \tilde{\mathbf{G}}^k = \int_0^{\Delta T_s} e^{\mathbf{A}^k\tau} d\tau,\ \tilde{\mathbf{B}}^k = \tilde{\mathbf{G}}^k\mathbf{B}^k \tag{4.14}$$

Nótese que la Ec. (4.13) permite determinar el estado $\mathbf{x}^{k+1}$ en el instante de tiempo t^{k+1} dados el estado $\mathbf{x}^k$ en el instante de tiempo t^k y el control constante aplicado $\mathbf{u}^{k+1}$ (ver Fig. 3.2).

Iterando el modelo descripto por la Ec. (4.13) a lo largo de los instantes de tiempo consecutivos t^k con $k = 0, 1, \ldots, h_p - 1$, puede demostrarse que el vector de estados predichos $\hat{\mathbf{X}}$, definido en la Ec. (4.2), queda dado por:

$$\hat{\mathbf{X}} = \mathbf{P}\mathbf{x}^0 + \mathbf{H}_u\mathbf{U}_p + \mathbf{H}_g\mathbf{D} \tag{4.15}$$

donde las matrices $\mathbf{D}$, $\mathbf{P}$, $\mathbf{H}_u$ y $\mathbf{H}_g$ se calculan como sigue:

$$\mathbf{D} = \begin{bmatrix} \mathbf{d}^0 \\ \mathbf{d}^1 \\ \mathbf{d}^2 \\ \vdots \\ \mathbf{d}^{h_p-1} \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} \tilde{\mathbf{A}}^0 \\ \tilde{\mathbf{A}}^1\tilde{\mathbf{A}}^0 \\ \tilde{\mathbf{A}}^2\tilde{\mathbf{A}}^1\tilde{\mathbf{A}}^0 \\ \vdots \\ \tilde{\mathbf{A}}^{h_p-1} \cdot \ldots \cdot \tilde{\mathbf{A}}^0 \end{bmatrix} \tag{4.16}$$

$$\mathbf{H}_u = \begin{bmatrix} \tilde{\mathbf{B}}^0 & \mathbf{0} & \cdots & \mathbf{0} \\ \tilde{\mathbf{A}}^1\tilde{\mathbf{B}}^0 & \tilde{\mathbf{B}}^1 & \cdots & \mathbf{0} \\ \tilde{\mathbf{A}}^2\tilde{\mathbf{A}}^1\tilde{\mathbf{B}}^0 & \tilde{\mathbf{A}}^2\tilde{\mathbf{B}}^1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{A}}^{h_p-1} \cdots \tilde{\mathbf{A}}^0\tilde{\mathbf{B}}^0 & \tilde{\mathbf{A}}^{h_p-1} \cdots \tilde{\mathbf{A}}^2\tilde{\mathbf{B}}^1 & \cdots & \tilde{\mathbf{B}}^{h_p-1} \end{bmatrix} \tag{4.17}$$

$$\mathbf{H}_g = \begin{bmatrix} \tilde{\mathbf{G}}^0 & \mathbf{0} & \cdots & \mathbf{0} \\ \tilde{\mathbf{A}}^1\tilde{\mathbf{G}}^0 & \tilde{\mathbf{G}}^1 & \cdots & \mathbf{0} \\ \tilde{\mathbf{A}}^2\tilde{\mathbf{A}}^1\tilde{\mathbf{G}}^0 & \tilde{\mathbf{A}}^2\tilde{\mathbf{G}}^1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{A}}^{h_p-1} \cdots \tilde{\mathbf{A}}^0\tilde{\mathbf{G}}^0 & \tilde{\mathbf{A}}^{h_p-1} \cdots \tilde{\mathbf{A}}^2\tilde{\mathbf{G}}^1 & \cdots & \tilde{\mathbf{G}}^{h_p-1} \end{bmatrix} \tag{4.18}$$

4.3.3. Restricciones

Las restricciones en entradas, velocidad de cambio de entradas y estados se calculan de forma similar a la realizada en el capítulo anterior, es decir:

$$\mathbf{u}_m^k \leq \mathbf{u}^k \leq \mathbf{u}_M^k \quad k = 1, \cdots, h_p \tag{4.19}$$

$$\dot{\mathbf{u}}_m^k \leq \dot{\mathbf{u}}^k \leq \dot{\mathbf{u}}_M^k \quad k = 1, \cdots, h_p \tag{4.20}$$

y

$$\mathbf{x}_m^k \leq \mathbf{x}^k \leq \mathbf{x}_M^k \quad k = 1, \cdots, h_p \tag{4.21}$$

La Ec. (4.19) puede escribirse en forma matricial como:

$$\begin{bmatrix} \mathbf{I}_p \\ -\mathbf{I}_p \end{bmatrix} \mathbf{U}_p \leq \begin{bmatrix} \mathbf{U}_M \\ -\mathbf{U}_m \end{bmatrix} \tag{4.22}$$

donde $\mathbf{U}_M$ y $\mathbf{U}_m$ son vectores que contienen los límites superior e inferior de las entradas de control, respectivamente, e $\mathbf{I}_p$ es una matriz identidad de dimensión $(N_i \times h_p) \times (N_i \times h_p)$.

De forma similar, evaluando para $k = 1, \cdots, h_p$, la Ec. (4.20) se puede escribir como sigue:

$$\dot{\mathbf{U}}_m \leq \frac{\Delta \mathbf{U}_p}{\Delta T_s} \leq \dot{\mathbf{U}}_M \tag{4.23}$$

donde $\dot{\mathbf{U}}_M$ y $\dot{\mathbf{U}}_m$ son vectores que contienen los límites superior e inferior de las velocidades de cambio de las entradas de control, respectivamente, y donde

$$\Delta \mathbf{U}_p = \mathbf{E}\mathbf{U}_p + \mathbf{U}_0 \tag{4.24}$$

con

$$\mathbf{E} = \begin{bmatrix} \mathbf{I}_{N_i} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ -\mathbf{I}_{N_i} & \mathbf{I}_{N_i} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N_i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{I}_{N_i} & \mathbf{I}_{N_i} \end{bmatrix} \text{ y } \mathbf{U}_0 = \begin{bmatrix} -\mathbf{u}^0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \tag{4.25}$$

e $\mathbf{I}_{N_i}$ es una matriz identidad de dimensión $N_i \times N_i$. Luego, las restricciones de velocidad de cambio de entradas de control definidas en la Ec. (4.23) se pueden expresar, en forma matricial, como:

$$\begin{bmatrix} \mathbf{E} \\ -\mathbf{E} \end{bmatrix} \mathbf{U}_p \leq \begin{bmatrix} \dot{\mathbf{U}}_M \\ -\dot{\mathbf{U}}_m \end{bmatrix} \Delta T_s + \begin{bmatrix} -\mathbf{U}_0 \\ \mathbf{U}_0 \end{bmatrix} \tag{4.26}$$

Utilizando la definición del vector de estados predicho de Ec. (4.15), las restricciones en los estados de Ec. (4.21) se pueden escribir matricialmente

como:

$$\begin{bmatrix} \mathbf{H}_u \\ -\mathbf{H}_u \end{bmatrix} \mathbf{U}_p \leq \begin{bmatrix} \mathbf{X}_M \\ -\mathbf{X}_m \end{bmatrix} + \begin{bmatrix} -\mathbf{P} \\ \mathbf{P} \end{bmatrix} \mathbf{x}^0 + \begin{bmatrix} -\mathbf{H}_g \\ \mathbf{H}_g \end{bmatrix} \mathbf{D} \tag{4.27}$$

donde $\mathbf{X}_M$ y $\mathbf{X}_m$ son vectores que contienen los límites superior e inferior de los estados, respectivamente.

Agrupando las Ecs. (4.22), (4.26) y (4.27), las restricciones se pueden escribir en forma compacta como:

$$\mathbf{A}_{\text{ineq}} \mathbf{U}_p \leq \mathbf{b}_{\text{ineq}} \tag{4.28}$$

donde

$$\mathbf{A}_{\text{ineq}} = \begin{bmatrix} \mathbf{I}_p \\ -\mathbf{I}_p \\ \mathbf{E} \\ -\mathbf{E} \\ \mathbf{H}_u \\ -\mathbf{H}_u \end{bmatrix}, \ \mathbf{b}_{\text{ineq}} = \begin{bmatrix} \mathbf{U}_M \\ -\mathbf{U}_m \\ \dot{\mathbf{U}}_M \Delta T_s - \mathbf{U}_0 \\ -\dot{\mathbf{U}}_m \Delta T_s + \mathbf{U}_0 \\ \mathbf{X}_M - \mathbf{P}\mathbf{x}^0 - \mathbf{H}_g \mathbf{D} \\ -\mathbf{X}_m + \mathbf{P}\mathbf{x}^0 + \mathbf{H}_g \mathbf{D} \end{bmatrix} \tag{4.29}$$

4.4. Función Objetivo Cuadrática

De la misma manera que MPC, el método INL-MPC utiliza una función objetivo o función de costo $\mathscr{J}(\mathbf{x}, \mathbf{u})$ para determinar una secuencia de entradas de control óptima. Dicha función de costo $\mathscr{J}$ penaliza las desviaciones de los estados predichos $\mathbf{x}$ respecto de los estados deseados $\mathbf{x}_d$. También puede penalizar cambios en las entradas de control $\mathbf{u}$.

Una forma bastante general de la función objetivo $\mathscr{J}$ en tiempo continuo puede escribirse como:

$$\begin{aligned} \mathscr{J}(\mathbf{x}, \mathbf{u}) = \int_{t^0}^{t^{h_p}} \big[&(\mathbf{x}_d - \mathbf{x})^T \mathbf{Q}_x (\mathbf{x}_d - \mathbf{x}) + (\dot{\mathbf{x}}_d - \dot{\mathbf{x}})^T \mathbf{Q}_{\dot{x}} (\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \\ &(\mathbf{u}_d - \mathbf{u})^T \mathbf{R}_u (\mathbf{u}_d - \mathbf{u}) + (\dot{\mathbf{u}}_d - \dot{\mathbf{u}})^T \mathbf{R}_{\dot{u}} (\dot{\mathbf{u}}_d - \dot{\mathbf{u}}) \big] \, dt \end{aligned} \tag{4.30}$$

donde $\mathbf{Q}_x$ y $\mathbf{Q}_{\dot{x}}$ son matrices definidas semi-positivas y $\mathbf{R}_u$ y $\mathbf{R}_{\dot{u}}$ son matrices definidas positivas y $\mathbf{x}_d$, $\dot{\mathbf{x}}_d$, $\mathbf{u}_d$ y $\dot{\mathbf{u}}_d$ especifican los valores deseados de variables de estados, velocidad de cambio de variables de estados, entradas de control y

velocidad de cambio de entradas de control, respectivamente.

En este capítulo será suficiente considerar una función de costo reducida que solamente penalice las desviaciones de los estados y de la velocidad de cambio de las entradas de control. En consecuencia, las matrices $\mathbf{Q}_{\dot{x}}$ y $\mathbf{R}_{u}$ se asumen nulas. Adicionalmente, se desea obtener entradas de control con mínima variación, por lo tanto $\dot{\mathbf{u}}_d$ se adopta igual a cero. Con estos supuestos, la forma discreta de la Ec. (4.30) puede escribirse como:

$$\mathscr{J} = \sum_{k=1}^{h_p} [(\mathbf{x}_d^k - \mathbf{x}^k)^T \tilde{\mathbf{Q}}_x^k (\mathbf{x}_d^k - \mathbf{x}^k) + (\mathbf{u}^k - \mathbf{u}^{k-1})^T \tilde{\mathbf{R}}_{\dot{u}}^k (\mathbf{u}^k - \mathbf{u}^{k-1})] \tag{4.31}$$

donde

$$\tilde{\mathbf{Q}}_x^k = \Delta T_s \mathbf{Q}_x^k \text{ y } \tilde{\mathbf{R}}_{\dot{u}}^k = \frac{\mathbf{R}_{\dot{u}}^k}{\Delta T_s} \tag{4.32}$$

Utilizando la Ec. (4.24), la Ec. (4.31) puede escribirse en forma matricial como sigue

$$\mathscr{J}(\hat{\mathbf{X}}, \mathbf{U}_p) = (\mathbf{X}_d - \hat{\mathbf{X}})^T \check{\mathbf{Q}}_x (\mathbf{X}_d - \hat{\mathbf{X}}) + (\mathbf{E}\,\mathbf{U}_p + \mathbf{U}_0)^T \check{\mathbf{R}}_{\dot{u}} (\mathbf{E}\,\mathbf{U}_p + \mathbf{U}_0) \tag{4.33}$$

donde

$$\check{\mathbf{Q}}_x = \begin{bmatrix} \tilde{\mathbf{Q}}_x^1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}_x^2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \tilde{\mathbf{Q}}_x^{h_p} \end{bmatrix}, \check{\mathbf{R}}_{\dot{u}} = \begin{bmatrix} \tilde{\mathbf{R}}_{\dot{u}}^1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{R}}_{\dot{u}}^2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \tilde{\mathbf{R}}_{\dot{u}}^{h_p} \end{bmatrix} \tag{4.34}$$

y

$$\mathbf{X}_d = \begin{bmatrix} \mathbf{x}_d^1 \, \mathbf{x}_d^2 \, \ldots \, \mathbf{x}_d^{h_p} \end{bmatrix}^T \tag{4.35}$$

Reemplazando la Ec. (4.15) en la Ec. (4.33), operando y descartando los términos constantes, la función objetivo de Ec. (4.33) puede escribirse como una función solamente de la secuencia de control como sigue:

$$\mathscr{J}(\mathbf{U}_p) = \mathbf{g}^T \mathbf{U}_p + \mathbf{U}_p^T \mathbf{H} \mathbf{U}_p \tag{4.36}$$

donde

$$\mathbf{g} = 2 \left[\mathbf{H}_u^T \check{\mathbf{Q}}_x (\mathbf{P}\mathbf{x}^0 + \mathbf{H}_g \mathbf{D} - \mathbf{X}_d) + \mathbf{E}^T \check{\mathbf{R}}_{\dot{u}} \mathbf{U}_0 \right] \tag{4.37}$$

y

$$\mathbf{H} = \mathbf{H}_u^T \check{\mathbf{Q}}_x \mathbf{H}_u + \mathbf{E}^T \check{\mathbf{R}}_{\dot{u}} \mathbf{E} \tag{4.38}$$

Al igual que MPC, el método INL-MPC asume que los valores de las entradas de control permanecen constantes una vez que se ha alcanzado el horizonte de control t^{h_c}. Esto significa que los valores $\mathbf{U}_p(k)$ de la secuencia de control $\mathbf{U}_p$ se mantienen igual a $\mathbf{u}^{h_c}$ para $k = h_c + 1, \cdots, h_p$. En consecuencia, es posible expresar la secuencia completa de control $\mathbf{U}_p$ en términos de la secuencia de control efectiva $\mathbf{U}_c$ como sigue:

$$\mathbf{U}_p = \check{\mathbf{T}} \mathbf{U}_c \tag{4.39}$$

donde el vector de secuencia efectiva de control se define como

$$\mathbf{U}_c = \left[\mathbf{u}^1 \, \mathbf{u}^2 \, \ldots \, \mathbf{u}^{h_c}\right]^T \tag{4.40}$$

y donde la matriz $\check{\mathbf{T}}$ está dada por:

$$\check{\mathbf{T}} = \begin{bmatrix} \mathbf{I}_c \\ \mathbf{V} \end{bmatrix} \tag{4.41}$$

donde

$$\mathbf{I}_c = \begin{bmatrix} \mathbf{I}_{N_i} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N_i} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N_i} \end{bmatrix} \text{ y } \mathbf{V} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N_i} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N_i} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N_i} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N_i} \end{bmatrix} \tag{4.42}$$

En las ecuaciones anteriores, $\mathbf{I}_{N_i}$ es una matriz identidad de dimensión $N_i \times N_i$; $\mathbf{I}_c$ es también una matriz identidad de dimensión $(N_i \times h_c) \times (N_i \times h_c)$ y $\mathbf{V}$ es una matriz de dimensión $(N_i \times (h_p - h_c)) \times (N_i \times h_c)$. Notar que, mientras $\mathbf{U}_p$ es un vector de dimensión $h_p \times N_i$ que contiene la secuencia completa de controles, $\mathbf{U}_c$ es un vector de dimensión $h_c \times N_i$ que contiene los controles aplicados durante el horizonte de control $(k = 1, \cdots, h_c)$.

La Ec. (4.39) permite expresar la función de costo $\mathscr{J}(\mathbf{U}_p)$ y las restric-

ciones en términos de la secuencia de control efectiva $\mathbf{U}_c$ (reemplazando la Ec. (4.39) en las Ecs. (4.36) y (4.28)). Entonces, planteado como un problema de minimización, el problema de control óptimo del método INL-MPC puede escribirse como:

$$\begin{aligned} \min_{\mathbf{U}_c} \ \mathscr{J}(\mathbf{U}_c) = \ & \mathbf{g}^T\check{\mathbf{T}}\mathbf{U}_c + \mathbf{U}_c^T\check{\mathbf{T}}^T\mathbf{H}\check{\mathbf{T}}\mathbf{U}_c \\ & st. \ \mathbf{A}_{\text{ineq}}\check{\mathbf{T}}\mathbf{U}_c \leq \mathbf{b}_{\text{ineq}} \end{aligned} \tag{4.43}$$

La solución del problema de minimización definido en Ec. (4.43) da como resultado la secuencia de entradas de control óptima $\mathbf{U}_c^*$ que minimiza tanto las desviaciones de los estados respecto de sus valores deseados, como así también la velocidad de cambio de las entradas de control.

4.5. Metodología de Control

El método INL-MPC puede implementarse en una unidad de procesamiento para formar el sistema de control de un sistema físico arbitrario como se muestra en la Fig. 4.2. Brevemente, la operación *online* de la misma es como se indica a continuación. En primer lugar, se miden el vector de estado y el vector de entradas de control actuales del sistema físico real $\mathbf{x}^0$ y $\mathbf{u}^0$. Luego, utilizando un modelo matemático del sistema a controlar, el método INL-MPC computa la secuencia de control óptimo efectiva $\mathbf{U}_c^*$ resolviendo la Ec. (4.43). Como la función objetivo $\mathscr{J}$ penaliza las desviaciones del vector de estados del sistema linealizado respecto de sus valores de estados deseados $\mathbf{x}_d$, la secuencia de control $\mathbf{U}_c^*$ hallada, es aquella que minimiza dichas desviaciones. Solamente la entrada de control $\mathbf{u}^1 = \mathbf{U}_c^*(1)$, correspondiente al primer intervalo de muestreo, se aplica al sistema físico real. El proceso continúa desplazando el horizonte de tiempo un paso hacia adelante al próximo instante de muestreo $t^0 \leftarrow t^0 + \Delta T_s$.

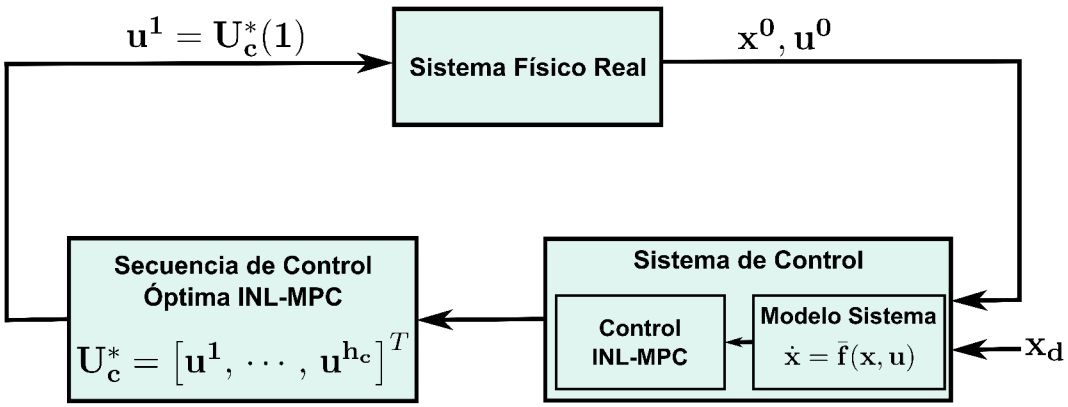


Figura 4.2: Esquema del lazo de control INL-MPC

4.6. Algoritmo

En el Cuadro 4.1 se formaliza el algoritmo INL-MPC, detallándose los pasos involucrados por el mismo.

4.7. Conclusiones

En este capítulo se presentó un nuevo método de control predictivo para sistemas no-lineales genéricos denominado *Iterative non-Linear Model Predictive Control* (INL-MPC). La técnica INL-MPC utiliza un proceso de linealización generalizado de la dinámica de los sistemas no-lineales, a lo largo de trayectorias candidatas definidas iterativamente en espacio de estados. INL-MPC presenta varias ventajas: 1) Permite la inclusión de sistemas no-lineales genéricos, 2) Puede ser utilizado como un sistema de control unificado que es capaz de controlar la dinámica completa de un sistema, como por ejemplo de un UAV, 3) Transforma el problema del control óptimo no-lineal de un sistema no-lineal en una serie de sub-problemas de optimización cuadráticos iterativos y más sencillos de resolver, y 4) Las restricciones pueden incorporarse con facilidad, además, las velocidades de cambio de estados y entradas de control pueden ser penalizados, permitiendo tener en cuenta diferentes limitaciones físicas de los sistemas reales.

Cuadro 4.1: Algoritmo INL-MPC

Algoritmo
Paso 1: Inicializar las variables *tol* y *maxIter* las cuales indican la tolerancia aceptada para terminar el lazo de iteración y el máximo número de iteraciones, respectivamente. Inicializar el índice de iteración $i = 1$.
Paso 2: Inicializar la secuencia de entradas de control $\mathbf{U}_p^{(i)}$ definida en la Ec. (4.3) con el valor de la entrada de control actual $\mathbf{u}^0$ y considerar que se mantiene constante a lo largo del intervalo de predicción $\left[t^0, t^{h_p}\right]$.
Paso 3: Dados el estado actual del sistema físico real $\mathbf{x}^0$ y la secuencia de entradas de control $\mathbf{U}_p^{(i)}$, obtener el vector de estados predichos $\hat{\mathbf{X}}^{(i)}$ definido en la Ec. (4.4) asociado a la trayectoria en espacio de estados predicha $T(\mathbf{U}_p^{(i)})$.
Paso 4: Linealizar el sistema no-lineal a lo largo de la trayectoria en espacio de estados predicha $T(\mathbf{U}_p^{(i)})$ para calcular las matrices de espacio de estado del sistema definidas en la Ec. (4.12). Luego, computar las matrices de predicción definidas en las Ecs. (4.16)-(4.18).
Paso 5: Calcular las matrices de la función objetivo $\mathscr{J}$ definidas en las Ecs. (4.37) y (4.38) y las matrices de restricciones de Ec. (4.29) a fin de establecer las matrices del problema de minimización de Ec. (4.43).
Paso 6: Computar la entrada de control óptima $\mathbf{U}_c^*$ con un algoritmo de programación cuadrática adecuado. Recordar que las entradas de control se mantienen constantes para $k > h_c$ por lo tanto la secuencia de entradas de control completa se puede calcular como $\mathbf{U}_p^* = \check{\mathbf{T}}\mathbf{U}_c^*$.
Paso 7: *Si,* $\left\|\mathbf{U}_p^* - \mathbf{U}_p^{(i)}\right\| \geq tol$ y $i \leq maxIter$ incrementar el índice de iteración $i \leftarrow i+1$, actualizar la secuencia de entradas de control $\mathbf{U}_p^{(i)} = \mathbf{U}_p^*$ para la próxima iteración y regresar al **Paso 3** *Si no*, terminar el lazo de iteración e ir al **Paso 8**.
Paso 8: Aplicar la primer entrada de control $\mathbf{u}^1 = \mathbf{U}_c^*(1)$ al sistema físico real, mover el horizonte de tiempo un paso hacia adelante al próximo instante de muestreo $t^0 \leftarrow t^0 + \Delta T_s$, reiniciar el índice de iteración i y regresar al **Paso 2**

Capítulo 5

Un Nuevo Método de Control Predictivo para Sistemas no-Lineales con Función de Costo no-Lineal (INL-MPC-J)

En este capítulo se presenta una nueva metodología que permite el tratamiento de funciones objetivos $\mathscr{J}$ no-lineales extendiendo la técnica INL-MPC presentada en el capítulo anterior. El nuevo método se denomina aquí INL-MPC-J en referencia a la inclusión de una función de costo no-lineal en el control INL-MPC.

Este capítulo se organiza como sigue: en la sección 5.1 se introduce la metodología propuesta. En la sección 5.2 se presenta la función de costo no-lineal genérica utilizada en el método INL-MPC-J. En la sección 5.3 se presenta el funcionamiento de la unidad de control basada en el método INL-MPC-J. Finalmente, en la sección 5.4 se formaliza el algoritmo INL-MPC-J detallándose los pasos que involucra el mismo.

5.1. Introducción al Método INL-MPC-J

En el caso de las técnicas MPC e INL-MPC, presentadas en los capítulos 3 y 4, respectivamente, las funciones objetivos utilizadas para obtener las secuencias de controles óptimas son funciones cuadráticas que penalizan las desviaciones de los estados y de la velocidad de cambio de las entradas de control respecto de sus valores deseados. Sin embargo, existen situaciones más complejas en las cuales se desea incluir en el proceso de optimización variables que no forman parte del vector de estados del sistema, pero que sí son funciones no-lineales del mismo. En este capítulo se presenta una metodología que permite considerar dichas variables. Para ello se plantea una función objetivo no-lineal que penaliza las desviaciones de funciones no-lineales del vector de estados y de la velocidad de cambio de las entradas de control respecto de sus valores deseados. Aplicando el proceso de linealización generalizado a dichas funciones, se logra transformar el problema de control óptimo no-lineal en un problema cuadrático, más sencillo de resolver y similar a los presentados en los capítulos anteriores.

5.2. Función Objetivo no-Lineal

La función objetivo $\mathscr{J}$ presentada en la Ec. (4.30) corresponde a una función de costo cuadrática que considera las desviaciones de: variables de estado, velocidad de cambio de variables de estado, entradas de control y velocidad de cambio de entradas de control respecto de sus valores deseados. Sin embargo, existen situaciones en las cuales es necesario utilizar una función objetivo más general que permita incorporar funciones no-lineales dependientes de dichas variables. En el marco de esta tesis será suficiente considerar una función objetivo reducida que solamente penalice las desviaciones de funciones no-lineales de variables de estado y de velocidad de cambio de entradas de control, respecto de sus valores deseados. En consecuencia, la función de costo $\mathscr{J}$ puede

escribirse como:

$$\begin{aligned}\mathscr{J}(\mathbf{x},\mathbf{u}) = \int_{t^0}^{t^{h_p}} \big[(\mathbf{c}_d - \mathscr{C}(\mathbf{x}))^T \mathbf{Q}_{\mathscr{C}} (\mathbf{c}_d - \mathscr{C}(\mathbf{x})) + \\ + (\mathbf{g}_d - \mathscr{G}(\dot{\mathbf{u}}))^T \mathbf{Q}_{\mathscr{G}} (\mathbf{g}_d - \mathscr{G}(\dot{\mathbf{u}}))\big]\, dt\end{aligned} \tag{5.1}$$

donde $\mathscr{C}(\mathbf{x})$ es una función no-lineal del vector de estados $\mathbf{x}$ y $\mathscr{G}(\dot{\mathbf{u}})$ es una función no-lineal de la velocidad de cambio de las entradas de control $\dot{\mathbf{u}}$. $\mathbf{c}_d$ y $\mathbf{g}_d$ especifican los valores deseados de $\mathscr{C}(\mathbf{x})$ y $\mathscr{G}(\dot{\mathbf{u}})$, respectivamente. $\mathbf{Q}_{\mathscr{C}}$ y $\mathbf{Q}_{\mathscr{G}}$ son matrices definidas semi-positiva y positiva, respectivamente.
Notación: para simplificar la notación, de aquí en adelante se omitirán las dependencias de $\mathbf{x}$ y $\dot{\mathbf{u}}$, es decir las funciones $\mathscr{C}(\mathbf{x})$ y $\mathscr{G}(\dot{\mathbf{u}})$ se notarán como $\mathscr{C}$ y $\mathscr{G}$, respectivamente.

La Ec. (5.1) se puede escribir en forma discreta como:

$$\begin{aligned}\mathscr{J}(\mathbf{x},\mathbf{u}) = \sum_{k=0}^{h_p} \big[(\mathbf{c}_d^k - \mathscr{C}^k)^T \mathbf{Q}_{\mathscr{C}}^k (\mathbf{c}_d^k - \mathscr{C}^k) + \\ + (\mathbf{g}_d^k - \mathscr{G}^k)^T \mathbf{Q}_{\mathscr{G}}^k (\mathbf{g}_d^k - \mathscr{G}^k)\big]\, \Delta T_s\end{aligned} \tag{5.2}$$

donde $\mathscr{C}^k = \mathscr{C}(\mathbf{x}^k)$ y $\mathscr{G}^k = \mathscr{G}(\dot{\mathbf{u}}^k)$.

Aplicando el proceso de linealización a lo largo de trayectorias en espacio de estados predichas, explicado en la sección 4.3.2, a la función no-lineal $\mathscr{C}$, la misma se puede aproximar como:

$$\mathscr{C} \approx \mathbf{C}_x^k \mathbf{x} + \mathbf{c}_0^k \tag{5.3}$$

donde

$$\mathbf{C}_x^k = \mathbf{C}_x^{k,(i)} = \frac{\partial \mathscr{C}}{\partial \mathbf{x}}\Big|_{\mathbf{x}^{k,(i)}} \text{ y } \mathbf{c}_0^k = \mathbf{c}_0^{k,(i)} = \mathscr{C}(\mathbf{x}^{k,(i)}) - \mathbf{C}_x^k \mathbf{x}^{k,(i)} \tag{5.4}$$

Se puede verificar fácilmente, que la versión discreta de la Ec. (5.3) es:

$$\mathscr{C}^k = \mathbf{C}_x^k \mathbf{x}^k + \mathbf{c}_0^k \tag{5.5}$$

Análogamente, se aplica el proceso de linealización generalizado a la función no-lineal $\mathscr{G}$, obteniéndose:

$$\mathscr{G} \approx \mathbf{G}_{\dot{u}}^k \dot{\mathbf{u}} + \mathbf{g}_0^k \tag{5.6}$$

donde

$$\mathbf{G}_{\dot{u}}^{k} = \mathbf{G}_{\dot{u}}^{k,(i)} = \frac{\partial \mathscr{G}}{\partial \dot{\mathbf{u}}}|_{\dot{\mathbf{u}}^{k,(i)}} \text{ y } \mathbf{g}_0^k = \mathbf{g}_0^{k,(i)} = \mathscr{G}(\dot{\mathbf{u}}^{k,(i)}) - \mathbf{G}_{\dot{u}}^{k}\dot{\mathbf{u}}^{k,(i)} \tag{5.7}$$

Puede verificarse fácilmente, que la versión discreta de la Ec. (5.6) es la siguiente:

$$\mathscr{G}^k = \mathbf{G}_{\dot{u}}^{k}\dot{\mathbf{u}}^k + \mathbf{g}_0^k \tag{5.8}$$

Ahora bien, $\dot{\mathbf{u}}^k \approx \frac{\mathbf{u}^k - \mathbf{u}^{k-1}}{\Delta Ts} = \frac{\Delta \mathbf{u}^k}{\Delta Ts}$, por lo tanto, la Ec. (5.6) se puede reescribir como:

$$\mathscr{G}^k = \mathbf{G}_{\dot{u}}^{k}\frac{\Delta \mathbf{u}^k}{\Delta T_s} + \mathbf{g}_0^k \tag{5.9}$$

Reemplazando las Ecs. (5.5) y (5.9) en la Ec. (5.2), se obtiene que:

$$\begin{aligned}\mathscr{J}(\mathbf{x},\mathbf{u}) = \textstyle\sum_{k=1}^{h_p} \Big[&(\mathbf{c}_d^k - \mathbf{C}_x^k\mathbf{x}^k - \mathbf{c}_0^k)^T\tilde{\mathbf{Q}}_{\mathscr{C}}^k(\mathbf{c}_d^k - \mathbf{C}_x^k\mathbf{x}^k - \mathbf{c}_0^k) + \\ &+ (\mathbf{g}_d^k\Delta T_s - \mathbf{G}_{\dot{u}}^k\Delta\mathbf{u}^k - \mathbf{g}_0^k\Delta T_s)^T\tilde{\mathbf{Q}}_{\mathscr{G}}^k(\mathbf{g}_d^k\Delta T_s - \mathbf{G}_{\dot{u}}^k\Delta\mathbf{u}^k - \mathbf{g}_0^k\Delta T_s)\Big]\end{aligned} \tag{5.10}$$

donde

$$\tilde{\mathbf{Q}}_{\mathscr{C}}^k = \Delta T_s\mathbf{Q}_{\mathscr{C}}^k \text{ y } \tilde{\mathbf{Q}}_{\mathscr{G}}^k = \frac{\mathbf{Q}_{\mathscr{G}}^k}{\Delta T_s} \tag{5.11}$$

En forma matricial, la Ec. (5.10) se puede escribir como:

$$\begin{aligned}\mathscr{J}(\hat{\mathbf{X}},\mathbf{U}_p) = &(\tilde{\mathbf{c}}_d - \tilde{\mathbf{C}}_x\hat{\mathbf{X}} - \tilde{\mathbf{c}}_0)^T\check{\mathbf{Q}}_{\mathscr{C}}(\tilde{\mathbf{c}}_d - \tilde{\mathbf{C}}_x\hat{\mathbf{X}} - \tilde{\mathbf{c}}_0) + \\ &+(\Delta T_s\tilde{\mathbf{g}}_d - \tilde{\mathbf{G}}_{\dot{u}}\Delta\mathbf{U}_p - \Delta T_s\tilde{\mathbf{g}}_0)^T\check{\mathbf{Q}}_{\mathscr{G}}(\Delta T_s\tilde{\mathbf{g}}_d - \tilde{\mathbf{G}}_{\dot{u}}\Delta\mathbf{U}_p - \Delta T_s\tilde{\mathbf{g}}_0)\end{aligned} \tag{5.12}$$

donde $\tilde{\mathbf{c}}_d = [\mathbf{c}_d^1 \ \mathbf{c}_d^2 \ \cdots \ \mathbf{c}_d^{h_p}]^T$, $\tilde{\mathbf{g}}_d = [\mathbf{g}_d^1 \ \mathbf{g}_d^2 \ \cdots \ \mathbf{g}_d^{h_p}]^T$, $\tilde{\mathbf{c}}_0 = [\mathbf{c}_0^1 \ \mathbf{c}_0^2 \ \cdots \ \mathbf{c}_0^{h_p}]^T$, $\tilde{\mathbf{g}}_0 = [\mathbf{g}_0^1 \ \mathbf{g}_0^2 \ \cdots \ \mathbf{g}_0^{h_p}]^T$ y

$$\tilde{\mathbf{C}}_x = \begin{bmatrix} \mathbf{C}_x^1 & 0 & \cdots & 0 \\ 0 & \mathbf{C}_x^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{C}_x^{h_p} \end{bmatrix} \quad \tilde{\mathbf{G}}_{\dot{u}} = \begin{bmatrix} \mathbf{G}_{\dot{u}}^1 & 0 & \cdots & 0 \\ 0 & \mathbf{G}_{\dot{u}}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{G}_{\dot{u}}^{h_p} \end{bmatrix} \tag{5.13}$$

y

$$\check{\mathbf{Q}}_{\mathscr{C}} = \begin{bmatrix} \tilde{\mathbf{Q}}_{\mathscr{C}}^{1} & 0 & \cdots & 0 \\ 0 & \tilde{\mathbf{Q}}_{\mathscr{C}}^{2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{\mathbf{Q}}_{\mathscr{C}}^{h_p} \end{bmatrix} \quad \check{\mathbf{Q}}_{\mathscr{G}} = \begin{bmatrix} \tilde{\mathbf{Q}}_{\mathscr{G}}^{1} & 0 & \cdots & 0 \\ 0 & \tilde{\mathbf{Q}}_{\mathscr{G}}^{2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{\mathbf{Q}}_{\mathscr{G}}^{h_p} \end{bmatrix} \tag{5.14}$$

Como se vio en el capítulo anterior, el vector de estados predicho $\hat{\mathbf{X}}$ y la variación de la secuencia de control $\Delta\mathbf{U}_p$ se pueden calcular como:

$$\hat{\mathbf{X}} = \mathbf{P}\mathbf{x}^0 + \mathbf{H}_u\mathbf{U}_p + \mathbf{H}_g\mathbf{D} \tag{5.15}$$

y

$$\Delta\mathbf{U}_p = \mathbf{E}\mathbf{U}_p + \mathbf{U}_0 \tag{5.16}$$

Finalmente, reemplazando las Ecs. (5.15) y (5.16) en la Ec. (5.12), operando y descartando los términos constantes, la Ec. (5.12) se puede escribir en forma cuadrática como una función paramétrica en las acciones de control, es decir:

$$\mathscr{J}(\mathbf{U}_p) = \mathbf{g}^T\mathbf{U}_p + \mathbf{U}_p^T\mathbf{H}\mathbf{U}_p \tag{5.17}$$

donde

$$\begin{aligned} \mathbf{g} = 2\Big\{&\mathbf{H}_u^T\Big[\hat{\mathbf{Q}}_{\mathscr{C}}(\mathbf{P}\mathbf{x}^0 + \mathbf{H}_g\mathbf{D}) + \tilde{\mathbf{C}}_x^T\check{\mathbf{Q}}_{\mathscr{C}}(\tilde{\mathbf{c}}_0 - \tilde{\mathbf{c}}_d)\Big] + \\ &+\mathbf{E}^T\Big[\tilde{\mathbf{G}}_{\dot{u}}^T\check{\mathbf{Q}}_{\mathscr{G}}\Delta T_s(\tilde{\mathbf{g}}_0 - \mathbf{g}_d) + \hat{\mathbf{Q}}_{\mathscr{G}}\mathbf{U}_0\Big]\Big\} \end{aligned} \tag{5.18}$$

y

$$\mathbf{H} = \mathbf{H}_u^T\hat{\mathbf{Q}}_{\mathscr{C}}\mathbf{H}_u + \mathbf{E}^T\hat{\mathbf{Q}}_{\mathscr{G}}\mathbf{E} \tag{5.19}$$

con $\hat{\mathbf{Q}}_{\mathscr{C}} = \tilde{\mathbf{C}}_x^T\check{\mathbf{Q}}_{\mathscr{C}}\tilde{\mathbf{C}}_x$ y $\hat{\mathbf{Q}}_{\mathscr{G}} = \tilde{\mathbf{G}}_{\dot{u}}^T\check{\mathbf{Q}}_{\mathscr{G}}\tilde{\mathbf{G}}_{\dot{u}}$.

El método INL-MPC-J también asume que los valores de las entradas de control permanecen constantes una vez que se ha alcanzado el horizonte de control t^{h_c}, es decir, que los valores $\mathbf{U}_p(k)$ de la secuencia de control $\mathbf{U}_p$ se mantienen igual a $\mathbf{u}^{h_c}$ para $k = h_c + 1, \cdots, h_p$. En consecuencia, es posible expresar la secuencia completa de control $\mathbf{U}_p$ en términos de la secuencia de control efectiva $\mathbf{U}_c$ como se hizo en la sección 4.4, es decir:

$$\mathbf{U}_p = \check{\mathbf{T}}\mathbf{U}_c \tag{5.20}$$

donde la matriz $\check{\mathbf{T}}$ fue definida en la Ec. (4.41) y la secuencia efectiva de control $\mathbf{U}_c$ fue definida en la Ec. (4.40).

Finalmente, planteado como un problema de minimización, el problema de control óptimo del método INL-MPC-J se puede expresar en términos de la secuencia de control efectiva $\mathbf{U}_c$ como:

$$\begin{aligned} \min_{\mathbf{U}_c} \ \mathscr{J}(\mathbf{U}_c) = \ & \mathbf{g}^T\check{\mathbf{T}}\mathbf{U}_c + \mathbf{U}_c^T\check{\mathbf{T}}^T\mathbf{H}\check{\mathbf{T}}\mathbf{U}_c \\ & st. \ \mathbf{A}_{\text{ineq}}\check{\mathbf{T}}\mathbf{U}_c \leq \mathbf{b}_{\text{ineq}} \end{aligned} \tag{5.21}$$

donde $\mathbf{A}_{\text{ineq}}$ y $\mathbf{b}_{\text{ineq}}$ son las matrices de restricciones definidas en la Ec. (4.29) de la sección 4.3.3. La solución del problema de minimización de la Ec. (5.21) da como resultado la secuencia de entradas de control óptima $\mathbf{U}_c^*$ que minimiza tanto las desviaciones de la función no-lineal de los estados $\mathscr{C}(\mathbf{x})$ respecto de sus valores deseados, como así también de la función no-lineal de la velocidad de cambio de las entradas de control $\mathscr{G}(\dot{\mathbf{u}})$ respecto de sus valores deseados.

5.3. Metodología de Control

El método INL-MPC-J puede implementarse en una unidad de procesamiento para formar el sistema de control de un sistema físico arbitrario como se muestra en la Fig. 5.1. Brevemente, la operación *online* de la misma es como se indica a continuación. En primer lugar, se miden el vector de estado y el vector de entradas de control actuales del sistema físico real $\mathbf{x}^0$ y $\mathbf{u}^0$, respectivamente. Luego, utilizando un modelo matemático del sistema a controlar, el control INL-MPC-J computa la secuencia de control óptimo efectiva $\mathbf{U}_c^*$ resolviendo la Ec. (5.21). Como la función objetivo $\mathscr{J}$ penaliza las desviaciones de la función no-lineal $\mathscr{C}(\mathbf{x})$ respecto de los valores deseados $\mathbf{c}_d$ y de la función no-lineal $\mathscr{G}(\dot{\mathbf{u}})$ respecto de los valores deseados $\mathbf{g}_d$, la secuencia de control $\mathbf{U}_c^*$ hallada, es aquella que minimiza dichas desviaciones. Solamente la entrada de control $\mathbf{u}^1 = \mathbf{U}_c^*(1)$, correspondiente al primer intervalo de muestreo, se aplica al sistema físico real. El proceso continúa desplazando el horizonte de tiempo un paso hacia adelante al próximo instante de muestreo $t^0 \leftarrow t^0 + \Delta T_s$.

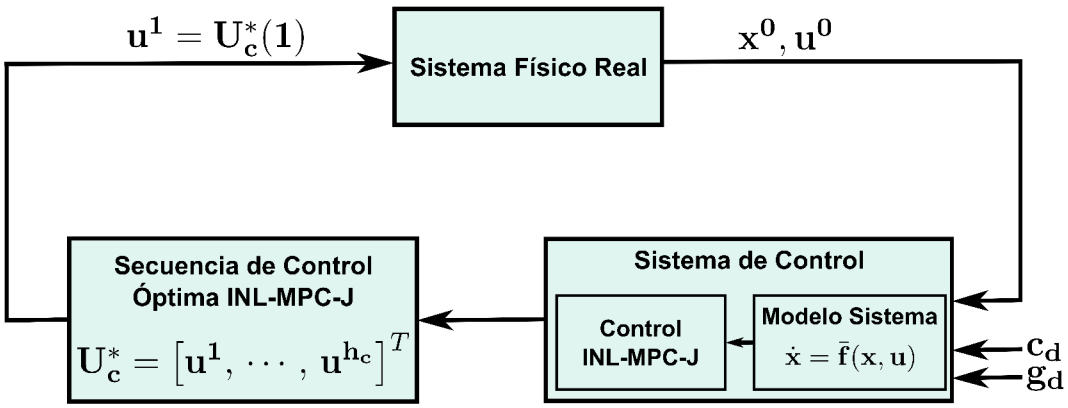


Figura 5.1: Esquema del lazo de control INL-MPC-J

5.4. Algoritmo

En el Cuadro 5.1 se formaliza el algoritmo INL-MPC-J, detallándose los pasos involucrados por el mismo.

5.5. Conclusiones

En este capítulo se presentó una nueva metodología que permite agregar al problema de control óptimo variables que no forman parte del vector de estados. Para ello se incluyeron en la función objetivo $\mathscr{J}$, funciones no-lineales de los estados y de la velocidad de cambio de las entradas de control, $\mathscr{C}(\mathbf{x})$ y $\mathscr{G}(\dot{\mathbf{u}})$, respectivamente, determinando así un problema de optimización no-lineal. Dicho problema se resuelve utilizando el concepto de linealización generalizada a lo largo de trayectorias en espacio de estados. Nuevamente, esto permite reducir el problema de control no-lineal a uno cuadrático similar a los estudiados en los capítulos anteriores.

Cuadro 5.1: Algoritmo INL-MPC-J

Algoritmo

Paso 1: Inicializar las variables *tol* y *maxIter* las cuales indican la tolerancia aceptada para terminar el lazo de iteración y el máximo número de iteraciones, respectivamente. Inicializar el índice de iteración $i = 1$.

Paso 2: Inicializar la secuencia de entradas de control $\mathbf{U}_p^{(i)}$ definida en la Ec. (4.3) con el valor de la entrada de control actual $\mathbf{u}^0$ y considerar que se mantiene constante a lo largo del intervalo de predicción $\left[t^0, t^{h_p}\right]$.

Paso 3: Dados el estado actual del sistema físico real $\mathbf{x}^0$ y la secuencia de entradas de control $\mathbf{U}_p^{(i)}$, obtener el vector de estados predichos $\hat{\mathbf{X}}^{(i)}$ definido en la Ec. (4.4) asociado a la trayectoria en espacio de estados predicha $T(\mathbf{U}_p^{(i)})$.

Paso 4: Linealizar el sistema no-lineal a lo largo de la trayectoria en espacio de estados predicha $T(\mathbf{U}_p^{(i)})$ para calcular las matrices de espacio de estado del sistema definidas en la Ec. (4.12). Luego, computar las matrices de predicción definidas en las Ecs. (4.16)-(4.18).

Paso 5: Linealizar las funciones $\mathscr{C}(\mathbf{x})$ y $\mathscr{G}(\dot{\mathbf{u}})$ de la función objetivo $\mathscr{J}$ a lo largo de la trayectoria en espacio de estados predicha $T(\mathbf{U}_p^{(i)})$. Calcular las matrices de la función objetivo $\mathscr{J}$ definidas en las Ecs. (5.18) y (5.19) y las matrices de restricciones de Ec. (4.29) a fin de establecer el problema de minimización definido en Ec. (5.21).

Paso 6: Computar la entrada de control óptima $\mathbf{U}_c^*$ con un algoritmo de programación cuadrática adecuado. Recordar que las entradas de control se mantienen constantes para $k > h_c$ por lo tanto la secuencia de entradas de control completa se puede calcular como $\mathbf{U}_p^* = \check{\mathbf{T}}\mathbf{U}_c^*$.

Paso 7: *Si,* $\left\|\mathbf{U}_p^* - \mathbf{U}_p^{(i)}\right\| \geq tol$ y $i \leq maxIter$ incrementar el índice de iteración $i \leftarrow i+1$, actualizar la secuencia de entradas de control $\mathbf{U}_p^{(i)} = \mathbf{U}_p^*$ para la próxima iteración y regresar al **Paso 3**

Si no, terminar el lazo de iteración e ir al **Paso 8**.

Paso 8: Aplicar la primer entrada de control $\mathbf{u}^1 = \mathbf{U}_c^*(1)$ al sistema físico real, mover el horizonte de tiempo un paso hacia adelante al próximo instante de muestreo $t^0 \leftarrow t^0 + \Delta T_s$, reiniciar el índice de iteración i y regresar al **Paso 2**.

Capítulo 6

Generación de Trayectorias de Navegación Utilizando la Técnica INL-MPC

Este capítulo presenta una novedosa metodología que permite el cálculo de trayectorias de navegación *online*, tanto en dos como en tres dimensiones (2D y 3D, respectivamente).

La organización del presente capítulo es la siguiente: en la sección 6.1 se introduce la temática. En la sección 6.2 se presentan los modelos de vehículos reducidos tipo partícula 2D y 3D, utilizados para la generación de trayectorias de navegación óptimas. En la sección 6.3 se expone la metodología propuesta para la generación de trayectorias de navegación mediante la utilización del método INL-MPC y los modelos de vehículos reducidos tipo partícula. Finalmente, en la sección 6.4 se detallan las conclusiones del capítulo.

6.1. Introducción

Los sistemas de navegación son aquellos que computan el camino o recorrido, es decir la trayectoria de navegación, que un determinado vehículo no-tripulado debe seguir. Dichos sistemas utilizan diversas metodologías para el cómputo de trayectorias de navegación, clasificándose principalmente, según

los algoritmos que utilicen, como *offline* y *online*. En el caso de los primeros, la trayectoria de navegación se computa en una etapa previa y la ruta no puede modificarse en presencia de ambientes dinámicos. Los algoritmos *online* son aquellos que permiten la modificación de las trayectorias de navegación en forma dinámica, por ejemplo en presencia de obstáculos, obteniéndose las mismas en tiempo de ejecución. A modo de ejemplo, se citan las referencias [28], [29] y [30]. En [28] se genera una trayectoria *offline* mediante la obtención de *splines* suaves utilizando técnicas de control óptimo y de programación matemática. El método se centra, en primer lugar, en la generación de curvas que pasen cerca de los *waypoints* (puntos de paso deseados) y luego se agregan las restricciones de interpolación a la formulación del problema. En el trabajo [29] se presenta una técnica para la generación de una trayectoria *online* partiendo de una serie *waypoints*. Empleando una metodología basada en la utilización de *B-Splines* es posible modificar la trayectoria de navegación en presencia de ambientes dinámicos. En [30], se presenta un algoritmo para generación de trayectorias *online* en dos dimensiones. El mismo propone que las transiciones de un segmento de trayectoria a otro se realicen en el menor tiempo posible. El cómputo de las trayectorias de navegación, para este caso, se realiza utilizando un proceso dinámico. Debe destacarse que los trabajos anteriormente mencionados no consideran la dinámica del vehículo no-tripulado que debe seguir la trayectoria de navegación.

En este capítulo se presenta un método *online* de generación de las trayectorias de navegación basado en la utilización de modelos de vehículos reducidos tipo partícula. La técnica propuesta se basa en el cómputo de una trayectoria de navegación a partir de un listado de *waypoints* por los cuales debe pasar la trayectoria de navegación deseada, considerando también, las restricciones impuestas por la dinámica propia del vehículo reducido tipo partícula. El problema de generación de trayectoria se plantea aquí como un problema de control óptimo no-lineal con restricciones, utilizándose la técnica de control INL-MPC presentada en el capítulo 4.

6.2. Modelado de Vehículos Reducidos tipo Partícula

Como se vio en el capítulo 2, la representación general de la dinámica de un sistema no-lineal arbitrario está dada por:

$$\dot{\mathbf{x}}(t) = \bar{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t)) \tag{6.1}$$

En particular, para el caso de una partícula, su modelo matemático es trivial y se reduce simplemente a:

$$\dot{\mathbf{x}} = \mathbf{v} \tag{6.2}$$

donde $\mathbf{x}$ es el vector de estado del vehículo reducido tipo partícula y está definido simplemente por su posición. A continuación se derivarán los modelos matemáticos para el caso de vehículos reducidos tipo partícula que se desplazan en el plano y en el espacio.

6.2.1. Vehículo Reducido tipo Partícula en 2D

En el caso de una partícula que se desplaza en el plano xy, su vector de estados $\mathbf{x}$ queda simplemente determinado por su posición, es decir

$$\mathbf{x} = [x_1 \quad x_2]^T = [x \quad y]^T \tag{6.3}$$

cuya dimensión es, por lo tanto, $N_s = 2$.

En la Fig. 6.1 se muestra un esquema del modelo 2D del vehículo reducido tipo partícula propuesto. Al descomponer el vector $\dot{\mathbf{x}}$ en sus componentes en los ejes x e y respectivamente, se obtiene que:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v\cos\delta \\ v\sin\delta \end{bmatrix} \tag{6.4}$$

donde v es el módulo del vector velocidad del vehículo y δ representa el ángulo de orientación en plano horizontal, determinado por el ángulo formado entre el vector velocidad $\mathbf{v}$ y el eje coordenado x. Adoptando el ángulo δ como entrada

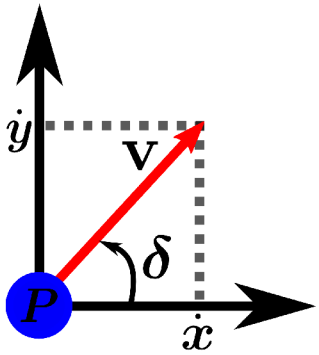


Figura 6.1: Modelo 2D de vehículo reducido tipo partícula

de control, es decir

$$\mathbf{u} = [u_1]^T = [\delta]^T \tag{6.5}$$

cuya dimensión es $N_i = 1$, la Ec.(6.4) se puede escribir en forma genérica como la Ec.(6.1), donde la función vectorial $\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$, que define la dinámica del vehículo reducido tipo partícula, queda determinada por:

$$\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v\cos\delta \\ v\sin\delta \end{bmatrix} \tag{6.6}$$

6.2.2. Vehículo Reducido tipo Partícula en 3D

Para el caso de una partícula que se desplaza en el espacio xyz, el vector de estados $\mathbf{x}$ queda determinado por:

$$\mathbf{x} = [x_1 \quad x_2 \quad x_3]^T = [x \quad y \quad z]^T \tag{6.7}$$

de dimensión $N_s = 3$.

En la Fig. 6.2 se puede observar un esquema del modelo 3D del vehículo reducido tipo partícula propuesto. Las componentes x, y y z del vector $\dot{\mathbf{x}}$ se pueden obtener como se muestra a continuación:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} v\cos\gamma\cos\delta \\ v\cos\gamma\sin\delta \\ v\sin\gamma \end{bmatrix} \tag{6.8}$$

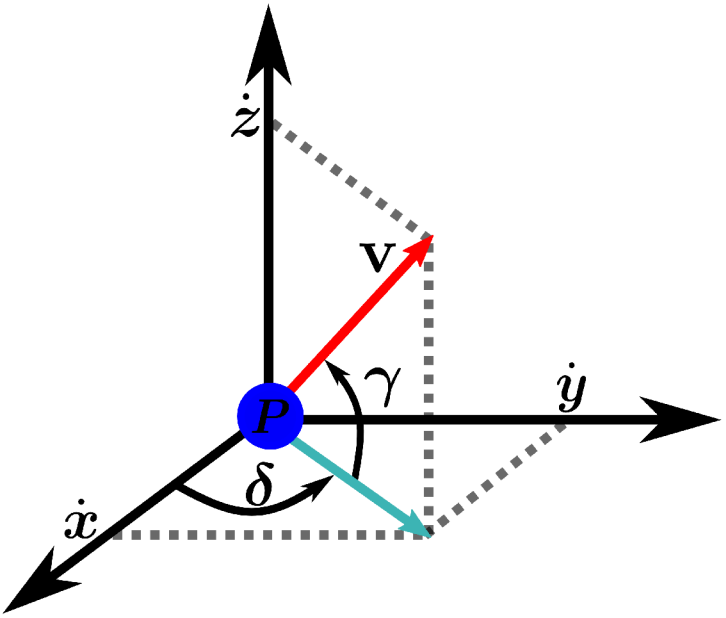


Figura 6.2: Modelo 3D de vehículo reducido tipo partícula

Nuevamente, v es el módulo del vector velocidad del vehículo y δ es el ángulo de orientación en plano horizontal. γ es el ángulo de ascenso y está definido por el ángulo formado entre el vector velocidad $\mathbf{v}$ y plano xy. Tomando como entradas de control a los ángulos γ y δ, el vector de entradas de control $\mathbf{u}$ se puede escribir como sigue:

$$\mathbf{u} = [u_1 \quad u_2]^T = [\gamma \quad \delta]^T \tag{6.9}$$

cuya dimensión es $N_i = 2$. La Ec.(6.8) también puede escribirse en forma genérica como la Ec.(6.1), siendo la función vectorial $\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ para este caso:

$$\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v \cos\gamma \cos\delta \\ v \cos\gamma \sin\delta \\ v \sin\gamma \end{bmatrix} \tag{6.10}$$

6.3. Generación de Trayectorias de Navegación Óptimas

La idea principal de la metodología propuesta consiste en generar una trayectoria navegación de mínimo recorrido, considerando las restricciones propias del modelo del vehículo reducido tipo partícula y penalizando las variaciones excesivas en las entradas de control.

En esta sección se describe el procedimiento empleado para la generación de trayectorias de navegación en el espacio tridimensional, ya que el caso bidimensional corresponde al caso particular en el cual el ángulo de ascenso γ es cero.

Si el modelo del vehículo reducido tipo partícula no tuviese restricciones, la trayectoria de navegación de mínimo recorrido es la recta que une la posición actual del vehículo reducido tipo partícula con un determinado *waypoint* (por ejemplo la línea punteada hacia $\mathbf{W}_1$ en la Fig. 6.3). Sin embargo, si el modelo del vehículo sí posee restricciones en su dinámica, entonces, la trayectoria de navegación óptima resultante puede diferir de la recta mencionada anteriormente. Ambas trayectorias de navegación se muestran en la Fig. 6.3. Nótese,

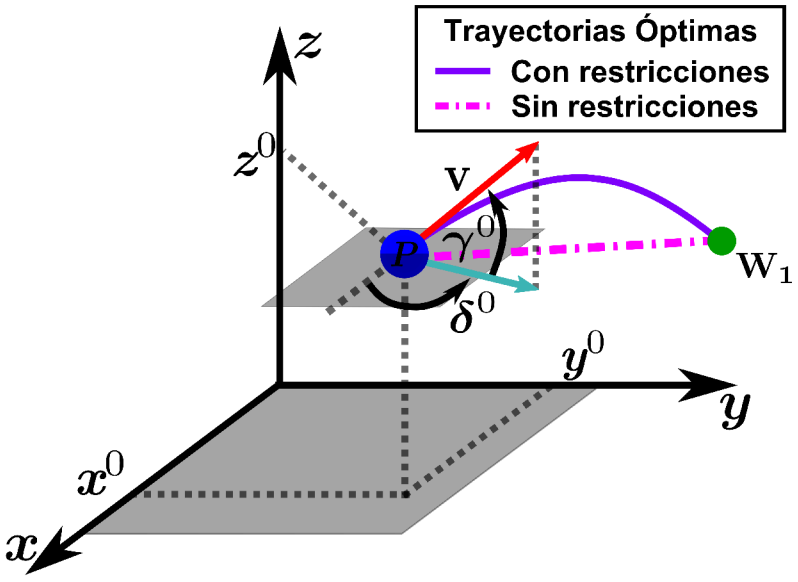


Figura 6.3: Trayectoria de navegación generada

que la trayectoria que describe el vehículo reducido tipo partícula queda determinada por la dirección del vector velocidad $\mathbf{v}$, o lo que es lo mismo, por los ángulos de ascenso γ y de orientación δ.

Dados el vector de estados inicial $\mathbf{x}^0 = [x^0 \quad y^0 \quad z^0]^T$ y el vector de ángulos iniciales $\mathbf{u}^0 = [\gamma^0 \quad \delta^0]^T$, existen infinitas trayectorias de navegación que unen el estado actual del vehículo reducido tipo partícula con el *waypoint* deseado, $\mathbf{W}_1$ en la Fig. 6.3. Como se ha mencionado anteriormente, se pretende hallar la trayectoria de navegación de mínimo recorrido que une el origen con el destino y que además tiene en cuenta las restricciones impuestas por la dinámica propia del vehículo en cuestión. Esto último puede lograrse aplicando el método INL-MPC, presentado en el capítulo 4, al modelo del vehículo reducido tipo partícula de Ec. (6.10), lo cual da como resultado, para este caso, una secuencia efectiva de control $\mathbf{U}_c^*$ que contiene los ángulos $\mathbf{u}^k = [\gamma^k \quad \delta^k]^T$ óptimos para $k = 1, \cdots, h_c$, es decir:

$$\mathbf{U}_c^* = \begin{bmatrix} \mathbf{u}^1 & \mathbf{u}^2 & \dots & \mathbf{u}^{h_c} \end{bmatrix}^T \tag{6.11}$$

que minimiza la distancia entre la posición actual del vehículo tipo partícula y el *waypoint* $\mathbf{W}_1$. Aplicando el vector de entradas de control $\mathbf{u}^1 = [\gamma^1 \quad \delta^1]^T$, correspondiente al primer instante de muestreo, al modelo del vehículo reducido tipo partícula definido en la Ec. (6.10), se puede hallar el nuevo estado de dicho vehículo mediante, por ejemplo, el algoritmo de integración presentado en la sección 2.5. El proceso se reinicia desplazando el horizonte de predicción al próximo instante de muestreo. Los sucesivos estados del vehículo reducido tipo partícula obtenidos son los que determinan la trayectoria de navegación resultante.

En el caso de contar con varios *waypoints*, la trayectoria de navegación resultante se puede determinar computando trayectorias de navegación parciales, por tramos, como se muestra en la Fig. 6.4. Como se puede observar en la Fig. 6.4, el estado actual del vehículo reducido tipo partícula $\mathbf{x}^0$ y el primer *waypoint* $\mathbf{W}_1$ definen el tramo de trayectoria de navegación parcial $\mathbf{T}_1$. Asimismo, cada par de *waypoints* $(\mathbf{W}_{i-1}, \mathbf{W}_i)$ definen el tramo de trayectoria $\mathbf{T}_i$, con $i = 2, \cdots, 4$. Así, por ejemplo, $\mathbf{T}_2$ es el tramo de trayectoria de navegación que

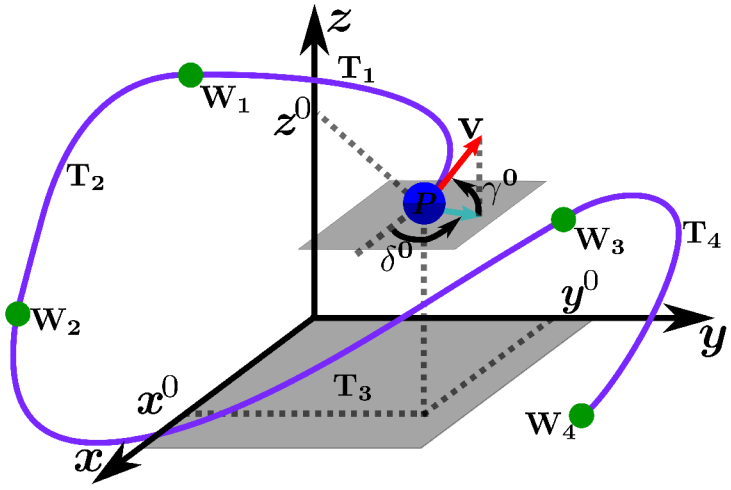


Figura 6.4: Trayectoria de navegación generada a partir de cuatro *waypoints*

une el *waypoint* $\mathbf{W}_1$ con el *waypoint* $\mathbf{W}_2$, y así sucesivamente hasta llegar al último *waypoint* del listado, $\mathbf{W}_4$ para este ejemplo. Cada tramo de trayectoria de navegación parcial $\mathbf{T}_i$, con $i = 1, \cdots, 4$, se computa aplicando INL-MPC al modelo del vehículo reducido tipo partícula, obteniéndose como resultado una secuencia efectiva de control $\mathbf{U}_c^*$ que contiene los ángulos $\mathbf{u}^k = [\gamma^k \quad \delta^k]^T$ óptimos para $k = 1, \cdots, h_c$, es decir:

$$\mathbf{U}_c^* = \begin{bmatrix} \mathbf{u}^1 & \mathbf{u}^2 & \dots & \mathbf{u}^{h_c} \end{bmatrix}^T \tag{6.12}$$

que minimiza la distancia entre la posición actual del vehículo tipo partícula y el *waypoint* $\mathbf{W}_i$ asociado al tramo de trayectoria de navegación $\mathbf{T}_i$, con $i = 1, \cdots, 4$. Nuevamente, aplicando al modelo reducido de vehículo tipo partícula de Ec. (6.10) el vector de entradas de control óptimo $\mathbf{u}^1 = [\gamma^1 \quad \delta^1]^T$, correspondiente al primer instante de muestreo, se puede hallar, mediante integración, el nuevo estado de dicho vehículo y el proceso se reinicia desplazando el horizonte de predicción al próximo instante de muestreo. Nuevamente, los sucesivos estados del vehículo reducido tipo partícula obtenidos son los que determinan la trayectoria de navegación resultante.

6.4. Conclusiones

En este capítulo se presentó una metodología *online* para la generación de trayectorias de navegación de mínimo recorrido, que tienen en cuenta las restricciones propias de la dinámica del modelo utilizado para la obtención de las mismas. El modelo empleado es un modelo no-lineal de un vehículo reducido tipo partícula que, al ser utilizado con INL-MPC, proporciona las coordenadas (x, y) y (x, y, z) que determinan las trayectorias 2D y 3D deseadas, respectivamente. La función de costo utilizada en INL-MPC minimiza, en este caso, la distancia entre la posición actual del vehículo reducido tipo partícula y los sucesivos *waypoints*, dando como resultado una trayectoria de navegación óptima que, considerando la dinámica del vehículo reducido tipo partícula, proporciona el camino más corto.

Capítulo 7

Sistema de Control de Vuelo Autónomo Unificado para UAVs usando INL-MPC

En este capítulo se presentan aplicaciones sobre el uso del método INL-MPC como un sistema de control de actitud y navegación unificado para UAVs. En las aplicaciones presentadas se utiliza como UAV el modelo matemático del avión Cessna 172, descripto en la sección 2.6 y en el apéndice A. En principio el método INL-MPC puede ser usado para realizar diversas tareas de vuelo autónomo. Aquí se presentarán las siguientes tres maniobras de vuelo autónomas:

1. Ascenso a una altitud deseada manteniendo velocidad constante
2. Cambio en la dirección de vuelo manteniendo velocidad y altitud constantes
3. Giro coordinado manteniendo velocidad y altitud constantes

Asimismo, en este capítulo se compara el desempeño de las técnicas MPC e INL-MPC mediante la realización de una maniobra autónoma de ascenso con aumento de velocidad y cambio en la dirección de la trayectoria de vuelo.

7.1. Metodología de Control

El sistema de control de actitud y navegación unificado permite a un vehículo aéreo no-tripulado volar y realizar diferentes maniobras autónomas, tales como ascensos, descensos, cambios de dirección y giros coordinados, de una manera óptima. El esquema de funcionamiento del control unificado basado en INL-MPC y aplicado al UAV se muestra en la Fig. 7.1. El lazo de control funciona de la siguiente manera: el UAV envía al controlador de actitud y navegación su vector de estado $\mathbf{x}^0$ y su vector de entradas de control $\mathbf{u}^0$ iniciales. El sistema de control de actitud y navegación recibe esta información, y utilizando un modelo matemático del UAV, computa la secuencia efectiva de control óptima $\mathbf{U}_c^*$ y envía al UAV real solamente el vector de entradas de control correspondiente al primer instante de muestreo, es decir $\mathbf{u}^1 = \mathbf{U}_c^*(1)$. Finalmente, el UAV real recibe el control óptimo computado y lo aplica. El proceso se reinicia desplazando el horizonte de tiempo un paso hacia adelante al próximo instante de muestreo.

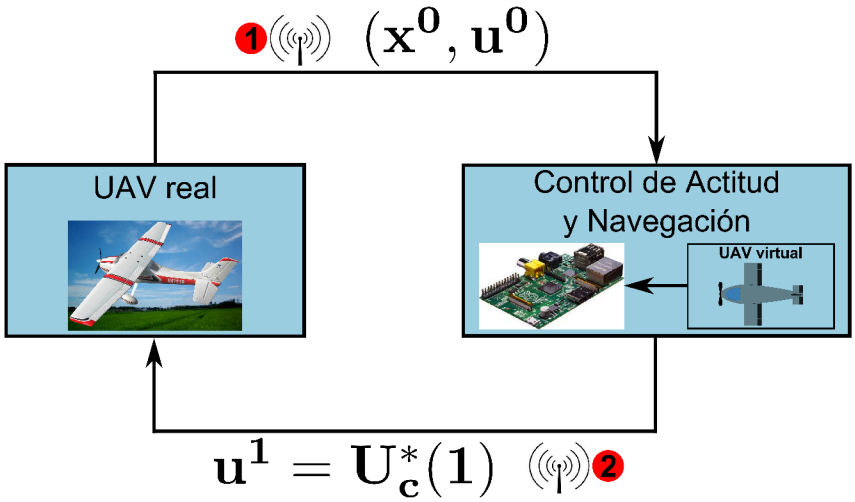


Figura 7.1: Esquema funcionamiento sistema de control unificado y UAV autónomo

7.2. Configuración de Variables Físicas y otros Parámetros para el Uso de INL-MPC

Se asume que la función vectorial $\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ que define la dinámica del UAV es la definida en Ec. (2.38). Se adopta un período de muestreo $\Delta T_s = 0{,}5$ [seg], horizonte de predicción $h_p = 20$ y horizonte de control $h_c = 17$. Tanto h_p como h_c se seleccionaron empíricamente, el aumento de los horizontes por encima de los valores adoptados no produjo cambios significativos en la respuesta del sistema, sin embargo, sí produce un aumento innecesario en los tiempos de cómputo.

Con el fin de ponderar de forma adecuada los estados físicos y las variaciones en las entradas de control con diferentes órdenes de magnitud, los pesos $\tilde{\mathbf{Q}}_x^k$ y $\tilde{\mathbf{R}}_{\dot{u}}^k$ definidos en la Ec. (4.32) son normalizados teniendo en cuenta valores típicos de desviaciones de estados $\mathbf{x}_{Typ}$ y de velocidad de cambio de entradas $\dot{\mathbf{u}}_{Typ}$. Las desviaciones típicas genéricas para la i-ésima componente del vector de estados $\mathbf{x}(i)$ y del vector de velocidad de cambio de entradas $\dot{\mathbf{u}}(i)$ se denota como $\mathbf{x}_{Typ}(i)$ y $\dot{\mathbf{u}}_{Typ}(i)$, respectivamente. Luego, los valores de las matrices definidas en la Ec. (4.32) se seleccionan como:

$$\tilde{\mathbf{Q}}_x^k(i,i) = \frac{\mathbf{Q}_x^k(i,i)\Delta T_s}{max\{\mathbf{x}(i) - \mathbf{x}_d(i), \mathbf{x}_{Typ}(i)\}^2} \tag{7.1}$$

y

$$\tilde{\mathbf{R}}_{\dot{u}}^k(i,i) = \frac{\mathbf{R}_{\dot{u}}^k(i,i)}{max\{\dot{\mathbf{u}}(i) - \dot{\mathbf{u}}_d(i), \dot{\mathbf{u}}_{Typ}(i)\}^2 \Delta T_s} \tag{7.2}$$

Además, $\mathbf{Q}_x^k(i,j) = 0$ y $\mathbf{R}_{\dot{u}}^k(i,j) = 0$ para $i \neq j$. Los valores típicos adoptados son los siguientes: $v_{t_{Typ}} = 5$ [m/seg], $\alpha_{Typ} = 1$ [deg], $\beta_{Typ} = 1$ [deg], $\phi_{Typ} = 5$ [deg], $\theta_{Typ} = 5$ [deg], $\psi_{Typ} = 5$ [deg], $p_{Typ} = 1$ [deg /seg], $q_{Typ} = 1$ [deg /seg], $r_{Typ} = 1$ [deg /seg], $x_{Typ} = 100$ [m], $y_{Typ} = 100$ [m], $z_{Typ} = 10$ [m], $thtl_{Typ} = 0{,}5$, $\delta_{e_{Typ}} = 15$ [deg], $\delta_{a_{Typ}} = 15$ [deg], $\delta_{r_{Typ}} = 15$ [deg].

Los pesos de las matrices $\mathbf{Q}_x^k(i,i)$ y $\mathbf{R}_{\dot{u}}^k(i,i)$ se definen para cada ejemplo. Sus valores de ponderación han sido seleccionados empíricamente simplemente pesando más los estados y entradas de acuerdo a la maniobra de vuelo deseada.

Para los ejemplos, las siguientes restricciones de entradas han sido utiliza-

das: $thtl_m = 0$, $thtl_M = 1$, $\delta_{em} = -20$ [deg] $\delta_{eM} = 20$ [deg], $\delta_{am} = -20$ [deg], $\delta_{aM} = 20$ [deg], $\delta_{rm} = -15$ [deg], $\delta_{rM} = 15$ [deg].

Las restricciones utilizadas en la velocidad de cambio de entradas son: $\dot{thtl}_m = -0{,}2$ [1/sec], $\dot{thtl}_M = 0{,}2$ [1/sec], $\dot{\delta}_{em} = -2$ [deg /sec], $\dot{\delta}_{eM} = 2$ [deg /sec], $\dot{\delta}_{am} = -2$ [deg /sec], $\dot{\delta}_{aM} = 2$ [deg /sec], $\dot{\delta}_{rm} = -2$ [deg /sec], $\dot{\delta}_{rM} = 2$ [deg /sec].

Finalmente, las restricciones de estados que fueron utilizadas son: $v_{t_m} = 10$ [m/sec], $v_{t_M} = 76$ [m/sec], $\alpha_m = -16$ [deg], $\alpha_M = 16$ [deg], $\beta_m = -15$ [m], $\beta_M = 15$ [deg], $h_m = 0$ [m], $h_M = 4000$ [m].

7.3. Primer Ejemplo: Maniobra de Ascenso a Velocidad Constante

La técnica INL-MPC presentada se aplica al UAV para realizar una maniobra autónoma de ascenso a velocidad constante. Se asume que la aeronave se encuentra inicialmente volando a una altitud $h = 1000\,[\text{m}]$ con una velocidad $v_t = 45\,[\text{m/sec}]$, y se comanda la realización de un ascenso a una nueva altitud de $h = 1500\,[\text{m}]$ mientras mantiene su velocidad constante. Para este propósito, los elementos del vector de valores deseados se adoptan como: $\mathbf{x}_d(1) = v_{t_d} = 45$, $\mathbf{x}_d(3) = \beta_d = 0$, $\mathbf{x}_d(4) = \phi_d = 0$, $\mathbf{x}_d(6) = \psi_d = 0$ y $\mathbf{x}_d(12) = h_d = 1500$. Los pesos de las matrices para la optimización se eligen como sigue: $\mathbf{Q}_x^k(1,1) = 10$, $\mathbf{Q}_x^k(3,3) = 10$, $\mathbf{Q}_x^k(4,4) = 10$, $\mathbf{Q}_x^k(6,6) = 10$, $\mathbf{Q}_x^k(12,12) = 10$ y $\mathbf{R}_{\dot{u}}^k(1,1) = 0{,}1$, $\mathbf{R}_{\dot{u}}^k(2,2) = 0{,}1$, $\mathbf{R}_{\dot{u}}^k(3,3) = 0{,}1$, $\mathbf{R}_{\dot{u}}^k(4,4) = 0{,}1$, con $k = 1, \cdots, h_p$. Notar que solamente los estados involucrados en la maniobra de ascenso han sido pesados: velocidad v_t, ángulo de deslizamiento β, ángulos de Euler ϕ y ψ y altura h. Los ángulos de Euler han sido pesados para que el UAV autónomo mantenga la condición de vuelo nivelado.

En la Fig. 7.2, se puede observar la evolución de la altitud y de la velocidad del UAV autónomo cuando la maniobra de ascenso indicada por el sistema de control es llevada a cabo. Notar cómo el sistema de control de actitud y navegación basado en INL-MPC permitió al UAV autónomo alcanzar la altitud deseada a la vez que mantuvo la velocidad prácticamente constante. En la Fig.

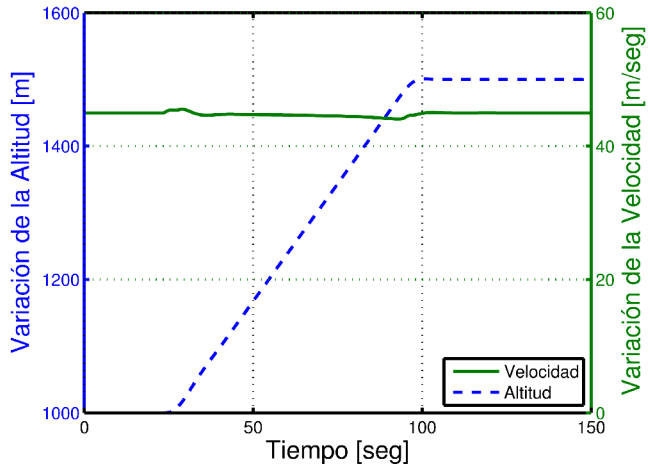


Figura 7.2: Evolución de la altitud y velocidad para un ascenso de 500 [m]

7.3, se muestra la evolución de la deflexión de las entradas de control. Se puede observar que la deflexión de la columna de propulsión aumenta hasta alcanzar su máximo valor para que el UAV ascienda con una máxima tasa de ascenso. Cuando el vehículo se encuentra próximo al valor de altitud deseada la deflexión de la columna de propulsión se reduce, hasta llegar prácticamente al valor inicial, para evitar que el UAV continúe subiendo. El elevador se mueve para producir una variación en el ángulo de ataque y poder así mantener la velocidad constante.

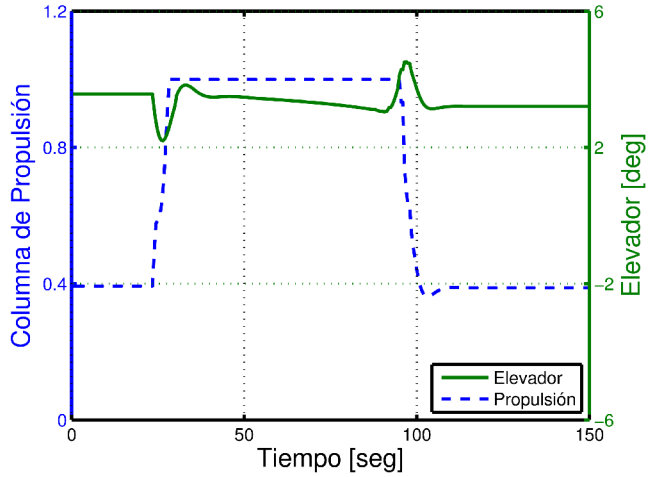


Figura 7.3: Evolución de las entradas de control para un ascenso de 500 [m]

7.4. Segundo Ejemplo: Cambio de Dirección en la Trayectoria de Vuelo a Velocidad y Altitud Constantes

Se considera que el UAV se encuentra nuevamente volando inicialmente a una altitud de $h = 1000\,[\text{m}]$ y velocidad de $v_t = 45\,[\text{m/sec}]$. El método INL-MPC se aplica al UAV autónomo para realizar un cambio de 90 [deg] en la dirección de la trayectoria de vuelo, manteniendo constantes tanto su altitud como su velocidad. En consecuencia, el vector de valores deseados se configura como sigue: $\mathbf{x}_d(1) = v_{t_d} = 45$, $\mathbf{x}_d(3) = \beta_d = 0$, $\mathbf{x}_d(6) = \psi_d = \frac{\pi}{2}$ y $\mathbf{x}_d(12) = h_d = 1000$. Los pesos de estados se eligen como sigue: $\mathbf{Q}_x^k(1,1) = 10$, $\mathbf{Q}_x^k(3,3) = 1$, $\mathbf{Q}_x^k(6,6) = 10$, $\mathbf{Q}_x^k(12,12) = 10$. Solamente los estados involucrados en esta maniobra han sido pesados: velocidad v_t, ángulo de deslizamiento β, ángulo de guiñada ψ y altitud h. Los pesos de entradas de eligen como sigue: $\mathbf{R}_{\dot{u}}^k(1,1) =$ 0,1, $\mathbf{R}_{\dot{u}}^k(2,2) =$ 0,1, $\mathbf{R}_{\dot{u}}^k(3,3) =$ 0,1, $\mathbf{R}_{\dot{u}}^k(4,4) =$ 0,1; con $k = 1, \cdots, h_p$. La evolución de la trayectoria del UAV autónomo se muestra en la Fig. 7.4. Como

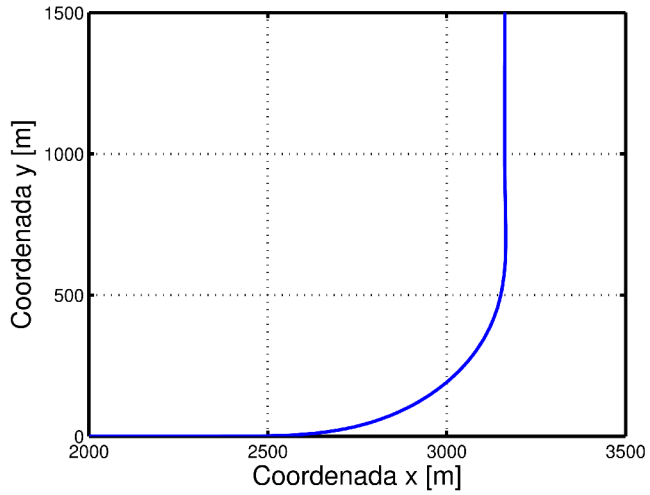
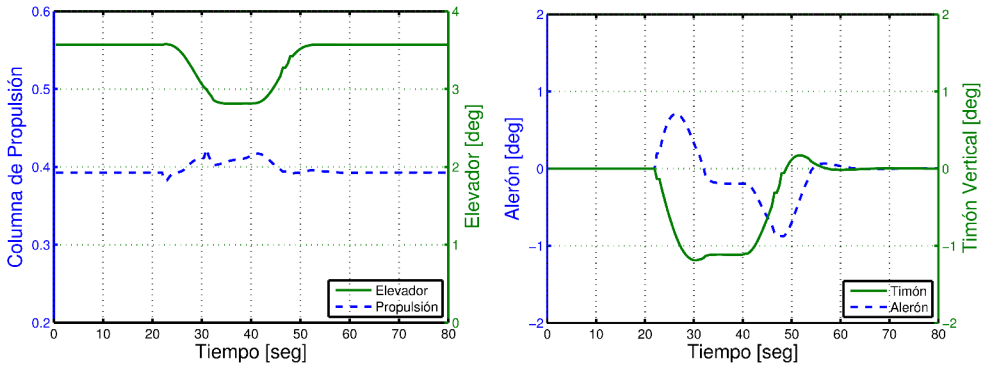


Figura 7.4: Cambio de 90 [deg] en la dirección de vuelo

se puede observar, el sistema de control de actitud y navegación basado en INL-MPC permitió al UAV autónomo realizar el cambio de noventa grados en la trayectoria de vuelo. La maniobra se llevó a cabo manteniendo tanto la

altitud como la velocidad constantes. La evolución de las cuatro entradas de



(a) Columna de Propulsión y Elevador (b) Alerón y Timón Vertical

Figura 7.5: Evolución de las entradas de control para un cambio de 90 [deg] en la dirección de vuelo

control involucradas en la maniobra se muestran en la Fig. 7.5. La columna de propulsión y el elevador se mueven conjuntamente para mantener constantes los valores de altitud y velocidad. El alerón y el timón vertical producen un cambio en los momentos de *roll* y de *yaw* y como resultado el UAV autónomo realiza el cambio de dirección deseado en la trayectoria de vuelo.

7.5. Tercer Ejemplo: Giro Coordinado a Velocidad y Altitud Constantes

La próxima maniobra autónoma a realizar es un giro coordinado a velocidad y altitud constantes. Nuevamente, el UAV se encuentra inicialmente volando a una altitud $h = 1000$ [m] y velocidad $v_t = 45$ [m/sec]. La técnica de control INL-MPC se aplica al UAV autónomo para realizar un giro coordinado manteniendo constantes tanto la altitud como la velocidad. Por lo tanto, el vector de valores de deseados se configura como sigue: $\mathbf{x}_d(1) = v_{t_d} = 45$, $\mathbf{x}_d(3) = \beta_d = 0$, $\mathbf{x}_d(4) = \phi_d = \frac{15\pi}{180}$, $\mathbf{x}_d(12) = h_d = 1000$. Los pesos de estados se adoptan como se muestra a continuación: $\mathbf{Q}_x^k(1,1) = 10$, $\mathbf{Q}_x^k(3,3) = 1$, $\mathbf{Q}_x^k(4,4) = 10$, $\mathbf{Q}_x^k(12,12) = 10$. Solamente los estados involucrados en esta maniobra han sido ponderados: velocidad v_t, ángulo de deslizamiento β, ángulo de *roll* ϕ y

altitud h. Los pesos de entradas se han elegido como sigue: $\mathbf{R}_{\dot{u}}^{k}(1,1) = 0{,}1$, $\mathbf{R}_{\dot{u}}^{k}(2,2) = 0{,}1$, $\mathbf{R}_{\dot{u}}^{k}(3,3) = 0{,}1$, $\mathbf{R}_{\dot{u}}^{k}(4,4) = 0{,}1$; con $k = 1, \cdots, h_p$. En la

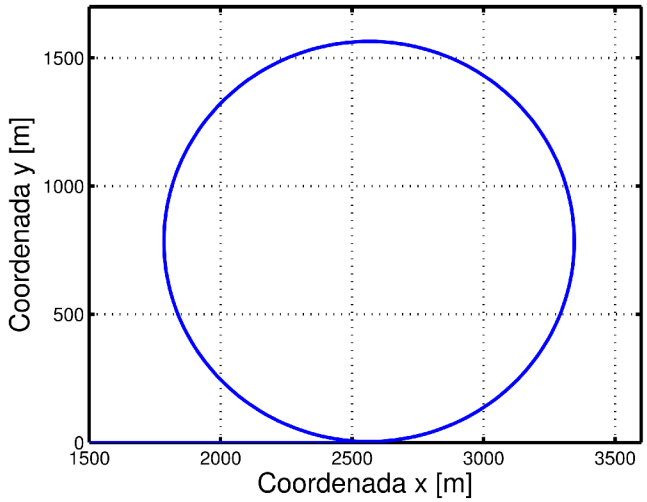
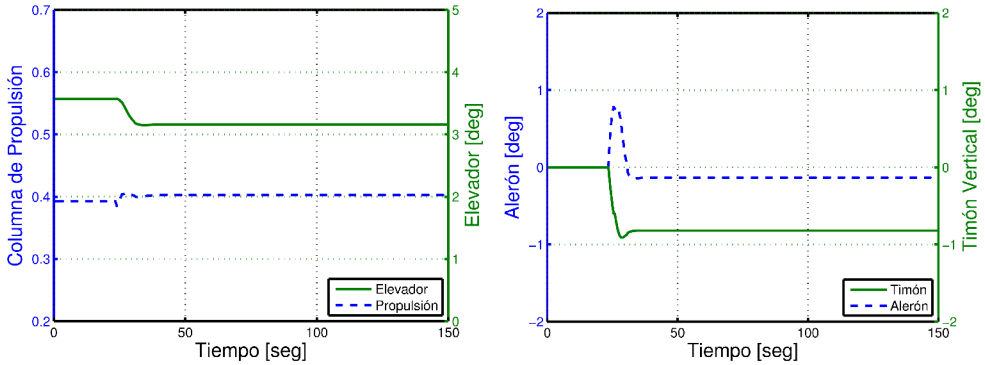


Figura 7.6: Evolución de la trayectoria de vuelo del UAV autónomo para un giro coordinado

Fig. 7.6 se puede observar que la trayectoria resultante que describe el UAV autónomo claramente corresponde a un giro coordinado. Esta maniobra se realizó manteniendo constante los valores de altitud y velocidad. La evolución



(a) Columna de Propulsión y Elevador (b) Alerón y Timón Vertical

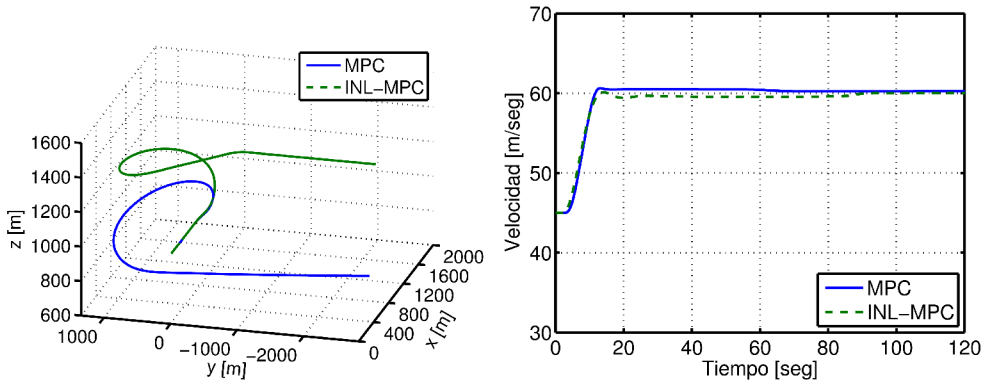
Figura 7.7: Evolución de las entradas de control para un giro coordinado

de las cuatro entradas de control se muestran en la Fig. 7.7. La columna de propulsión y el elevador se mueven en forma conjunta para mantener tanto la

altitud como la velocidad en valores constantes. Puede observarse que tanto el alerón como el timón vertical alcanzan una posición final que hace que el ángulo de *roll* tienda al valor deseado.

7.6. Comparación de las Técnicas de Control MPC e INL-MPC

En esta sección se compara la *performance* de la técnica de control INL-MPC con respecto a la técnica de control predictivo clásica (MPC). Para ello, se asume que el UAV se encuentra inicialmente volando a una altitud $h = 1000\,[\text{m}]$ con una velocidad $v_t = 45\,[\text{m/sec}]$. El sistema de control de actitud y navegación unificado basado en ambas técnicas (INL-MPC y MPC) se aplica al UAV autónomo para realizar una maniobra de ascenso a $h = 1500\,[\text{m}]$ con una velocidad $v_t = 60\,[\text{m/sec}]$ y un cambio en la dirección de vuelo de 270 [deg]. En consecuencia, el vector de valores deseados se configura como sigue: $\mathbf{x}_d(1) = v_{t_d} = 60$, $\mathbf{x}_d(3) = \beta_d = 0$, $\mathbf{x}_d(6) = \psi_d = \frac{3}{2}\pi$ y $\mathbf{x}_d(12) = h_d = 1500$. Los pesos de estados se eligen como sigue: $\mathbf{Q}_x^k(1,1) = 10$, $\mathbf{Q}_x^k(3,3) = 1$, $\mathbf{Q}_x^k(6,6) = 10$, $\mathbf{Q}_x^k(12,12) = 10$. Solamente los estados involucrados en esta maniobra han sido pesados: velocidad v_t, ángulo de deslizamiento β, ángulo de guiñada ψ y altitud h. Los pesos de entradas de eligen como sigue: $\mathbf{R}_{\dot{u}}^k(1,1) = 0{,}1$, $\mathbf{R}_{\dot{u}}^k(2,2) = 0{,}1$, $\mathbf{R}_{\dot{u}}^k(3,3) = 0{,}1$, $\mathbf{R}_{\dot{u}}^k(4,4) = 0{,}1$; con $k = 1, \cdots, h_p$. En la Fig.7.8a se puede observar el recorrido que describe el UAV autónomo, cuando el mismo es controlado con ambos métodos de control. Como se puede observar, solamente con la técnica de control INL-MPC se pudo realizar la maniobra de vuelo deseada. Cuando se utiliza el método MPC, si bien el vehículo logra modificar la dirección de vuelo aumentando la velocidad al valor deseado, la altitud del mismo, contrario a la maniobra deseada, comienza a disminuir. En la Fig. 7.9 se pueden observar las evoluciones de las entradas de control del UAV autónomo. En el caso de la columna de propulsión se puede observar, para el caso de MPC, que el valor de la misma comienza a descender notablemente, lo cual provoca que el UAV pierda altitud. Esto se puede ver en la Fig. 7.9a. Con la maniobra presentada anteriormente, queda en evidencia la ventaja de



(a) Variación Altitud (b) Variación Velocidad

Figura 7.8: Evolución de la altitud, velocidad y ángulo de orientación ψ con controles MPC e INL-MPC

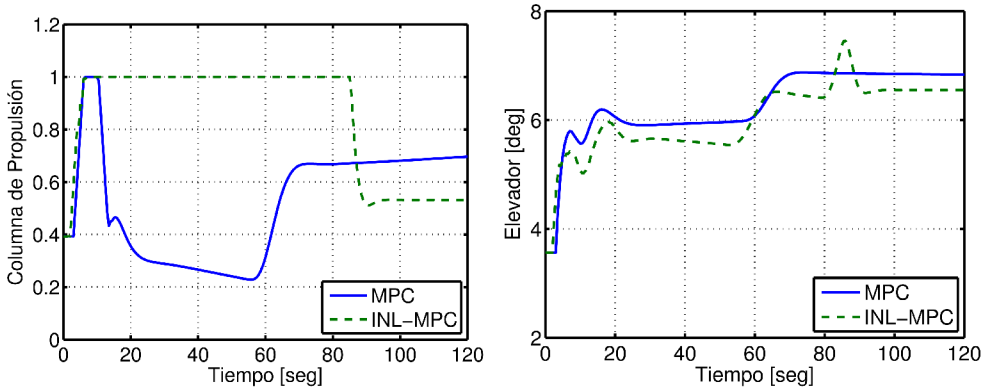
utilizar un método de control diseñado para sistemas no-lineales, como es el INL-MPC, el cual fue presentado en detalle en el capítulo 4.

7.7. Comentarios Adicionales

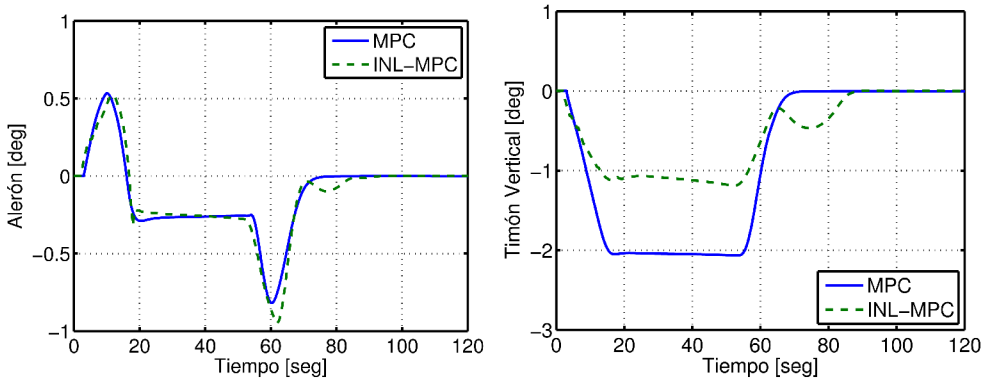
Para las aplicaciones presentadas en la sección anterior, se evaluó y se testeó que la formulación de cada sub-problema de minimización iterativo, definidos en la Ec. (4.43), estén bien planteados. Es decir, se verificó que en cada iteración se está resolviendo un sub-problema *convexo correcto*. Debido a esta convexidad, se puede garantizar la terminación del problema de optimización. En consecuencia, los problemas de optimización resultantes del método INL-MPC están bien planteados.

En cuanto a la selección del período de muestreo ΔT_s, debe mencionarse que, como la máxima frecuencia natural del sistema modelado se encuentra siempre por debajo de $f_{max} = 0{,}71\,[\text{Hz}]$, el teorema del muestreo requiere que $\Delta T_s \leq 0{,}7\,[\text{sec}]$, por lo tanto se adoptó $\Delta T_s = 0{,}5\,[\text{sec}]$.

Respecto a la implementación, tanto el modelo del UAV como el algoritmo INL-MPC han sido programados en C++, utilizando la librería *LTensor* para los cómputos matriciales [50], siendo ejecutados los mismos en unidades de cómputo independientes y en *tiempo real*. Para resolver el problema de optimi-



(a) Deflexión Columna Propulsión (b) Deflexión Elevador



(c) Deflexión Alerón (d) Deflexión Timón Vertical

Figura 7.9: Evolución de las entradas de control con métodos MPC e INL-MPC

zación cuadrático (Ec. (4.43) y **Paso 6** del algoritmo dado en el Cuadro 4.1) se realiza una llamada a la función *quadprog* de Matlab (accediéndose desde C++ como una librería dinámica). Dentro de esta función el algoritmo de *punto de interior* fue elegido.

En este capítulo, la técnica INL-MPC se emplea como un sistema de control de actitud y navegación unificado que es capaz de controlar tanto la dinámica como la navegación del UAV, teniendo en cuenta la dinámica completa, de 6-DOF, del mismo. Esto es muy importante ya que no es necesario desacoplar el modelo del vehículo aéreo en sus modos dinámicos, como por ejemplo el longitudinal y el transversal, y en consecuencia los acoples existentes entre modos son tenidos en cuenta. Notar que, el sistema de control de actitud y navegación controla al modelo completo del UAV (6-DOF con 12 variables de estado y 4 entradas de control), con lo cual, la carga computacional de este controlador puede ser, en general, elevada. Sin embargo, para las maniobras presentadas anteriormente, el tiempo de cómputo máximo de la implementación del algoritmo INL-MPC es $\Delta t_{\text{cpu}} = 0{,}3\,[\text{sec}]$ (con tres iteraciones en el lazo de optimización de trayectoria). Esto es siempre menor que la tasa de refresco del control $\Delta T_s = 0{,}5\,[\text{sec}]$, garantizando una simulación y control en *tiempo real*.

7.8. Conclusiones

El desempeño del método INL-MPC utilizado como un sistema de control de actitud y navegación unificado para UAVs, se demostró mediante la realización de tres maniobras autónomas diferentes, usando un modelo completo de UAV, de 6-DOF. En todos los casos, las maniobras autónomas fueron realizadas satisfactoriamente.

Capítulo 8

Sistema de Control de Vuelo Autónomo Desacoplado para UAVs usando INL-MPC e INL-MPC-J

En este capítulo se presenta un sistema de control de vuelo autónomo desacoplado para UAVs basado en los métodos INL-MPC e INL-MPC-J. Para ello, el sistema de control de vuelo se desacopla en dos sub-sistemas: uno de actitud y otro de navegación. El controlador de vuelo resultante se utiliza para realizar maniobras de generación y seguimiento de trayectorias de navegación (2D y 3D con uno o más *waypoints*). El vehículo aéreo se modela utilizando el modelo matemático del avión Cessna 172 descripto en la sección 2.6 y en el apéndice A.

8.1. Metodología de Control

Para que los UAVs puedan volar autónomamente, es indispensable que los mismos cuenten con un adecuado sistema de control de vuelo. En el capítulo anterior se discutió un sistema de control unificado donde un único controlador realiza las tareas de control de actitud y de navegación. En este capítulo se

presenta una segunda perspectiva basada en el desacople del sistema de control en dos controladores independientes, uno para controlar la actitud del UAV y otro para obtener las trayectorias de navegación. El control de navegación es el encargado de determinar la trayectoria o ruta que el UAV autónomo debe seguir. Por su parte, el control de actitud se encarga de computar los valores de deflexiones que deben tomar las superficies aerodinámicas (elevador, alerones y timón vertical) y la columna de propulsión para controlar la orientación y velocidad del UAV autónomo. El sistema de control propuesto tiene en cuenta la dinámica completa, de 6-grados de libertad (6-DOF), del UAV. Esto es muy ventajoso ya que no es necesario desacoplar el modelo en sus modos dinámicos, como por ejemplo el longitudinal y el transversal, y en consecuencia los acoples existentes entre modos son tenidos en cuenta. El diseño del sistema de control de navegación se realizó utilizando modelos de vehículos reducidos tipo partícula y el método INL-MPC presentado en el capítulo 4, permitiendo la generación de diferentes trayectorias de navegación óptimas, de forma dinámica. Para el diseño del controlador de actitud se utilizó la técnica INL-MPC-J presentada en el capítulo 5. En este caso, a diferencia del sistema de control unificado, los dos controladores se ejecutan de forma independiente en paralelo, reduciéndose, en consecuencia, el costo computacional del sistema de control global.

Notación: En este capítulo se subrayarán los vectores asociados al sistema de control de navegación para distinguirlos de los vectores asociados al sistema de control de actitud. En consecuencia, según la notación adoptada, se tiene para el controlador de navegación que: $\underline{\mathbf{x}}$ es el vector de estados, $\underline{\mathbf{x}}_d$ es el vector de estados deseados, $\underline{\mathbf{u}}$ es el vector de entradas de control, $\underline{\dot{\mathbf{u}}}$ es el vector velocidad de cambio de entradas de control y $\underline{\mathbf{U}}_p^*$ es la secuencia de entradas de control óptima.

El esquema de funcionamiento del control desacoplado se puede observar en la Fig. 8.1. El lazo de control funciona de la siguiente manera: el UAV envía al controlador de actitud y al controlador de navegación su vector de estado $\mathbf{x}^0$ y su vector de entradas de control $\mathbf{u}^0$, actuales. Dado el estado recibido del vehículo y el *waypoint* $\mathbf{W}_i$ a alcanzar, el sistema de control de navegación brinda como *output* la orientación del vector velocidad deseado

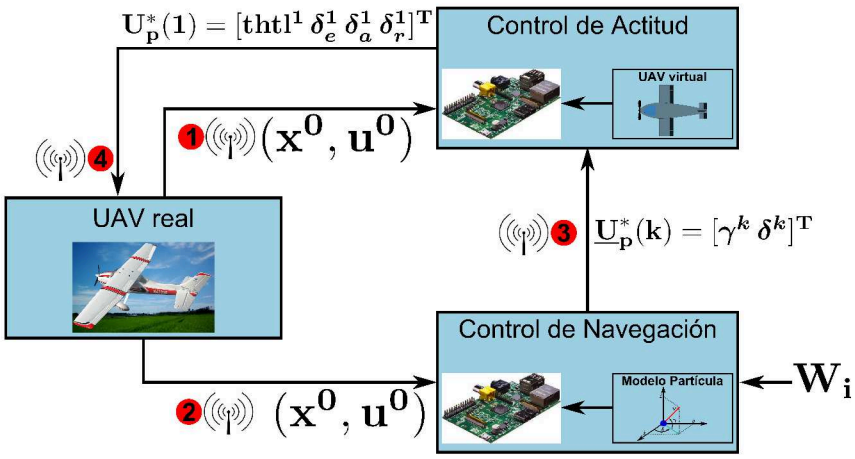


Figura 8.1: Esquema funcionamiento sistema de control desacoplado y UAV autónomo

que el vehículo deberá tener para alcanzar dicho *waypoint*. Para ello, dicho controlador configura su vector de estados y usando INL-MPC computa la secuencia de orientación (del vector velocidad) deseada $\underline{\mathbf{U}}_p^* = \check{\mathbf{T}}\underline{\mathbf{U}}_c^*$ conteniendo los ángulos de ascenso y de orientación óptimos

$$\underline{\mathbf{U}}_p^*(k) = [\gamma^k \quad \delta^k]^T \tag{8.1}$$

con $k = 1, \cdots, h_p$, que llevarán al vehículo hacia los *waypoints* deseados. Esta información es transmitida al controlador de actitud. Los ángulos γ y δ son los ángulos de ascenso y de orientación definidos en el capítulo 6. Por otro lado, el control de actitud dado el estado recibido del vehículo da como *output* la deflexión de las superficies de control y de la columna de propulsión que llevarán al UAV a moverse hacia los *waypoints* deseados. Para ello, dicho controlador recibe la secuencia de ángulos óptimos $\underline{\mathbf{U}}_p^*$ y los utiliza para configurar su vector de valores deseados. Aplicando el método INL-MPC-J al UAV virtual, el controlador de actitud computa la secuencia de control óptima $\mathbf{U}_p^* = \check{\mathbf{T}}\mathbf{U}_c^*$ que contiene las deflexiones óptimas de las superficies de control del UAV:

$$\mathbf{U}_p^*(k) = [thtl^k \quad \delta_e^k \quad \delta_a^k \quad \delta_r^k]^T \tag{8.2}$$

con $k = 1, \cdots, h_p$, que dirigirán al UAV hacía a los *waypoints* deseados. Sola-

mente se envía al UAV real el vector de entradas de control correspondiente al primer instante de muestreo, es decir $\mathbf{U}_p^*(1) = [thtl^1 \quad \delta_e^1 \quad \delta_a^1 \quad \delta_r^1]^T$. Finalmente, el UAV recibe dicho valor de control y lo aplica. El proceso se reinicia desplazando el horizonte de tiempo un paso hacia adelante al próximo instante de muestreo.

El controlador de actitud necesita minimizar la diferencia entre los ángulos de orientación (γ, δ) del UAV y sus valores deseados (γ_d, δ_d), como estas variables físicas no forman parte del vector de estados del UAV (ver Ec. (2.16)), el esquema INL-MPC no puede utilizarse. Sin embargo si dichas variables físicas se expresan matemáticamente en términos de las variables de estados, el esquema INL-MPC-J sí podría usarse. A continuación se presenta la metodología que posibilita esta última implementación.

8.2. Cómputo de los Ángulos de Ascenso y de Orientación de una Aeronave

En esta sección se obtienen la expresiones matemáticas de los ángulos de ascenso $\gamma = \bar{\gamma}(\mathbf{x})$ y de orientación en el plano $\delta = \bar{\delta}(\mathbf{x})$ en términos las variables de estado $\mathbf{x}$ del UAV.

El vector velocidad de un vehículo aéreo, $\mathbf{v}_T = [v_t \quad 0 \quad 0]^T$, se encuentra expresado en el W-*Frame*. Las componentes de dicho vector, expresadas en el NED-*Frame*, es decir $\mathbf{v}_{t_{NED}} = [v_{t_{NED_x}} \quad v_{t_{NED_y}} \quad v_{t_{NED_z}}]^T$, permiten obtener las expresiones para el ángulo de ascenso $\bar{\gamma}(\mathbf{x})$ y el ángulo de orientación en el plano $\bar{\delta}(\mathbf{x})$ del UAV. El vector $\mathbf{v}_{t_{NED}}$ en el NED-*Frame* se puede hallar mediante la realización de dos transformaciones consecutivas: 1) transformación de W-*Frame* a B-*Frame*, mediante la matriz de rotación $\mathbf{B}_{\mathrm{W}}^{\mathrm{B}}$ definida en la Ec. (2.22), y 2) transformación de B-*Frame* a NED-*Frame*, mediante la matriz de rotación $\mathbf{B}_{\mathrm{B}}^{\mathrm{NED}}$, la cual puede hallarse tomando la transpuesta de la matriz de rotación definida en la Ec. (2.20), es decir:

$$\mathbf{v}_{t_{NED}} = \mathbf{B}_{\mathrm{B}}^{\mathrm{NED}} \mathbf{B}_{\mathrm{W}}^{\mathrm{B}} \mathbf{v}_t \tag{8.3}$$

Haciendo los productos matriciales correspondientes, cada una de las componentes del vector $\mathbf{v}_{t_{NED}}$ se pueden escribir como:

$$\begin{bmatrix} v_{t_{NED_x}} \\ v_{t_{NED_y}} \\ v_{t_{NED_z}} \end{bmatrix} = \begin{bmatrix} \mathrm{c}_\theta\mathrm{c}_\psi\mathrm{c}_\alpha\mathrm{c}_\beta + (\mathrm{s}_\phi\mathrm{s}_\theta\mathrm{c}_\psi - \mathrm{c}_\phi\mathrm{s}_\psi)\mathrm{s}_\beta + (\mathrm{s}_\phi\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{c}_\psi)\mathrm{s}_\alpha\mathrm{c}_\beta \\ \mathrm{c}_\theta\mathrm{s}_\psi\mathrm{c}_\alpha\mathrm{c}_\beta + (\mathrm{s}_\phi\mathrm{s}_\theta\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{c}_\psi)\mathrm{s}_\beta + (\mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi)\mathrm{s}_\alpha\mathrm{c}_\beta \\ -\mathrm{s}_\theta\mathrm{c}_\alpha\mathrm{c}_\beta + \mathrm{s}_\phi\mathrm{c}_\theta\mathrm{s}_\beta + \mathrm{c}_\phi\mathrm{c}_\theta\mathrm{s}_\alpha\mathrm{c}_\beta \end{bmatrix} v_t \quad (8.4)$$

La tasa de ascenso del UAV, como se describe en [8], se puede calcular como:

$$\dot{h} = -v_t \mathrm{s}_{\bar{\gamma}} \quad (8.5)$$

donde el signo negativo se debe a la elección $h = -z_\mathrm{D}$. $v_{t_{NED_z}}$ también representa la tasa de ascenso del UAV, por lo tanto se tiene que $\dot{h} = v_{t_{NED_z}}$. En consecuencia, se tiene que:

$$-v_t \mathrm{s}_{\bar{\gamma}} = (-\mathrm{s}_\theta\mathrm{c}_\alpha\mathrm{c}_\beta + \mathrm{s}_\phi\mathrm{c}_\theta\mathrm{s}_\beta + \mathrm{c}_\phi\mathrm{c}_\theta\mathrm{s}_\alpha\mathrm{c}_\beta)v_t \quad (8.6)$$

de donde se puede obtener la expresión para el ángulo de ascenso del UAV $\bar{\gamma}(\mathbf{x})$ como sigue:

$$\bar{\gamma}(\mathbf{x}) = \mathrm{asin}(\mathrm{s}_\theta\mathrm{c}_\alpha\mathrm{c}_\beta - \mathrm{s}_\phi\mathrm{c}_\theta\mathrm{s}_\beta - \mathrm{c}_\phi\mathrm{c}_\theta\mathrm{s}_\alpha\mathrm{c}_\beta) \quad (8.7)$$

El ángulo de orientación en el plano del UAV, $\bar{\delta}(\mathbf{x})$, se puede computar a partir del ángulo formado entre $v_{t_{NED_x}}$ y $v_{t_{NED_y}}$, es decir:

$$t_{\bar{\delta}} = \frac{v_{t_{NED_y}}}{v_{t_{NED_x}}} \quad (8.8)$$

de donde se puede obtener el ángulo $\bar{\delta}(\mathbf{x})$ como sigue:

$$\bar{\delta}(\mathbf{x}) = \mathrm{atan}\left(\frac{\mathrm{c}_\theta\mathrm{s}_\psi\mathrm{c}_\alpha\mathrm{c}_\beta + (\mathrm{s}_\phi\mathrm{s}_\theta\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{c}_\psi)\mathrm{s}_\beta + (\mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi)\mathrm{s}_\alpha\mathrm{c}_\beta}{\mathrm{c}_\theta\mathrm{c}_\psi\mathrm{c}_\alpha\mathrm{c}_\beta + (\mathrm{s}_\phi\mathrm{s}_\theta\mathrm{c}_\psi - \mathrm{c}_\phi\mathrm{s}_\psi)\mathrm{s}_\beta + (\mathrm{s}_\phi\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{c}_\psi)\mathrm{s}_\alpha\mathrm{c}_\beta}\right) \quad (8.9)$$

Las expresiones de Ecs. (8.7) y (8.9) serán utilizadas, como se verá más adelante, en el sistema de control de actitud del UAV.

8.3. Configuración de Variables Físicas y otros Parámetros para el Uso de INL-MPC e INL-MPC-J

Se asume que la función vectorial $\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ que define la dinámica del UAV es la presentada en Ec. (2.38). Tanto para el controlador de navegación como para el controlador de actitud, se adoptaron los siguientes parámetros: $h_p = 20$, $h_c = 17$, $\Delta T_s = 0{,}5$. La selección de h_p y h_c se realizó empíricamente, observando la ausencia de cambios significativos en la respuesta de los sistemas al aumentar dichos valores por encima de los adoptados.

Para el diseño del controlador de navegación, las matrices de pesos $\tilde{\mathbf{Q}}_x^k$ y $\tilde{\mathbf{R}}_u^k$ se adoptan según las Ecs. (7.1) y (7.2) con el fin de ponderar adecuadamente los estados físicos y las variaciones en las entradas de control del modelo de vehículo reducido tipo partícula. Los valores de estados típicos adoptados para dicho modelo de vehículo son los siguientes: $x_{Typ} = 100\,[\text{m}]$, $y_{Typ} = 100\,[\text{m}]$ y $z_{Typ} = 10\,[\text{m}]$. Los valores de ángulos de orientación y de ascenso típicos adoptados son: $\delta_{Typ} = 15\,[\text{deg}]$ y $\gamma_{Typ} = 15\,[\text{deg}]$, respectivamente. En cuanto a las restricciones del modelo de vehículo reducido tipo partícula, solamente se limitó la velocidad de cambio de las entradas de control, es decir, $\dot{\gamma}_m = -15\,[\text{deg}/\text{seg}]$, $\dot{\gamma}_M = 15\,[\text{deg}/\text{seg}]$, $\dot{\delta}_m = -15\,[\text{deg}/\text{seg}]$ y $\dot{\delta}_M = 15\,[\text{deg}/\text{seg}]$.

Para el diseño del controlador de actitud, las matrices de pesos $\tilde{\mathbf{Q}}_{\mathscr{C}}^k$ y $\tilde{\mathbf{Q}}_{\mathscr{G}}^k$ también se normalizan según las Ecs. (7.1) y (7.2). Los valores típicos y de restricciones adoptados para el UAV son los detallados en la sección 7.2 del capítulo anterior.

Tanto para el controlador de navegación como para el controlador de actitud, las matrices de pesos del problema de optimización se definen para cada aplicación, seleccionándose empíricamente los valores de ponderación.

8.4. Trayectorias de Navegación 2D

8.4.1. Navegación 2D hacia un *waypoint* $\mathbf{W}_1$

En esta subsección se describe el funcionamiento de ambos controladores (actitud y navegación) para comandar el UAV hacia un único *waypoint* $\mathbf{W}_1 = [4000 \;\; 2000]^T$ [m]. Se asume que el UAV se encuentra inicialmente volando a una altitud $h = 1000\,[\text{m}]$ y velocidad $v_t = 45\,[\text{m/seg}]$, y en las coordenadas $x_\text{N} = 1000\,[\text{m}]$ e $y_\text{E} = 0\,[\text{m}]$.

El vector de valores de deseados del controlador de navegación se configura como: $\underline{\mathbf{x}}_d(1) = x_d = \mathbf{W}_1(1) = 4000$ y $\underline{\mathbf{x}}_d(2) = y_d = \mathbf{W}_1(2) = 2000$. El valor deseado de la velocidad de cambio de la orientación $\underline{\dot{\mathbf{u}}}_d = \dot{\delta}_d$ se configuró de la siguiente manera: $\underline{\dot{\mathbf{u}}}_d = 0$ si $\|\underline{\mathbf{x}}_d - \underline{\mathbf{x}}\|_2 > 100\,[\text{m}]$ y $\underline{\dot{\mathbf{u}}}_d = 18\,[\text{deg/seg}]$ en caso contrario. Esto se escogió así para que, una vez alcanzado un entorno del *waypoint* $\mathbf{W}_1$, la trayectoria de navegación describa una circunferencia en el plano, de radio constante que cruce dicho *waypoint*. Los pesos adoptados para las matrices involucradas en el proceso de optimización son los siguientes: $\mathbf{Q}_x^k(1,1) = 10$, $\mathbf{Q}_x^k(2,2) = 10$, $\mathbf{R}_u^k(1,1) = 5$.

El UAV autónomo envía al controlador de navegación su vector de estados actual $\mathbf{x}^0$. Con estos datos, el método INL-MPC presente en el controlador de navegación obtiene la secuencia de control óptimo $\underline{\mathbf{U}}_p^*$. Para el caso 2D, dicha secuencia de control óptimo contiene los ángulos de orientación en el plano horizontal, en consecuencia la Ec. (8.1) se reduce a:

$$\underline{\mathbf{U}}_p^*(k) = \delta^k \tag{8.10}$$

Los ángulos $\underline{\mathbf{U}}_p^*(k)$ son los que llevarán al vehículo hacia $\mathbf{W}_1$. Esta información es transmitida al controlador de actitud.

La función $\mathscr{C}(\mathbf{x})$ requerida por el método INL-MPC-J es la siguiente:

$$\mathscr{C}(\mathbf{x}) = \begin{bmatrix} \bar{\delta}(\mathbf{x}) \\ \mathbf{x} \end{bmatrix} \tag{8.11}$$

donde $\bar{\delta}(\mathbf{x})$ es la función no-lineal definida en la Ec. (8.9) que determina el

ángulo de orientación en el plano del UAV y $\mathbf{x}$ es el vector de estados del UAV definido en Ec. (2.16). Asimismo, se asume que la función $\mathscr{G}(\dot{\mathbf{u}})$ utilizada en problema de optimización de Ec. (5.21) es

$$\mathscr{G}(\dot{\mathbf{u}}) = \dot{\mathbf{u}} \tag{8.12}$$

Los vectores de valores deseados $\tilde{\mathbf{c}}_d$ y $\tilde{\mathbf{g}}_d$ se configuran de la siguiente manera: $\tilde{\mathbf{c}}_d = [\mathbf{c}_d^1 \;\; \mathbf{c}_d^2 \;\; \cdots \;\; \mathbf{c}_d^{h_p}]^T$ con $\mathbf{c}_d^k = [\underline{\mathbf{U}}_p^*(k) \;\; \mathbf{x}_d^k]^T$, siendo $\underline{\mathbf{U}}_p^*(k)$ el vector definido en Ec. (8.10) y $\mathbf{x}_d^k$ el k-*ésimo* vector de estados deseados del UAV autónomo cuyas componentes de importancia son $\mathbf{x}_d^k(1) = v_{t_d} = 45$, $\mathbf{x}_d^k(3) = \beta_d = 0$, $\mathbf{x}_d^k(4) = \phi_d = 0$ y $\mathbf{x}_d^k(12) = h_d = 1000$. Como se desea obtener entradas de control con mínima variación, el vector $\tilde{\mathbf{g}}_d$ se asume nulo. La matriz $\mathbf{Q}_{\mathscr{C}}^k$ se configura como sigue: $\mathbf{Q}_{\mathscr{C}}^k(1,1) = 1$, $\mathbf{Q}_{\mathscr{C}}^k(2,2) = 50$, $\mathbf{Q}_{\mathscr{C}}^k(4,4) = 5$, $\mathbf{Q}_{\mathscr{C}}^k(5,5) = 0{,}1$, $\mathbf{Q}_{\mathscr{C}}^k(13,13) = 100$. Los pesos de la matriz de entradas se adoptan como: $\mathbf{Q}_{\mathscr{G}}^k(1,1) = 5$, $\mathbf{Q}_{\mathscr{G}}^k(2,2) = 5$, $\mathbf{Q}_{\mathscr{G}}^k(3,3) = 5$, $\mathbf{Q}_{\mathscr{G}}^k(4,4) = 5$, con $k = 1, \cdots, h_p$. El ángulo β ha sido pesado para que el UAV autónomo tenga mínimo deslizamiento.

Con sus parámetros configurados como se indica arriba, el sistema de control de actitud computa la secuencia de control $\mathbf{U}_p^*$ y envía al UAV las entradas de control óptimas correspondientes al primer instante de muestreo, es decir $\mathbf{U}_p^*(1)$ para que el UAV lo aplique y su ángulo de orientación en el plano $\bar{\delta}(\mathbf{x})$ se aproxime al ángulo de orientación en el plano δ computado por el sistema de control de navegación. Como se puede apreciar, el proceso de generación y seguimiento de trayectorias de navegación presentado es dinámico debido al constante intercambio de datos entre el UAV autónomo y los controladores, se realiza *online*, y permite además, considerar las restricciones propias de la dinámica de los modelos utilizados.

En la Fig. 8.2 se puede observar la trayectoria real del UAV. Como se puede ver, la misma alcanza el *waypoint* $\mathbf{W}_1$ deseado, describiendo luego una circunferencia de radio constante que cruza el entorno de dicho *waypoint*. En la figura mencionada, también se muestra el vector velocidad inicial $\mathbf{v}$ del UAV que determina la dirección a partir de la cual comienza a moverse el vehículo. En el caso presentado, la misma apunta hacia el Este, es decir, $\delta^0 = 0\,[\text{deg}]$,

determinando así una trayectoria de navegación que, inicialmente, posee esa dirección. En la Fig. 8.3 se pueden observar las respuestas de las entradas de

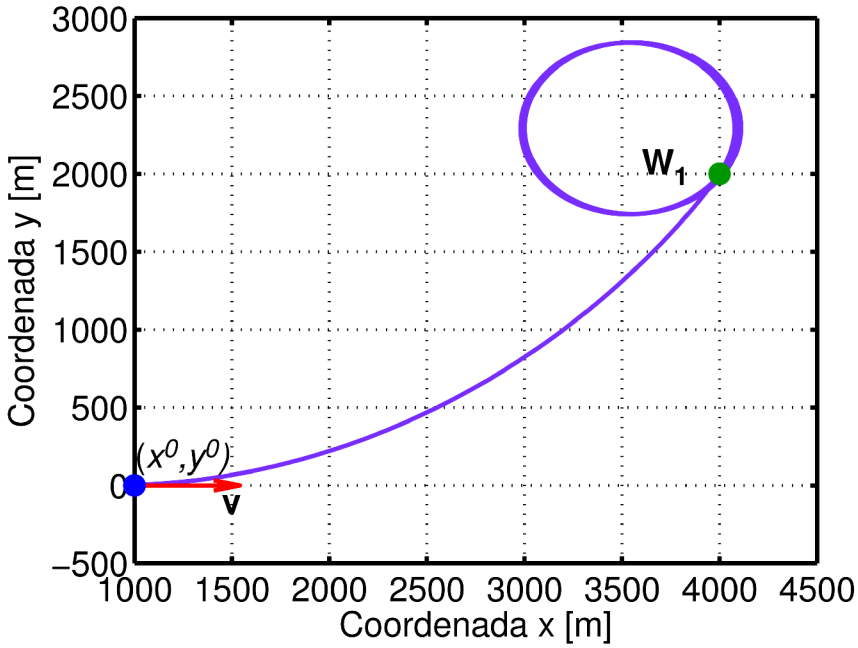
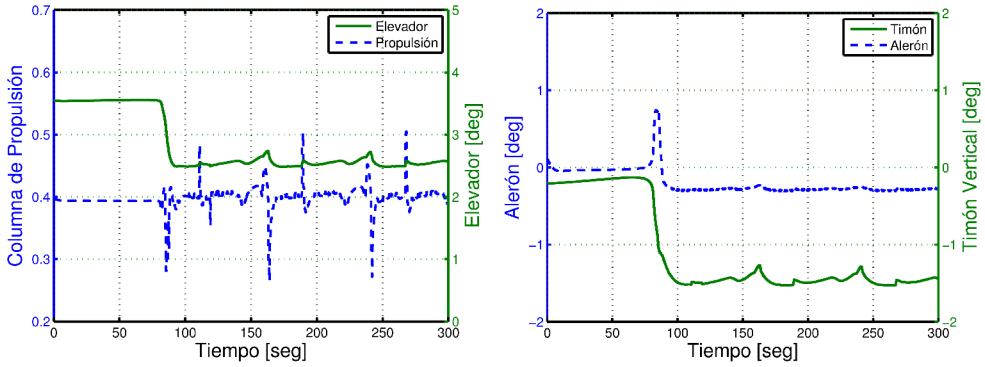


Figura 8.2: Trayectoria de navegación 2D con ángulo $\delta^0 = 0\,[\text{deg}]$ y un único *waypoint*

control del UAV autónomo. En la Fig. 8.3a se puede ver que tanto la columna de propulsión como el elevador presentan valores constantes hasta aproximadamente $t = 80\,[\text{seg}]$, instante en el cual el UAV alcanza el *waypoint* $\mathbf{W}_1$. Cuando el UAV comienza a describir la circunferencia de radio constante, ambas entradas de control modifican sus valores para tratar de conservar la altitud y la velocidad prácticamente constantes. En la Fig. 8.3b se muestran la deflexión del alerón y del timón vertical. Como se puede ver, estas entradas de control se mantienen prácticamente constantes y con valores pequeños hasta que el UAV alcanza $\mathbf{W}_1$. A partir de ese instante, ambas entradas de control modifican sus valores para que el UAV autónomo comience a describir el movimiento circular.



(a) Columna de Propulsión y Elevador (b) Alerón y Timón Vertical

Figura 8.3: Entradas de control: seguimiento trayectoria 2D con un único *waypoint*

8.4.2. Navegación 2D hacia múltiples *waypoints*

En la Fig. 8.4 se puede observar la trayectoria de navegación obtenida al utilizar dos *waypoints*: $\mathbf{W}_1 = [4000 \quad 2000]^T$ [m] y $\mathbf{W}_2 = [2000 \quad -1000]^T$ [m]. El controlador de navegación computa en cada instante de tiempo la secuencia de ángulos de orientación deseados δ_d que harían que el vehículo se dirija hacia los *waypoints* $\mathbf{W}_1$ y $\mathbf{W}_2$. La trayectoria de navegación resultante se obtiene mediante la concatenación de dos trayectorias de navegación parciales: $\mathbf{T}_1$ entre $\mathbf{x}^0$ y $\mathbf{W}_1$, y $\mathbf{T}_2$ entre $\mathbf{W}_1$ y $\mathbf{W}_2$. Como se puede observar en la Fig. 8.4, se obtiene como resultado una trayectoria de navegación óptima que pasa por el entorno de los *waypoints* $\mathbf{W}_1$ y $\mathbf{W}_2$, y que, una vez alcanzado éste último, describe una circunferencia de radio constante que cruza el *waypoint* final ($\mathbf{W}_2$ en este caso). En la Fig. 8.5 se muestran las evoluciones de las entradas de control del UAV comandadas por el controlador de actitud. Al igual que en el caso anterior, la columna de propulsión y el elevador se mueven en forma conjunta para mantener la altitud y la velocidad constante. En la Fig. 8.5b, se puede observar cómo el alerón y el timón vertical se mueven para modificar la trayectoria de vuelo del UAV para que el mismo se dirija en dirección a los *waypoints* deseados.

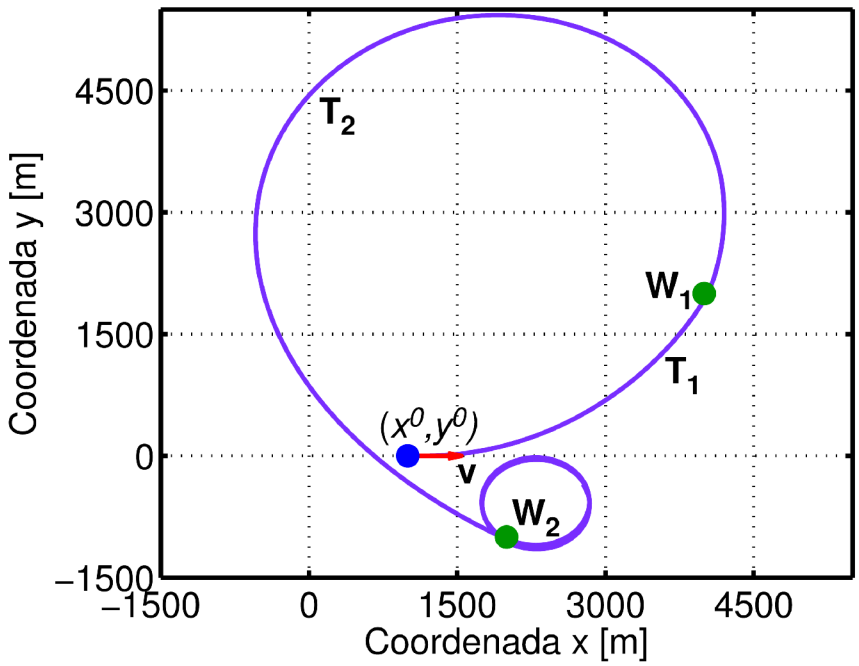
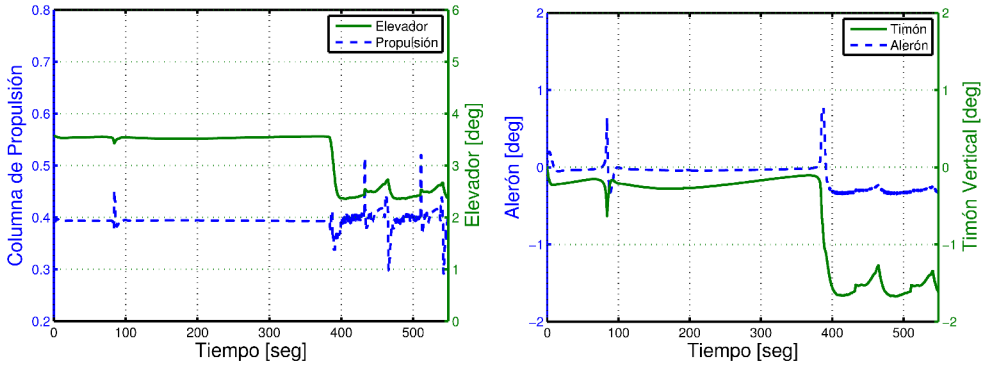


Figura 8.4: Trayectoria de navegación 2D con ángulo $\delta^0 = 0\,[\text{deg}]$ y dos *waypoints*



(a) Columna de Propulsión y Elevador (b) Alerón y Timón Vertical

Figura 8.5: Entradas de control: seguimiento trayectoria 2D con dos *waypoints*

8.5. Trayectorias de Navegación 3D hacia múltiples *waypoints*

En esta sección se discute el caso más general de navegación hacia múltiples *waypoints*. En particular se describe la configuración y funcionamiento

de ambos controladores (actitud y navegación) para realizar una maniobra de vuelo autónomo hacia dos *waypoints*: $\mathbf{W}_1 = [4000 \quad 2000 \quad 1500]^T$ [m] y $\mathbf{W}_2 = [2000 \quad -1000 \quad 1200]^T$ [m]. En este caso, también se asume que el UAV se encuentra volando inicialmente con una velocidad $v_t = 45$ [m/seg] y altitud $h = 1000$ [m] constantes, y que se encuentra en las coordenadas $x_{\mathrm{N}} = 1000$ [m], $y_{\mathrm{E}} = 0$ [m] y $h = 1000$ [m].

El vector de valores deseados del controlador de navegación se configura como: $\underline{\mathbf{x}}_d(1) = x_d = 4000$, $\underline{\mathbf{x}}_d(2) = y_d = 2000$ y $\underline{\mathbf{x}}_d(3) = z_d = 1500$, para el tramo $\mathbf{T}_1$ y como $\underline{\mathbf{x}}_d(1) = x_d = 2000$, $\underline{\mathbf{x}}_d(2) = y_d = -1000$ y $\underline{\mathbf{x}}_d(3) = z_d = 1200$ para el tramo $\mathbf{T}_2$ (ver Fig. 8.6). En este caso, el controlador de navegación tiene dos *outputs*: el ángulo de ascenso γ y el ángulo de orientación en el plano δ. El valor deseado de sus velocidades de cambio $\dot{\underline{\mathbf{u}}}_d = [\dot{\gamma}_d \quad \dot{\delta}_d]^T$ se adoptó de la siguiente manera: configurado el vector $\underline{\mathbf{x}}_d$ para el último tramo de trayectoria ($\mathbf{T}_2$ en este caso), $\dot{\underline{\mathbf{u}}}_d = 0$ si $\|\underline{\mathbf{x}}_d - \underline{\mathbf{x}}\|_2 > 100$ [m] y $\dot{\underline{\mathbf{u}}}_d(2) = \dot{\delta}_d = 18$ [deg /seg] en caso contrario. Esto se escogió así para que, una vez alcanzado el *waypoint* final deseado, la trayectoria describa una circunferencia de radio constante en el plano xy que cruce dicho *waypoint*. Los pesos involucrados en el proceso de optimización son los siguientes: $\mathbf{Q}_x^k(1,1) = 10$, $\mathbf{Q}_x^k(2,2) = 10$, $\mathbf{Q}_x^k(3,3) = 200$, $\mathbf{R}_{\dot{u}}^k(1,1) = 5$, $\mathbf{R}_{\dot{u}}^k(2,2) = 5$.

El UAV autónomo envía al controlador de navegación su vector de estados actual $\mathbf{x}^0$. Con estos datos, el método INL-MPC presente en el controlador de navegación obtiene la secuencia de control óptimo $\underline{\mathbf{U}}_p^*$ conteniendo los ángulos de ascenso y de orientación óptimos definidos en Ec. (8.1) que llevarán al vehículo primero hacia $\mathbf{W}_1$ y luego hacia $\mathbf{W}_2$. Esta información es transmitida al controlador de actitud.

La función $\mathscr{C}(\mathbf{x})$ requerida por el método INL-MPC-J es la siguiente:

$$\mathscr{C}(\mathbf{x}) = \begin{bmatrix} \bar{\gamma}(\mathbf{x}) \\ \bar{\delta}(\mathbf{x}) \\ \mathbf{x} \end{bmatrix} \tag{8.13}$$

donde $\bar{\gamma}(\mathbf{x})$ la función no-lineal definida en la Ec. (8.7) que determina el ángulo de ascenso del UAV, $\bar{\delta}(\mathbf{x})$ la función no-lineal definida en la Ec. (8.9) que

determina el ángulo de orientación en el plano del UAV y $\mathbf{x}$ es el vector de estados del UAV definido en Ec. (2.16). Asimismo, se asume que la función $\mathscr{G}(\dot{\mathbf{u}})$ del problema de optimización de Ec. (5.21) es

$$\mathscr{G}(\dot{\mathbf{u}}) = \dot{\mathbf{u}} \tag{8.14}$$

Los vectores de valores deseados $\tilde{\mathbf{c}}_d$ y $\tilde{\mathbf{g}}_d$ se configuran como: $\tilde{\mathbf{c}}_d = [\mathbf{c}_d^1 \;\; \mathbf{c}_d^2 \;\; \cdots \;\; \mathbf{c}_d^{h_p}]^T$ con $\mathbf{c}_d^k = [\underline{\mathbf{U}}_p^*(k) \;\; \mathbf{x}_d^k]^T$, siendo $\underline{\mathbf{U}}_p^*(k)$ el vector definido en Ec. (8.1) y $\mathbf{x}_d^k$ el k-*ésimo* vector de estados deseados del UAV autónomo cuyas componentes de importancia son $\mathbf{x}_d^k(1) = v_{t_d} = 45$, $\mathbf{x}_d^k(3) = \beta_d = 0$ y $\mathbf{x}_d^k(4) = \phi_d = 0$. Como se desea obtener entradas de control con mínima variación, el vector $\tilde{\mathbf{g}}_d$ se asume nulo. La matriz $\mathbf{Q}_{\mathscr{C}}^k$ se configura como sigue: $\mathbf{Q}_{\mathscr{C}}^k(1,1) = 20$, $\mathbf{Q}_{\mathscr{C}}^k(2,2) = 1$, $\mathbf{Q}_{\mathscr{C}}^k(3,3) = 50$, $\mathbf{Q}_{\mathscr{C}}^k(5,5) = 5$, $\mathbf{Q}_{\mathscr{C}}^k(6,6) = 0{,}1$. Los pesos de la matriz de entradas se adoptan como: $\mathbf{Q}_{\mathscr{G}}^k(1,1) = 5$, $\mathbf{Q}_{\mathscr{G}}^k(2,2) = 5$, $\mathbf{Q}_{\mathscr{G}}^k(3,3) = 5$, $\mathbf{Q}_{\mathscr{G}}^k(4,4) = 5$, con $k = 1, \cdots, h_p$. El ángulo β ha sido pesado para que el UAV realice una maniobra con mínimo deslizamiento.

Con sus parámetros configurados como se indica arriba, el sistema de control de actitud computa la secuencia de control $\mathbf{U}_p^*$ y envía al UAV real las entradas de control óptimas correspondientes al primer instante de muestreo, es decir $\mathbf{U}_p^*(1)$ para que el UAV lo aplique y sus ángulos de ascenso y de orientación $\bar{\gamma}(\mathbf{x})$ y $\bar{\delta}(\mathbf{x})$ se aproximen a los ángulos de ascenso y de orientación γ y δ computados por sistema de control de navegación.

En la Fig. 8.6 se puede observar la trayectoria de navegación resultante. Como se puede ver, la misma atraviesa los *waypoints* $\mathbf{W}_1$ y $\mathbf{W}_2$ deseados. Una vez alcanzado $\mathbf{W}_2$, la trayectoria de navegación describe una circunferencia en el plano xy, de radio constante, que cruza el entorno del *waypoint* $\mathbf{W}_2$. En la figura mencionada, también se muestra el vector velocidad inicial $\mathbf{v}$ del UAV que determina la dirección a partir de la cual comienza a moverse el vehículo. En la Fig. 8.7 se muestran las evoluciones de las entradas de control del UAV. Como se puede notar en la Fig. 8.7a, cuando el UAV sigue la trayectoria ascendente, la columna de propulsión aumenta a su valor al máximo para que el mismo ascienda con una máxima tasa de ascenso. Una vez alcanzado el *waypoint* $\mathbf{W}_1$, la columna de propulsión disminuye su valor al mínimo para que

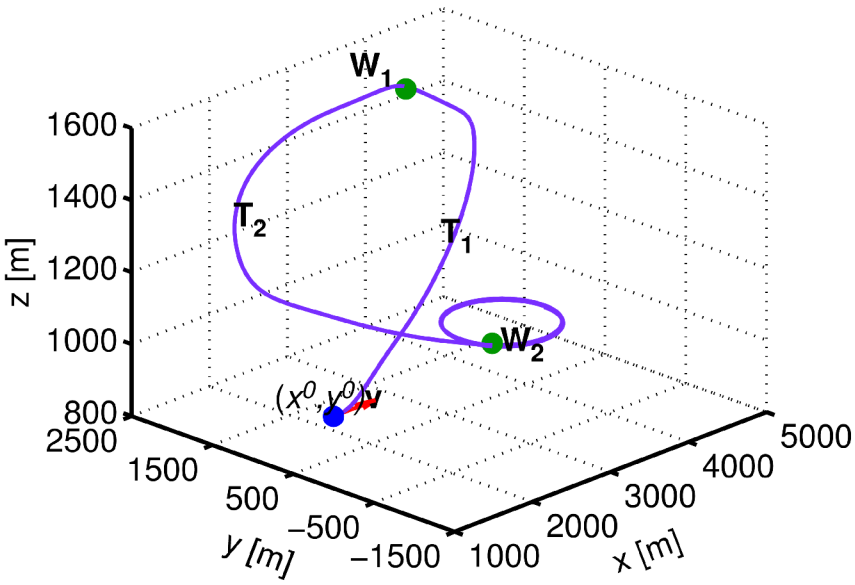
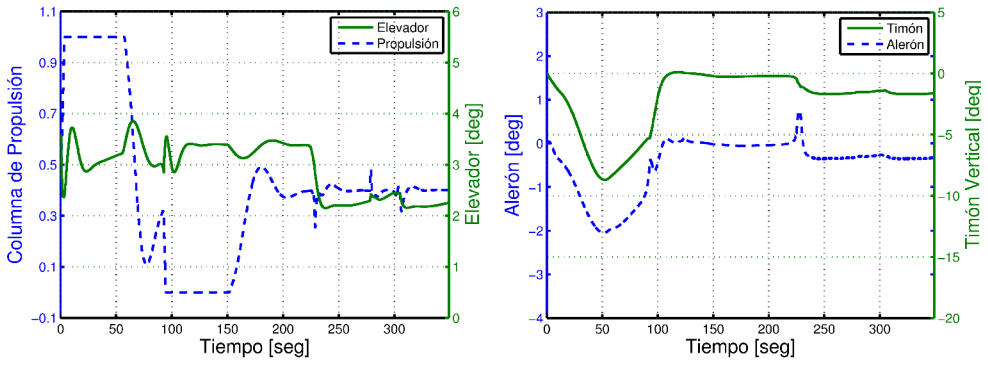


Figura 8.6: Trayectoria 3D óptima generada con dos *waypoints* y ángulo $\delta^0 = 0$ [deg]

el UAV descienda lo más rápido posible y cruce por $\mathbf{W}_2$. Una vez alcanzado el último *waypoint* tanto el elevador como la columna de propulsión presentan variaciones pequeñas para mantener constantes altitud y velocidad. En la Fig. 8.7b, se puede observar como el alerón y el timón vertical se mueven para dirigir al UAV autónomo en la dirección, primero de $\mathbf{W}_1$, luego de $\mathbf{W}_2$ y finalmente describir la circunferencia.



(a) Columna de Propulsión y Elevador (b) Alerón y Timón Vertical

Figura 8.7: Entradas de control: seguimiento trayectoria 3D con dos *waypoints*

8.6. Comentarios Adicionales

Al desacoplar el sistema de control de actitud y de navegación, es posible reducir el vector de estados $\mathbf{x}$ del UAV virtual, de dimensión $N_s = 12$, a un vector de estados reducido $\breve{\mathbf{x}}$ de dimensión $N_s = 9$, el cual se define como:

$$\begin{aligned} \breve{\mathbf{x}} &= [x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5 \;\; x_6 \;\; x_7 \;\; x_8 \;\; x_9]^T \\ &= [v_t \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; p \;\; q \;\; r]^T \end{aligned} \tag{8.15}$$

Como puede observarse, se han eliminado las variables de posición x_{N}, y_{E} y h del vector de estados del UAV virtual. En consecuencia, la dinámica del modelo reducido del UAV queda definida por la siguiente función vectorial:

$$\bar{\mathbf{f}}(\breve{\mathbf{x}}, \mathbf{u}) = \begin{bmatrix} (u\dot{u} + v\dot{v} + w\dot{w})/v_t \\ (u\dot{w} - w\dot{u})/(u^2 + w^2) \\ (\dot{v}v_t - v\dot{v}_t)/(v_t^2 \cos\beta) \\ p + \tan\theta(q\sin\phi + r\cos\phi) \\ q\cos\phi - r\sin\phi \\ (q\sin\phi + r\cos\phi)/\cos\theta \\ (c_1 r + c_2 p)q + c_3 M_x + c_4 M_z \\ c_5 pr - c_6(p^2 - r^2) + c_7 M_y \\ (c_8 p - c_2 r)q + c_4 M_x + c_9 M_z \end{bmatrix} \tag{8.16}$$

Al reducir el tamaño del vector de estados, se reduce el tamaño de la ecuación de estados y en consecuencia, se reduce el tamaño de las matrices involucradas en el método INL-MPC-J. En la Fig. 8.8 se puede observar el costo computacional demandado por el sistema de control de actitud, cuando se utilizan los modelos de UAVs de 9 y 12 estados. Al utilizar el UAV virtual reducido, es decir con 9 estados, se produjo una reducción en el tiempo de cómputo del controlador de actitud de aproximadamente el 40 %.

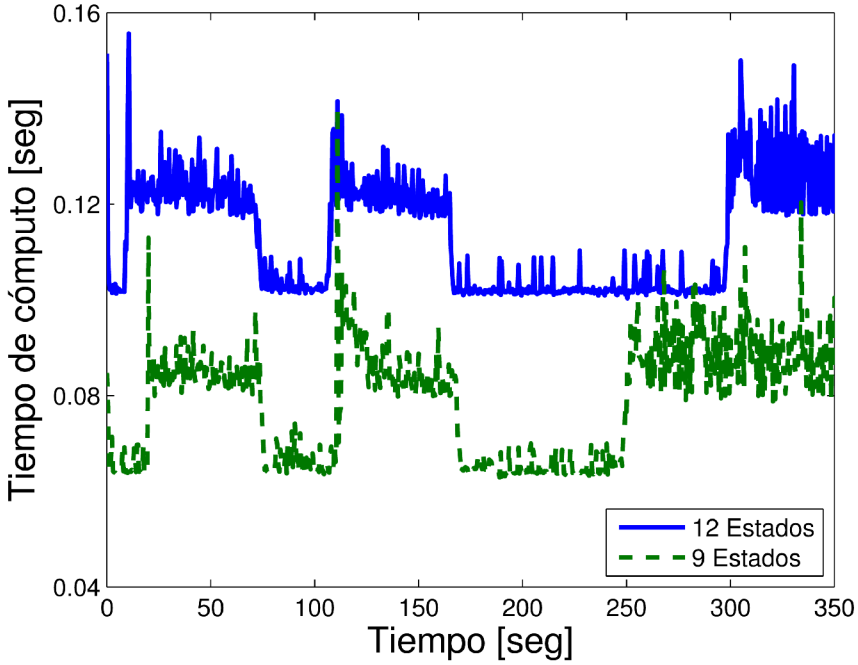


Figura 8.8: Comparación tiempos de cómputo

8.7. Conclusiones

En este capítulo, se desacopló el sistema de control de vuelo autónomo en dos controladores independientes, uno de navegación y otro de actitud. El controlador de navegación se diseñó utilizando INL-MPC con la metodología de generación de trayectorias de navegación presentada en el capítulo 6. El controlador de actitud se diseñó utilizando el método INL-MPC-J presentado en el capítulo 5. Para el diseño de éste último, fue necesario, determinar las expresiones matemáticas de los ángulos de ascenso y de orientación del UAV. Estas expresiones permitieron construir la función no-lineal $\mathscr{C}(\mathbf{x})$ utilizada en el método INL-MPC-J.

El desempeño de la metodología propuesta se demostró mediante la generación y seguimiento de diferentes trayectorias de navegación, tanto en 2D como en 3D. En todos los casos, las trayectorias de navegación óptimas obtenidas cruzan los *waypoints* especificados y, una vez alcanzado el último *waypoint*, describen una circunferencia en el plano xy de radio constante que cruza al mismo. Es importante destacar que, debido al continuo intercambio de datos

entre el UAV real y los controladores, el cómputo de la trayectoria de navegación es un proceso dinámico, se computa *online*, y además, se consideran las restricciones de los modelos utilizados.

Finalmente, se demostró que, el desacople del sistema de control de navegación y de actitud tiene como ventaja adicional la posibilidad de reducir el tamaño del vector de estados del UAV virtual, con lo que se logra una reducción adicional en el costo computacional del controlador de actitud.

Capítulo 9

Plataforma de Simulación, Visualización y Control de Vuelo

En este capítulo se presentan los aspectos de diseño de la plataforma de Simulación, Visualización y Control de vuelo (plataforma SVC) *Excalibur*. La misma posibilita el diseño, validación y evaluación de distintos sistemas de control automático. Además, permite la visualización en **tiempo real** del funcionamiento de los algoritmos de control aplicados a un determinado UAV virtual autónomo.

La organización de este capítulo es la siguiente: en el capítulo 9.1 se presenta una introducción a la plataforma SVC y los simuladores de vuelo. En en el capítulo 9.2 se presentan los aspectos de diseño de la plataforma. El capítulo 9.3 muestra cómo se diseñó y se implementó el módulo estación remota en la plataforma *Excalibur*. En el capítulo 9.4 se describe el UAV, mientras que en los capítulos 9.5 y 9.6 se presentan los sistemas de control de actitud y de navegación, respectivamente.

9.1. Introducción

La plataforma SVC *Excalibur* ha sido desarrollada como parte de esta tesis en el CIMEC [51, 52], en conjunto con el Ing. Pablo S. Rojas Fredini y el Dr. Alejandro Limache, para extender las capacidades con que cuentan los simuladores de vuelo convencionales. Estos últimos, en general, funcionan solo en modo manual. Utilizando un *joystick* o un teclado estándar, se pueden enviar entradas manuales de control para simular las deflexiones de las superficies aerodinámicas y de la columna de propulsión del UAV. Existen numerosos simuladores de vuelo cuya utilidad es muy variada. Se pueden dividir en dos grandes grupos:

- Simuladores de vuelo de acceso público, entre los cuales pueden encontrarse tanto versiones comerciales [53, 54] como versiones gratuitas [55]. Estas últimas tienen la ventaja de poseer su código fuente abierto y además se encuentran disponible en numerosas plataformas.

- Simuladores de vuelo de índole profesional (conocidos como *Level D Full Flight Simulator*), los cuales crean una réplica del avión y del medio incluyendo el movimiento propio del aeronave. Generalmente esto se logra disponiendo de una réplica de la cabina del avión y de un sistema de visualización dentro de una plataforma con movimiento. Son pocas las empresas que pueden construir estos simuladores, entre las cuales se pueden destacar [56, 57, 58].

La plataforma *Excalibur*, además del manejo manual, puede ser utilizada en modo automático ya que la misma cuenta con un módulo de control de actitud y de navegación, tanto unificado como desacoplado. Asimismo, podría ser utilizada tanto con modelos virtuales de UAVs como con vehículos aéreos reales. En el marco de la presente tesis, la plataforma *Excalibur* se utilizó con modelos virtuales de UAVs, permitiendo recrear por computadora el comportamiento de un avión real no-tripulado.

9.2. Diseño de la Plataforma

En esta sección se presentan los aspectos de diseño de la plataforma SVC *Excalibur*. En la Fig. 9.1 se puede observar un esquema sencillo de la plataforma de vuelo diseñada, siendo la misma configurada con un sistema de control de actitud y de navegación desacoplado, mientras que en la Fig. 9.2 se configura utilizando un sistema de control de actitud y de navegación unificado. En ambas figuras, además, se exhiben las conexiones existentes entre los distintos módulos (o sub-sistemas) que la componen. Para su diseño, se eligieron los

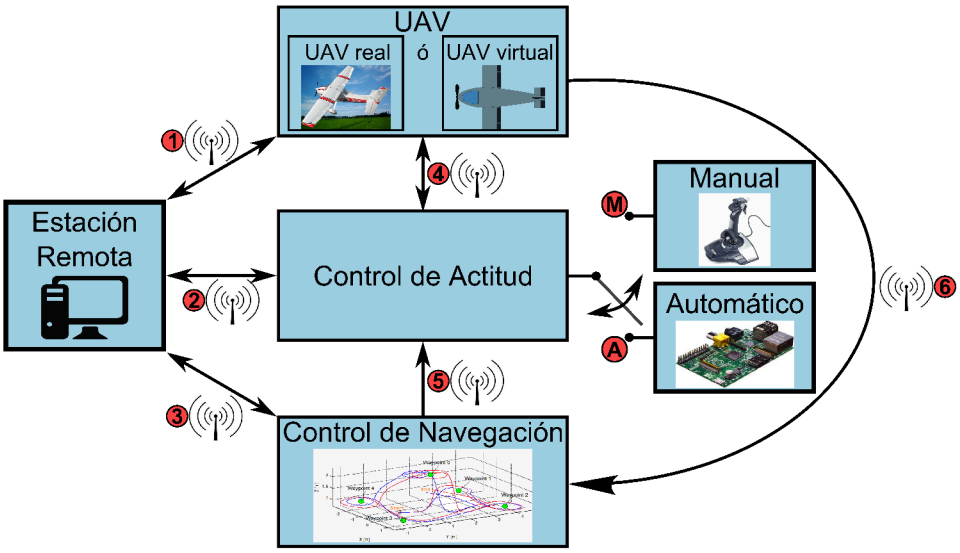


Figura 9.1: Esquema de la plataforma SVC *Excalibur* con sistema de control de navegación y de actitud desacoplado

lenguajes de programación C++ y C# y se optó por utilizar una estructura modular, que facilita la modificación independiente de cada uno de los módulos. Por ejemplo, si se desea evaluar un sistema de control de actitud basado en otras técnicas de control, simplemente se debe modificar el módulo **Control de Actitud** de Fig. 9.1. Además, esta estructura modular posibilita el desacople de la plataforma, permitiendo la ejecución de cada uno de los diferentes módulos en computadoras independientes. Esta propiedad es muy importante ya que, al implementar cada módulo en un procesador distinto, disminuye notablemente el costo computacional del sistema global, permitiendo la ejecución

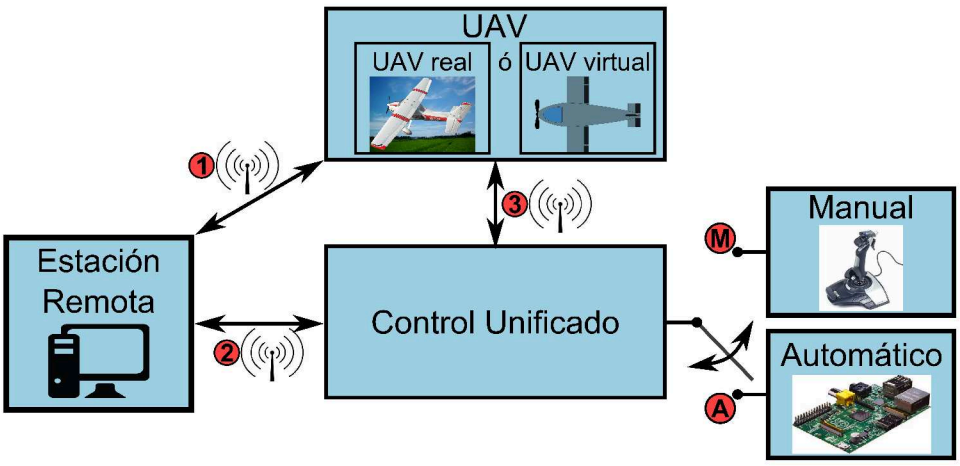


Figura 9.2: Esquema de la plataforma SVC *Excalibur* con sistema de control unificado

de la plataforma SVC en **tiempo real**.

Esta plataforma puede utilizarse tanto en modo manual (llave en la posición M de Figs. 9.1 y Fig. 9.2) como en modo automático (llave en la posición A de Figs. 9.1 y Fig. 9.2). Cuando se utiliza en modo manual, las entradas de control se comandan mediante *hardware*. Para el presente caso, las mismas se simulan utilizando un *Joystick* similar al que se utiliza en aviones de combate. El mismo permite comandar las deflexiones del elevador, alerones, timón vertical y también de la columna de propulsión. Eventualmente, es posible enviar estas entradas manuales a la plataforma SVC mediante un teclado estándar. Para administrar los dispositivos de entrada de *hardware* se utilizó la conocida librería multi-plataforma *OIS* [59], la cual permite leer el estado de los dispositivos y enviar efectos de *force-feedback* a aquellos que lo soporten. En la Fig. 9.3 se muestra una fotografía de la plataforma de vuelo, en la cual se puede observar el *Joystick* utilizado y la ejecución de la misma en dos computadoras diferentes. La utilización de la plataforma en modo automático, se analizará en detalle en la sección 9.5.

Como se puede observar en la Fig. 9.1, la plataforma SVC configurada con un sistema de control de actitud y navegación desacoplado, consta de cuatro módulos interconectados entre sí: 1) **Estación remota**, 2) **UAV**, 3) **Control de Actitud** y 4) **Control de Navegación**. La comunicación entre dichos

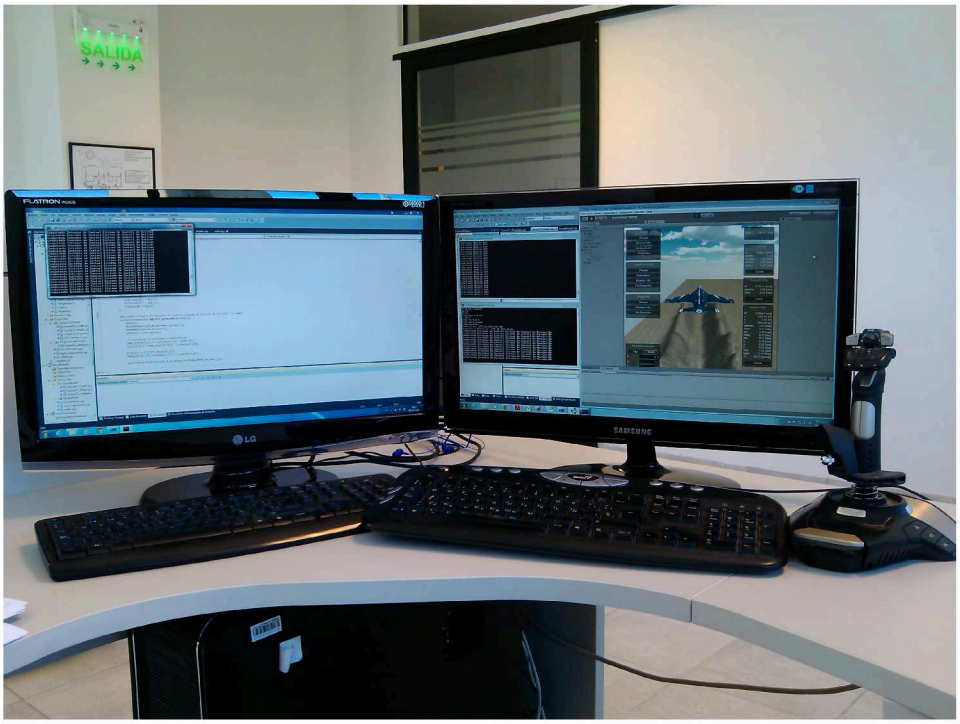


Figura 9.3: Fotografía de la plataforma SVC *Excalibur*

módulos se establece mediante una red de área local (LAN), utilizando protocolos de comunicación TCP y UDP. El envío y recepción de datos se realiza en forma asíncrona mediante la utilización de las funciones provistas para tal fin por las bibliotecas *Boost C++* [60]. Los enlaces existentes entre el módulo **Estación Remota**, el **UAV** y los módulos **Control de Actitud** y **Control de Navegación** (numerados como (1), (2) y (3), respectivamente) son bidireccionales, utilizándose el protocolo TCP para el envío de datos desde la **Estación Remota** hacia los módulos anteriormente mencionados, y el protocolo UDP para el envío desde el **UAV** y los módulos **Control de Actitud** y **Control de Navegación** hacia la **Estación Remota**. Los enlaces (4), (5) y (6) también utilizan el protocolo de comunicación UDP.

El intercambio de datos entre los distintos módulos se realiza de la siguiente manera: a través del enlace (1), el **UAV** envía su vector de estados $\mathbf{x}$ y su vector de entradas de control $\mathbf{u}$ a la **Estación Remota** para visualizar sus valores en la pantalla principal de la plataforma. La **Estación Remota** envía al **UAV**

comandos de control y de configuración. Mediante el enlace de comunicación (2), el módulo **Control de Actitud** envía el vector de entradas de control óptimas computadas a la **Estación Remota** para visualizar sus valores en la pantalla principal de la plataforma *Excalibur*. La **Estación Remota** envía al **Control de Actitud** comandos de control y de configuración. Con el enlace (3), el módulo **Control de Navegación** envía el vector de ángulos de control óptimo computado para ser visualizados en la pantalla principal de la plataforma SVC. La **Estación Remota** envía al **Control de Navegación** comandos de control y de configuración. A través del enlace de comunicación (4) el **UAV** envía al **Control de Actitud** sus vectores de estados y de controles actuales para que el módulo de **Control de Actitud**, estando configurado en su modo automático, es decir con la llave en la posición (A), compute la secuencia de entradas de control óptimas y luego la envíe al **UAV** para que éste último las aplique a su modelo de vehículo. Mediante el enlace (5), el módulo **Control de Navegación** envía al **Control de Actitud** la secuencia de ángulos de orientación y/o de ascenso óptimos computados para que el **Control de Actitud** configure su vector de valores deseados con los ángulos recibidos. Con el enlace (6) el **UAV** envía al módulo **Control de Navegación** sus vectores de estados y de controles actuales para que el mencionado módulo configure las variables que sean necesarias antes de iniciar el cómputo de las trayectorias de navegación.

A continuación se muestra un extracto del código utilizado para el intercambio de datos. El mismo se implementa en el **UAV** y en los módulos **Control de Actitud** y **Control de Navegación**.

```
1  // ***************************recepción de datos vía tcp****************************
2  void OnReceiveTCPCommand(const boost::system::error_code &err,
3                           const std :: size_t  bytes_transferred ){
4   ReceiveSocket->async_read_some(boost::asio:: buffer (&dataReceived, sizeof (dataReceived)),
5                                  OnReceiveTCPCommand);
6  }
7
8  void OnReceiveTCPConnection(const boost::system::error_code &err){
9   ReceiveSocket->async_read_some(boost::asio:: buffer (&dataReceived, sizeof (dataReceived)),
10                                 OnReceiveTCPCommand);
```

```
11 }
12
13 void ListenTCP(){
14  ReceiveAcceptor->async_accept(*ReceiveSocket,OnReceiveTCPConnection);
15  ioServiceTCP->run();
16 }
17
18 // ****************************envío asíncrono vía udp*****************************
19 void handle_sendData(const boost::system::error_code& err,std::size_t len){
20   if (err == 0){
21     memcpy(dataSent_Buf.data(),&dataSent,sizeof(dataSent));
22     SendSocket->open(ip::udp::v4());
23     SendSocket->async_send_to(boost::asio::buffer(dataSent_Buf),*Endpoint,
24                               handle_sendData);
25     SendSocket->close();
26   }
27 }
28
29 void ListenSendAsyncData(){
30   SendSocket->open(ip::udp::v4());
31   SendSocket->async_send_to(boost::asio::buffer(dataSent_Buf),*Endpoint,
32                             handle_sendData);
33   SendSocket->close();
34   ioServiceAsyncSend->run();
35 }
36
37 // ****************************recepción asíncrona vía udp************************
38 void handle_receiveData(const boost::system::error_code& err,std::size_t len){
39   if (err == 0){
40     ip::udp::endpoint temp;
41     memcpy(&dataReceived,dataReceived_Buf.data(),sizeof(dataReceived));
42     ReceiveSocket->async_receive_from(boost::asio::buffer(dataReceived_Buf),temp,0,
43                                       handle_receiveData);
44   }
45 }
46
47 void ListenReceiveAsyncData(){
48   ip::udp::endpoint temp;
49   ReceiveSocket->async_receive_from(boost::asio::buffer(dataReceived_Buf),temp,0,
```

```
50                                    handle_receiveData);
51    ioServiceAsyncReceive->run();
52 }
```

Las funciones **OnReceiveTCPCommand**, **OnReceiveTCPConnection** y **ListenTCP** se encargan de la recepción, vía protocolo TCP, de los datos enviados por la **Estación Remota** a los distintos sub-sistemas. La recepción de datos por protocolo TCP se realiza de la siguiente forma: *ioServiceTCP* es un objeto del tipo *io_service*. La clase *io_service* es quien proporciona la funcionalidad básica de entrada salida. El llamado a la función *run()* del objeto *ioServiceTCP* lanza el bucle de procesamiento de eventos y se queda "escuchando" la existencia de una conexión TCP a través de la función *OnReceiveTCPConnection*. Una vez establecida la conexión, cada vez que haya datos en el *buffer*, se realiza un llamado a la función *OnReceiveTCPCommand* y se copian los datos del *buffer* a la variable *dataReceived*.

Las funciones **handle_sendData** y **ListenSendAsyncData** manejan el envío asíncrono de datos vía protocolo UDP, cuyo funcionamiento, básicamente, es el siguiente: *ioServiceAsyncSend* es un objeto del tipo *io_service*. Al ejecutar la función *run()* de dicho objeto se lanza el bucle de procesamiento de eventos y se queda "escuchando" la existencia datos para enviar vía UDP. Si hay datos para enviar, se ejecuta la función *handle_SendData* y se copian los datos de la variable *dataSent* al *buffer* de salida *dataSent_Buf*.

Las funciones **handle_receiveData** y **ListenReceiveAsyncData** manejan la recepción asíncrona de datos vía protocolo UDP, cuyo funcionamiento, básicamente, es el que se detalla a continuación: *ioServiceAsyncReceive* es un objeto del tipo *io_service*. Al ejecutar la función *run()* de dicho objeto se lanza el bucle de procesamiento de eventos y se queda "escuchando" la existencia de datos en el *buffer* de entrada. Si hay datos, se ejecuta la función *handle_receiveData* y se copian los datos del *buffer* de entrada *dataReceived_Buf* a la variable *dataReceived*.

En las secciones que siguen se realiza una descripción de cada uno de los módulos que componen a la plataforma SVC.

9.3. Diseño del Módulo Estación Remota

La **Estación Remota** es el módulo en el cual se ejecuta el entorno de visualización. El mismo fue diseñado utilizando el software *Unity3D* [61] y, para la implementación de las funciones que el mismo utiliza, se utilizó el lenguaje de programación C#. Es este módulo el que permite visualizar el terreno y el UAV virtual.

El software de visualización desarrollado permite el envío de comandos de configuración desde la **Estación Remota** hacia el **UAV** y hacia los módulos **Control de Actitud** y **Control de Navegación**, como ser, por ejemplo, cambio de modo manual a automático y viceversa, modificación de parámetros del algoritmo de control, realización de maniobras pre-establecidas en vuelo autónomo, seguimiento de trayectorias de navegación, entre otros. Tanto el envío de comandos como la modificación de parámetros se realiza en tiempo de ejecución, lo cual es muy ventajoso ya que permite evaluar la respuesta de los diferentes módulos mientras se realiza una determinada simulación de vuelo. Además, el módulo **Estación Remota** procesa los datos recibidos de otros módulos, como por ejemplo los valores de estados y entradas de control, trayectorias de navegación generadas, movimiento del UAV, entre otros, para su posterior visualización en la pantalla de la plataforma. En síntesis, este módulo hace las veces de interfaz entre el usuario y la plataforma.

El envío de datos desde la **Estación Remota** hacia los restantes módulos se realiza utilizando el protocolo de comunicación TCP. La recepción de la información proveniente del **UAV** y de los módulos **Control de Actitud** y **Control de Navegación** se procesa en la **Estación Remota** utilizando el protocolo de comunicación UDP. A continuación se presenta el extracto de código, implementado en la **Estación Remota**, que permite el envío y recepción de datos.

```
1 // ****************************envío de datos vía tcp*****************************
2 void SendCommand (DataSend data){
3   if(tcpClient.Connected){
4     byte[] arr = StructureToByteArray(data);
5     netStream.Write(arr, 0, arr.Length);
6   }else{
```

```
7       Debug.Log("No se puede enviar comando al control, la conexión no fue abierta");
8     }
9   }
10
11  byte[] StructureToByteArray(DataSend obj){
12    byte [] bytes;
13    int doubleSize = 8;
14    int intSize = 4;
15    int boolSize = 1;
16    int size = (obj.X0.Length + obj.U0.Length + 1)*doubleSize + 2*boolSize;
17    int offset = 0;
18
19    bytes= new byte[size];
20    offset = 0;
21
22    Buffer.BlockCopy(obj.X0,0,bytes, offset ,obj.X0.Length*doubleSize);
23    offset += obj.X0.Length*doubleSize;
24
25    Buffer.BlockCopy(obj.U0,0,bytes, offset ,obj.U0.Length*doubleSize);
26    offset += obj.U0.Length*doubleSize;
27
28    Buffer.BlockCopy(BitConverter.GetBytes(obj.mass),0, bytes, offset ,doubleSize);
29    offset += doubleSize;
30
31    Buffer.BlockCopy(BitConverter.GetBytes(obj.Pausa),0, bytes, offset ,boolSize);
32    offset += boolSize;
33
34    Buffer.BlockCopy(BitConverter.GetBytes(obj.Auto),0, bytes, offset ,boolSize);
35    offset += boolSize;
36
37    return bytes;
38  }
```

Las funciones **SendCommand** y **StructureToByteArray** se utilizan para enviar datos vía protocolo TCP al **UAV** y a los módulos **Control de Actitud** y **Control de Navegación**. Cuando se detecta una conexión TCP y hay datos para enviar, la función *SendCommand* se ejecuta y llama a la función *StructureToByteArray*, la cual se encarga de copiar los datos a enviar, mediante la función *BlockCopy*, en un arreglo de bytes. Luego, mediante la sentencia

netStream.Write se envía el arreglo de bytes *arr*.

```
// ***************************recepción de datos vía udp***************************
void Update(){
  if(receiveBytes!=null){
    data = new Data();
    ByteArrayToStructure(receiveBytes, ref data);
    receiveBytes=null;
  }
}

void ByteArrayToStructure(byte [] bytearray, ref Data obj){
  int doubleSize = 8;
  int offset = 0;

  Buffer.BlockCopy(bytearray, offset ,obj.Xstate,0, obj.Xstate.Length*doubleSize);
  offset += obj.Xstate.Length*doubleSize;

  Buffer.BlockCopy(bytearray, offset ,obj.Uctrl,0, obj.Uctrl.Length*doubleSize);
  offset += obj.Uctrl.Length*doubleSize;
}
```

Las funciones **Update** y **ByteArrayToStructure** se utilizan para la recepción, vía protocolo UDP, de los datos enviados por el **UAV** y por los módulos **Control de Actitud** y **Control de Navegación**. La función *Update* se ejecuta periódicamente. Cuando el arreglo de *bytes* denominado *receiveBytes* contiene datos, se llama a la función *ByteArrayToStructure*, la cual se encarga de copiar los datos recibidos desde el arreglo de *bytes* a las variables correspondientes.

En la Fig. 9.4 se muestra la pantalla principal de la plataforma SVC desarrollada. A continuación se describen brevemente cada una de las partes que la componen. En el recuadro (1) se puede apreciar el tiempo de simulación transcurrido. En el recuadro (2) se muestran los botones, asociados al sistema de control de actitud, que permiten la comunicación entre la **Estación Remota** y el módulo **Control de Actitud**. Por ejemplo, el botón **Pausar** permite detener la ejecución del sistema de control de actitud. El botón **Mostrar Info** despliega la ventana (5), en la cual se visualizan los valores que poseen las

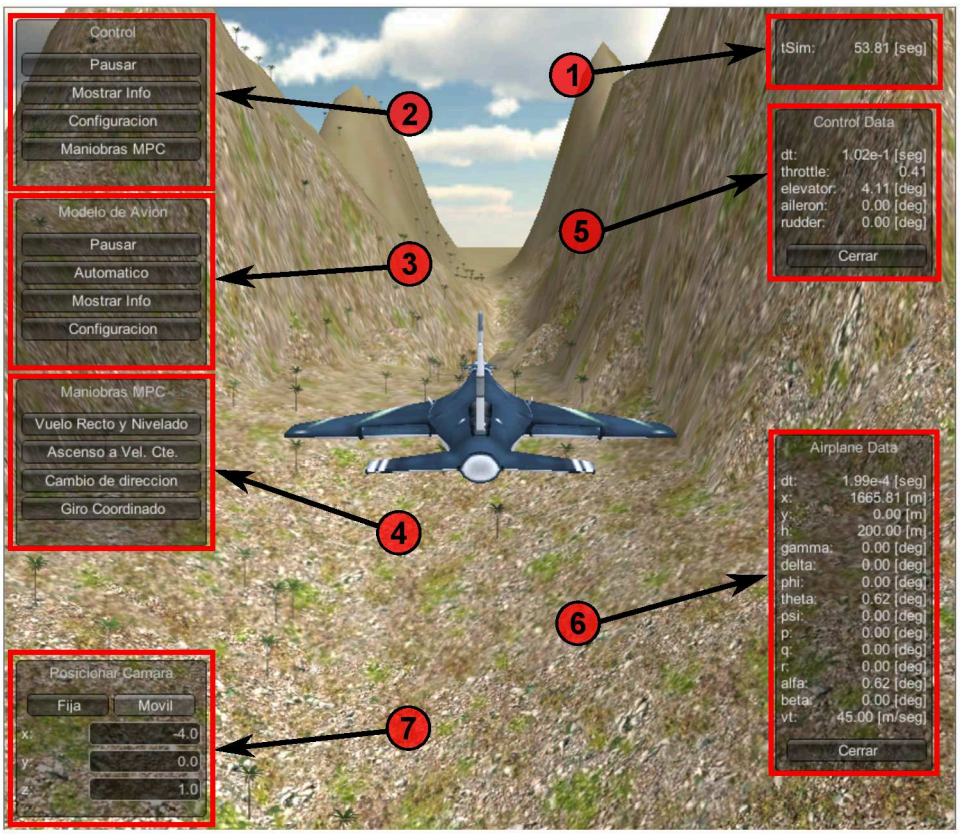


Figura 9.4: Pantalla principal de la plataforma SVC *Excalibur*

entradas de control computadas. Además se muestra el tiempo que demanda la ejecución del lazo de control. El botón **Configuración** permite modificar los parámetros de los métodos de control INL-MPC e INL-MPC-J, como por ejemplo, configurar diferentes valores deseados, modificar los valores de los pesos de las matrices $\mathbf{Q}_x^k$, $\mathbf{R}_{\dot{u}}^k$, $\mathbf{Q}_{\mathscr{C}}^k$ y $\mathbf{Q}_{\mathscr{G}}^k$, horizontes de predicción h_p y de control h_c, entre otros. Al presionar el botón **Maniobras MPC** se alterna entre el sistema de control de actitud y navegación unificado y desacoplado. En el recuadro 4 se muestra el panel de de control que permite seleccionar diferentes maniobras pre-establecidas. El recuadro 3 contiene los botones que permiten la comunicación entre la **Estación Remota** y el **UAV**. Al presionar el botón **Pausar**, se detiene la ejecución del UAV virtual. El botón **Automático** permite alternar entre el modo manual y el modo automático. **Mostrar Info** despliega la ventana 6, en la cual muestran los valores que toman las

variables de estado y otras variables de interés. Allí también se muestra el tiempo de cómputo que demanda el lazo de simulación del UAV virtual. El botón **Configuración** permite modificar las condiciones iniciales $\mathbf{x}^0$ y $\mathbf{u}^0$ del UAV virtual. También es posible modificar, desde allí, el valor de su masa. La ventana 7 permite modificar la posición de la cámara, pudiendo además alternar entre una cámara fija o una cámara móvil.

En el recuadro 8 de la Fig. 9.5 se muestra el panel de control asociado a la generación de trayectorias de navegación. El botón **Pausar** permite detener la ejecución del controlador de navegación. Como en los casos anteriores, el botón **Mostrar Info** despliega la ventana 10 en donde se detallan los valores que toman los ángulos de control (ángulos de ascenso y de orientación) y el algoritmo INL-MPC. Asimismo, se muestra el tiempo de cómputo que demanda la ejecución del lazo de control. El botón **Configuración** permite definir un listado de *waypoints* (hasta un máximo de 10). También permite modificar los parámetros característicos del algoritmo de control INL-MPC como ser las matrices de peso $\mathbf{Q}_x^k$ y $\mathbf{R}_u^k$, horizontes de predicción h_p y de control h_c, entre otros. La trayectoria de navegación óptima computada se puede visualizar también en la Fig. 9.5, siendo la misma señalada como 9.

9.4. Diseño del Módulo UAV

El **UAV** describe el modelo de vehículo que se utiliza para realizar las diferentes simulaciones de vuelo. Las ecuaciones de movimiento, definidas en la Ec. (2.38), se procesan dentro de este módulo mediante la integración temporal de las mismas. Como resultado, se obtiene la evolución del vector de estados $\mathbf{x}$ del UAV virtual, ante la entrada de control $\mathbf{u}$. A continuación se muestra un extracto del código implementado en el **UAV** que permite simular el modelo del vehículo aéreo para obtener así la evolución de las variables de su vector de estados.

```
1 void Simulate(double _dt){
2   if (!pausa){
3     tiempo+=_dt;
4     RungeKutta.Integrate(Xstate, Uctrl, _dt);
```

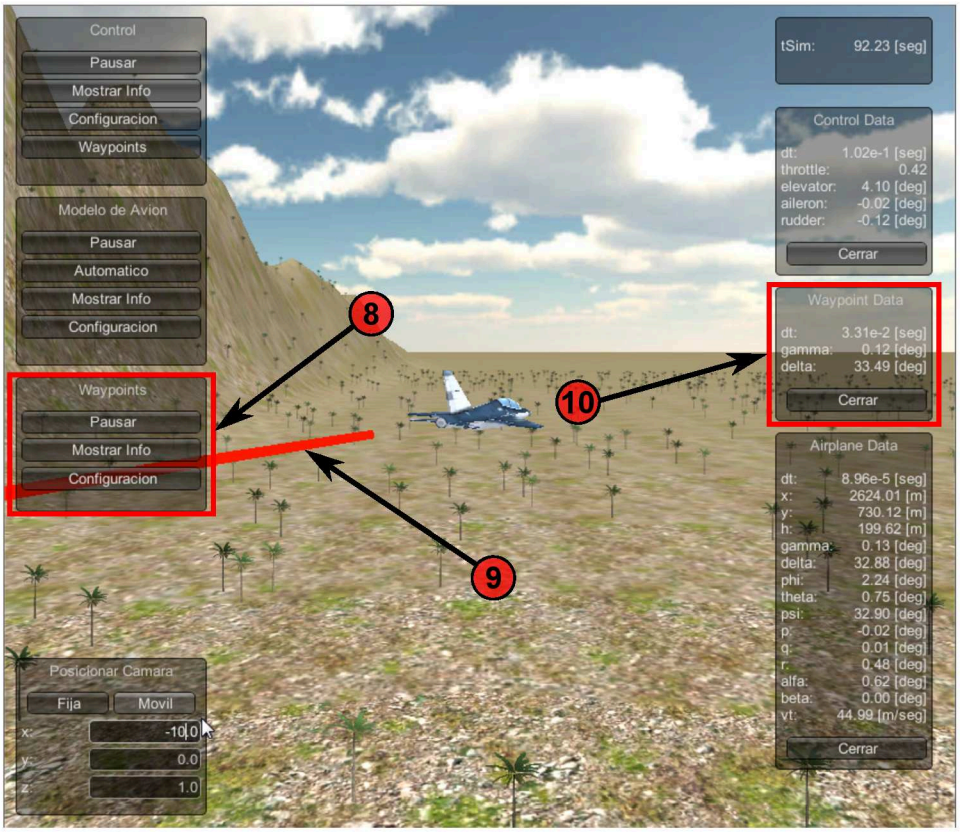


Figura 9.5: Pantalla principal de la plataforma SVC *Excalibur*

```
5       computeRates(Xstate,Uctrl,XstateDot);
6     }
7   }
8
9   void computeRates(DTensor1 &_X, DTensor1 &_U, DTensor1 &_Xdot){
10    Index <'i'> iT; Index <'j'> jT; Index <'k'> kT;
11    double dynPres, ps, rho, sigma;
12    double wingChord = pCessna172_Properties->wingChord;
13    double wingSpan = pCessna172_Properties->wingSpan;
14    double wingArea = pCessna172_Properties->wingArea;
15    double m = mass;
16    double rm = 1/m;
17
18    double vt=_X(VT); double alpha =_X(ALPHA); double beta=_X(BETA);
19    double phi=_X(PHI); double theta=_X(THETA); double psi=_X(PSI);
20    double p=_X(P); double q=_X(Q); double r=_X(R);
```

```
21  double alt=_X(ZE);
22  double thtl=_U(0); double elevDeg=_U(1); double ailDeg=_U(2); double rdrDeg=_U(3);
23  double flapDeg = 0.0;
24
25  if(alpha>=alphaMax) std::cout << "Alpha max exceeded" << endl;
26  if(alpha<=alphaMin) std::cout << "Alpha min exceeded" << endl;
27  if(beta>=betaMax) std::cout << "Beta max exceeded" << endl;
28  if(beta<=betaMin) std::cout << "Beta min exceeded" << endl;
29
30  angularVel_B(0)=p; angularVel_B(1)=q; angularVel_B(2)=r;
31
32  // Actualización de matrices de rotación
33  updateB2Ematrix(theta,phi,psi,B2E); updateE2Bmatrix(theta,phi,psi,E2B);
34  updateB2Wmatrix(alpha,beta,B2W); updateW2Bmatrix(alpha,beta,W2B);
35  updateB2Smatrix(alpha,B2S); updateS2Bmatrix(alpha,S2B);
36  // Expreso p, q y r en ejes viento
37  angularVel_W(iT) = B2W(iT,jT) * angularVel_B(jT);
38
39  ComputeUbVbWb(_X,ub,vb,wb);
40
41  AtmosphereModel(vt,alt,dynPres,ps,rho,sigma);
42  EngineModel(thtl,Thrust(0));
43
44  // Cálculo de coeficientes aerodinámicos
45  double CD = pCessna172_Coeffs->GetDragCoeff(alpha,beta,elevDeg,flapDeg);
46  double CY = pCessna172_Coeffs->GetSideForceCoeff(alpha,beta,rdrDeg,flapDeg,
47                                                   wingSpan,vt,p,r);
48
49  double Cl = pCessna172_Coeffs->GetRollCoeff(alpha,beta,ailDeg,vt,wingSpan,p,r,
50                                              rdrDeg,flapDeg);
51  double Cn = pCessna172_Coeffs->GetYawCoeff(alpha,beta,ailDeg,wingSpan,vt,p,r,
52                                             rdrDeg);
53
54  double sth = sin(theta); double cth = cos(theta);
55  double sph = sin(phi); double cph = cos(phi);
56  double spsi = sin(psi); double cpsi = cos(psi);
57
58  ForceW(0)=-dynPres * wingArea * CD;
59  ForceW(1)= dynPres * wingArea * CY;
```

```
60
61  double alphaDotOld=0;
62  double dif=10.0;
63  double alphaDot;
64  while(abs(dif)>=0.001){
65    double CL1 = pCessna172_Coeffs->GetLiftCoeff(alpha,alphaDotOld,elevDeg,flapDeg,
66                                                 wingChord,vt,q);
67    ForceW(2) = -dynPres * wingArea * CL1;
68    ForceB(iT) = W2B(iT,jT) * ForceW(jT);
69    ForceB(0)+= Thrust(0);
70    double ubDot = r*vb - q*wb - GRAVITY*sth + rm*ForceB(0);
71    double wbDot = q*ub - p*vb + GRAVITY*cth*cph + rm*ForceB(2);
72    double alphaDotNew = (ub * wbDot - wb * ubDot)/(ub*ub + wb*wb);
73    dif = alphaDotNew - alphaDotOld;
74    alphaDotOld = alphaDotNew;
75    alphaDot = alphaDotNew;
76  }
77
78  double CL = pCessna172_Coeffs->GetLiftCoeff(alpha,alphaDot,elevDeg,flapDeg,
79                                              wingChord,vt,q);
80  ForceW(2) = -dynPres * wingArea * CL;
81
82  // vector de fuerzas en ejes cuerpo
83  ForceB(iT) = W2B(iT,jT) * ForceW(jT);
84  ForceB(0)+= Thrust(0);
85
86  double Cm = pCessna172_Coeffs->GetPitchCoeff(alpha,alphaDot,elevDeg,flapDeg,
87                                               wingChord,vt,q);
88
89  double ubDot = r*vb - q*wb - GRAVITY*sth + rm*ForceB(0);
90  double vbDot = p*wb - r*ub + GRAVITY*cth*sph + rm*ForceB(1);
91  double wbDot = q*ub - p*vb + GRAVITY*cth*cph + rm*ForceB(2);
92
93  _Xdot(VT) = (ub * ubDot + vb * vbDot + wb * wbDot)/vt;
94  _Xdot(BETA) = (vbDot * vt + vb * _Xdot(VT))/(vt*vt*cos(beta));
95  _Xdot(ALPHA) = (ub*wbDot - wb*ubDot)/(ub*ub+wb*wb);
96  _Xdot(PHI) = p + tan(theta) * (q * sin(phi) + r * cos(phi));
97  _Xdot(THETA) = q * cos(phi) - r * sin(phi);
98  _Xdot(PSI) = (q * sin(phi) + r * cos(phi))/cos(theta);
```

```
99
100  MomentB(0)= dynPres * wingArea * wingSpan * Cl;
101  MomentB(1)= dynPres * wingArea * wingChord * Cm;
102  MomentB(2)= dynPres * wingArea *wingSpan * Cn;
103
104  _Xdot(P) = (c(1)*p +c(0)*r + c(3) * pCessna172_Properties->He)*q +
105                    (c(2) *MomentB(0) + c(3)* MomentB(2));
106  _Xdot(Q) = (c(4)*p - c(6)*pCessna172_Properties->He)*r + c(5)*(r*r - p*p)+
107                     MomentB(1)*c(6);
108  _Xdot(R) = (c(7)*p - c(1)*r +c(8)*pCessna172_Properties->He)*q +
109                    (c(3)*MomentB(0)+ c(8)*MomentB(2));
110
111  double T1 = sph*cpsi;
112  double T2 = cph * sth;
113  double T3 = sph * spsi;
114  double S1 = cth * cpsi;
115  double S2 = cth * spsi;
116  double S3 = T1 * sth - cph * spsi;
117  double S4 = T3 * sth + cph * cpsi;
118  double S5 = sph * cth;
119  double S6 = T2 * cpsi + T3;
120  double S7 = T2 * spsi - T1;
121  double S8 = cph * cth;
122
123  _Xdot(XE) = ub*S1 + vb*S3 + wb*S6;
124  _Xdot(YE) = ub*S2 + vb*S4 + wb*S7;
125  _Xdot(ZE) = ub*sth - vb*S5 - wb*S8;
126 }
```

La función **Simulate** constituye el lazo principal del módulo **UAV**. La misma se invoca en cada instante de muestreo. En el método **Integrate** de la clase **RungeKutta** se implementa el algoritmo de integración *Runge-Kutta* de cuarto orden, presentado en la sección 2.5. La llamada a este método permite integrar la ecuación de estados del UAV virtual para obtener la evolución de los sucesivos estados del mismo. En la función **ComputeRates** se implementa la ecuación $\dot{\mathbf{x}} = \bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ donde la función vectorial $\bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ es la definida en la Ec. (2.38). Esto permite obtener, en cada instante de muestreo, la velocidad de cambio del vector de estados, es decir, de $\dot{\mathbf{x}}$. Resumiendo, la función

ComputeRates define la dinámica del UAV.

9.5. Diseño del Módulo Control

En el módulos **Control de Actitud** y **Control Unificado** de Figs. 9.1 y 9.2, respectivamente, se computan las entradas de control que el **UAV** debe aplicar. Ambos módulos se pueden ejecutar tanto en modo manual (llave en la posición M de Figs. 9.1 y Fig. 9.2) como en modo automático (llave en la posición A de Figs. 9.1 y Fig. 9.2). La utilización de la plataforma en modo manual se presentó en la sección 9.2. Cuando la misma se utiliza en modo automático, las entradas de control que el **UAV** debe aplicar se sintetizan a través de *software*, es decir, mediante la implementación de algún algoritmo de control. En el caso particular del módulo **Control Unificado**, el mismo se diseñó utilizando el algoritmo INL-MPC presentado en el capítulo 4, mientras que el módulo **Control de Actitud** se diseñó utilizando la técnica de control INL-MPC-J presentada en el capítulo 5. En modo automático, la plataforma SVC permite evaluar mediante la realización de maniobras pre-definidas o bien mediante el seguimiento de trayectorias de navegación óptimas, el comportamiento autónomo de un UAV real o virtual. Además, permite verificar que las maniobras de vuelo se realicen en **tiempo real**.

Como se mencionó anteriormente, el intercambio de datos entre los distintos módulos se realiza de forma asíncrona, enviándose los mismos con una tasa pre-definida. Las estructuras de datos utilizadas para el envío y recepción de información para el caso del enlace 4 de Fig. 9.1 (siendo su análogo en Fig. 9.2 el enlace 3) son las siguientes:

```
// ************************* datos recibidos del UAV ****************************
  struct DataReceivedFromUAV{
    double Xstate[12];
    double Uctrl[4];
  };

// ************************** datos enviados al UAV *****************************
  struct DataSentToUAV{
    double Uctrl[4];
```

```
10 };
```

En la estructura de datos **DataReceivedFromUAV**, se configura el vector de estados **Xstate** y el vector de entradas de control **Uctrl** con el vector de estados y el vector de control recibidos desde el **UAV**, respectivamente. En la estructura **DataSentToUAV** el módulo **Control de Actitud** (**Control Unificado** en el caso del enlace 3) configura el vector de entradas de control **Uctrl** con el vector correspondiente al primer instante de muestreo de la secuencia de entradas de control óptima computada, es decir $\mathbf{Uctrl} = \mathbf{U}_c^*(1)$, para ser luego enviada al **UAV**.

9.6. Diseño del Módulo Control de Navegación

El módulo **Control de Navegación** es donde se calculan las trayectorias de navegación óptimas, utilizándose la metodología descripta en el capítulo 6 e implementándose en el mismo el algoritmo de control INL-MPC.

En las figuras que se muestran a continuación se presenta la secuencia de pasos que permiten configurar, con ayuda de la plataforma SVC, los parámetros necesarios para generar una trayectoria de navegación óptima. Para este ejemplo, se utiliza como UAV autónomo un modelo sencillo de aeronave. Asimismo, se considera que la plataforma *Excalibur* está configurada como en Fig. 9.1, es decir con un sistema de control de actitud y navegación desacoplado.

Al presionar el botón **Configuración**, señalado en la Fig. 9.6 con 1, se despliega la ventana de configuración de *waypoints* como se muestra en la Fig. 9.7. Al presionar el botón **Configurar Waypoints** de la ventana 2, se despliega la ventana que permite el ingreso de hasta un máximo de 10 *waypoints*. Esto se puede ver en la Fig. 9.8. En el recuadro 3 se debe ingresar el número de *waypoints* que se van a configurar. En 4 se definen las coordenadas $\mathbf{W}_i = [x_i \;\; y_i \;\; z_i]^T$ del i-*ésimo waypoint*, con $i = 0, \cdots, 9$. Para este ejemplo se definieron tres puntos de paso: $\mathbf{W}_1 = [1 \;\; 1 \;\; 0]^T$, $\mathbf{W}_2 = [-1 \;\; -1 \;\; 1]^T$ y $\mathbf{W}_3 = [2 \;\; 2 \;\; 2]^T$. Al presionar el botón aceptar se envía el listado de *waypoints* al módulo **Control de Navegación** para que el mismo

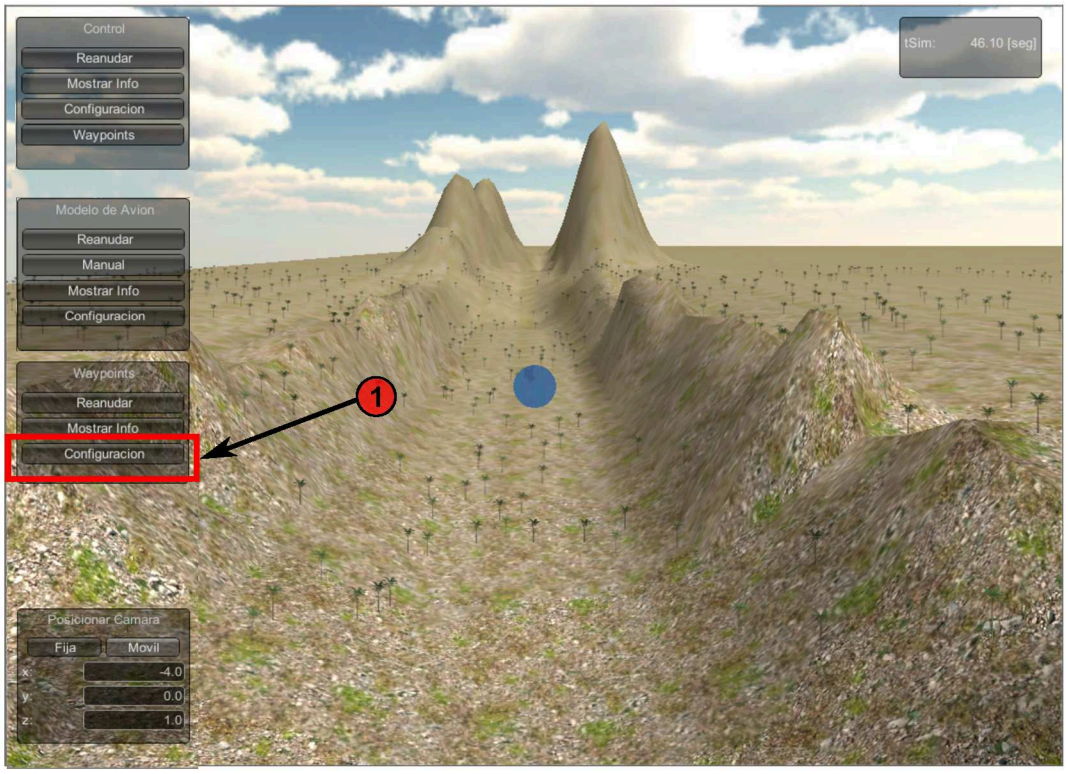


Figura 9.6: Selección de la opción configuración de *waypoints*

reconfigure su vector de valores deseados. Los *waypoints* definidos se dibujan en la pantalla principal, como se puede ver en la Fig. 9.9 señalados con 5. En la Fig 9.10 se muestra en 6, la trayectoria de navegación óptima resultante que pasa por los *waypoints* definidos, y que además describe una circunferencia en el plano xy de radio constante que atraviesa un entorno del *waypoint* final. En la Fig. 9.11, en 8 se muestra una vista superior de la trayectoria de navegación obtenida. Para realizar esto, solamente es necesario modificar la posición de la cámara ingresando las coordenadas deseadas en la ventana 7.

9.7. Conclusiones

En este capítulo se presentaron los detalles de diseño de la plataforma SVC *Excalibur*, la cual ha sido desarrollada en el marco de la presente te-

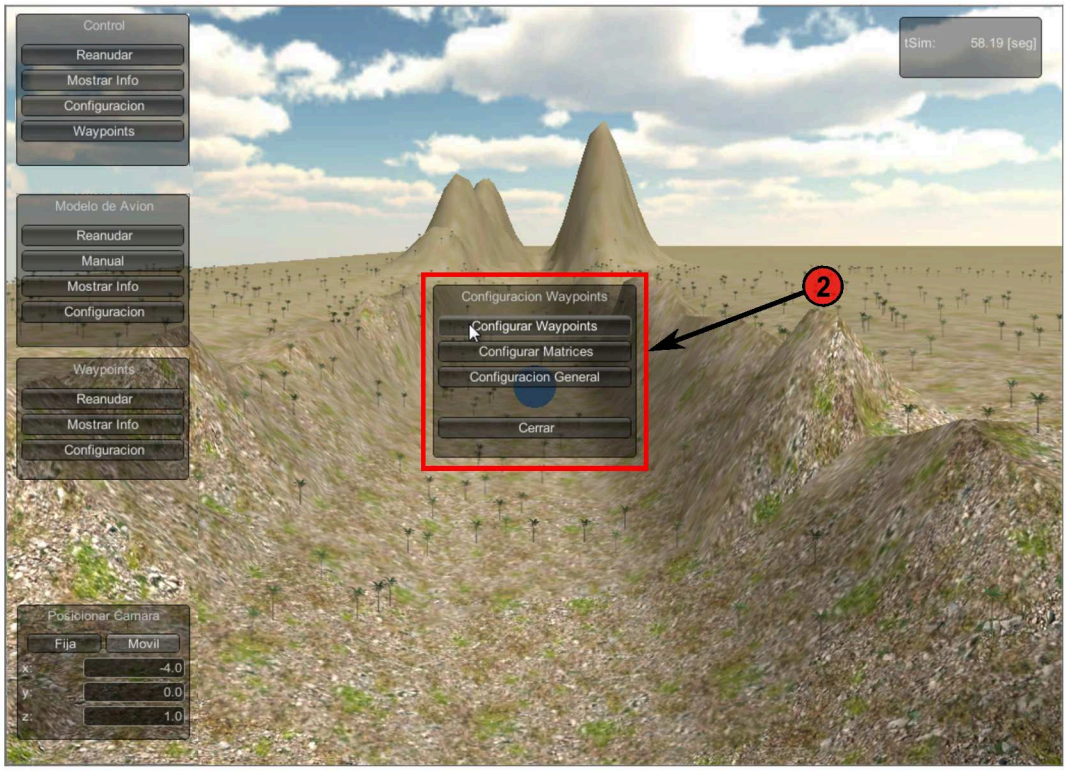


Figura 9.7: Ventana principal de configuración de *waypoints*

sis. Mediante la utilización de la plataforma se logró recrear por computadora el comportamiento de un UAV real de orden completo, con 6-grados de libertad. La plataforma mencionada, hace las veces de interfaz entre usuario y computadora, permitiendo comandar al UAV tanto en modo manual como en modo automático. Además, permite la modificación de diferentes parámetros del UAV virtual y de los controladores en tiempo de ejecución. La estructura modular con que se diseñó permite desacoplar cada uno de los módulos presentados, posibilitando la modificación de cada uno de ellos de forma totalmente independiente. Asimismo, se logró reducir la carga computacional de la plataforma SVC mediante la ejecución de cada uno de los módulos en procesadores independientes. La plataforma *Excalibur* se utilizó exhaustivamente como banco de pruebas para el diseño, validación y evaluación de los sistemas de control de actitud y navegación, unificados y desacoplados, presentados en los capítulos 3, 4, 5 y 6.

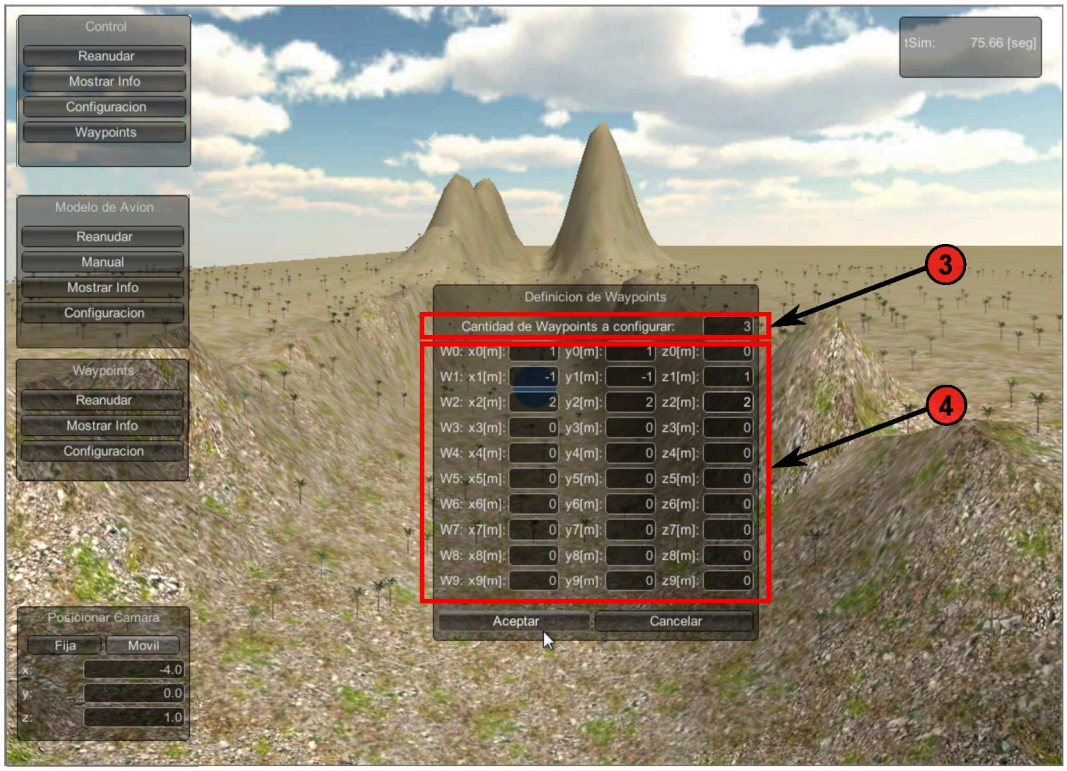


Figura 9.8: Configuración listado de *waypoints*

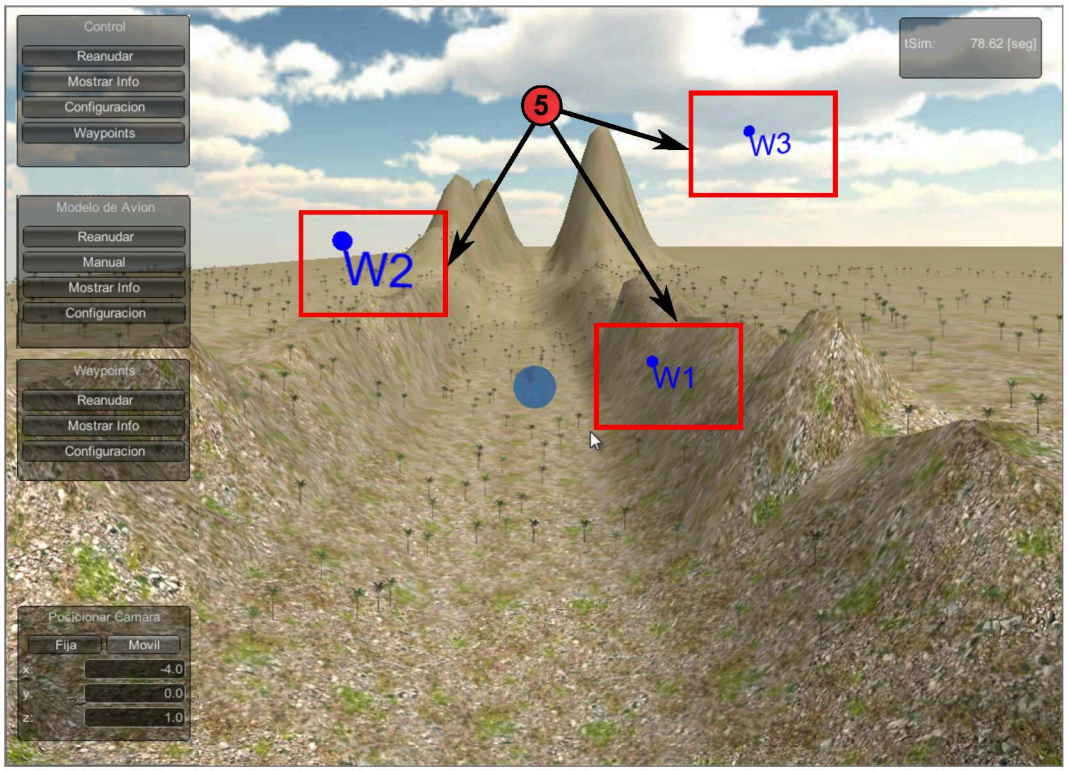


Figura 9.9: Visualización de los *waypoints* en la pantalla principal de la plataforma *Excalibur*

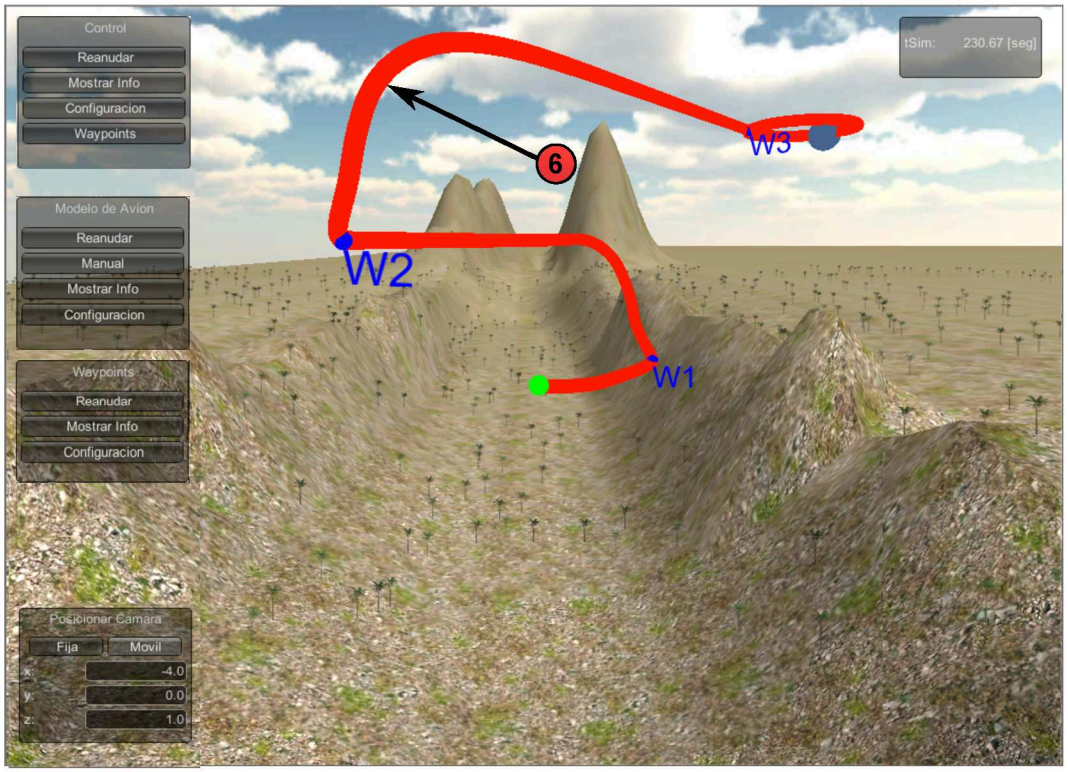


Figura 9.10: Visualización de la trayectoria de navegación en la pantalla principal de la plataforma *Excalibur*

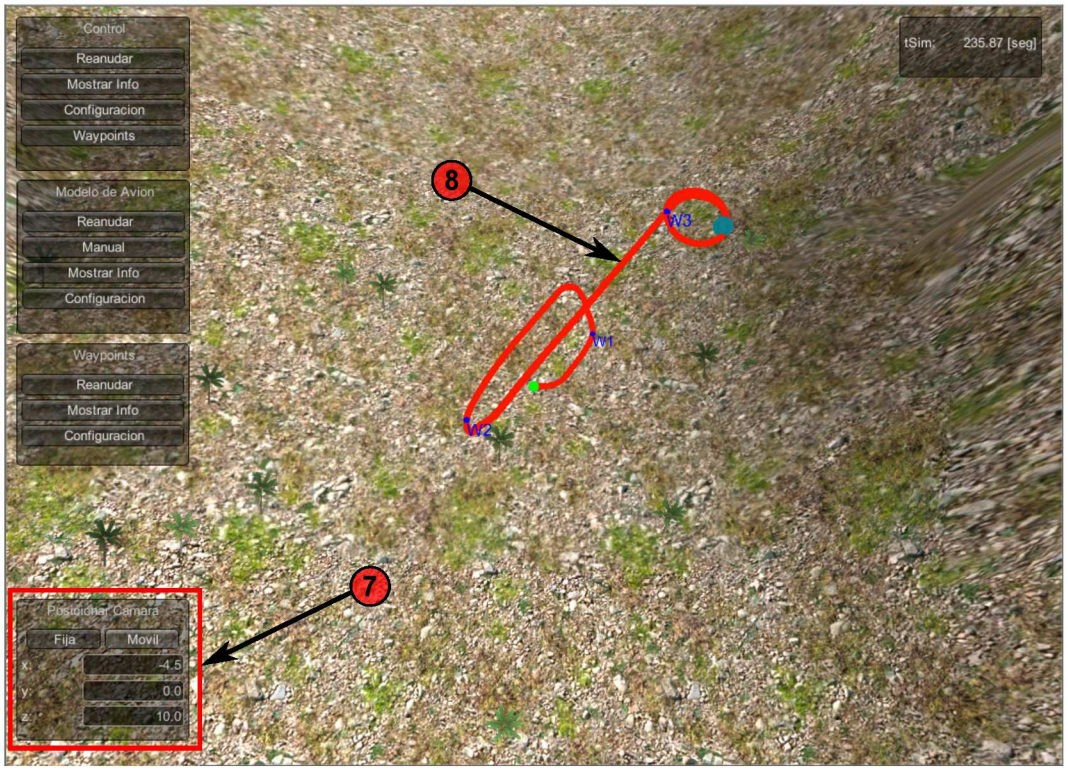


Figura 9.11: Vista superior de la trayectoria de navegación

Capítulo 10

Conclusiones y Trabajo Futuro

10.1. Conclusiones

Las contribuciones de la presente tesis son principalmente cuatro: 1) Desarrollo del método de control INL-MPC, 2) Generalización del método INL-MPC dando origen al método INL-MPC-J, el cual permite la inclusión de funciones de costo no-lineales arbitrarias, 3) Desarrollo de una metodología *online* para la generación y seguimiento de trayectorias de navegación óptimas, tanto en 2D como en 3D, y 4) Desarrollo de la plataforma de simulación, visualización y control de vuelo *Excalibur*, la cual se utiliza, en el marco de esta tesis para diseñar, validar y evaluar sistemas de control de actitud y navegación tanto unificados como desacoplados, para vuelo autónomo de UAVs.

En el capítulo 2 se realizó una revisión acerca de las representaciones generales de los sistemas lineales y no-lineales. Asimismo, se presentó el modelo matemático de un avión de orden completo, de 6-DOF, el cual fue utilizado para modelar la dinámica del UAV virtual empleado en los capítulos posteriores.

En el capítulo 3 se realizó una revisión de la técnica de control MPC clásica, ya que la misma se usó como base para desarrollar los métodos de control INL-MPC e INL-MPC-J descriptos en los capítulos 4 y 5, respectivamente.

El método de control INL-MPC es una técnica de control predictivo no-lineal genérico que extiende las capacidades del MPC clásico con el objetivo

de controlar sistemas genéricos con dinámicas no-lineales. El mismo utiliza un proceso de linealización generalizado de la dinámica no-lineal del sistema a lo largo de trayectorias candidatas definidas iterativamente en el espacio de estados. Cuenta con importantes ventajas: 1) Permite la inclusión de sistemas no-lineales genéricos, 2) Puede ser utilizado como un sistema de control unificado permitiendo considerar los acoplamientos existentes entre los diferentes modos dinámicos del sistema, 3) Transforma el problema de control óptimo no-lineal de un sistema no-lineal en una serie de sub-problemas de optimización cuadráticos, iterativos más sencillos de resolver, y 4) La incorporación de las restricciones puede realizarse con facilidad, además las velocidades de cambio de estados y entradas de control pueden ser penalizadas, permitiendo tener en cuenta las limitaciones físicas de los sistemas reales. En el capítulo 7, el método INL-MPC se utilizó como un sistema de control unificado de vuelo autónomo. Con este sistema de control se realizaron diferentes maniobras de vuelo autónomo como ser: vuelo recto y nivelado, ascensos y descensos, cambio de velocidad, cambio de dirección, etc.

El método general INL-MPC-J permite incorporar al problema de control variables arbitrarias que no forman parte del vector de estados ni del vector velocidad de cambio de entradas de control, pero que son funciones no-lineales de las mismas. Con INL-MPC-J se pueden utilizar funciones de costo no-lineales que penalizan las variaciones de las funciones no-lineales anteriormente mencionadas respecto de valores deseados. El problema de optimización no-lineal resultante se resuelve utilizando el proceso de linealización generalizado (a lo largo de trayectorias candidatas en espacio de estados) sobre la función de costo. Con esto, el problema de optimización no-lineal se reduce a uno equivalente cuadrático y más sencillo de resolver.

En el capítulo 6 se presentó una metodología que permite la generación y seguimiento de trayectorias de navegación óptimas. La misma se basa en el cómputo de una trayectoria de navegación *online* utilizando modelos no-lineales de vehículos reducidos tipo partícula y el método INL-MPC. La metodología propuesta permite la obtención de trayectorias de navegación de mínimo recorrido que tienen en cuenta las restricciones propias de la dinámica del modelo de vehículo utilizado. Primero se computaron trayectorias de nave-

gación en dos dimensiones, extendiéndose luego al espacio tridimensional. En el capítulo 8, la metodología propuesta se utilizó para diseñar un sistema de control de navegación, el cual se usó en conjunto con un sistema de control de actitud para diseñar un sistema de control desacoplado de vuelo autónomo, capaz de controlar la dinámica completa, de 6-DOF, de un UAV. Con este sistema de control se realizaron diferentes maniobras de generación y seguimiento de trayectorias, tanto en 2D como en 3D y con múltiples puntos de paso.

En el capítulo 9 se presentaron los aspectos de diseño de la plataforma de simulación, visualización y control de vuelo *Excalibur*. La plataforma hace las veces de interfaz entre usuario, el UAV y los sistemas de control, permitiendo comandar al UAV tanto modo manual como en modo automático. Se diseñó con una estructura modular que permite desacoplar cada uno de los módulos que la componen, posibilitando la modificación independiente de cada uno de ellos. Cada uno de los módulos se ejecuta en unidades de procesamiento independientes. Dicha plataforma se utilizó exhaustivamente para diseñar, validar y evaluar los sistemas de control de actitud y navegación, unificados y desacoplados, y permitió, además, verificar que las maniobras se ejecuten en tiempo real.

10.2. Trabajo Futuro

Como continuación, profundización y ampliación de la investigación realizada en esta tesis de doctorado se propone a futuro:

1. Analizar las propiedades teóricas más relevantes (factibilidad, convergencia, estabilidad, optimalidad y robustez) de los métodos ya desarrollados (INL-MPC e INL-MPC-J)

2. Utilizando las técnicas INL-MPC e INL-MPC-J, analizar el comportamiento del UAV en las fases de despegue y aterrizaje autónomo

3. Implementar y evaluar los métodos de control desarrollados en un modelo a escala de un UAV

4. Desarrollar nuevos métodos de control basados en control predictivo que

   a) Permitan incorporar múltiples objetivos operacionales (correspondientes a diferentes fases de misiones) utilizando mecanismos de adaptación basados en conmutación de funciones objetivos y restricciones

   b) Permitan controlar el sistema en diferentes condiciones operacionales utilizando mecanismos de adaptación basados en conmutación e interpolación de modelos lineales

   c) Utilicen una red de comunicación en el enlace sensor-controlador-actuador (*Networked Control Systems* - NCS) para reducir la carga computacional de la computadora a bordo del UAV y garanticen la operación del sistema en presencia de fallas en las comunicaciones

5. Analizar las propiedades teóricas más relevantes de los posibles nuevos métodos propuestos

Apéndice A

Modelado del Avión Cessna 172

A continuación se detallan los parámetros característicos del avión Cessna 172: Masa = 680,39 [kg], CG = $[1{,}0414 \;\; 0{,}0, \;\; 0{,}9271]^T$ [m], S = 16,16 [m^2], b = 10,9 [m], m.a.c = 1,5 [m], J_{xz} = 0,0 [$\frac{kg}{m^2}$], J_y = 1824,8712 [$\frac{kg}{m^2}$], J_z = 2666,8066 [$\frac{kg}{m^2}$], J_x = 1285,2734 [$\frac{kg}{m^2}$]

Las fuerzas aerodinámicas F_x^{aero}, F_y^{aero} y F_z^{aero} se calculan como:

$$F_x^{\text{aero}} = \bar{q}SC_d,\ F_y^{\text{aero}} = \bar{q}SC_y \text{ y } F_z^{\text{aero}} = \bar{q}SC_L \tag{A.1}$$

donde $\bar{q}$ es la presión dinámica y S es el área del ala. C_d, C_y y C_L son los coeficientes adimensionales de las fuerzas aerodinámicas; C_d el es coeficiente de arrastre (*drag*), C_y es el coeficiente de fuerza lateral (*sideforce*) y C_L es el coeficiente de sustentación (*lift*).

Los momentos aerodinámicos M_x^{aero}, M_y^{aero} y M_z^{aero} con respecto al CG del avión, se calculan como se muestra a continuación:

$$M_x^{\text{aero}} = \bar{q}SbC_l,\ M_y^{\text{aero}} = \bar{q}S\bar{c}C_m \text{ y } M_z^{\text{aero}} = \bar{q}SbC_n \tag{A.2}$$

donde $\bar{c}$ es la cuerda media aerodinámica (m.a.c), b es la envergadura del avión C_l, C_m y C_n son los coeficientes adimensionales de los momentos aerodinámicos. C_l es el coeficiente de alabeo (*roll*), C_m es el coeficiente de cabeceo (*pitch*) y C_n es el coeficiente de guiñada (*yaw*).

Se asume que los coeficientes aerodinámicos pueden calcularse utilizando

las siguientes expresiones (ver [8, 62]):

$$\begin{aligned}
C_d &= C_{d0} + \Delta C_d(\alpha) + C_{d_\beta}\beta \\
C_L &= \Delta C_L(\alpha) + C_{L_{\delta_e}}\delta_e + \tfrac{\bar{c}}{2v_t}(C_{L_{\dot{\alpha}}}\dot{\alpha} + C_{L_q}q) \\
C_y &= \Delta C_y(\beta) + C_{y_{\delta_r}}\delta_r + \tfrac{b}{2v_t}(C_{y_p}p + C_{y_r}r) \\
C_l &= \Delta C_l(\beta) + \tfrac{b}{2v_t}(C_{l_p}p + C_{l_r}r) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \\
C_m &= C_{m_0} + C_{m_\alpha}\sin(\alpha) + \tfrac{\bar{c}}{2v_t}(C_{m_q}q + C_{m_{\dot{\alpha}}}\dot{\alpha}) + C_{m_{\delta_e}}\delta_e \\
C_n &= \Delta C_n(\beta) + \tfrac{b}{2v_t}(C_{n_p}p + C_{n_r}r + C_{n_{\delta_a}}\delta_a + C_{n\delta_r}\delta_r)
\end{aligned} \tag{A.3}$$

Los coeficientes en el lado izquierdo de la Ec. (A.3) pueden obtenerse utilizando los coeficientes constantes que se listan a continuación: $C_{d_0} = 0{,}027$, $C_{d_\beta} = 0{,}17$, $C_{y_{\delta_r}} = 0{,}1870$, $C_{L_{\delta_e}} = 0{,}43$, $C_{L_{\dot{\alpha}}} = 1{,}7$, $C_{L_q} = 3{,}9$, $C_{l_p} = -0{,}484$, $C_{l_{\delta_a}} = 0{,}229$, $C_{l_{\delta_r}} = 0{,}0147$, $C_{m_0} = 0{,}1$, $C_{m_\alpha} = -1{,}8$, $C_{m_q} = -12{,}4$, $C_{m_{\dot{\alpha}}} = -7{,}27$, $C_{m_{\delta_e}} = -1{,}1220$, $C_{n_p} = -0{,}0278$, $C_{n_r} = -0{,}0937$, $C_{n_{\delta_a}} = -0{,}0053$, $C_{n_{\delta_r}} = -0{,}0430$, y mediante interpolación de las tablas presentadas en los Cuadros A.1-A.3. Estos valores han sido obtenidos de las plataformas *open-source* FlightGear Flight Simulator [55] y JSBSim [63] para el caso del modelo del avión Cessna 172, y pueden descargarse de [64].

Cuadro A.1: Coeficientes Aerodinámicos Dependientes de β

β [rad]	$\Delta C_y(\beta)$	$\Delta C_n(\beta)$	C_{l_β}
-0.349	0.137	-0.021	0.032
0.0	0.0	0.0	0.0
0.349	-0.137	0.021	-0.032

Cuadro A.2: Coeficientes de Derivadas Dinámicas

α [rad]	C_{y_p}	C_{y_r}	C_{l_r}
0.0	-0.075	0.214	0.08
0.094	-0.145	0.267	0.187

Para el modelo de propulsión del avión Cessna 172, se adoptó un modelo básico y sencillo para facilitar la reproducción de resultados. Se asumió que la fuerza de empuje (*thrust*) T es directamente proporcional a la deflexión de la

Cuadro A.3: Coeficientes Aerodinámicos dependientes de α

α [rad]	$\Delta C_d(\alpha)$
-0.087	0.004
-0.07	0.001
-0.052	0.0001
-0.035	0.0003
-0.018	0.002
0.0	0.005
0.018	0.01
0.035	0.016
0.052	0.024
0.07	0.033
0.087	0.044
0.105	0.057
0.122	0.071
0.14	0.086
0.157	0.096
0.175	0.107
0.192	0.118
0.209	0.13
0.227	0.142
0.244	0.157
0.262	0.173
0.279	0.178
0.297	0.172
0.314	0.162
0.332	0.148
0.349	0.11

α [rad]	$\Delta C_L(\alpha)$
-0.09	-0.22
0.00	0.25
0.09	0.73
0.10	0.83
0.12	0.92
0.14	1.02
0.16	1.08
0.17	1.13
0.19	1.19
0.21	1.25
0.24	1.35
0.26	1.44
0.28	1.47
0.30	1.43
0.32	1.38
0.34	1.30
0.36	1.15

columna de propulsión (entrada de control *thtl*), es decir:

$$T = a \cdot thtl \tag{A.4}$$

Además, se considera que esta fuerza de empuje solamente genera una fuerza a lo largo del eje $\hat{x}_{\text{Body}}$; es decir, $F_x^{\text{engine}} = T$. Se adopta un valor $a = 1690{,}0$ [N].

Apéndice B

Modelado de un Avión Basado en su Geometría

A continuación se describe una metodología alternativa para calcular las fuerzas y momentos aerodinámicas actuantes en un avión. El método propuesto se basa, principalmente, en la partición geométrica del avión en N elementos (o secciones), similar a lo realizado en [65]. Cada uno de los N elementos poseen características aerodinámicas, pudiendo así generar fuerzas y momentos aerodinámicos individuales, que actúan en cada uno de ellos.

Notación: De aquí en adelante se utilizarán superíndices para indicar la pertenencia de una determinada propiedad a la j-*ésima* sección del avión. Por ejemplo, el valor de la variable genérica Ξ para el j-*ésimo* elemento del avión se notará como Ξ^j.

En la Fig. B.1 se puede observar, en color azul, un posible esquema de subdivisión de la geometría del avión. Como se puede apreciar en dicha figura, el avión se dividió en 7 partes. A continuación se describirán las secciones elegidas para la parte derecha del avión. La parte izquierda se realiza de forma análoga. El ala derecha se dividió en dos: con (1) se indica la parte del ala que incluye el *flap* derecho, con (2) se señala la división del ala que incluye el alerón derecho. De igual forma se divide el ala izquierda. En (3) se muestra la subdivisión que incluye la parte derecha de la cola del avión. La parte izquierda de la misma, se divide de igual forma. Finalmente, con (4) se muestra la

sección compuesta por la cola vertical. Por supuesto que el nivel de detalle con el cual se subdivide al avión depende del grado de precisión que se quiere obtener en el modelado del mismo, es decir, a mayor número de divisiones más preciso será el modelo del avión. Al subdividir la geometría del avión en N

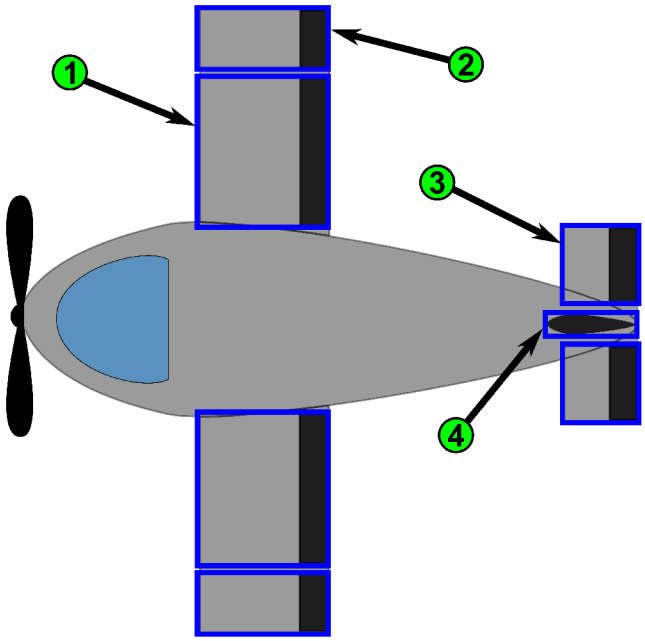


Figura B.1: Partes en que se divide el avión para su modelado

partes, se puede considerar que cada una de ellas tienen asociadas: un vector velocidad $\mathbf{v}_R^j$, ángulo de ataque α^j, ángulo de deslizamiento β^j, coeficientes aerodinámicos C_d^j, C_y^j y C_L^j, fuerzas aerodinámicas $F_x^{\text{aero}^j}$, $F_y^{\text{aero}^j}$ y $F_z^{\text{aero}^j}$ y momentos aerodinámicos $M_x^{\text{aero}^J}$, $M_y^{\text{aero}^j}$ y $M_z^{\text{aero}^j}$, donde $j = 1, \cdots, N$. Si se suman estas N contribuciones de fuerzas y momentos aerodinámicos generados por cada sección del avión, se obtienen las fuerzas y momentos resultantes que describen el movimiento del avión completo. La metodología para el cómputo de dichas cantidades es la siguiente:

1. Dividir la geometría del avión en N partes, cada una de las cuales representa una sección alar más pequeña con capacidad de generar fuerza de sustentación.

2. Calcular la velocidad relativa $\mathbf{v}_R^j$, con $j = 1, \cdots, N$, de cada una de las

secciones en que se dividió el avión. Para ello, se debe tener en cuenta que, como el avión es modelado como un sólido rígido, el cálculo de la velocidad relativa para la j-*ésima* división del avión se puede calcular como:

$$\mathbf{v}_R^j = \mathbf{v}_{CG} + \boldsymbol{\omega} \times \mathbf{r}^j \tag{B.1}$$

donde $\mathbf{v}_R^j$ es la velocidad relativa del j-*ésimo* elemento del avión, $\mathbf{v}_{CG}$ es el vector que representa la velocidad del viento y la dirección de vuelo, $\boldsymbol{\omega}$ es el vector velocidad angular del avión y $\mathbf{r}^j$ es el vector que mide la distancia desde el origen del sistema coordenado del avión al centro aerodinámico (ca^j) de la j-*ésima* subdivisión del avión.

3. Calcular el ángulo de ataque α^j y el ángulo de deslizamiento β^j para cada una de las N divisiones geométricas del avión.

4. Con los ángulos α^j y β^j, y utilizando la información provista por ensayos en túnel de viento para el/los perfiles alares con que fue diseñado el avión, computar los coeficientes adimensionales de sustentación, arrastre y fuerza lateral C_L^j, C_d^j y C_y^j, respectivamente, con $j = 1, \cdots, N$.

5. Calcular las fuerzas aerodinámicas $F_x^{\text{aero}^j}$, $F_y^{\text{aero}^j}$ y $F_z^{\text{aero}^j}$, $j = 1, \cdots, N$, para cada una de las particiones del avión, como sigue:

$$F_x^{\text{aero}^j} = \bar{q}^j S^j C_d^j,\ F_y^{\text{aero}^j} = \bar{q}^j S^j C_y^j \text{ y } F_z^{\text{aero}^j} = \bar{q}^j S^j C_L^j \tag{B.2}$$

donde $\bar{q}^j$ y S^j son la presión dinámica y el área, respectivamente, de la j-*ésima* sección del avión.

6. Las fuerzas aerodinámicas resultantes, actuantes en el avión completo, se pueden obtener sumando las N contribuciones de cada una de las partes en que se dividió la geometría del avión, es decir:

$$F_x^{\text{aero}} = \sum_{j=1}^{N} F_x^{\text{aero}^j},\ F_y^{\text{aero}} = \sum_{j=1}^{N} F_y^{\text{aero}^j} \text{ y } F_z^{\text{aero}} = \sum_{j=1}^{N} F_z^{\text{aero}^j} \tag{B.3}$$

7. Habiendo calculado los N vectores de fuerzas aerodinámicas $\mathbf{F}^{\text{aero}^j} =$

$[F_x^{\text{aero}^j}\ F_y^{\text{aero}^j}\ F_z^{\text{aero}^j}]^T$ para cada sección del avión, calcular la distancia existente entre el CG del avión y los centros aerodinámicos de las N secciones del avión. Los momentos aerodinámicos de cada partición se obtienen como:

$$\mathbf{M}^{\text{aero}^j} = (\text{ca}^j - CG) \times \mathbf{F}^{\text{aero}^j} \tag{B.4}$$

donde $\mathbf{M}^{\text{aero}^j} = [M_x^{\text{aero}^j}\ M_y^{\text{aero}^j}\ M_z^{\text{aero}^j}]^T$.

8. Los momentos aerodinámicas resultantes, respecto al CG del avión, se pueden obtener sumando los N momentos de cada de las secciones en que se dividió la geometría del avión, es decir:

$$M_x^{\text{aero}} = \sum_{j=1}^{N} M_x^{\text{aero}^j},\ M_y^{\text{aero}} = \sum_{j=1}^{N} M_y^{\text{aero}^j} \text{ y } M_z^{\text{aero}} = \sum_{j=1}^{N} M_z^{\text{aero}^j} \tag{B.5}$$

Bibliografía

[1] Kimon P Valavanis and Kimon P Valavanis. *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*. Springer Publishing Company, Incorporated, 2007.

[2] Kimon P Valavanis, Paul Y Oh, and Les A Piegl. *Unmanned Aircraft Systems: International Symposium On Unmanned Aerial Vehicles, UAV 08*. Springer, 2008.

[3] Patrick Doherty and Piotr Rudol. A uav search and rescue scenario with human body detection and geolocalization. In *AI 2007: Advances in Artificial Intelligence*, pages 1–13. Springer, 2007.

[4] Chunhua Zhang and John M Kovacs. The application of small unmanned aerial systems for precision agriculture: a review. *Precision agriculture*, 13(6):693–712, 2012.

[5] Stuart M Adams, Marc L Levitan, and Carol J Friedland. High resolution imagery collection utilizing unmanned aerial vehicles (uavs) for post-disaster studies. *Bridges*, 10:777–793, 2014.

[6] Iván Maza, Fernando Caballero, Jesús Capitán, JR Martínez-de Dios, and Aníbal Ollero. Experimental results in multi-uav coordination for disaster management and civil security applications. *Journal of intelligent & robotic systems*, 61(1-4):563–585, 2011.

[7] A Ollero and L Merino. Unmanned aerial vehicles as tools for forest-fire fighting. *Forest Ecology and Management*, 234(1):263, 2006.

[8] Brian L. Stevens and Frank L. Lewis. *Aircraft Control and Simulation.* John Wiley & Sons Inc., 2003.

[9] J. Roskam. *Airplane flight dynamics and automatic flight controls.* DAR-corporation, 2001.

[10] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *Int. J. Rob. Res.*, 31(5):664–674, 2012.

[11] A. E. Jr. Bryson. New concepts in control theory. *J. Guid.*, 8:417–425, 1985.

[12] A. E. Jr. Bryson and Yu-Chi Ho. *Applied Optimal Control.* Taylor & Francis, 1975.

[13] J. Borggaard, J. Burkardt, M. Gunzburger, and J. Peterson. *Optimal Design and Control.* Proceedings of the Workshop on Optimal Design and Control, 1994.

[14] R. Hilscher and V. Zeidan. Discrete optimal control: The accessory problem and necessary optimality conditions. *Journal of Mathematical Analysis and Applications*, 243:429–452, 2000.

[15] F.L. Lewis and V.L Syrmos. *Optimal control.* Wiley-Interscience, 1995.

[16] F.L. Lewis. *Applied optimal control and estimation.* Prentice Hall PTR Upper Saddle River, NJ, USA, 1992.

[17] J. Maciejowski. *Predictive control: with constraints.* Prentice Hall, 2002.

[18] James Blake Rawlings and David Q Mayne. *Model predictive control: Theory and design.* Nob Hill Pub., 2009.

[19] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[20] Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667 – 682, 1999.

[21] Jay H Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415–424, 2011.

[22] Eduardo F Camacho and Carlos Bordons. Control predictivo: Pasado, presente y futuro. *RIAII*, 1(3):5–28, 2010.

[23] Basil Kouvaritakis and Mark Cannon. *Non-linear Predictive Control: theory and practice*. Number 61. IET, 2001.

[24] Frank Allgower, Rolf Findeisen, and Zoltan K Nagy. Nonlinear model predictive control: From theory to application. *Journal-Chinese Institute of Chemical Engineers*, 35(3):299–316, 2004.

[25] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control*. Springer, 2011.

[26] José R Cueli and Carlos Bordons. Iterative nonlinear model predictive control. stability, robustness and applications. *Control Engineering Practice*, 16(9):1023–1034, 2008.

[27] M.H. Murillo, A.C. Limache, P.S. Rojas Fredini, and L.L. Giovanini. Generalized nonlinear optimal predictive control using iterative state-space trajectories: Applications to autonomous flight of uavs. *International Journal of Control, Automation and Systems*, 13(2), April 2015.

[28] Magnus Egerstedt and Clyde F Martin. Optimal trajectory planning and smoothing splines. *Automatica*, 37(7):1057–1064, 2001.

[29] T Arney. Dynamic path planning and execution using b-splines. In *Information and Automation for Sustainability, 2007. ICIAFS 2007. Third International Conference on*, pages 1–6. IEEE, 2007.

[30] Erik P Anderson, Randal W Beard, and Timothy W McLain. Real-time dynamic trajectory smoothing for unmanned air vehicles. *Control Systems Technology, IEEE Transactions on*, 13(3):471–477, 2005.

[31] Eric F Sorton and Sonny Hammaker. Simulated flight testing of an autonomous unmanned aerial vehicle using flightgear. *American Institute of Aeronautics and Astronautics, AIAA 2005*, 7083, 2005.

[32] Sefer Kurnaz, Omer Cetin, and Okyay Kaynak. Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles. *Expert Systems with Applications*, 37(2):1229–1234, 2010.

[33] S. Skogestad and I. Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.

[34] R. Anderson, E. Bakolas, D. Milutinović, and P. Tsiotras. Optimal feedback guidance of a small aerial vehicle in a stochastic wind. *Journal of Guidance, Control, and Dynamics*, pages 1–11, 2013.

[35] J. Wilburn, J. Cole, M. G. Perhinschi, and B. Wilburn. Comparison of a fuzzy logic controller to a potential field controller for real-time uav navigation. In *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, M., August 2012. AIAA. AIAA 2012-4907.

[36] A. Bemporad and M. Morari. Robust model predictive control: A survey, in robustness in identification and control. *Lecture Notes in Control and Information Sciences*, 245:207–226, 1999.

[37] B. Kouvaritakis and M. Cannon. Nonlinear predictive control: theory and practice, 2001.

[38] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng. A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 2980–2985, 2007.

[39] Zhi jun Yang, Xiao hui Qi, and Gan li Shan. Simulation of flight control laws design using model predictive controllers. *Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation*, pages 4213–4218, 2009.

[40] H. Chao, Y. Cao, and Y. Chen. Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, 8(1):36–44, 2010.

[41] I. Kaminer, A. Pascoal, E. Xargay, N. Hovakimyan, C. Cao, and V. Dobrokhodov. Path following for small unmanned aerial vehicles using l1 adaptive augmentation of commercial autopilots. *Journal of guidance, control, and dynamics*, 33(2):550–564, 2010.

[42] C. Jones, A. Domahidi, M. Morari, S. Richter, F. Ullmann, M. Zeilinger, et al. Fast predictive control: Real-time computation and certification. In *Nonlinear Model Predictive Control*, volume 4, pages 94–98, 2012.

[43] K. Alexis, G. Nikolakopoulos, and A. Tzes. Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances. *Control Engineering Practice*, 19(10):1195–1207, 2011.

[44] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[45] P. W. Gibbens and E. D. Medagoda. Efficient model predictive control algorithm for aircraft. *Journal of guidance, control, and dynamics*, 34(6):1909–1915, 2011.

[46] I. Prodan, R. Bencatel, S. Olaru, J. Sousa, C. N. Stoica, and S. Niculescu. Predictive control for autonomous aerial vehicles trajectory tracking. In *Nonlinear Model Predictive Control*, volume 4, pages 508–513, 2012.

[47] T. Keviczky and G. J. Balas. Receding horizon control of an f-16 aircraft: A comparative study. *Control Engineering Practice*, 14(9):1023 – 1033, 2006.

[48] Kwangjin Yang, Yeonsik Kang, and Salah Sukkarieh. Adaptive nonlinear model predictive path-following control for a fixed-wing unmanned aerial vehicle. *International Journal of Control, Automation and Systems*, 11(1):65–74, 2013.

[49] N. Slegers, J. Kyle, and M. Costello. Nonlinear model predictive control technique for unmanned air vehicles. *Journal of guidance, control, and dynamics*, 29(5):1179–1188, 2006.

[50] https://code.google.com/p/ltensor/.

[51] Limache, Rojas, and Murillo. Diseño de un moderno simulador de vuelo en tiempo real, 2010.

[52] Limache, Murillo, Rojas, and Giovanini. Aspectos de diseño de un simulador de vuelo, 2011.

[53] `http://www.x-plane.com/`.

[54] `http://www.microsoft.com/games/flightsimulatorx/`.

[55] `http://www.flightgear.org/`.

[56] `http://www.rockwellcollins.com/`.

[57] `http://www.cae.com/en/`.

[58] `http://www.flightsafety.com/`.

[59] `http://www.wreckedgames.com/`.

[60] `http://www.boost.org/`.

[61] `http://unity3d.com/`.

[62] Duke, Antoniewicz, and Krambeer. Derivation and definition of a linear aircraft model, 1988.

[63] `http://jsbsim.sourceforge.net/`.

[64] http://sourceforge.net/projects/jsbsim/files/.

[65] David M Bourg. *Physics for game developers*. O'Reilly Media, Inc., 2002.