Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN

INGENIERÍA DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA

TRABAJO FIN DE MÁSTER

DESARROLLO DE UNA ESTRATEGIA DE CONTROL INTELIGENTE SOBRE UNA PLATAFORMA DE SIMULACIÓN DE UN HELICÓPTERO (TWIN ROTOR)

Estudiante *Rivera Navarro, Julen*
Director/Directora *Irigoyen Gordo, Eloy*
Departamento Ingeniería de Sistemas y Automática
Curso académico *2020-2021*

Bilbao, 10, Septiembre, 2021

AGRADECIMIENTOS

Mi más sincero agradecimiento a todos aquellos que me han ayudado a lo largo de este camino, tanto a amigos como familia, y en especial al Grupo de Control Inteligente, más concretamente a Eloy Irigoyen Gordo y Mikel Larrea Sukia, por su ayuda durante la ejecución del proyecto.

RESUMEN TRILINGÜE

1. Resumen

Hoy en día, con la continua aparición de nuevas tecnologías y estrategias de control, surge la problemática de combinar ambas para el desarrollo de nuevos sistemas avanzados de control, tanto en procesos productivos como en aplicaciones de diferentes áreas. A medida que los dispositivos de control ganan en prestaciones, se ofrecen novedosas alternativas a las técnicas existentes. Prueba de ello es que las técnicas procedentes de la Computación Inteligente ganan terreno en el diseño de controladores más robustos y eficaces, en el análisis de información procedente de procesos o plantas, o incluso en los procesos de optimización y tomas de decisión.

Por tanto, el principal objetivo del presente trabajo es avanzar en el estudio de una estrategia de control basada en técnicas inteligentes para una plataforma Twin-Rotor, simulando el control de un helicóptero. Dicha estrategia ya ha sido contrastada en sistemas monovariables (SISO) y se propone dar el salto a un sistema multivariable (MIMO).

2. Laburpena

Gaur egun sortzen ari diren teknologia berriak zein kontrol estrategiak direla eta, hauek konbinatzeko arazoak sortu dira kontrol sistema aurreratu berriak garatzerako orduan, produkzio prozesuetan baita arlo desberdinetako aplikazioetan ere. Kontrol gailuek errendimendua hobetzen duten heinean, teknika berriak eskaintzen dira. Hori egiaztatu da, Konputazio Adimenean parte hartzen duten teknikek indarra hartzen ari direlako, kontrolagailu sendoagoak eta eraginkorragoak diseinatzerakoan, lantegietako informazioa aztertzerakoan zein optimizazio edo erabakiak hartzeko prozesuetan.

Ondorioz, lan honen helburu nagusia Twin-Rotor izeneko plataforma baterako adimen teknika ezberdinetan oinarritutako estrategiak ikertzea da, helikoptero baten kontrola simulatuz. Estrategia hau dagoeneko kontrastatu da sistema mono-aldagaietan (SISO). Azkenik, lan honekin lortu nahi dena sistema multi-aldagaietan (MIMO) erabiltzea da.

3. **Abstract**

Nowadays, due to the continuous appearance of new technologies and control strategies, the problem of combining both for the development of new advanced control systems appears, in production processes and in applications in different areas. As control devices gain in performance, novel alternatives to existing techniques are offered. Proof of this is that the techniques from Intelligent Computing are gaining ground in the design of more robust and efficient controllers, in the analysis of information from processes or plants, or even in the optimization and decision-making processes.

Therefore, the main objective of this work is to advance in the study of a control strategy based on intelligent techniques for a Twin-Rotor platform, simulating the control of a helicopter. This strategy has already been tested in monovariable systems (SISO) and it is proposed to make the leap to a multivariable system (MIMO).

Palabras Clave: Algoritmo Genético Multiobjetivo, Control Predictivo, Twin Rotor, Sistema MIMO, RNA.

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ACRÓNIMOS

MPC	Control Predictivo Basado en Modelo
NMPC	Control Predictivo Basado en Modelo No Lineal
AG	Algoritmo Genético
MOGA	Algoritmo Genético Multiobjetivo
DM	Decision Maker
LD	Lógica Difusa
RNA	Red Neuronal Artificial
MIMO	Sistema Multivariable
SISO	Sistema Monovariable
TR	Twin-Rotor
NARX	RNA No-Lineal Auto-Regresiva con Entradas Externas
SMC	Control en Modo Deslizante
VSCS	Control de Sistemas de Estructura Variable
iMO-NMPC	Intelligent Multi-Objective Nonlinear Model Predictive Control

1. INTRODUCCIÓN

El presente trabajo ha sido desarrollado dentro del laboratorio del Grupo de Investigación de Control Inteligente (GICI) del Departamento de Ingeniería de Sistemas y Automática de la Universidad del País Vasco. El propósito de este grupo es avanzar en el estudio de nuevas estrategias de control incorporando técnicas de Computación Inteligente a las formas de control ya establecidas, tanto dentro de la industria como en el ámbito de la investigación.

La integración de nuevas tecnologías en los sistemas avanzados de control, sigue siendo hoy en día un reto actual, a medida que los dispositivos de control ganan en prestaciones y se ofrecen novedosas alternativas a las técnicas existentes. Prueba de ello es que las técnicas procedentes de la Computación Inteligente [5], [8], [10], [15], [16], [18], [19], tales como las Redes Neuronales Artificiales (RNA) [7], [9], los Algoritmos Genéticos (AG) [1], [5], [6], [15], [18], [19] o la Lógica Difusa (LD) [1], [15], permiten desarrollar sistemas de control lo suficientemente eficaces y robustos, sin necesidad de conocer por completo la dinámica de la planta. Esto se debe a que generalmente, no es posible obtener un modelo exacto de la misma, ya sea por complejidad o por incertidumbres derivadas del deterioro de los componentes.

Por otro lado, el uso del control predictivo es una realidad dentro del mundo de la industria del proceso. Su uso se ha ido extendiendo dentro del entorno industrial debido a su sencillez y robustez.

Es por ello, que el objetivo del presente trabajo es desarrollar una estrategia avanzada de control que combine las virtudes de ambos mundos, definiendo un controlador predictivo basado en las técnicas de computación inteligente anteriormente descritas.

En trabajos previos, dicha estrategia ya ha sido contrastada en sistemas monovariables (SISO) [25], y se propone dar el salto a un sistema multivariable (MIMO), en los cuales el acoplamiento de las variables del sistema es uno de los principales problemas a la hora de realizar el control de la planta. Para ello, el presente proyecto se

desarrollará sobre una maqueta denominada Twin-Rotor, la cual consiste en un par de rotores que simulan el comportamiento de un helicóptero.

Para ello, se dispondrá de las herramientas informáticas Matlab y Simulink desarrolladas por MathWorks®, tanto para el desarrollo del proyecto como para mostrar los resultados finales obtenidos.

A continuación, se resume brevemente el contenido de cada uno de los apartados de la presente memoria del proyecto:

- **Capítulo 1.** En el presente capítulo, se realiza una breve introducción del trabajo llevado a cabo, explicando de forma general en qué se basa el mismo y de dónde parte.
- **Capítulo 2.** En el segundo capítulo, se presentará el contexto del proyecto desarrollado, detallando el trabajo previo desarrollado en el grupo de investigación. Así como se describirá la maqueta empleada para la realización del trabajo.
- **Capítulo 3.** En el tercer capítulo, se definen los objetivos a cumplir a lo largo del desarrollo del proyecto, definiendo el objetivo principal, así como los objetivos parciales.
- **Capítulo 4.** En el cuarto capítulo, se describen los beneficios que aporta la finalización del proyecto, diferenciándolos entre beneficios técnicos y beneficios económicos.
- **Capítulo 5.** En este capítulo, se recogen las estrategias presentes en la literatura con relación al control y modelado de sistemas multivariables y así como la técnica iMO-NMPC desarrollada por el GICI, desde la cual se ha desarrollado el trabajo. Además, se presenta el fundamento teórico del control predictivo basado en modelo (MPC), al ser utilizado a lo largo del trabajo.
- **Capítulo 6.** En este capítulo, se muestra el desarrollo completo de la solución. Para ello se describen con detalle los diferentes pasos llevados a cabo para afrontar la consecución de los objetivos marcados al inicio del proyecto.

- **Capítulo 7.** En este capítulo, se muestran los resultados obtenidos a lo largo del desarrollo del proyecto, para ello, se muestran diferentes gráficas.
- **Capítulo 8.** En este capítulo, se describen las diferentes fases llevadas a cabo a la hora de la ejecución del trabajo, para ello, se muestra un diagrama de Gantt.
- **Capítulo 9.** En el presente capítulo, se analizan las conclusiones obtenidas, así como se presentan los futuros trabajos que se pueden abordar tras la finalización de este.
- **Referencias.** Por último, se muestran las referencias bibliográficas consultadas a lo largo del proyecto.
- **ANEXO I.** En dicho anexo, se presentan el código desarrollado a lo largo del proyecto. Se muestran los códigos más relevantes del mismo. Todo el código se encuentra alojado en el repositorio virtual del grupo de investigación en *GitHub* [25].

2. CONTEXTO

Hoy en día, con el continuo avance de la tecnología y el desarrollo de nuevas estrategias de control avanzado, tanto en el ámbito de la investigación como en la propia industria, se ha conseguido controlar de forma satisfactoria sistemas multivariables que anteriormente presentaban grandes dificultades en su control, debido a su carácter no lineal o su facilidad a volverse inestable.

En consecuencia, el control predictivo basado en modelo, tanto el lineal (MPC) como el no lineal (NMPC), han tomado mucha fuerza en los últimos años, debido a su simplicidad y a su robustez [1], [3], [7], [9], [11], [13], [17], [19]. Dicha estrategia de control, utiliza un modelo para predecir las salidas del proceso y calcular la acción de control futura a través de la minimización de una función de coste, siendo capaz de anticiparse a los cambios de referencia. A diferencia de los controladores convencionales como el controlador proporcional integral derivativo (PID), que actúan al producirse un error de seguimiento de la referencia.

Por otro lado, a fin de abordar el control de sistemas complejos, los ingenieros de control se han inspirado en el comportamiento del ser humano para dotar de "inteligencia" a los controladores, de ahí, surgen las técnicas de Computación Inteligente o *Soft Computing* anteriormente mencionadas [5], [8], [10], [15], [16], [18], [19]. Mediante esta "inteligencia" se le dota al lazo de control de capacidad de aprendizaje, adaptación, anticipación y predicción.

Por tanto, con el fin de combinar las virtudes del control predictivo no lineal y de las técnicas de *Soft Computing,* el Grupo de Control Inteligente desarrolló la técnica denominada iMO-NMPC [1], [19]. Dicha técnica trata el problema de optimización multiobjetivo a través del uso de algoritmos genéticos, generando una serie de posibles acciones de control. Por último, con la ayuda del *Decision Maker*, se selecciona una de las posibles soluciones, atendiendo a diferentes métodos de decisión, dependiendo de las características del problema a controlar. Por tanto, el controlador diseñado opera en línea (*On Line)*.

Para la realización del proyecto, en el laboratorio se dispone de la maqueta *Twin Rotor MIMO System 33-949S* diseñada por la empresa *Feedback Instruments [22]*, que simula el comportamiento de un helicóptero. La maqueta dispone de dos grados de libertad, pudiendo girar en el plano horizontal mediante el ángulo *yaw* y en el plano vertical mediante el ángulo *pitch*. A continuación, en la Figura 1, se muestra la maqueta.

Figura 1. Twin Rotor MIMO System 33-949S de Feedback Instruments.

A pesar de disponer de la maqueta real, el trabajo ha sido desarrollado en un entorno de simulación, a través de las herramientas informáticas de Matlab y Simulink de *MathWorks*®. Para simular el comportamiento del Twin-Rotor (TR), se han empleado las ecuaciones que describen la dinámica de la planta que define *Feedback Instruments*.

En conclusión, el principal problema que presenta dicha maqueta es que se trata de un sistema no lineal multivariable, por lo tanto, presenta una gran complejidad a la hora de su control. Por tanto, combinando las técnicas anteriormente descritas y partiendo de la técnica iMO-NMPC ya desarrollada dentro del grupo [1], [19], [25], se trata de desarrollar una estrategia de control avanzada capaz de controlar el Twin-Rotor de forma satisfactoria, aportando estabilidad y robustez al mismo.

3. OBJETIVOS Y ALCANCE

Como se ha comentado anteriormente, al tratarse de una planta con una dinámica no lineal y un sistema MIMO fuertemente acoplado, el principal objetivo del presente TFM es avanzar en el estudio de una estrategia de control de control para sistemas no lineales multivariables, basada en técnicas inteligentes, como Redes Neuronales Artificiales, Algoritmos Genéticos y la Lógica Difusa, aprovechando los beneficios que aportan cada una de ellas.

Anteriormente, dicha técnica fue desarrollada y contrastada en sistemas SISO por el Grupo de Investigación de Control Inteligente, obteniendo buenos resultados. Por lo tanto, el siguiente paso es dar el salto hacia sistemas MIMO, donde el acoplamiento entre las variables es uno de los principales problemas a la hora de realizar un control satisfactorio.

Por otro lado, para alcanzar de forma satisfactoria el objetivo principal, es necesario alcanzar una serie de objetivos de carácter secundario, pero de vital importancia:

1. **Estudiar cuáles son los componentes inteligentes de la estrategia iMO-NMPC.**

En primer lugar, para poder desarrollar una nueva estrategia de control inteligente para sistemas multivariables, es necesario identificar y estudiar los diferentes componentes que conforman la estrategia iMO-NMPC para sistemas SISO. De este modo, se podrá dar el salto a sistemas MIMO de forma satisfactoria.

2. **Integración de modelos MIMO basados en Redes Neuronales Artificiales y estudio de diferentes arquitecturas neuronales.**

Tras comprobar el correcto funcionamiento de las RNA para sistemas SISO, se deberán realizar entrenamientos para sistemas MIMO, comprobando su eficacia ante este tipo de sistemas. Una vez contrastada, se probarán y estudiarán diferentes arquitecturas neuronales, con el objetivo de obtener la RNA que mejores resultados posibles aporte.

3. **Analizar y diseñar diferentes métodos de decisión a la hora de seleccionar las acciones de control del Frente de Pareto.**

El *Decision Maker* cumple un papel muy importante a la hora de realizar el control multiobjetivo, por lo tanto, se probarán diferentes métodos de decisión, dependiendo del objetivo buscado, ya sea un control suave o un control algo más agresivo, pero más preciso.

4. **Experimentación y validación de la propuesta de control inteligente sobre la maqueta de un Twin-Rotor.**

Por último, tras desarrollar la estrategia iMO-NMPC para sistemas MIMO, el siguiente paso será comprobar su correcto funcionamiento. Se definirán diferentes referencias con el objetivo de ver cómo reacciona el sistema. Una vez contrastado, el siguiente paso sería probar la estrategia en la maqueta real que hay en el laboratorio del Grupo de Control Inteligente.

4. BENEFICIOS

Los beneficios obtenidos con la finalización del proyecto pueden ser agrupados en diferentes categorías:

4.1 Beneficios técnicos

Al haber sido contrastada la estrategia iMO-NMPC en sistemas monovariables, al ser modificada para poder actuar sobre la plataforma Twin-Rotor, se habría avanzado en el control de sistemas de mayor complejidad, pudiendo probarse en otros sistemas multivariables o con alta no linealidad.

Por otro lado, con la consecución de los objetivos marcados, se podrían diseñar nuevas líneas de investigación dentro del Grupo de Control Inteligente, pudiendo probarse la estrategia de control sobre la maqueta real.

4.2 Beneficios económicos

Una de las principales características de las redes neuronales artificiales es la capacidad de aprendizaje de las dinámicas de sistemas no lineales, en este caso, del Twin-Rotor. Por tanto, al emplear este tipo de redes para modelar sistemas complejos, se eliminaría la necesidad de tener que conocer las ecuaciones matemáticas que definen la dinámica de la planta para poder llegar a controlarla, considerando el sistema como una caja negra. Por tanto, se reduciría el tiempo necesario para modelar la planta.

Por otro lado, al emplear un controlador predictivo, se puede llevar a cabo un control más suave de la planta, evitando cambios bruscos en la variación de la señal de control, lo que supone un menor gasto energético. Además, al poder aplicar restricciones al controlador, se evita que el actuador se sature, aumentando la vida útil del mismo.

Por último, aunque los controladores NMPC producen un mayor coste computacional, al emplear una red neuronal artificial como modelo de predicción de la planta, el tiempo de cómputo total es menor que si se emplearan las ecuaciones matemáticas para realizar las predicciones.

5. ESTADO DEL ARTE

5.1 Modelado de sistemas MIMO

Antes de comprender y de poder realizar el control de sistemas multivariables, es necesario obtener un modelo que describa el comportamiento físico de la planta. Para la obtención del mismo, los autores han empleado diferentes métodos.

A continuación, se definen los métodos más usados para el modelado de sistemas multivariables, en concreto para el modelado del Twin-Rotor.

5.1.1 Modelo en espacio de estados

Se trata del modelo más empleado, a través de las ecuaciones matemáticas que definen la dinámica de la planta, se definen las variables de estado a utilizar y posteriormente se definen las ecuaciones de estado.

En el caso del Twin-Rotor, generalmente se emplean 6 variables de estado (*pitch, yaw, dpitch, dyaw, tau1* y *tau2)*, siguiendo las ecuaciones que *Feedback Instruments* pone a nuestra disposición a través del manual de usuario del TR, obtenidas a través de aplicar el equilibrio de momentos tanto en el plano vertical como en el plano horizontal [22].

Algunos autores emplean dichas ecuaciones [3], [8], [11], [13], [17], [21], [28], [29] mientras que otros obtienen las ecuaciones en espacio de estados aplicando *Euler-Lagrange* para calcular las velocidades angulares y lineales de los rotores [2].

Tras disponer de dichas ecuaciones algunos autores linealizan las ecuaciones de estado en torno a un punto de operación (generalmente (0,0,0,0,0,0)), con el objetivo de definir las matrices de estado (A, B, C, D) que definen a la planta [2], [3], [4], [11], [17], [21], [29]. De esta forma, consiguen un modelo lineal del Twin-Rotor, pudiendo emplear un controlador lineal, como, por ejemplo, un MPC.

5.1.2 Modelo neuronal

Actualmente, varios de autores están empleando redes neuronales artificiales con el fin de modelar sistemas multivariables. Así, la red actúa como un modelo en caja negra (únicamente se conocen las salidas en función del valor de las entradas, Figura 2), en el que no se conocen las físicas internas de la planta. De este modo, no es necesario conocer las ecuaciones que describen la dinámica interna de la planta.

Figura 2. Esquema modelo caja negra.

El uso de redes neuronales artificiales (RNA) se debe principalmente a las características propias de las redes en el modelado de sistemas dinámicos, como la capacidad de aproximar funciones no lineales con precisiones arbitrariamente pequeñas.

En consecuencia, las redes NARX (No-Lineal Auto-Regresiva con Entradas Externas) están consiguiendo gran popularidad en el modelado de sistemas no lineales, debido a las características anteriormente descritas y a ser redes no recurrentes. Dichas redes vienen definidas por la siguiente ecuación matemática de carácter no lineal:

$$\varphi(t) = [y(t-1) \dots y(t-n), u(t-d) \dots u(t-d-m)]^T \quad (1)$$

donde, como su propio nombre indica, las entradas *u(t)*, son entradas externas.

Por tanto, algunos autores han decidido emplear redes ARX o NARX para aproximar dinámicas no lineales [1], [4], [9], [19], [27], [36], mientras que otros autores han optado por emplear redes convencionales como son las redes perceptrón multicapa (MLP) [12]. Por otro lado, autores como [27], han entrenado RNA con el fin de diseñar observadores neuronales en espacio de estados, combinando las RNA con el modelado en espacio de estados.

5.1.3 Modelo en caja gris

El modelo en caja gris consiste en un método intermedio entre el modelo en caja blanca y el modelo en caja negra, en el que se combinan las características de ambos, siendo necesario conocer ciertas características de la planta a modelar.

En el caso del Twin-Rotor, las características físicas ya conocidas del TR, como son las ecuaciones propuestas por *Feedback Instruments* [22], que describen la relación entre las entradas y las salidas, son combinadas con comportamientos y características obtenidas a través de mediciones y de métodos experimentales. De este modo, se consigue un modelo en caja blanca mejorado, es decir, un modelo en caja gris, el cual posee información que el modelo inicial no tenía [12], [20], [28].

5.2 Control de sistemas MIMO

Debido a la complejidad en el control de sistemas multivariables, al haber acoplamiento entre las variables y debido al carácter (en general) no lineal de los mismos, como se ha comentado en capítulos anteriores, los autores han desarrollado diferentes estrategias de control con las que enfrentarse ante estas dificultades y así, lograr el control óptimo de sistemas de dinámica compleja.

A través de las referencias bibliográficas estudiadas, se han observado dos estrategias de control predominantes para el control de sistemas multivariables, más concretamente en el control de la maqueta Twin-Rotor. Estas técnicas son el control predictivo basado en modelo (MPC) y el control en modo deslizante (SMC).

5.2.1 Control predictivo basado en modelo

Se trata de la técnica de control más empleada por los autores para el control de sistemas multivariables [1], [3], [9], [11], [13], [16], [17], [18], [19]. Para comprender la importancia y el uso del control predictivo, es necesario comprender su origen. Surgió en el ámbito industrial, con el fin de dar solución a los principales problemas que tenían los controladores PID tradicionales como:

- Sintonización no intuitiva de los controladores.
- Restricciones físicas de los procesos.
- Control de sistemas no lineales.
- Necesidad de ajustar el coste económico de la producción.

En consecuencia, surgió el Control Predictivo basado en Modelo (MPC), para sistemas lineales y el Control Predictivo basado en Modelo No Lineal (NMPC), para sistemas no lineales; permitiendo la optimización de procesos multivariables y un mejor aprovechamiento de los actuadores al permitir aplicar restricciones (tanto en las entradas como en las salidas) al control.

Por tanto, para el control de sistemas como el Twin-Rotor, de carácter multivariable y altamente no lineal, varios autores han optado por el empleo de la estrategia de control NMPC [3], [7], [9], [11], [13], [17], [18], [19].

Dicha estrategia presenta el siguiente diagrama de bloques:

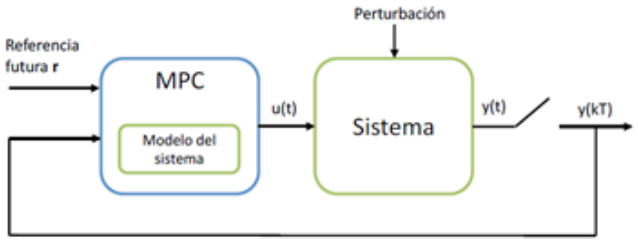

Figura 3. Diagrama control MPC.

Por tanto, para desarrollar un controlador NMPC es necesario definir los siguientes elementos.

- **Modelo del sistema.**

El modelo de predicción es el pilar fundamental de las estrategias MPC. Este se usa para predecir la salida del sistema en un instante de muestreo *t* en el horizonte deslizante *h* para *t+k*.

Anteriormente, se han descrito los diferentes modelos que se emplean para describir la dinámica del TR. En el caso de los controladores NMPC, los modelos más empleados son:

- **Modelo matemático.** La mayor parte de los autores desarrollan el control predictivo empleando las ecuaciones matemáticas del TR [3], [11], [13], [17], [18], [19].
- **Modelo neuronal.** Al combinar el control NMPC tradicional con las redes neuronales artificiales, surge el denominado Controlador Predictivo Neuronal (*Neural Network Predictive Controller NNPC*). En dicho controlador, el modelo de predicción empleado es una RNA entrenada para aproximar el comportamiento del Twin-Rotor [1], [4], [7], [14].

- **Función objetivo.**

Aunque cada estrategia predictiva define su propia función de coste, la mayoría de los autores incorporan los siguientes términos:

- Error futuro de seguimiento.
- Esfuerzo de control.

La forma habitual de la función de coste, suele ser una función cuadrática descrita en forma matricial como la siguiente:

$$J = (\hat{y} - w)^T R(\hat{y} - w) + u^{+T} Q u^{+} \quad (2)$$

donde *R* y *Q* son las matrices de ponderación del seguimiento de la referencia y de la variación de la señal de control respectivamente, $\hat{y}$ es la predicción de las salidas, *w* es la referencia conocida y u^{+} es la secuencia de control óptima.

A través de las matrices de ponderación, el control producido será diferente, pudiendo realizar una sintonización intuitiva. Por ejemplo, siendo R = [r_k] y Q = [q_k], se puede dar lo siguiente:

- Si $r_k > q_k$ el controlador presentará menos error de seguimiento y un control más agresivo.
- Si $r_k < q_k$ el controlador presentará más error de seguimiento y un control más suave.

- Si uno de los dos es 0, se eliminará el factor asociado a la minimización del mismo.

- **Ley de control.**

La obtención de la señal de control *u(t)* requiere resolver de un problema de minimización, para ello, se dispone de un optimizador.

El optimizador debe obtener la secuencia óptima de control u^+ a través de la minimización del siguiente problema, teniendo en cuenta las restricciones tanto de las entradas como de las salidas:

$$V = \min_u J\left(u^+, y(t)\right) \tag{3}$$

$$s.a.$$
$$\underline{y} \leq y(t+k) \leq \overline{y}$$
$$\underline{u} \leq u(t+k) \leq \overline{u}$$
$$\underline{\Delta u} \leq \Delta u(t+k) \leq \overline{\Delta u}$$

Por último, tras calcular la secuencia óptima de control, se seleccionará el primer instante u*(k), el cual será la acción de control a aplicar.

5.2.2 Control en modo deslizante

El éxito de los controladores de estructura variable (*Variable Structure Control Systems VSCS*), concretamente los controladores en modo deslizante (*Sliding Mode Control SMC*) reside en lograr que el sistema controlado se comporte tal y como se requiere aún en presencia de dinámicas no-modeladas, variaciones en los parámetros del sistema o incluso aproximaciones del modelo, realizadas habitualmente a la no linealidad del sistema a gobernar.

Esto lo consiguen a través de variar la estructura del controlador en función de determinadas condiciones, con el objetivo de obtener el comportamiento deseado del sistema.

La principal característica, por tanto, que presentan estos controladores es su robustez, tanto ante incertidumbres de tipo paramétrico o de dinámicas no modeladas como ante perturbaciones.

Por tanto, la ley de control conmutará entre dos expresiones en función de alguna condición, convirtiendo la estructura del sistema en variable. Entonces, si la frecuencia de conmutación es la adecuada, se puede alcanzar el objetivo fijado. Para ello, la base del diseño del SMC reside en la elección adecuada de la variable conmutante *s(x,t)*. Cuando *s(x,t)=0* (Figura 4), se dice que el sistema se encuentra en régimen deslizante, por tanto, los estados del sistemas convergerán a los valores deseados, tendiendo el error a 0.

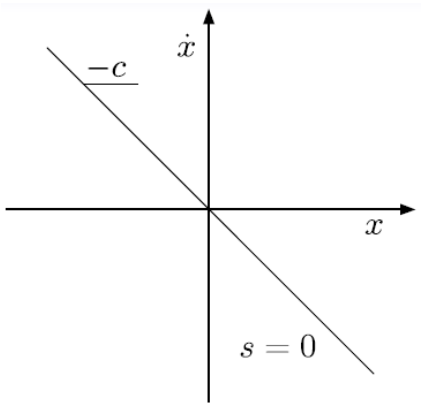

Figura 4. Superficie deslizante.

En consecuencia, varios autores han decidido emplear el control en modo deslizante para controlar la maqueta del Twin-Rotor, obteniendo resultados aceptables [3], [21].

Para el control del TR, los autores definen tres variables conmutantes S_1, S_2 y S_3, para cada uno de los subsistemas vertical y horizontal. Por tanto, finalmente se emplean 6 variables conmutantes en total.

Para garantizar la estabilidad de ambos subsistemas, se aplica la función candidata de Lyapunov a ambos subsistemas.

Subsistema vertical:

$$V_v = \frac{S_{1v}^2 + S_{2v}^2 + S_{3v}^2}{2} > 0 \tag{4}$$

Subsistema horizontal:

$$V_h = \frac{S_{1h}^2 + S_{2h}^2 + S_{3h}^2}{2} > 0 \tag{5}$$

Por tanto, para que ambos subsistemas sean estables y el sistema converja a 0, su derivada temporal deberá ser definida negativa. Por consiguiente:

Subsistema horizontal:

$$\dot{V}_h = S_{1h}\dot{S_{1h}} + S_{2h}\dot{S_{2h}} + S_{3h}\dot{S_{3h}} < 0 \quad (6)$$

Subsistema vertical:

$$\dot{V}_v = S_{1v}\dot{S_{1v}} + S_{2v}\dot{S_{2v}} + S_{3v}\dot{S_{3v}} < 0 \quad (7)$$

Cumpliendo las Ecuaciones 5 y 6, el error del sistema tenderá a 0 y su estabilidad estará garantizada.

5.3 iMO-NMPC

Dentro del Grupo de Investigación de Control Inteligente (GICI), se desarrolló la técnica denominada iMO-NMPC (*Intelligent Multi-Objective Nonlinear Model Predictive Control)* para el control de sistemas multivariables [1], [19], combinando el NMPC con las técnicas de Computación Inteligente, con el fin de desarrollar una estrategia avanzada de control capaz de trabajar *on line*.

Dicha estrategia de control, presenta la siguiente estructura [1]:

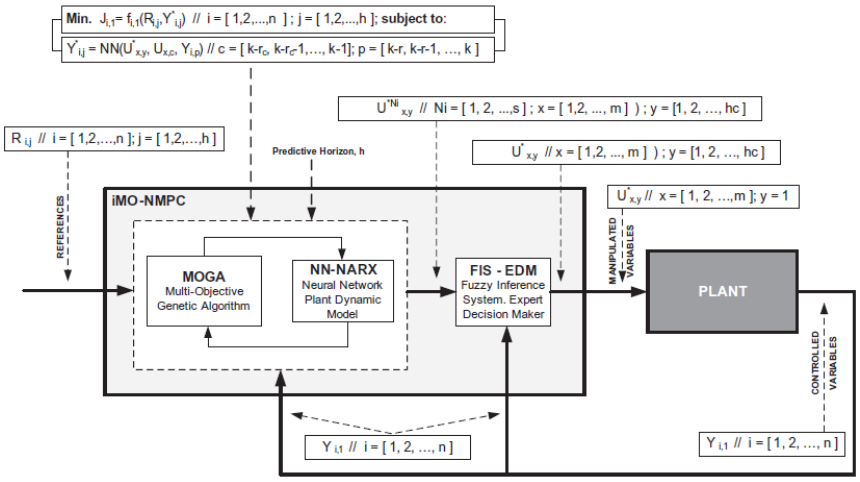

Figura 5. Estructura y diagrama de bloques iMO-NMPC.

Como se puede observar a través de la Figura 5, está compuesto por tres elementos:

- **Optimizador.**

El optimizador realiza un problema de optimización multiobjetivo, en el que se enfrentan la minimización del error de seguimiento de la referencia y la minimización de la variación de la señal de control. La función de coste que da lugar a dicho problema de optimización es similar a la Ec. (3).

En dicha estrategia, el optimizador utilizado es un algoritmo genético multiobjetivo (MOGA), el cual calcula la secuencia de control óptima u^+ en función del horizonte de control *Hu*, definiendo las soluciones no dominadas que forman el frente de Pareto (Figura 6 [1]).

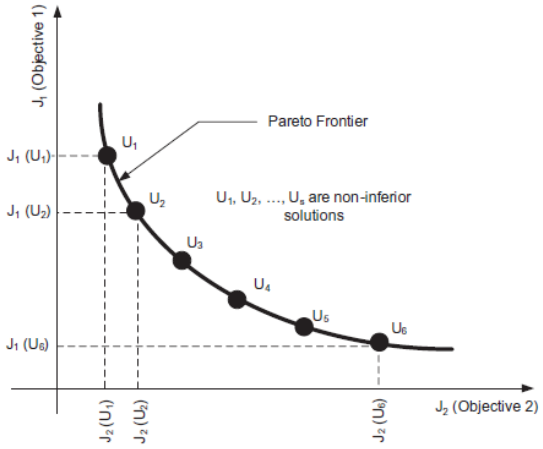

Figura 6. Ejemplo de Frente de Pareto de dos objetivos enfrentados.

En conclusión, el uso de MOGA en problemas de optimización multiobjetivo es debido a la gran complejidad de dichos problemas y a la capacidad de los MOGA de obtener soluciones ante dichos problemas. Es por ello, que otros autores han decidido solucionar este tipo de problemas con MOGA [6], [15], [16], [19].

- **Modelo de predicción.**

El modelo de predicción que emplea la técnica iMO-NMPC es una red NARX, debido a su capacidad de aproximar sistemas no lineales (como se ha comentado anteriormente).

La red es usada por el MOGA para evaluar cada uno de los individuos de la población, con el fin de encontrar el mejor individuo.

- **Sistema de decisión.**

Tras definir el Frente de Pareto, se emplea un Sistema de Decisión (*Decision Maker)*, para seleccionar una solución entre las posibles soluciones del Frente de Pareto. Al tratarse de un problema de solo dos objetivos, con una inspección del Frente de Pareto es suficiente, sin necesidad de emplear técnicas muy complejas.

Para ello, hay diferentes métodos usados por los autores como pueden ser los Sistemas Expertos o La Lógica Difusa [1], [15] o el criterio de Nash [18]. Otros autores en cambio, seleccionan la solución en función de la mínima distancia al origen [16], [19].

6. DESARROLLO DE LA SOLUCIÓN

6.1 Estudio dinámico del Twin-Rotor

En primer lugar, se decidió estudiar la dinámica de la maqueta, con el fin de conocer su comportamiento y su carácter no lineal, para seguidamente, diseñar un controlador predictivo basado en modelo no lineal (NMPC), el cual empleará el modelo en espacio de estados del Twin-Rotor. Para ello, *Feedback Instruments* pone a nuestra disposición el manual de usuario de su maqueta, de donde se han obtenido las ecuaciones, a pesar de que otros autores emplean otras ecuaciones [2], [19], se decidió usar las que propone el fabricante.

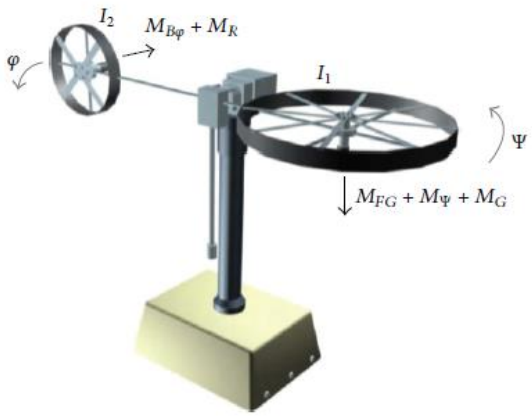

Figura 7. Momentos de inercia en ambos planos.

Entonces, dividiendo el Twin-Rotor en dos partes, por un lado, el rotor principal (plano vertical) y por otro, el rotor de la cola (plano horizontal). En la anterior figura, se muestran los momentos en ambos planos según [22], donde aplicando el equilibrio de momentos, se definen las ecuaciones en cada uno de los planos.

En el plano vertical:

$$I_1\ddot{\psi} = M_1 - M_{FG} - M_{B\psi} - M_G \tag{8}$$

donde

$$M_1 = a_1\tau_1^2 + b_1\tau_1 \tag{9}$$

$$M_{FG} = M_g \sin(\psi) \tag{10}$$

$$M_{B\psi} = B_{1\psi}\,\dot{\psi} - \frac{0.0326}{2}\sin(2\psi)\,\dot{\varphi}^2 \tag{11}$$

$$M_G = k_{gy} M_1 \dot{\varphi}\cos(\psi) \tag{12}$$

Por otro lado, en el plano horizontal:

$$I_2\ddot{\varphi} = M_2 - M_{B\varphi} - M_R \tag{13}$$

donde

$$M_2 = a_2\tau_2^2 + b_2\tau_2 \tag{14}$$

$$M_{B\varphi} = B_{1\varphi}\,\dot{\varphi} \tag{15}$$

$$M_R = k_c\frac{T_0 s + 1}{T_p s + 1}M_1 \tag{16}$$

Derivando la ecuación (16), se obtienen las ecuaciones de los momentos de los rotores:

$$\tau_1 = \frac{k_1}{T_{11}s + T_{10}}u_1 \tag{17}$$

$$\tau_2 = \frac{k_2}{T_{21}s + T_{20}}u_2 \tag{18}$$

Por último, uniendo todas las ecuaciones, se obtienen las ecuaciones que definen la dinámica del TR:

$$\frac{d\dot{\psi}}{dt} = \frac{a_1}{I_1}\tau_1^2 + \frac{b_1}{I_1}\tau_1 - \frac{M_g}{I_1}\sin(\psi) + \frac{0.0326}{2I_1}\sin(2\psi)\,\dot{\varphi}^2 - \frac{B_{1\psi}}{I_1}\dot{\psi} - \frac{k_{gy}}{I_1}\cos(\psi)\,(a_1\tau_1^2 + b_1\cdot\tau_1)\dot{\varphi} \tag{19}$$

$$\frac{d\dot{\varphi}}{dt} = \frac{a_2}{I_2}\tau_2^2 + \frac{b_2}{I_2}\tau_2 - \frac{B_{1\varphi}}{I_2}\dot{\varphi} - \frac{1.75k_c}{I_2}(a_1\tau_1^2 + b_1\tau_1) \tag{20}$$

$$\frac{d\tau_1}{dt} = -\frac{T_{10}}{T_{11}}\tau_1 + \frac{k_1}{T_{11}}u_1 \tag{21}$$

$$\frac{d\tau_2}{dt} = -\frac{T_{20}}{T_{21}}\tau_2 + \frac{k_2}{T_{21}}u_2 \tag{22}$$

Una vez obtenidas las ecuaciones, se han definido las variables de estado, las entradas y las salidas del sistema.

$$x = \begin{pmatrix} \psi \\ \dot{\psi} \\ \varphi \\ \dot{\varphi} \\ \tau_1 \\ \tau_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \qquad u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \qquad y = \begin{pmatrix} \psi \\ \varphi \end{pmatrix}$$

Donde ψ corresponde al ángulo *pitch*, φ al ángulo *yaw*, τ_1 y τ_2 al par aplicado por los motores principal y de la cola respectivamente; por último, u_1 y u_2 corresponden a las tensiones aplicadas a cada uno de los motores.

Tras definir las variables de estado, las ecuaciones de estado resultantes son las siguientes:

$$\frac{dx_1}{dt} = x_2 \tag{23}$$

$$\frac{dx_2}{dt} = \frac{a_1}{I_1}x_5^2 + \frac{b_1}{I_1}x_5 - \frac{M_g}{I_1}\sin(x_1) - \frac{B_{1\psi}}{I_1}x_2 + \frac{0.0326}{2I_1}\sin(2x_1)\,x_4^2 - \frac{k_{gy}}{I_1}\cos(x_1)\,(a_1x_5^2 + b_1x_5)x_4 \tag{24}$$

$$\frac{dx_3}{dt} = x_4 \tag{25}$$

$$\frac{dx_3}{dt} = \frac{a_2}{I_2}x_6^2 + \frac{b_2}{I_2}x_6 - \frac{B_{1\varphi}}{I_2}x_4 - \frac{1.75k_c}{I_2}\left(a_1x_5^2 + b_1x_5\right) \tag{26}$$

$$\frac{dx_5}{dt} = -\frac{T_{10}}{T_{11}}x_5 + \frac{k_1}{T_{11}}u_1 \tag{27}$$

$$\frac{dx_6}{dt} = -\frac{T_{20}}{T_{21}}x_6 + \frac{k_2}{T_{21}}u_2 \tag{28}$$

A continuación, se muestran los demás parámetros del TR:

Tabla 1. Parámetros del Twin-Rotor.

I_1 Momento de inercia rotor vertical	$6.8\cdot10^{-2}$ kg·m^2
I_2 Momento de inercia rotor horizontal	$2\cdot10^{-2}$ kg·m^2
a_1 Característica estática	0.0135
a_2 Característica estática	0.0924
b_1 Característica estática	0.02
b_2 Característica estática	0.09

M_g Momento gravitatorio	0.32 N·m
$B_{1\psi}$ Función de momento de fricción vertical	$6 \cdot 10^{-3}$ N·m·s/rad
$B_{1\varphi}$ Función de momento de fricción horizontal	$1 \cdot 10^{-3}$ N·m·s/rad
$B_{2\psi}$ Función de momento de fricción vertical	$1 \cdot 10^{-1}$ N·m·s^2/rad
$B_{2\varphi}$ Función de momento de fricción horizontal	$1 \cdot 10^{-2}$ N·m·s^2/rad
k_{gy} Parámetro momento giroscópico	0.05 s/rad
k_1 Ganancia rotor principal	1.1
k_2 Ganancia rotor de la cola	0.8
k_c Ganancia de impulso de reacción cruzada	-0.2
T_{11} Denominador rotor principal	1.1
T_{10} Denominador rotor principal	1
T_{21} Denominador rotor de la cola	1
T_{20} Denominador rotor de la cola	1
T_P Parámetro momento reacción cruzada	2
T_0 Parámetro momento reacción cruzada	3.5

Tras obtener las ecuaciones de estado, ya se dispone de un modelo de predicción con el que diseñar el controlador predictivo.

6.2 Estudio y diseño del controlador NMPC

Para el diseño del controlador NMPC, se ha decidido trabajar en el entorno de Matlab/Simulink; en Matlab se realizará la inicialización de los parámetros y la definición del controlador, mientras que en Simulink se simulará el control del TR y se graficarán las variables del proceso necesarias. Para ello, se ha empleado el bloque NMPC de la librería de Simulink. Antes de diseñar el controlador, se hizo un estudio de los parámetros configurables del controlador [23].

Tras comprender la configuración del mismo, se han definido los parámetros del Twin-Rotor y se ha creado el objeto *nlmpc* que corresponde al controlador predictivo en un *script*, mostrado en el Anexo I. Para definir el controlador, se han configurado los siguientes parámetros y funciones:

- Horizonte de predicción *h*.
- Horizonte de control *hu*.
- Número de variables de estado y sus nombres, 6.
- Número de entradas y sus nombres, 2.
- Número de salidas, 2.
- Función correspondiente al modelo de predicción, se han empleado las ecuaciones (23) a (28).
- Función que representa las salidas del sistema, en este caso son *pitch* y *yaw*.
- Tiempo de muestreo *Ts*, 0.1 s.
- Ponderación del seguimiento de la referencia *R*.
- Ponderación de la señal de control *Q*.
- Restricciones, en este caso, solo se han considerado restricciones en las señales de control. Los motores eléctricos de la maqueta trabajan en el rango de ±2.5 V.

Por otro lado, se han definido las referencias, tanto para el ángulo *pitch* como para el ángulo *yaw*. Cabe destacar que las señales de referencia han sido suavizadas, empleando funciones sigmoideas, de forma que el cambio de referencia no sea muy brusco. De esta forma, se simularía como sería un cambio de referencia en la realidad y se evitaría que el sistema se inestabilice fácilmente. A continuación, se muestran las referencias:

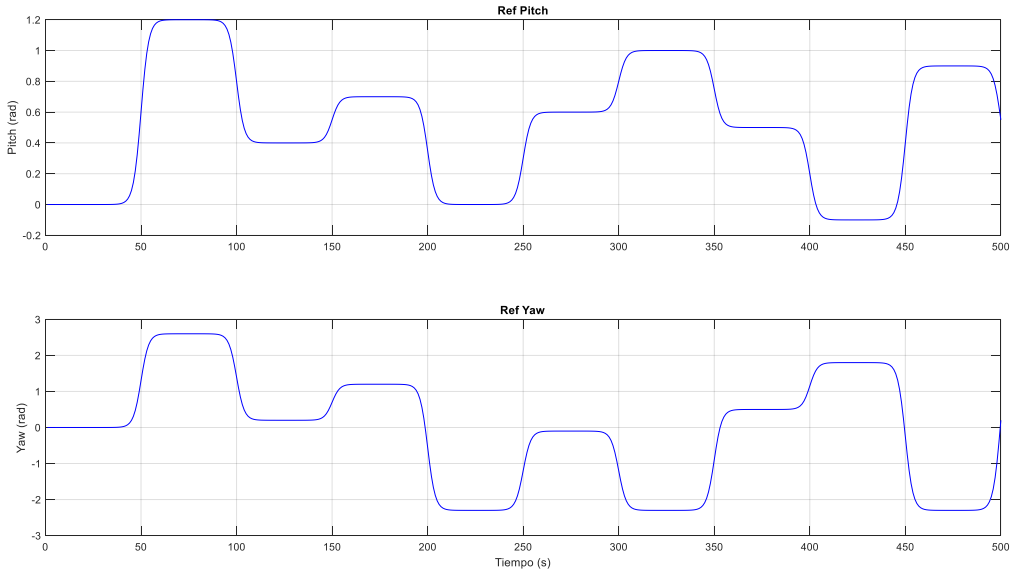

Figura 8. Referencias iniciales para pitch y yaw, respectivamente.

Por tanto, una vez inicializados todos los parámetros necesarios, se ha definido el esquema de Simulink donde se realizarán las simulaciones. A continuación, se muestra el mismo:

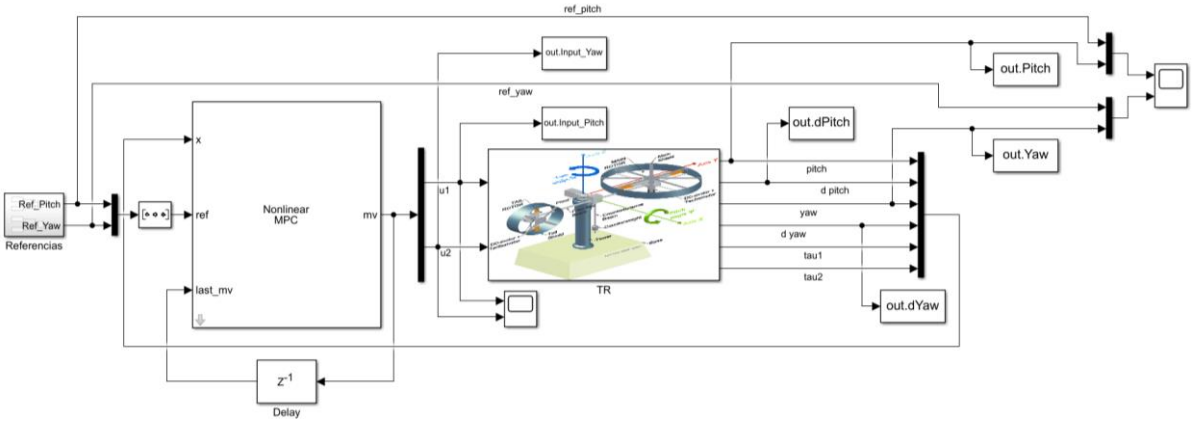

Figura 9. Esquema de Simulink NMPC.

Como se observa en la anterior figura, el bloque TR corresponde a la planta "real", por lo tanto, es necesario implementarla para que actúe como tal. Para ello, se han desarrollado dos estrategias, por un lado, se ha desarrollado una *s-function* mediante un *script* la cual emplea las ecuaciones del modelo de predicción; por otro lado, se ha definido la planta mediante un diagrama de bloques, partiendo de las mismas ecuaciones. En este caso, se ha decidido emplear el diagrama de bloques al ser más visual y más fácil de modificar. En la siguiente figura, se muestra el mismo:

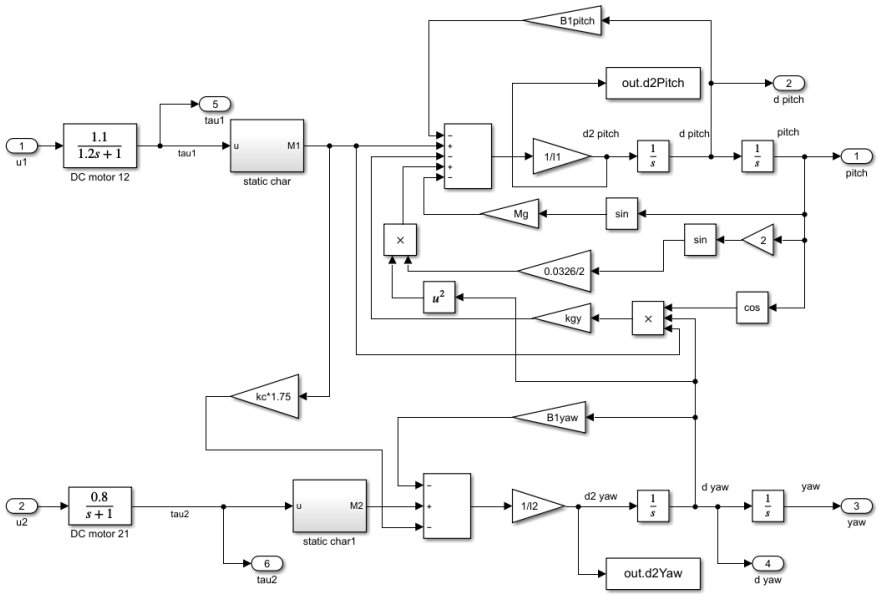

Figura 10. Diagrama de bloques que simula el TR real.

Tras realizar varias simulaciones, se ha comprobado el gran problema que presentan los sistemas MIMO: el fuerte acoplamiento entre el *pitch* y el *yaw*, al producirse un gran cambio en el ángulo *yaw*, este produce un gran error de seguimiento en el ángulo *pitch*, ya que el *pitch* depende de la velocidad del *yaw*, como muestra la ecuación (19).

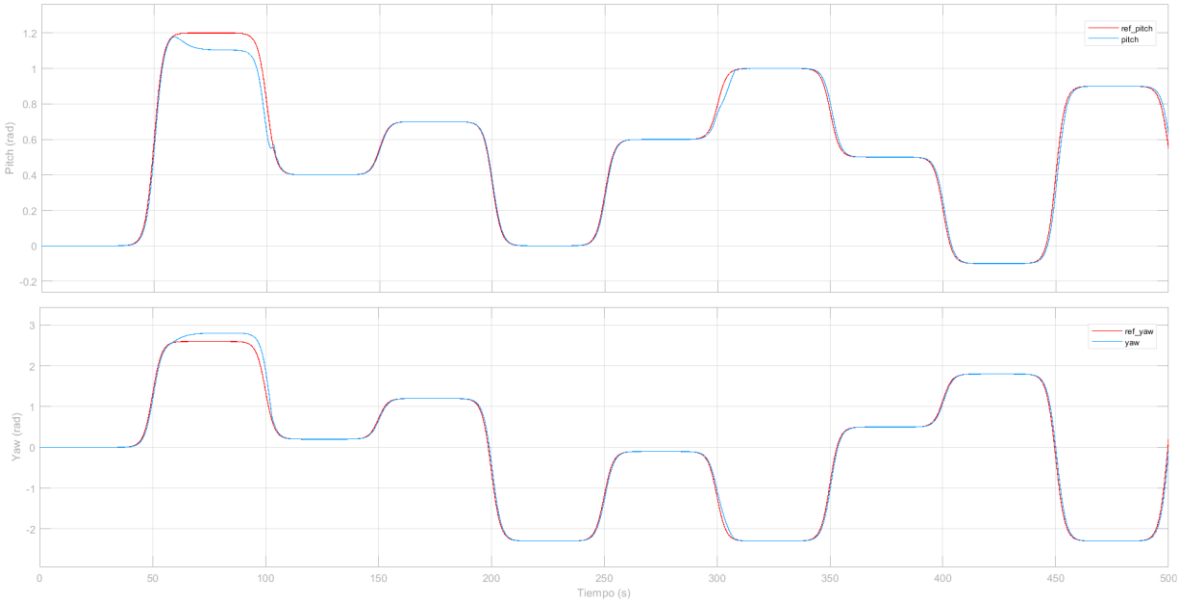

Figura 11. Pitch y yaw ante referencias iniciales.

Por otra parte, como se observa en la anterior figura, en algunos instantes, el sistema no es capaz de alcanzar la referencia marcada, debido a que se produce una

saturación de uno de los actuadores, en concreto en el del *yaw*, produciéndose un gran error de seguimiento en ambos ángulos. Para evitar esto, se ha decidido reducir la amplitud de las dos referencias, de este modo, el sistema sería capaz de alcanzar las referencias marcadas sin necesidad de que los actuadores alcancen sus valores mínimos y máximos. En la siguiente figura, se muestra el fenómeno descrito.

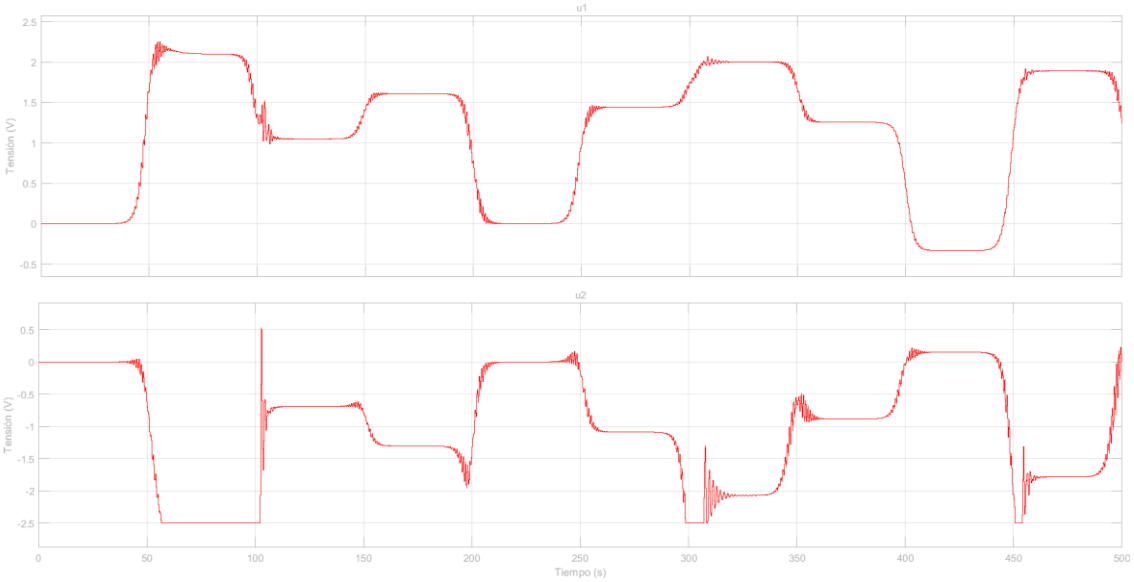

Figura 12. Saturación de los controladores.

6.3 Entrenamiento de modelos neuronales

Tras comprobar el correcto control del NMPC, el siguiente paso que se ha dado ha sido entrenar una serie de RNA de forma que simulen el comportamiento del Twin-Rotor, de modo que se puedan usar como modelo de predicción dentro del controlador NMPC. Para ello, se ha dispuesto del *toolbox* de Matlab para el entrenamiento de RNA. En este caso, se ha empleado una red NARX, debido a su eficacia a la hora de reproducir el comportamiento de sistemas no lineales. Por tanto, se ha empleado el comando *narxnet*, configurándolo sus parámetros atendiendo a [23].

Se han definido dos lotes de referencias, unas para el entrenamiento de las redes y otro para la validación de las mismas. Además, estas han sido modificadas con respecto a las iniciales: por un lado, se ha aumentado el tiempo de simulación de 500 a 1000 segundos, de esta forma, se disponen de más muestras de entrenamiento para la red; por otro lado, se han barrido más zonas de funcionamiento, en consecuencia, la RNA será capaz de realizar una mejor predicción de la dinámica del sistema. Además, se ha

reducido la amplitud de las mismas, como se ha comentado en el apartado anterior. Por tanto, las nuevas referencias son las siguientes:

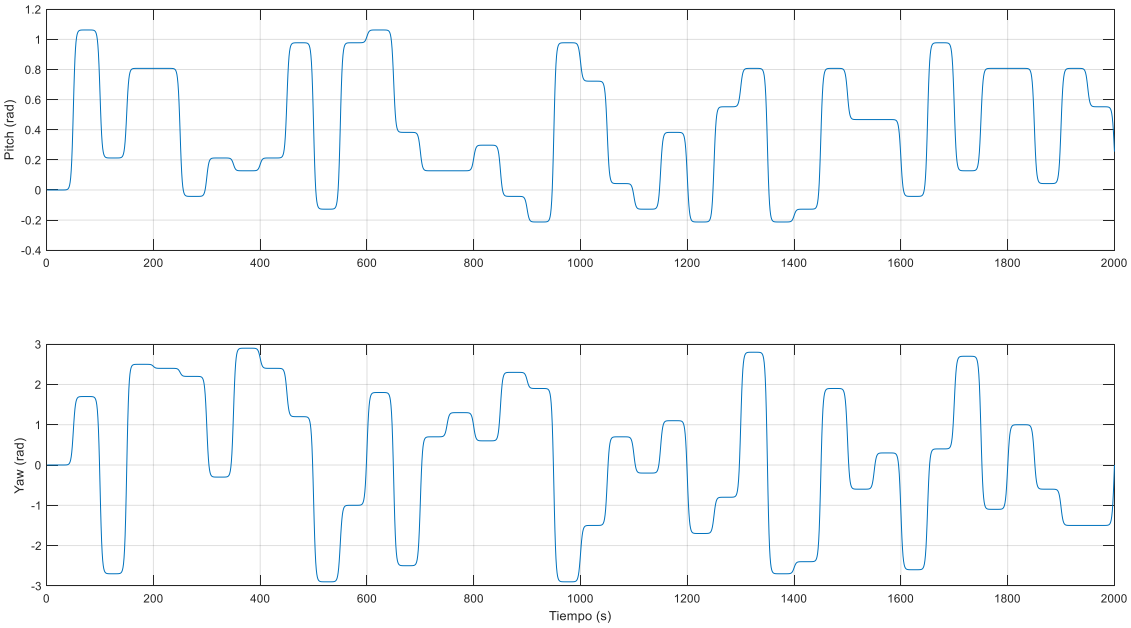

Figura 13. Nuevas referencias pitch y yaw para el entrenamiento.

Para entrenar la RNA, es necesario disponer de los datos de entrada aplicados y los datos de salida deseados, de forma que esta actúe como una caja negra. Por tanto, para generar los datos de entrenamiento, se ha empleado el modelo de Simulink del apartado anterior. Por tanto, empleando unos horizontes de predicción y de control de 7, con una ponderación de las referencias de 3 y con una ponderación de las señales de control de 0.1, las salidas ante dichas referencias son las siguientes:

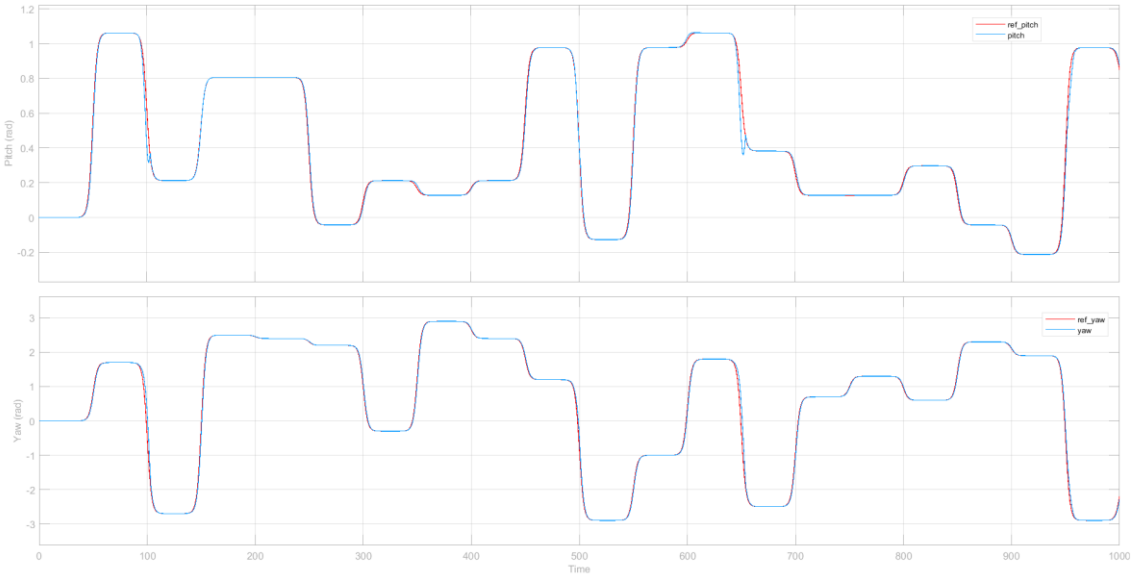

Figura 14. Pitch y yaw ante la referencia para el entrenamiento.

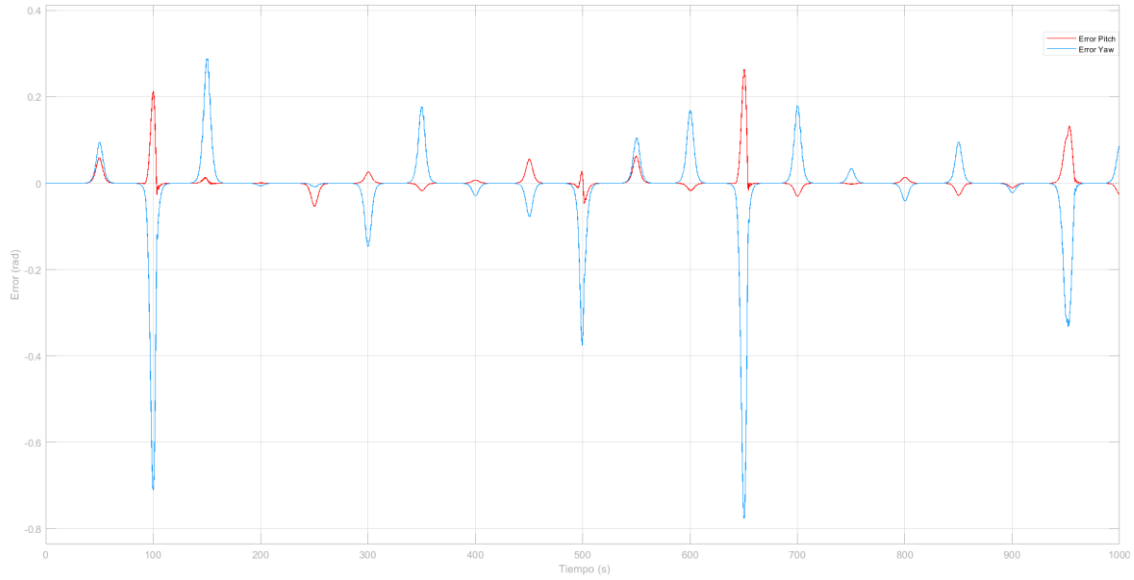

Figura 15. Error de pitch y yaw ante la referencia para el entrenamiento.

Como se observa, el error cometido tras los cambios mencionados es prácticamente inexistente, únicamente se observa error cuando se produce un cambio rápido en el *yaw*. Por tanto, las tensiones aplicadas para el seguimiento de estas referencias serán las entradas aplicadas a la RNA para su entrenamiento, mientras que las salidas (*pitch* y *yaw*) serán las salidas deseadas de la RNA. A continuación, se muestran las tensiones aplicadas.

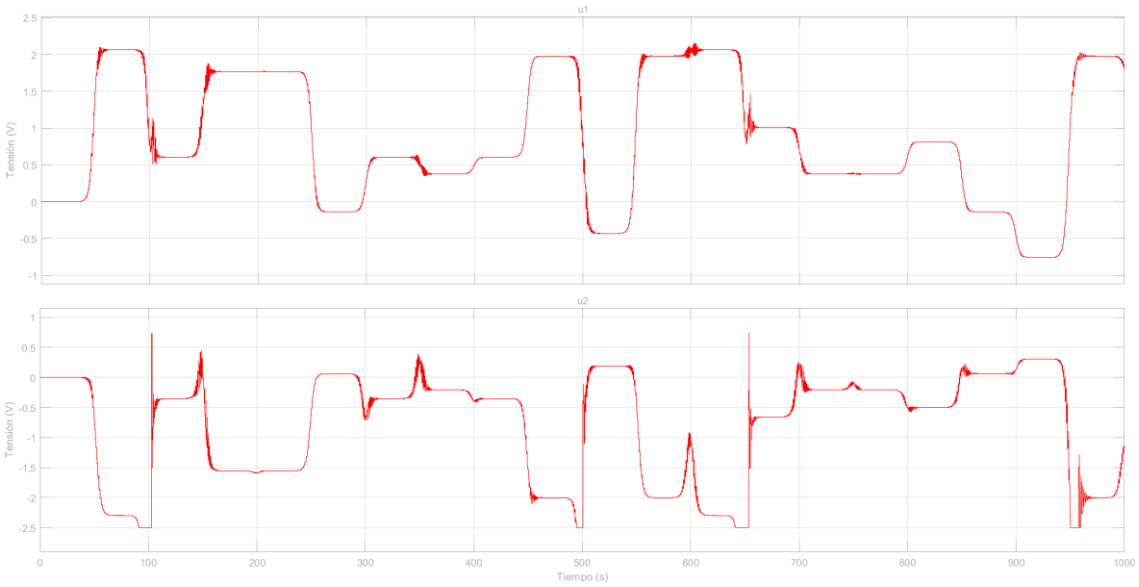

Figura 16. Tensiones para el entrenamiento de las RNA.

Por tanto, al ser el tiempo de muestreo, tanto del *solver* de Matlab como del controlador NMPC, al haber simulado 1000 segundos, se disponen de 10000 muestras para el entrenamiento de las redes, como se muestra en el apartado anterior. Dicha cantidad, se considera que es suficiente para un correcto entrenamiento de las RNA.

La RNA entrenada, actuará como modelo de predicción del bloque NMPC. Como el bloque que ofrece Simulink trabaja con modelos en espacio de estados, la RNA deberá actuar del mismo modo. En el caso del modelo matemático, este trabajaba con 6 variables de estado (pitch, yaw, dpitch, dyaw, tau1 y tau2); sin embargo, la RNA trabajará con 4 variables de estado, sin tener en cuenta el par de cada uno de los motores.

Por tanto, antes de comenzar a entrenar redes, se ha estudiado cómo trabaja internamente el bloque NMPC, ya que la RNA entrenada, deberá ser incorporada en el objeto NMPC para que actúe como modelo de predicción del TR.

Tras depurar el código, se ha observado que cada vez que se llama a la función *modelo_TR* (Anexo I), esta recibe las variables de estado en el instante *k*. Entonces, la función calcula las derivadas de cada una de las variables de estado (*dx/dt*) siguiendo las ecuaciones del modelo matemático, las cuales son las salidas de dicha función, para después calcular las predicciones las variables de estado en el instante siguiente (*x(k+1)*). Este cálculo lo realiza siguiendo la siguiente ecuación:

$$x(k+1) = x(k) + \frac{dx}{dt} \tag{29}$$

Por tanto, para que la RNA pueda actuar como modelo de predicción de forma correcta, se ha de desarrollar una función que, a través de las salidas de la red, calcule las derivadas de las variables de estado. Por tanto, para el cálculo de dichas variaciones, se emplearán las siguientes ecuaciones:

$$x = \begin{pmatrix} \psi \\ \dot{\psi} \\ \varphi \\ \dot{\varphi} \end{pmatrix} \rightarrow \dot{x} = \begin{pmatrix} \dot{\psi} \\ \ddot{\psi} \\ \dot{\varphi} \\ \ddot{\varphi} \end{pmatrix} = \begin{pmatrix} \dot{\psi} \\ (\hat{\psi} - \psi) - \dot{\psi} \\ \dot{\varphi} \\ (\hat{\varphi} - \varphi) - \dot{\varphi} \end{pmatrix} \tag{30}$$

Atendiendo a las ecuaciones anteriores, se observa que las derivadas de las variables de estado corresponden a las velocidades y las aceleraciones de los ángulos *pitch* y *yaw*. Por tanto, la RNA deberá realizar predicciones lo suficientemente precisas, de modo que, a través de estas, se calculen las velocidades y aceleraciones con suficiente precisión.

Entonces, el siguiente paso antes de comenzar a entrenar las redes neuronales artificiales, es definir la estructura de las mismas. Es decir, se han de decidir las entradas, las salidas y el número de neuronas en la capa oculta.

Para ello, se dispone de dos entradas, *u1* y *u2*, y dos salidas, *pitch* y *yaw*. Por tanto, la red dispondrá de al menos cuatro entradas (2 correspondientes a las entradas y 2 correspondientes a las salidas realimentadas con un retardo), pudiendo ser ampliadas mediante retardos, tanto de las entradas como de las salidas; y dos salidas. Las variables de estado correspondientes a las velocidades *dpitch* y *dyaw y a las aceleraciones d2pitch* y *d2yaw*, se obtendrán siguiendo las siguientes fórmulas:

$$dpitch(k) = \frac{pitch(k) - pitch(k-1)}{Ts} \tag{31}$$

$$dyaw(k) = \frac{yaw(k) - yaw(k-1)}{Ts} \tag{32}$$

$$d^2pitch(k) = \frac{dpitch(k) - dpitch(k-1)}{Ts} \tag{33}$$

$$d^2yaw(k) = \frac{dyaw(k) - dyaw(k-1)}{Ts} \quad (34)$$

Por tanto, se han diseñado y estudiado diferentes configuraciones de redes NARX, con el fin de obtener un modelo de predicción preciso. En el siguiente capítulo, se definen las distintas redes probadas, así como la red neuronal que mejores resultados ha ofrecido.

Tras entrenar las redes, estas han sido validadas. Para la validación de las mismas, se han empleado dos métodos diferentes. Por un lado, se ha empleado el método compacto, que realiza la validación de todas las muestras simultáneamente; y, por otro lado, se ha empleado el método Paso a Paso, el cual para cada muestra individualmente, realiza los cálculos internos que realizaría la RNA.

Además de comprobar la viabilidad de las RNA a través de un *script*, se ha decidido simular su comportamiento dentro del esquema de Simulink. Para incorporar la red neuronal, se ha empleado el bloque *NNET* de Matlab. Para ello, se dispuso del siguiente esquema:

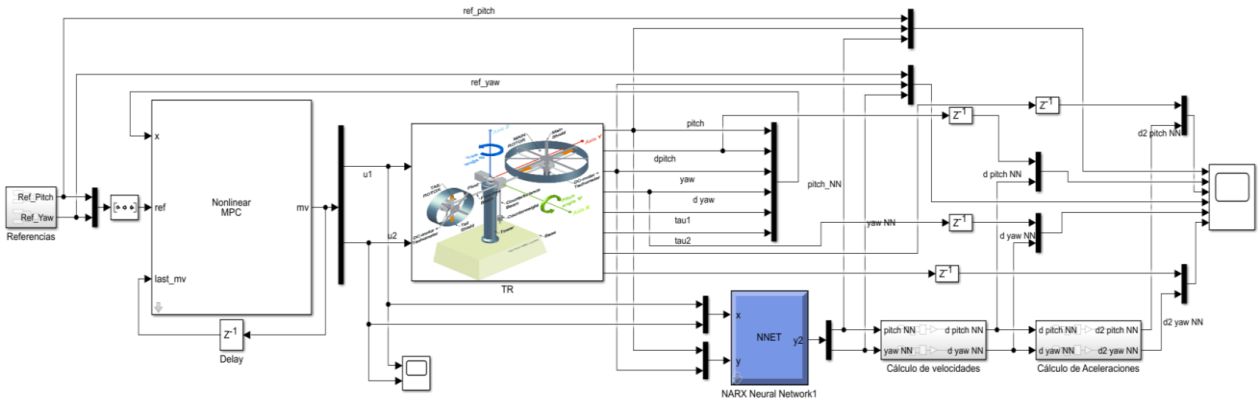

Figura 17. Esquema comparación modelo matemático y RNA, velocidades y aceleraciones.

Cabe destacar que el bloque NNET tiene en cuenta los retardos con los que fue entrenada la red, por lo tanto, únicamente es necesario conectar las entradas y salidas empleadas, sin incluir los retardos.

En primer lugar, se realizó el esquema únicamente calculando y comparando las velocidades. Al comprobar que las predicciones se realizaban con una buena precisión, se realizó el mismo procedimiento para las aceleraciones.

6.4 Incorporación de la RNA como modelo de predicción del NMPC

El siguiente paso llevado a cabo, ha sido la incorporación de la red que mejores resultados ha dado como modelo de predicción del bloque NMPC, con el fin de reemplazar el modelo matemático anteriormente utilizado por el modelo neuronal entrenado.

Tras comprobar que la RNA ofrece predicciones correctas, se ha desarrollado la función *NN_TR* que define el modelo neuronal de predicción. Dicha función emplea el método *Paso a Paso* para el cálculo de las salidas de la red, para después calcular las variaciones de las variables de estado.

Para el correcto funcionamiento del modelo neuronal, se necesitan las siguientes variables (correspondientes a los parámetros de la mejor RNA): u*1(k)* y *u2(k), pitch(k-1), yaw(k-1)*, *pitch(k-2)* y *yaw(k-2),* necesarias tanto para actuar como entradas de la red, como para el cálculo de las variaciones de las variables de estado. Por tanto, la función debe recibir dichas variables. Sin embargo, al emplear el código ya desarrollado por Matlab, este solo permite recibir *u(k)* y *x(k)*.

Para tratar de solucionar dicho problema, se han probado las siguientes formas de almacenar las variables:

- **Variables globales.** Sin embargo, como el script del modelo de predicción es llamado en varias ocasiones, no se almacenaban de forma correcta las variables.
- **Aumento de variables de estado.** Se han aumentado el número de estados a 8 (4 variables de estado en k y 4 variables de estado en k-1 usando retardos) para poder pasar como datos de entrada las salidas en el instante anterior. Sin embargo, al tener que definir las ecuaciones de estado de los nuevos estados, no ha sido posible su implementación.

A pesar de probar dichas configuraciones, no fue posible realizar el control del Twin-Rotor de forma correcta. Se podría haber modificado los scripts desarrollados por Matlab, sin embargo, hubiera sido necesario bajar a niveles de programación inferiores, lo cual no es un objetivo del presente trabajo.

6.5 Modelo híbrido

6.5.1 Con predicciones de 2 variables de estado

La siguiente prueba que se ha llevado a cabo es el desarrollo de un modelo híbrido. Dicho modelo combinará el modelo matemático en espacio de estados con la RNA. Para ello, calculará *dx/dt* mediante el modelo matemático, sin embargo, para su cálculo no empleará el valor de *pitch* y de *yaw* de la simulación; si no que empleará las predicciones de los mismos, para ello se dispone de la RNA. Así se comprobará el comportamiento de ambos modelos trabajando simultáneamente.

Cabe destacar que dicho modelo empleará 6 variables de estado, al igual que el modelo matemático. Por tanto, cada instante de muestreo, se realimentan al bloque NMPC las 6 variables de estado y las tensiones. Después, se pasan los valores de *u1*, *u2*, *pitch* y *yaw* a la red como entradas de forma que realice las predicciones de *pitch* y *yaw*. Finalmente, dichos valores se emplean en el cálculo de dx/dt.

6.5.2 Con predicciones de 4 variables de estado

El siguiente paso ha sido emplear la red neuronal artificial para predecir tanto los ángulos como las velocidades de los mismos (mediante el cálculo de las mismas usando los ángulos), para después calcular *dx/dt.*

En este caso, la RNA deberá ser lo suficientemente precisa como para calcular a través de las predicciones de *pitch* y *yaw*, las velocidades de cada uno de los ángulos descritos.

6.6 iMO-NMPC

Ante la imposibilidad de implementar un controlador NMPC basado en un modelo puramente neuronal, al usar los bloques y funciones desarrollados por MathWorks, se ha decidido modificar el rumbo del TFM. Entonces, el nuevo objetivo ha

sido desarrollar el controlador NMPC sin usar la herramienta de Simulink, únicamente empleando *scripts*.

Por tanto, se ha continuado la estrategia de control *iMO-NMPC (Intelligent Multi-Objective Nonlinear Model Predictive Control)* desarrollado dentro del *GICI* [1], [19]. Dicha estrategia emplea un controlador NMPC convencional, sin embargo, el optimizador en este caso es un algoritmo genético multiobjetivo (MOGA). Dicho MOGA realiza un problema de optimización multiobjetivo (se enfrentan la minimización del error de seguimiento de la referencia y minimización de la variación de la señal de control), generando el Frente de Pareto, donde se visualizan las soluciones no dominadas.

Por último, mediante el *Decision Maker*, se selecciona una de las soluciones empleando diferentes métodos de decisión (mínima distancia al origen, mínimo error de seguimiento, etc.).

El objetivo final, al igual que el apartado anterior, es modificar la estrategia de control de modo que se obtenga una estrategia avanzada que utilice una RNA como modelo de predicción de la planta y además esta trabaje con sistemas multivariables, ya que la estrategia desarrollada anteriormente trabajaba con sistemas monovariables [25].

Cabe destacar que en principio el método de decisión empleado ha sido el de mínima distancia al origen y que el tamaño del cromosoma depende del horizonte de control *Hu*.

6.6.1 SISO matemático

En primer lugar, se decidió conocer cómo trabaja el controlador previamente diseñado [25], conociendo sus diferentes elementos y los parámetros configurables del mismo. Para ello, se dispuso de un sistema no lineal monovariable, siendo el modelo de predicción empleado un modelo matemático. Por tanto, la planta viene definida mediante la siguiente ecuación:

$$y(k) = \frac{1.5 \cdot y(k-1) \cdot y(k-2)}{1 + y(k-1)^2 + y(k-2)^2} + 0.7 \cdot \sin\left(0.5 \cdot \big(y(k-1) + y(k-2)\big)\right) \cdot \cos\left(0.5 \cdot \big(y(k-1) + y(k-2)\big)\right) + 1.2 \cdot u(k-1) \qquad (35)$$

La evaluación de los objetivos, por un lado, la minimización del error de seguimiento (Objetivo 1) y por otro la minimización de la variación de la señal de control (Objetivo 2), es la siguiente:

$$Objetivo\ 1\ \rightarrow\ J_1 = \sum_{i=1}^{H} \left(Ref_y(k) - y(k)\right)^2 \tag{36}$$

$$Objetivo\ 2\ \rightarrow J_2 = \sum_{i=2}^{H} \left(u(k) - u(k-1)\right)^2 \tag{37}$$

6.6.2 SISO RNA

El siguiente paso antes de probar la estrategia en un sistema multivariable, ha sido reemplazar el modelo de predicción matemático por un modelo neuronal. Para ello, se ha entrenado una red neuronal artificial para reproducir el comportamiento de la planta no lineal mostrada en el apartado anterior.

Para ello, se ha empleado una RNA con la siguiente estructura, 5 neuronas en la capa oculta y con un retardo en la salida:

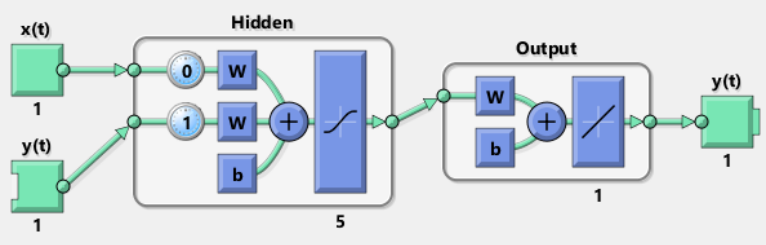

Figura 18. Estructura RNA para iMO-NMPC SISO.

6.6.3 MIMO matemático sin acoplamiento

Tras comprobar el correcto funcionamiento de la estrategia iMO-NMPC en un sistema monovariable, el siguiente paso ha sido comprobar su funcionamiento ante un sistema multivariable, de dos entradas y dos salidas.

Sin embargo, antes de pasar al sistema TR, el cual es un sistema MIMO fuertemente acoplado, se ha probado en un sistema MIMO desacoplado, de dos ecuaciones no lineales como las del sistema SISO.

Para poder aplicar el optimizador al sistema MIMO, se ha modificado el cromosoma que emplea el AG. A partir de ahora, la mitad del cromosoma corresponden a las entradas calculadas para la primera salida; mientras que la otra mitad corresponden a las entradas de la otra salida. Por tanto, el cromosoma ahora es el doble de grande, al haber dos entradas. Por ejemplo, si se dispone de un horizonte de control de 4, el cromosoma es el siguiente:

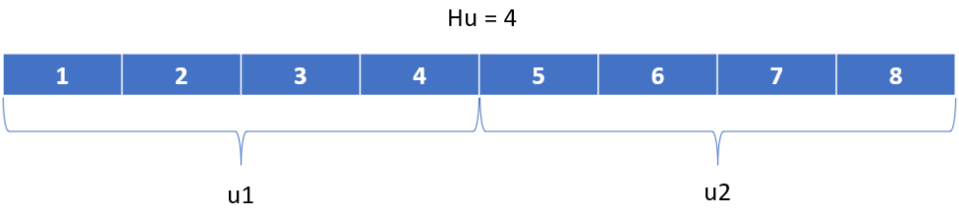

Figura 19. División del cromosoma para Hu = 4.

Por otro lado, se ha modificado la función que evalúa los objetivos de cada uno de los individuos (Anexo I). Ahora se tienen en cuenta los errores de seguimiento de las dos referencias en conjunto y las variaciones de las señales de control en conjunto:

$$J_1 = \sum_{i=1}^{H} \left(Ref_{\psi}(k) - \psi(k)\right)^2 + \left(Ref_{\varphi}(k) - \varphi(k)\right)^2 \tag{38}$$

$$J_2 = \sum_{i=2}^{H} \left(u_1(k) - u_1(k-1)\right)^2 + \left(u_2(k) - u_2(k-1)\right)^2 \tag{39}$$

En este caso, las salidas del sistema son las siguientes:

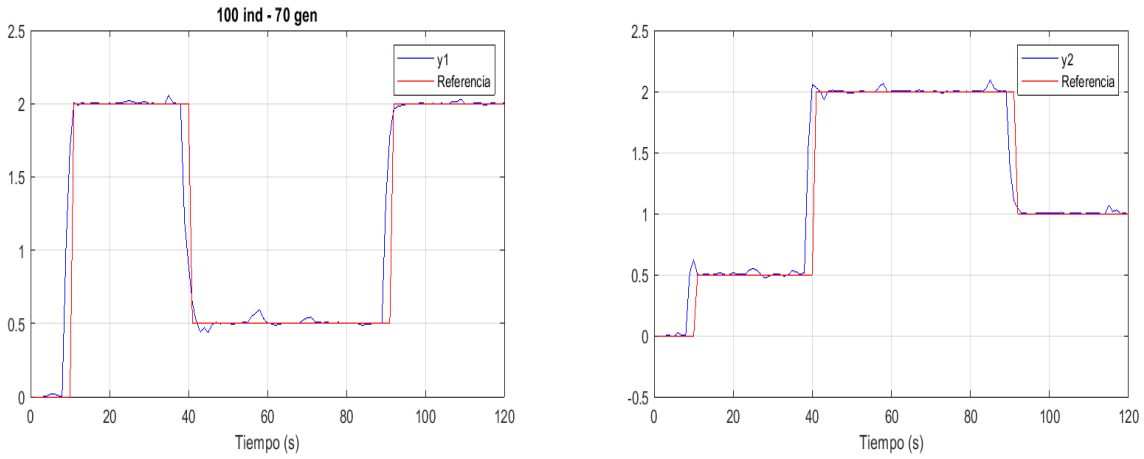

Figura 20. Salidas iMO-NMPC MIMO Matemático sin acoplamiento 100 individuos, 70 generaciones.

Observando la anterior figura, se puede comprobar que el seguimiento de cada una de las referencias se realiza de forma correcta, por lo tanto, la división del cromosoma en dos partes ha sido satisfactoria.

6.6.4 MIMO matemático Twin-Rotor

El siguiente paso, ha sido realizar el control iMO-NMPC en la plataforma Twin Rotor, para ello, se han empleado las ecuaciones matemáticas que describen la dinámica del mismo, tanto para definir el modelo de predicción de la estrategia de control como para definir la planta real.

En trabajos anteriores, al trabajar con el bloque NMPC en Simulink, este ya realizaba la discretización del modelo automáticamente. En este caso, sin embargo, al trabajar con *scripts*, es necesario realizar la discretización manualmente, empleando una función.

Por lo tanto, se ha observado en los ejemplos que proporciona Matlab cómo discretiza las plantas en espacio de estados [26]. En dichos ejemplos, Matlab emplea la discretización multipaso hacia adelante de Euler [31], para ello, emplea la siguiente función matemática:

$$x(k+1) = x(k) + \frac{Ts}{N} \cdot x(k) \tag{40}$$

donde *Ts* es el periodo de discretización, *x(k)* son las variables de estado y *N* es 21el número de pasos de integración.

6.6.5 MIMO RNA

Como último paso en la realización del proyecto, se ha incorporado la RNA que mejores resultados ha ofrecido a la hora de aproximar el comportamiento del Twin-Rotor (Figura 22) como modelo de predicción de la estrategia iMO-NMPC.

7. ANÁLISIS DE RESULTADOS

7.1 Estructuras de RNA

Como se ha comentado en el capítulo anterior, se probaron diferentes estructuras de RNA con el fin de diseñar una lo suficientemente precisa en la predicción de los ángulos, para así disponer de la mayor precisión posible a la hora de calcular las velocidades y las aceleraciones a partir de las mismas.

En primer lugar, se ha decidido emplear una red lo más simple posible: 10 neuronas en la capa oculta, sin retardos en la entrada y con un retardo en la salida (es obligatorio emplear al menos un retardo de la salida). De este modo se ha comprobado la eficacia de la red, realizando las predicciones de *pitch* y de *yaw* y para, además de conocer cómo se configura la misma.

En la siguiente figura, se muestra la validación de la misma:

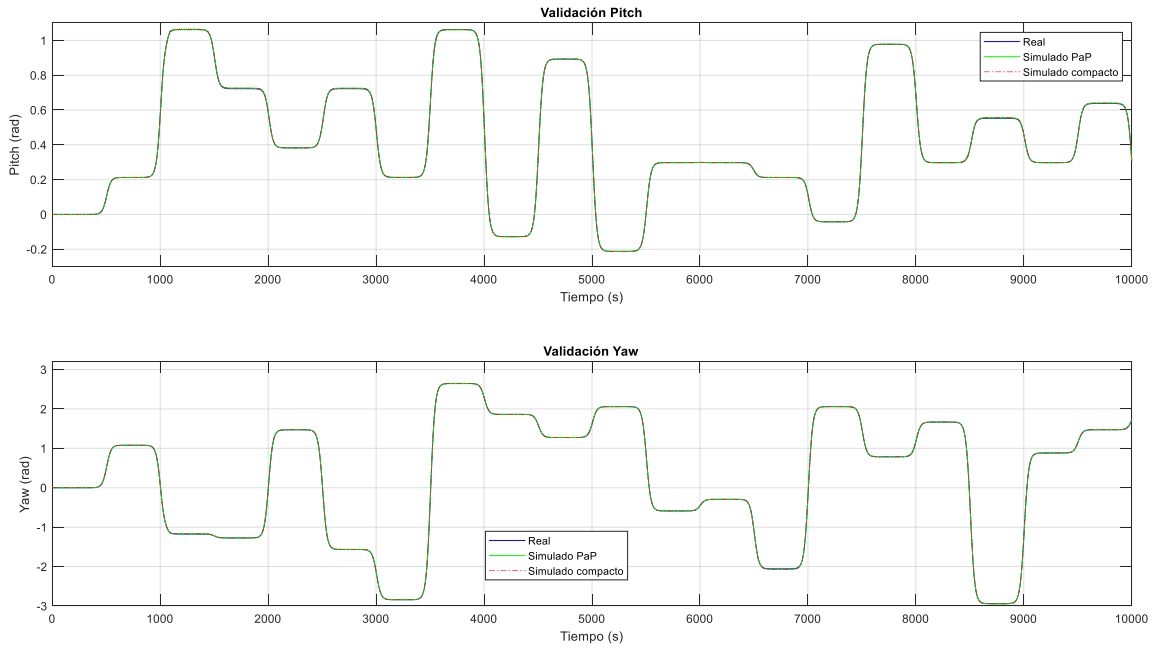

Figura 21. Validación RNA de 10 neuronas en la capa oculta, sin retardos en la entrada y con un retardo en la salida.

Como se puede observar, la red, aun siendo simple, es capaz de predecir las salidas de forma correcta, tanto mediante el método compacto usando todas las muestras, como mediante el método *Paso a Paso*.

Sin embargo, este proceso se ha realizado con redes de diferentes arquitecturas, variando el número de neuronas en la capa oculta y el número de entradas a la red, para así obtener una red cuyas aceleraciones y velocidades predichas sean próximas a las reales, hasta encontrar la red que mejores resultados ha ofrecido. A continuación, se muestran las diferentes configuraciones probadas:

Tabla 2. Distintas configuraciones de RNA probadas.

Nº	Neuronas Capa Oculta	Retardos Entrada	Retardos Salida	MSE 1	MSE 2	MSE 3	MSE 4	MSE 5	MSE Medio
1	20	0	1	3.7027e-6	3.6938e-6	3.7547e-6	3.8600e-6	3.5017e-6	3.7026e-6
2	20	0:2	1:2	3.0835e-9	2.5469e-9	2.3335e-9	3.2383e-9	3.2699e-9	2,8944e-9
3	20	0	1:2	2.4869e-8	2.3271e-8	2.6169e-8	2.7861e-8	2.4506e-8	2,5335e-8
4	30	0	1:2	2.1141e-8	2.1250e-8	2.5131e-8	1.9771e-8	2.2700e-8	2,1997e-8
5	30	0:1	1	1.2658e-6	1.1799e-6	1.0008e-6	1.1818e-6	1.0368e-6	1.1330e-6

De las redes anteriormente mostradas, las redes que mejores resultados han ofrecido han sido las redes 2 y 4, como muestran sus valores de error cuadrático medio (MSE). Sin embargo, se ha decidido emplear la red 4. De este modo, no es necesario almacenar valores de las entradas anteriores, únicamente de las salidas, reduciendo la complejidad de la escritura del código.

Dicha red presenta la siguiente estructura:

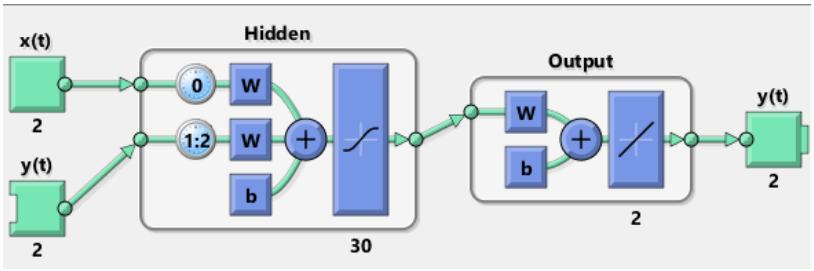

Figura 22. Estructura RNA con mejores resultados.

La red seleccionada presenta 30 neuronas en la capa oculta, no dispone de entradas con retardo, pero sí dispone de salidas realimentadas con retardo, de los dos instantes anteriores. Por tanto, por un lado, recibe u*1(k)* y *u2(k)* como entradas y por otro recibe *pitch(k-1), yaw(k-1)*, *pitch(k-2)* y *yaw(k-2)*.

A continuación, se muestran los resultados obtenidos:

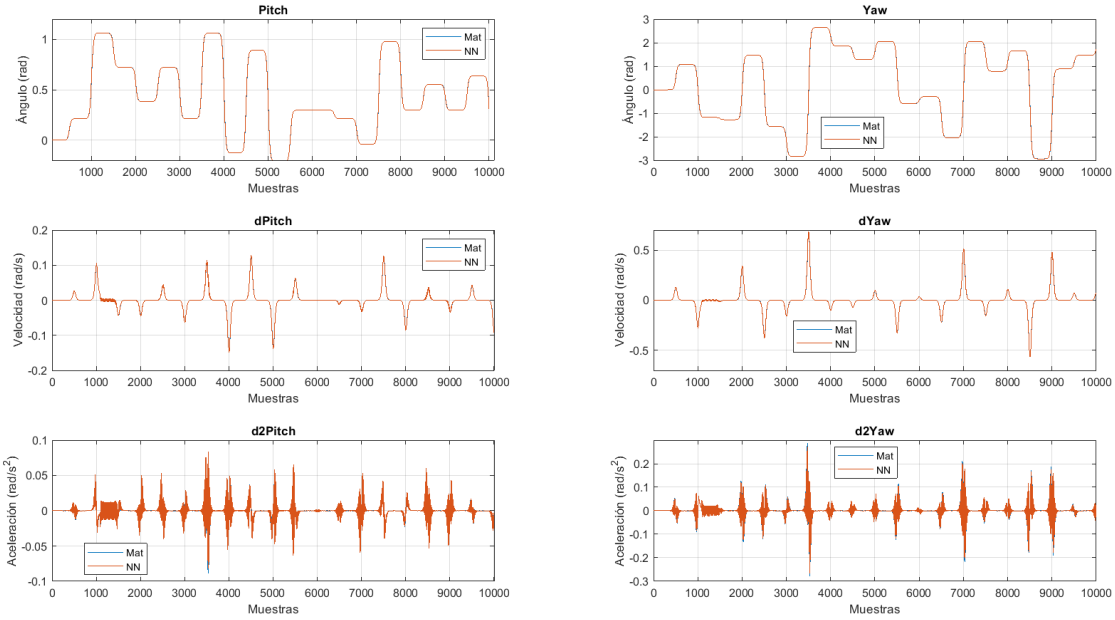

Figura 23. Comparación modelo matemático y RNA, velocidades y aceleraciones.

Como se puede observar, la RNA posee una precisión tal que se pueden deducir las aceleraciones y las velocidades con suficiente precisión a través de las predicciones de los ángulos. Para poder observar mejor la precisión de los cálculos, se presenta un *zoom* sobre las gráficas de las aceleraciones.

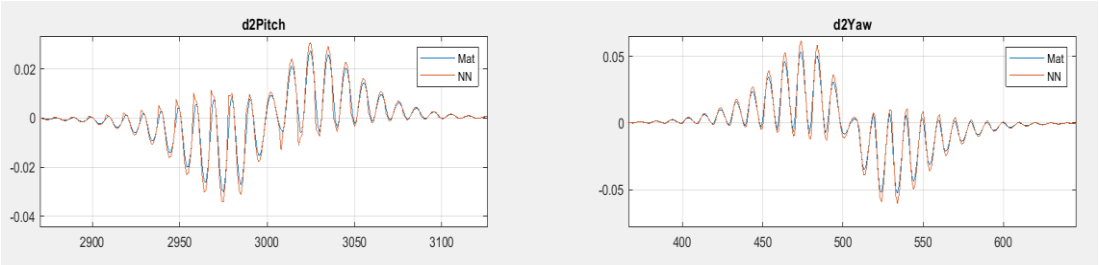

Figura 24. Zoom gráfica aceleraciones.

Por otro lado, los resultados obtenidos a través del esquema de Simulink, son los siguientes:

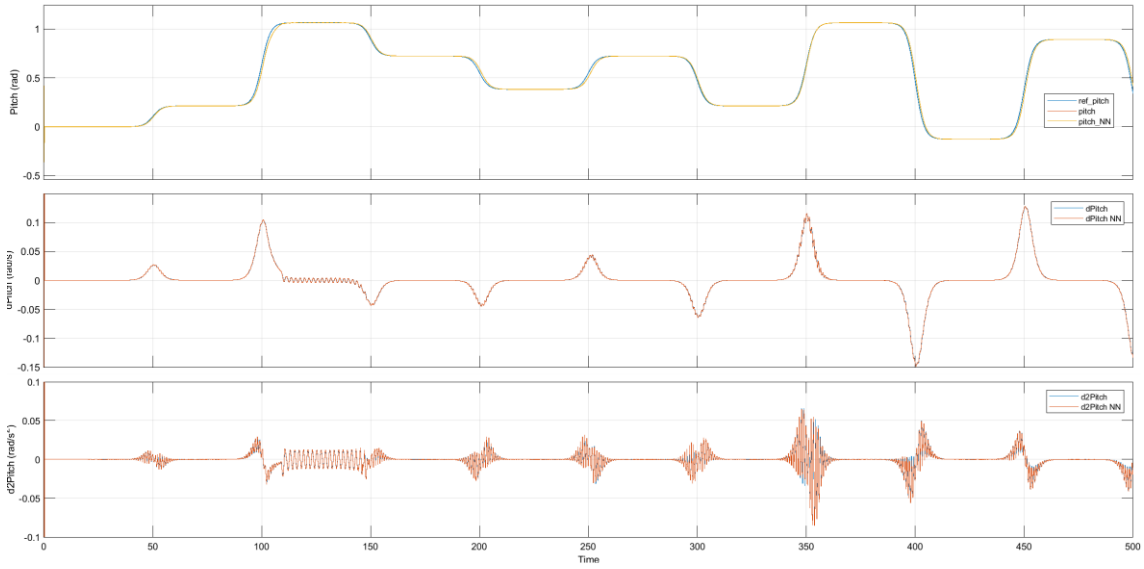

Figura 25. Pitch y sus derivadas comparación NMPC matemático y RNA.

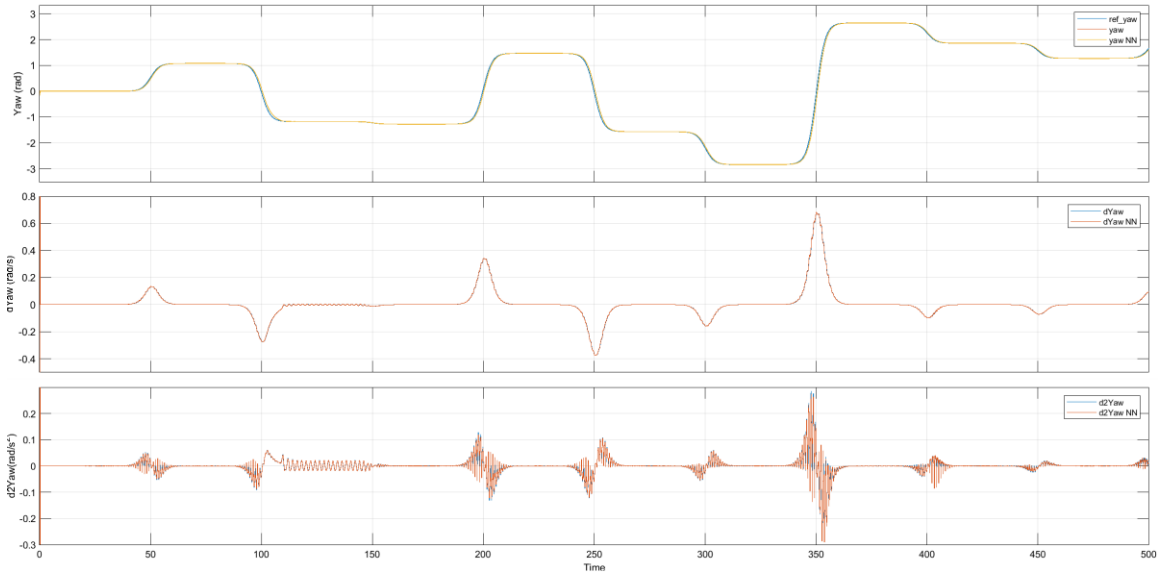

Figura 26. Yaw y sus derivadas, comparación NMPC matemático y RNA.

Como se observa, el resultado obtenido a través de ambos métodos es idéntico, por lo tanto, la RNA puede actuar como modelo de predicción de forma correcta, al predecir tanto los ángulos *pitch* y *yaw* como sus respectivas derivadas con la suficiente exactitud.

7.2 Modelo híbrido

En este apartado, se muestran las salidas empleando el método híbrido anteriormente descrito, combinando la RNA entrada con el modelo matemático.

En primer lugar, se muestran las salidas empleando el modelo híbrido que predice únicamente los ángulos.

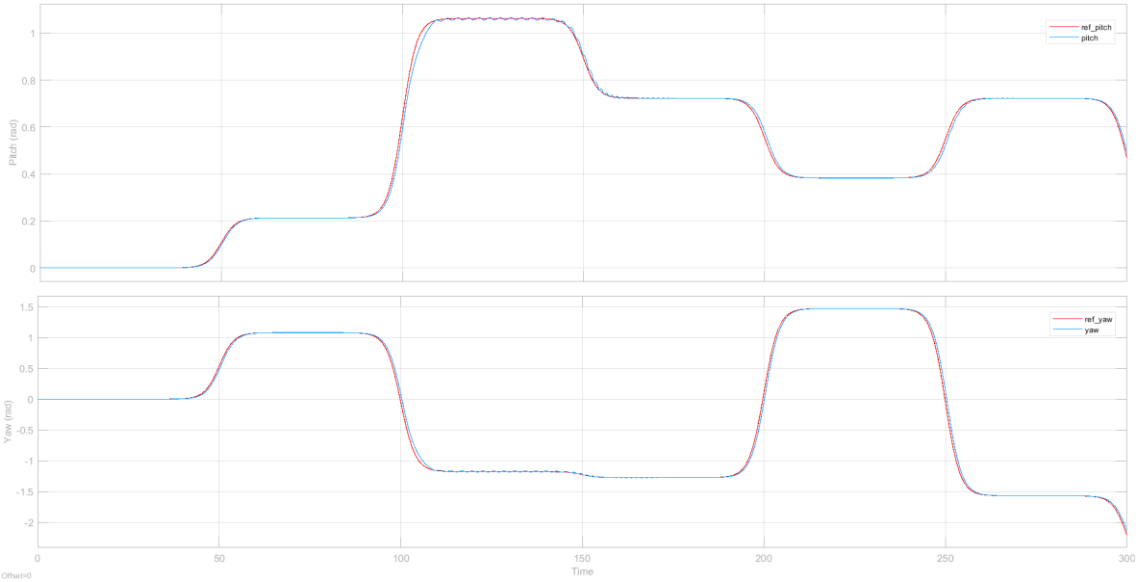

Figura 27. Pitch y yaw modelo híbrido, con predicciones de ángulos.

El error que presenta es algo mayor que en el caso de emplear el modelo puramente matemático, sin embargo, sigue a la referencia correctamente.

Por otro lado, las salidas con el modelo que emplea las predicciones tanto de los ángulos como de las velocidades son las siguientes:

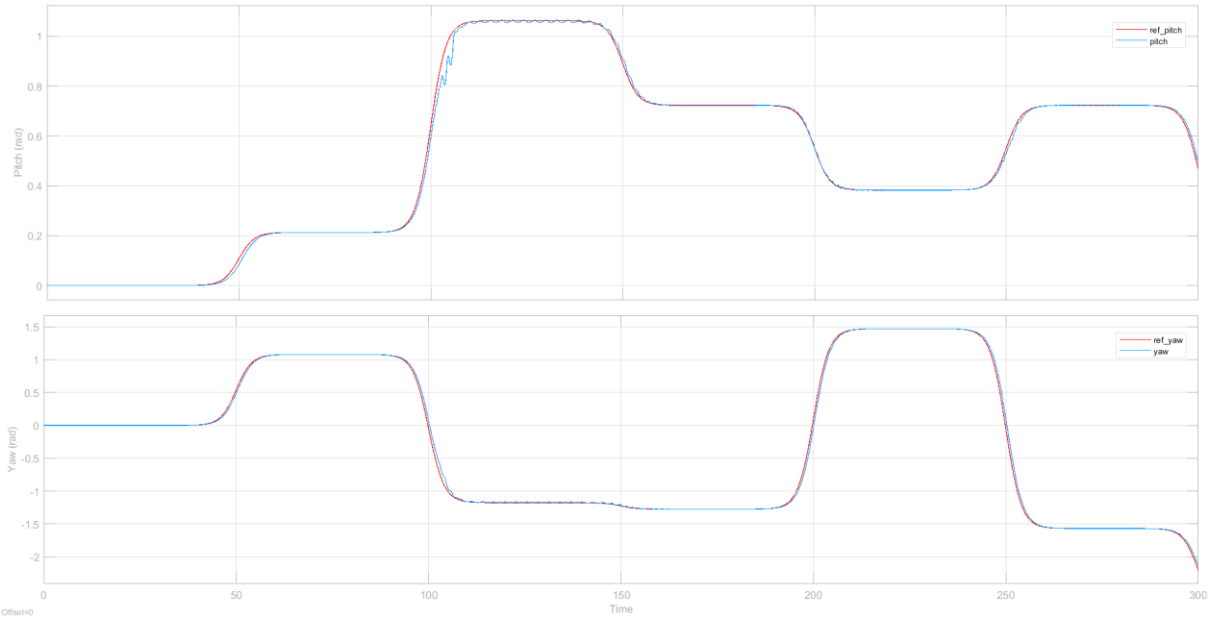

Figura 28. Pitch y yaw modelo híbrido, con predicciones de ángulos y velocidades.

Como se observa (Figura 28), el modelo híbrido trabaja de forma correcta, únicamente se producen errores grandes cuando se produce un gran cambio de referencia en el yaw, al igual que ocurría en anteriores simulaciones. Además, el error es algo mayor que en el de la Figura 27.

Cabe destacar que cada vez que se llama al *script* del modelo, este recibe como parámetros de entrada las mediciones de *u(k)* e *y(k)*, por tanto, al ejecutar la red, esta da como salidas las predicciones de *y(k+1)*, por tanto, los cálculos de las variaciones de las variables de estado no se están realizando con los valores temporales exactos. Sin embargo, se observa el potencial y la robustez de las redes neuronales artificiales, siendo estas capaces de trabajar con valores desfasados, dando valores muy cercanos a los valores reales.

7.3 iMO-NMPC

En este apartado, se muestran los resultados obtenidos en el desarrollo de la estrategia iMO-NMPC. Para ello, se presentan las gráficas de las diferentes configuraciones probadas.

7.3.1 SISO matemático

Tras varias simulaciones, modificando el número de generaciones y de individuos, el resultado obtenido es el siguiente:

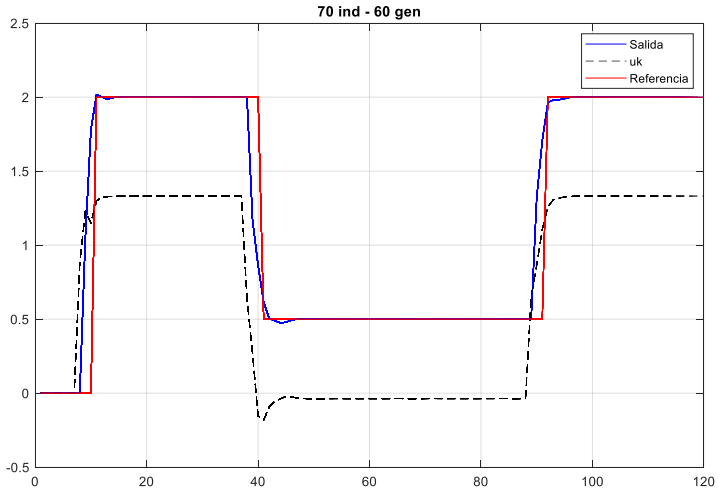

Figura 29. Salida iMO-NMPC SISO Matemático, 70 individuos 60 generaciones.

Como se observa, el sistema es capaz de seguir a la referencia de forma correcta, al tratarse de un sistema sencillo; además de ser capaz de adelantarse a los cambios de referencia debido a la estrategia NMPC.

Por otro lado, cabe mencionar que, al tratarse de un sistema SISO, el aumentar el número de generaciones o de individuos no produce cambios significativos, por lo que es mejor emplear la menor cantidad posible de ambos; de esta forma, se reduce considerablemente el gasto computacional.

7.3.2 SISO RNA

Tras incorporar la RNA como modelo de predicción del NMPC, los resultados obtenidos han sido los siguientes:

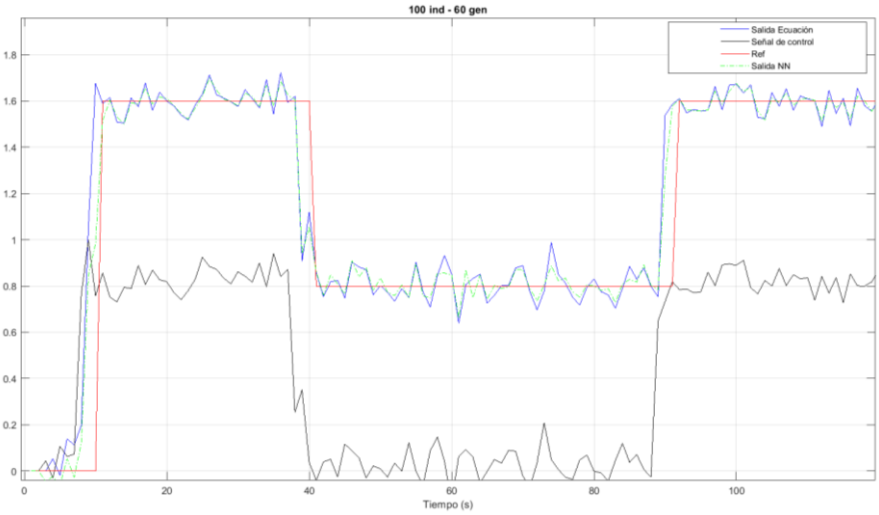

Figura 30. Salida iMO-NMPC SISO RNA 100 individuos, 60 generaciones.

Como se observa, el error en estado estacionario es mayor comparado con el modelo puramente matemático, sin embargo, la salida de la planta real y la salida predicha por la RNA es similar.

Por tanto, para reducir el error cometido, se han aumentado el número de individuos y el número de generaciones, de este modo, al disponer de más cantidad, se promueve la obtención de mejores individuos, promoviendo así, la localización de mínimos globales, evitando el estancar la búsqueda en un mínimo local.

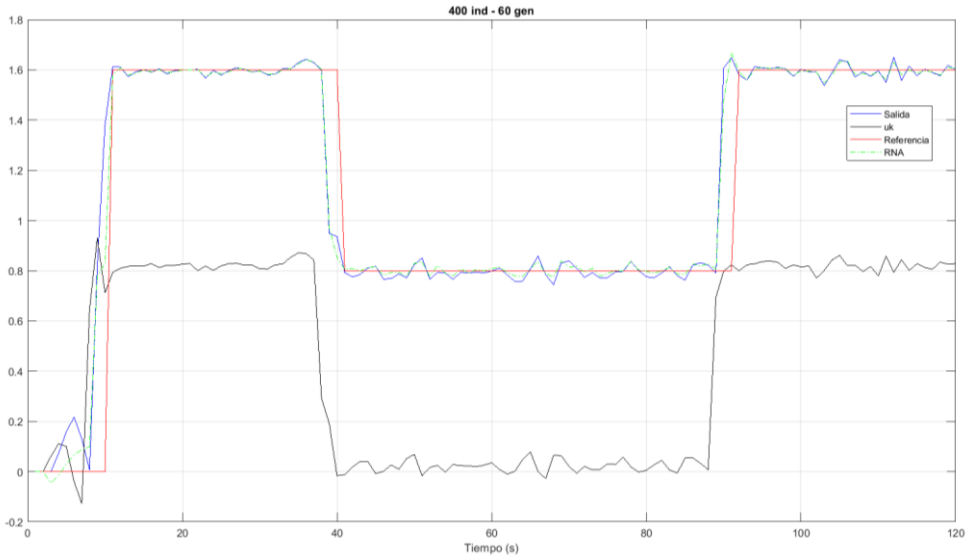

Figura 31. Salida iMO-NMPC SISO RNA 400 individuos, 60 generaciones.

Al aumentar la cantidad de individuos, se han encontrado mejores candidatos, por lo que el error ha disminuido como muestra la anterior gráfica.

7.3.3 MIMO matemático

Aplicando la estrategia de control iMO-NMPC ante el sistema Twin-Rotor, tras haber probado el comportamiento de la estrategia para un sistema multivariable desacoplado usando el modelo de predicción matemático, las salidas han sido las mostradas a continuación:

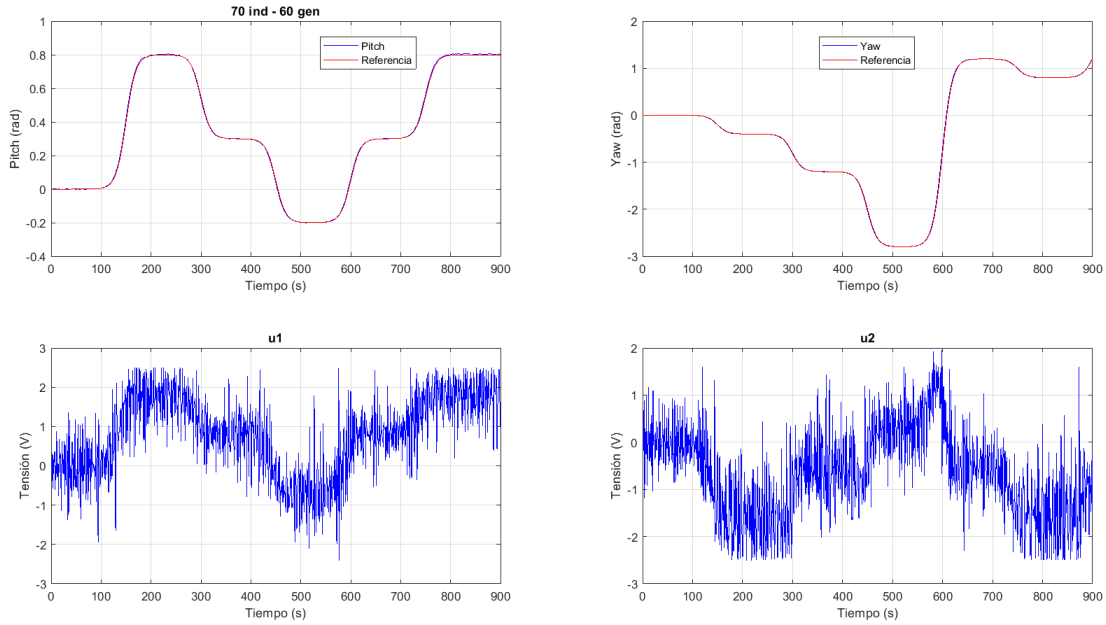

Figura 32. Salidas iMO-NMPC MIMO Matemático 70 individuos, 60 generaciones.

Ambas salidas siguen la referencia de forma correcta, por lo tanto, el control es satisfactorio. En este caso, el método de decisión empleado es el de mínimo error de seguimiento, es por ello que se observa como la acción de control varía mucho, para evitar que el error sea elevado.

Sin embargo, se podría emplear otro método de decisión, por ejemplo, penalizando más la variación de la acción de control que el error de seguimiento de la referencia, de modo que se reduzca la amplitud de la variación de la acción de control. Teniendo así un control algo más suave.

7.3.4 MIMO RNA

Una vez incorporada la red NARX como modelo de predicción del controlador iMO-NMPC, se han probado diferentes estrategias para tratar de lograr un correcto control de la planta mediante el uso de RNA. Sin embargo, no se ha logrado tal objetivo, debido a la complejidad que presenta la planta a controlar.

Las estrategias que se han probado (sin éxito) han sido las siguientes:

- **Suavizado de las referencias.** Se han suavizado aún más las referencias, de modo que el cambio de referencia sea lo menos brusco posible.

- **Seguimiento de una sola referencia.** Se ha probado a dejar una referencia a 0, de modo que se realizaría el seguimiento de la otra referencia.
- **Entrada tipo escalón.** Se ha probado a usar una entrada escalón, sin embargo, el sistema se sale de rango.

A pesar de que las estrategias probadas, explicadas anteriormente, no han sido un éxito, se ha comprobado si la RNA es capaz de predecir las salidas de forma correcta. Para ello, se ha vuelto a usar el modelo matemático como modelo de predicción del controlador; mientras, la RNA, únicamente realizará la predicción de las salidas empleando las salidas reales.

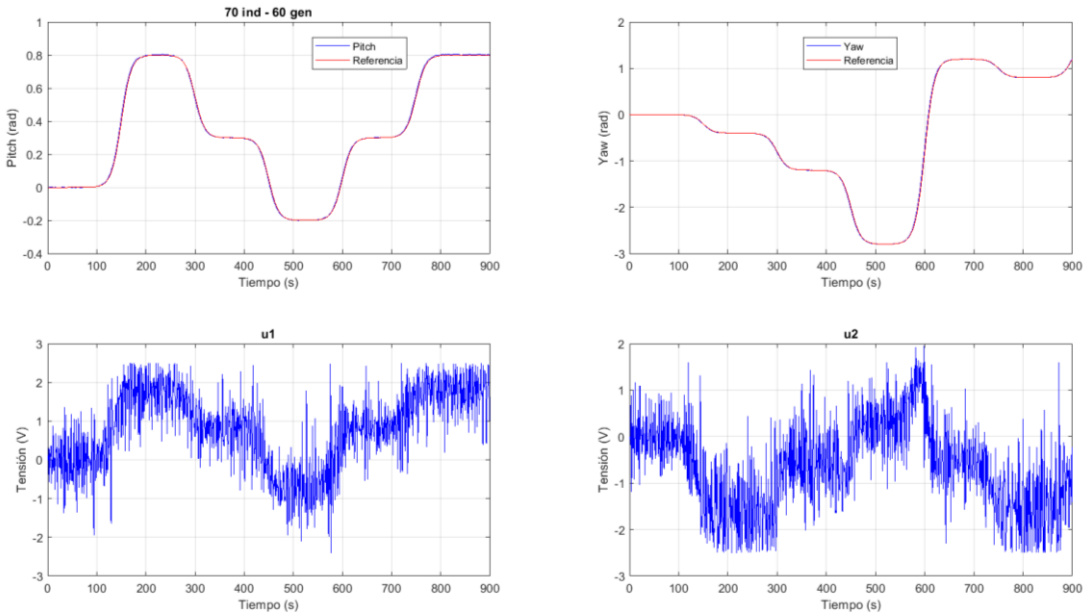

Figura 33. Salidas iMO-NMPC sin usar la RNA como modelo de predicción.

Se observa que la RNA aproxima las salidas de forma correcta, por lo que, en futuros trabajos, se tratará de modificar el código de forma que esta pueda actuar como modelo de predicción del controlador predictivo.

8. PLANIFICACIÓN

Para la finalización de dicho proyecto y la consecución de los objetivos marcados al inicio del mismo, este se ha dividido en varias fases, sin embargo, cabe destacar que algunas fases han sido llevadas a cabo simultáneamente:

- **Fase I. Búsqueda y análisis de información.** Búsqueda de referencias bibliográficas relacionadas con la temática del trabajo, tanto desarrolladas dentro del GICI, como fuera del mismo, para el desarrollo del estado del arte.
- **Fase II. Desarrollo del trabajo.** En esta fase, se concentran tanto el entrenamiento de RNA como el diseño de estrategias avanzadas para el control de la maqueta Twin-Rotor (NMPC, NMPC Neuronal, NMPC Híbrido, iMO-NMPC).
- **Fase III. Análisis de resultados y conclusiones.** A partir de los resultados obtenidos, se han desarrollado las conclusiones y se han comprobado los objetivos alcanzados.
- **Fase IV. Redacción memoria final.** Tras la finalización del proyecto, se ha desarrollado esta memoria, correspondiente al trabajo desarrollado.
- **Fase V. Presentación.** Por último, se ha desarrollado la presentación correspondiente a la defensa del TFM.

A continuación, se muestra el diagrama de Gantt correspondiente al alcance de cada una de las fases del trabajo, dividido por semanas. El proyecto se comenzó la semana del 26 de abril de 2021 y se concluyó la semana del 6 de septiembre de 2021, durando alrededor de 19 semanas.

Tabla 3. Diagrama de Gantt

Semana	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
Fase I																			
Fase II																			
Fase III																			
Fase IV																			
Fase V																			

9. CONCLUSIONES Y TRABAJOS FUTUROS

9.1 Conclusiones

Tras la finalización del proyecto, en lo que respecta a la estrategia iMO-NMPC, se ha logrado dar el salto de un sistema monovariable (SISO) a un sistema multivariable (MIMO), como se había marcado antes del inicio del mismo. Dicha estrategia ha presentado grandes resultados, por tanto, la división del cromosoma en dos ha resultado satisfactoria (Figura 19).

Por otro lado, cabe mencionar la dificultad de controlar un sistema multivariable como el presente empleando redes neuronales artificiales como modelo de predicción, al no haber sido posible de implementar.

En lo que respecta a la aproximación del Twin-Rotor a través de las redes neuronales artificiales, cabe destacar su robustez, dado que, en ocasiones, a pesar de trabajar con variables que no están correctamente alineadas en el tiempo, aproximan las salidas con una gran exactitud. Además, el cálculo de las derivadas de los ángulos *pitch* y *yaw* a través de dichas predicciones, resultan ser valores muy próximos a los valores reales (Figura 23).

En conclusión, con la realización del presente proyecto, se han realizado grandes avances dentro del Grupo de Investigación en Control Inteligente (GICI), los cuales permitirán desarrollar nuevos trabajos y líneas de investigación dentro del mismo, partiendo del presente trabajo, en el siguiente apartado, se presentan los posibles trabajos a futuro.

9.2 Trabajos futuros

En primer lugar, se podría continuar con el desarrollo de la técnica iMO-NMPC, con el fin de tratar de incorporar una RNA como modelo de predicción de la misma, al no haberse conseguido en el presente trabajo. De este modo, se podría contrastar dicha estrategia en otros sistemas no lineales.

Por otro lado, una vez desarrollada y contrastada dicha técnica dentro del entorno de simulación de Matlab/Simulink, esta podría ser contrastada en la maqueta

real del laboratorio. De esta forma, se comprobaría la robustez de la misma ante las no linealidades y el deterioro que pueda presentar el Twin-Rotor real. Para ello, se debería entrenar una nueva RNA que aproxime la dinámica de la planta real.

Además, se podrían probar diferentes configuraciones de RNA o diferentes divisiones del cromosoma, con el fin de realizar un estudio sobre el tiempo de ejecución necesario en cada caso.

En lo que respecta al uso del bloque NMPC de Simulink, se podría seguir investigando sobre cómo incorporar la RNA como modelo del bloque, por ejemplo, modificando el código desarrollado por *MathWorks®*, con el fin de controlar el Twin-Rotor en el entorno de Simulink.

REFERENCIAS

[1] J. J. Valera, V. Gómez-Garay, E. Irigoyen, F. Artaza and M. Larrea, "Intelligent Multi-Objective Nonlinear Model Predictive Control (iMO-NMPC): Towards the 'on-line' optimization of highly complex control problems," Expert Syst. Appl., vol. 39, no. 7, pp. 6527–6540, 2012.

[2] A. Tastemirov, A. Lecchini-Visintini and R. M. Morales-Viviescas, "Complete dynamic model of the Twin Rotor MIMO System (TRMS) with experimental validation," Control Eng. Pract., vol. 66, no. July, pp. 89–98, 2017.

[3] M. Ilyas, N. Abbas, M. Ubaidullah, W. A. Imtiaz, M. A. Q. Shah and K. Mahmood, "Control Law Design for Twin Rotor MIMO System with Nonlinear Control Strategy," Discret. Dyn. Nat. Soc., vol. 2016, pp. 13–22, 2016.

[4] K. Manu, E. Agarwal and M. Vashisht, "Discrete-Time Chebyshev Neural Observer for Twin Rotor MIMO System", Int. J. Eng. Res., vol. V4, no. 08, pp. 303–307, 2015.

[5] N. M. H. Norsahperi and K. A. Danapalasingam, "Particle swarm-based and neuro-based FOPID controllers for a Twin Rotor System with improved tracking performance and energy reduction," ISA Trans., vol. 102, pp. 230–244, 2020.

[6] E. Irigoyen, M. Larrea, J. Valera, V. Gomez, and F. Artaza, "A hybridized neuro-genetic solution for controlling industrial R 3 workspace," Neural Netw. World, vol. 20, no. 7, pp. 811–824, 2010.

[7] C. H. Lu and C. C. Tsai, "MIMO Neural-Network Predictive Controller Design," IECON Proc. (Industrial Electron. Conference)., vol. 2, pp. 1733–1738, 2004.

[8] S. Abderrahmene, C. Mohammed, K. Abderrahmane and H. Rachida, "Neural network NARMA-L2 control of a Twin Rotor MIMO System," 2019 Int. Conf. Adv. Electr. Eng. ICAEE 2019, 2019.

[9] K. Viana, M. Larrea, E. Irigoyen, M. Diez and A. Zubizarreta, "MIMO Neural Models for a Twin-Rotor Platform: Comparison Between Mathematical Simulations and Real Experiments", vol. 1268 AISC. Springer International Publishing, 2021.

[10] E. Larzabal, J. Cubillos, M. Larrea, E. Irigoyen and J. Valera, "Soft computing testing in real industrial platforms for process intelligent control", Advances in Intelligent Systems and Computing, vol. 188 (2013), pp. 221-230.

[11] A. Ulasyar and H. S. Zad, "Robust & optimal model predictive controller design for twin rotor MIMO system," ELECO 2015 - 9th Int. Conf. Electr. Electron. Eng., pp. 854–858, 2016.

[12] A. Rahideh, M. H. Shaheed and H. J. C. Huijberts, "Dynamic modelling of a TRMS using analytical and empirical approaches," Control Eng. Pract., vol. 16, no. 3, pp. 241–259, 2008.

[13] R. Raghavan and S. Thomas, "Practically Implementable Model Predictive Controller for a Twin Rotor Multi-Input Multi-Output System," J. Control. Autom. Electr. Syst., vol. 28, no. 3, pp. 358–370, 2017.

[14] A. A. Aldair, "Hardware Implementation of the Neural Network Predictive Controller for Coupled Tank System," Am. J. Electr. Electron. Eng., vol. 2, no. 2, pp. 40–47, 2014.

[15] M. Dendaluce, J.J. Valera, V. Gómez-Garay, E. Irigoyen and E. Larzabal, "Microcontroller implementation of a multi objective genetic algorithm for real-time intelligent control", International Joint Conference SOCO13-CISIS13-ICEUTE13, pp. 71–80. Springer, 2014.

[16] M. Larrea, E. Larzabal, E. Irigoyen, J. J. Valera, and M. Dendaluce, "Implementation and testing of a soft computing-based model predictive control on an industrial controller," J. Appl. Log., vol. 13, no. 2, pp. 114–125, 2015.

[17] R. G. Tiwalkar, S. S. Vanamane, S. S. Karvekar and S. B. Velhal, "Model predictive controller for position control of twin rotor MIMO system," IEEE Int. Conf. Power, Control. Signals Instrum. Eng. ICPCSI 2017, pp. 952–957, 2018.

[18] J. Carrillo-Ahumada, G. Reynoso-Meza, S. García-Nieto, J. Sanchis and M. A. García-Alvarado, "Sintonización de controladores Pareto-óptimo robustos para sistemas multivariables. Aplicación en un helicóptero de 2 grados de libertad," RIAI - Rev. Iberoam. Autom. e Inform. Ind., vol. 12, no. 2, pp. 177–188, 2015.

[19] A. Gutierrez, E. Irigoyen, E. Larzabal, J.J. Valera and M. Larrea, "PRIMEROS RESULTADOS DE UN CONTROL GENÉTICO PREDICTIVO SOBRE MAQUETA DE HELICOPTERO (TWINROTOR)", Actas de las XXXV Jornadas de Automática, 2014.

[20] P. Chalupa, J. Přikryl and J. Novák, "Modelling of twin rotor MIMO system," Procedia Eng., vol. 100, no. January, pp. 249–258, 2015.

[21] Q. V Nguyen et al., "Multiple Sliding Surface Control Approach to Twin Rotor MIMO Systems", International Journal of Fuzzy Logic and Intelligent Systems, vol. 14, no. 3 September 2014, pp. 171-180, 2014.

[22] "Twin Rotor MIMO System Control Experiments 33-949S", de Feedback Instruments. Recuperado 20 de mayo de 2021, website: http://www.cpdee.ufmg.br/~palhares/33-942rotor.pdf

[23] Nonlinear MPC, de MathWorks®. Recuperado 14 de mayo de 2021, website: https://es.mathworks.com/help/mpc/ug/nonlinear-mpc.html

[24] Nonlinear autoregressive neural network with external input, de MathWorks®. Recuperado 18 de junio de 2021, website: https://es.mathworks.com/help/deeplearning/ref/narxnet.html?searchHighlight=narxnet&s_tid=srchtitle

[25] Repositorio GitHub "NMPC Twin-Rotor", de Grupo de Investigación de Control Inteligente UPV/EHU. Recuperado 15 de julio de 2021, website: https://github.com/GICI-UPV-EHU

[26] Nonlinear MPC Design - Examples, de MathWorks®. Recuperado 24 de junio de 2021, website: https://es.mathworks.com/help/mpc/examples.html?category=nonlinear-mpc-design&s_tid=CRUX_topnav

[27] Rivals, I. and Personnaz, L., "Black-box modeling with state-space neural networks". In: Zbikowski, R., Hint, K.J. (Eds.) Neural Adaptive Control Technology, World Scientific, pp. 237–264, 1996.

[28] A. Rahideh and M. H. Shaheed, "Dynamic modelling of a twin rotor MIMO system using grey box approach," 2008 5th International Symposium on Mechatronics and Its Applications*, pp. 1-6, 2008.*

[29] Christoph G., "Modelling, Simulation and Control of a Twin Rotor MIMO-System" [Tesis de Doctorado, Universidad Politécnica de Valencia]. Repositorio Universidad Politécnica de Valencia, website: https://riunet.upv.es/handle/10251/66346

[30] Petlenkov E. (2007), "Neural Networks Based Identification and Control of Nonlinear Systems: ANARX Model Based Approach", [Tesis de Doctorado, Tallin University of Technology, website: https://digikogu.taltech.ee/en/Download/f4a5b34f-51cb-4003-b4fd-b57657855780

[31] Zeltkevic M., (1998), "Forward and Backward Euler Methods", Recuperado 10 de julio de 2021., website: https://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node3.html

ANEXO I

1. Declaración objeto NMPC

```
nx = 6;
ny = 2;
nu = 2;

nlobj = nlmpc(nx,ny,nu);
nlobj.States(1).Name = 'pitch';
nlobj.States(2).Name = 'dpitch';
nlobj.States(3).Name = 'yaw';
nlobj.States(4).Name = 'dyaw';
nlobj.States(5).Name = 'tau1';
nlobj.States(6).Name = 'tau2';
nlobj.MV(1).Name = 'u1';
nlobj.MV(2).Name = 'u2';

% Modelo predictivo

nlobj.Model.StateFcn = @(x,u) modelo_TR(x,u);

nlobj.Model.OutputFcn = @(x,u) salida_TR(x,u);

nlobj.Model.IsContinuousTime =  true;
nlobj.Model.NumberOfParameters = 0;

%% Tuning MPC

R = [3 3];  % Ponderación referencia
Q = [0.1 0.1];  % Ponderación señal de control
h = 7; % Horizonte
hu = 7; % Horizonte de control

Ts = 0.1;

nlobj.Ts = Ts;
nlobj.PredictionHorizon = h;
nlobj.ControlHorizon = hu;
nlobj.Weights.OutputVariables = R;
nlobj.Weights.ManipulatedVariablesRate = Q;

%% NMPC Restricciones Señal de control

nlobj.MV(1).Min = -2.5;
nlobj.MV(1).Max = 2.5;
nlobj.MV(2).Min = -2.5;
nlobj.MV(2).Max = 2.5;
```

2. Función modelo de predicción matemático

```
function dxdt = modelo_TR(x,u)
% Estados

pitch = x(1);
dpitch = x(2);
yaw = x(3);
dyaw = x(4);
tau1 = x(5);
tau2 = x(6);

% Variables Manipuladas
u1 = u(1);
u2 = u(2);

% Ecuaciones No Lineales
dxdt = x(:);

% Son variaciones, en el siguiente instante x(k+1) = dxdt + x(k)
dxdt(1)= dpitch;
dxdt(2) = (0.0135*tau1*tau1 + 0.0924*tau1 - 0.32*sin(pitch) - 0.006*dpitch +
0.0063*sin(2*pitch)*dyaw*dyaw - 0.05*cos(pitch)*(0.0135*tau1*tau1 + 0.0924*tau1)*dyaw)/0.068;
dxdt(3)= dyaw;
dxdt(4) = (0.02*tau2*tau2 + 0.09*tau2 - 0.1*dyaw  + 0.35 *(0.0135*tau1*tau1 + 0.0924*tau1))/0.02;
dxdt(5) = -0.9091*tau1 + u1;
dxdt(6) = -tau2 + 0.8*u2;
```

3. Entrenamiento y validación RNA

```
load('Data/Input_Pitch');
load('Data/Input_Yaw');
load('Data/Target_Pitch');
load('Data/Target_Yaw');

inputData = [Input_Pitch, Input_Yaw]';   % Variables de entrada al sistema
targetData = [Target_Pitch, Target_Yaw]';  % Variables de salida del sistema

inputNet = con2seq(inputData);
targetNet = con2seq(targetData);

%% Creación de la NARX
numNeuronas = 30;

net = narxnet(0,1:2,numNeuronas);

%% Entrenamiento de la NARX

[x,initialInput,initialLayer,t] = preparets(net,inputNet,{},targetNet);

    net.divideParam.trainRatio = 100/100;
    net.divideParam.valRatio = 0/100;
    net.divideParam.testRatio = 0/100;

    net.trainParam.min_grad = 1e-100;
    net.trainParam.epochs = 1000;

%entrenamos la red
[net,tr] = train(net,x,t,initialInput,initialLayer);

yTrain= net(x,initialInput,initialLayer);

yF = cell2mat(yTrain);
targetData = cell2mat(targetNet);                    % Predicciones

%% Prueba de validación

load('Data/Input_Pitch_Val');
load('Data/Input_Yaw_Val');
load('Data/Target_Pitch_Val');
load('Data/Target_Yaw_Val');

inputDataV = [Input_Pitch_Val, Input_Yaw_Val]';   % Variable de entrada al sistema
targetDataV = [Target_Pitch_Val, Target_Yaw_Val]';  % Variable de salida del sistema

inputNetV = con2seq(inputDataV);
targetNetV = con2seq(targetDataV);

[x,initialInputV,initialLayerV,t] = preparets(net,inputNetV,{},targetNetV);

% Ejecución de red, en la manera compacta
    yValid= net(x,initialInputV,initialLayerV);
    yValid = cell2mat(yValid);    % Salida método compacto

%% Validación paso a paso

inputDataValPaP = [Input_Pitch_Val, Input_Yaw_Val]';
targetDataValPaP = [Target_Pitch_Val, Target_Yaw_Val]';

xk = [inputDataValPaP(1,1), inputDataValPaP(2,1)]';
yk_1 = [0, 0, 0, 0]';

yValPaP(:,1) = Paso_a_Paso_NARX_MIMO_con(xk,yk_1,net);

xk = [inputDataValPaP(1,2), inputDataValPaP(2,2)]';
yk_1 = [targetDataValPaP(1,1), targetDataValPaP(2,1), 0, 0]';

yValPaP(:,2) = Paso_a_Paso_NARX_MIMO_con(xk,yk_1,net);

for i=3:length(inputDataValPaP)

    xk = [inputDataValPaP(1,i), inputDataValPaP(2,i)]';

    yk_1 = [targetDataValPaP(1,i-1),targetDataValPaP(2,i-1), targetDataValPaP(1,i-2),
targetDataValPaP(2,i-2)]';

    yValPaP(:,i) = Paso_a_Paso_NARX_MIMO_con(xk,yk_1,net); % Salida método PaP
end
```

4. Función Paso a Paso

```
function outputs  = Paso_a_Paso_NARX_MIMO_con(xk,yk_1,net)

gainIn_xk  = net.inputs{1}.processSettings{1}.gain;
offsetIn_xk  = net.inputs{1}.processSettings{1}.xoffset;

gainIn_yk_1 = [net.inputs{2}.processSettings{1}.gain; net.inputs{2}.processSettings{1}.gain];
offsetIn_yk_1 = [net.inputs{2}.processSettings{1}.xoffset; net.inputs{2}.processSettings{1}.xoffset];

iNorm_input_xk  = normalizar(gainIn_xk,offsetIn_xk,xk(:));
iNorm_input_yk_1  = normalizar(gainIn_yk_1,offsetIn_yk_1,yk_1(:));

%% Cálculo de salidas de capa oculta

aXH = net.IW{1,1}*iNorm_input_xk;
aYH = net.IW{1,2}*iNorm_input_yk_1;

oH = tansig(aXH+aYH+net.b{1,1});
aO = net.LW{2,1}*oH + net.b{2,1};
oNorm = aO;
gainOut    = net.outputs{2}.processSettings{1}.gain;
offsetOut  = net.outputs{2}.processSettings{1}.xoffset;

outputs = desnormalizar(gainOut,offsetOut,oNorm);
```

5. Función de discretización multipaso hacia delante de Euler

```
function xk1 = modelo_TR_Discreto(xk,uk)

Ts = 0.1;   % Periodo de muestreo

xk1 = xk(:);

N = 10;    % N° de pasos de integración

dt = Ts/N;

uk1 = [uk(:);0];

for i = 1:N

xk1 = xk1 + dt*modelo_TR_Continuo(xk1,uk1);

end
```

6. Evaluación de objetivos MIMO

```
function yf = EvalObj_MIMO_NN (xf)

global ref;
global uk;
global yk;
global xk;
global k;
global H;
global Hc;
global yp;
global up;
global xp;
global ukopt;
global orden_sys;
global epred;
global epredNN;
global net;

%                Inicialización de variables

 for i=1:orden_sys

     yp(1,i) = yk(1,k-orden_sys-1+i);
     yp(2,i) = yk(2,k-orden_sys-1+i);
     xp(1,i) = xk(1,k-orden_sys-1+i);
     xp(2,i) = xk(2,k-orden_sys-1+i);
     xp(3,i) = xk(3,k-orden_sys-1+i);
     xp(4,i) = xk(4,k-orden_sys-1+i);
     xp(5,i) = xk(5,k-orden_sys-1+i);
     xp(6,i) = xk(6,k-orden_sys-1+i);
     up(1,i) = uk(1,k-orden_sys-1+i);
```

```
    up(2,i) = uk(2,k-orden_sys-1+i);
 end

 ukopt = zeros(2,H);
 ukopt(1,1:Hc) = xf(1,1:Hc);
 ukopt(2,1:Hc) = xf(1,Hc+1:2*Hc);

 for i=Hc/2+1:H

    ukopt(1,i) = ukopt(1,Hc);
    ukopt(2,i) = ukopt(2,Hc);

 end

 for i=(orden_sys+1):(orden_sys+H)

    up(1,i) = ukopt(1,i-orden_sys);
    up(2,i) = ukopt(2,i-orden_sys);

 end

%            Predicción sobre el horizonte

 for i=(orden_sys+1):(orden_sys+1+H)

    % Modelo neuronal
   yp(:,i) = Paso_a_Paso_NARX_MIMO_con([up(1,i-1); up(2,i-1)],[xp(1,i-1); xp(3,i-1); xp(1,i-2);
xp(3,i-2)]);

    % Modelo matemático

%     xp(:,i) = modelo_TR_Discreto(xp(:,i-1), up(:,i-1));
%
%     yp(1,i) = xp(1,i);
%     yp(2,i) = xp(3,i);

   yp(:,i) = yp(:,i) + epred;
 end

%        Primer Objetivo. Error trayectoria mínimo en Horizonte:
ref_pitch = ref(1,k:k+H);
ref_yaw = ref(2,k:k+H);
sum2 = 0;

for i=1:H
    sum2 = sum2 + (ref_pitch(i)-yp(1,i))^2 + (ref_yaw(i)-yp(2,i))^2;
end

yf(1) = sum2;

%          Segundo Objetivo. minimizacion incremento accion de control
sum4 = 0;

for i=2:H
    sum4 = sum4 + (up(1,i)-up(1,i-1))^2 + (up(2,i)-up(2,i-1))^2;
end

yf(2) = sum4;
```

7. iMO-NMPC

```
load('NN_JR_30_ENT_0_SAL_12.mat');   % Cargar modelo neuronal
global k;
global H;
global Hc;
global ref;
global uk;
global yk;
global xk;
global tm;
global t;
global yp;
global up;
global xp;
global orden_sys;
global epred;
global epredNN;
global net;
%                               PARAMETROS
orden_sys = 2;                   % orden del sistema
n= 900;                          % muestras de simulación
tm = 0.2;                        % tiempo de muestreo controlador
numPop = 80;                    % poblacion inicial
numGenerations = 80;            % numero de iteraciones (generaciones)
tsim = n * tm;                   % tiempo de simulación
H = 8;                           % horizonte de predicción
Hc = H;                          % horizonte de control
Num_Ent = 2;                     % nº de entradas

%          Inicialización de vectores y variables de programa
ref = zeros(2,n+H);
uk  = zeros(2,n);
xk  = zeros(6,n);
yk  = zeros(2,n);
ypk  = zeros(2,n);
OutNNpk = zeros(2,n);
t   = (0:tm:tsim-tm);
yp = zeros(2,H+5);
xp = zeros(6,H+5);
up = zeros(2,H+5);
epred = [0; 0];
epredNN = [0; 0];
RegParetoSet = cell(n,1);
RegNonInfSol = cell(n,1);

%         Generación de la referencia del control
load('Data/Ref_Pitch_Suavizada');
load('Data/Ref_Yaw_Suavizada');

ref(1,:) = Ref_Pitch_Suavizada(1:(n+H),2);
ref(2,:) = Ref_Yaw_Suavizada(1:(n+H),2);

%                               LAZO DE CONTROL
for k=orden_sys+1:n

    FitnessFunction = @EvalObj_MIMO_NN;     % nombre de la función de evaluación
    numberOfVariables = Num_Ent*Hc;         % número de variables del cromosoma

    for i=1:numberOfVariables

        lb(i)= -2.5;                  % limite inferior genes
        ub(i)= 2.5;                   % limite superior genes

    end

    A = []; b = [];                 % sin restricciones de desigualdad
    Aeq = []; beq = [];

    % Ejecución del Solver GAMULTIOBJ

    options = gaoptimset('PlotFcns',[]);
    options = gaoptimset(options,'PopulationSize',numPop);
    options = gaoptimset(options,'PopInitRange',[(-1);(1)]);
    options = gaoptimset(options,'CrossoverFcn',@crossoverheuristic);
    options = gaoptimset(options,'MutationFcn',@mutationadaptfeasible);
    options = gaoptimset(options,'TimeLimit',tm);
    options = gaoptimset(options,'Generations',numGenerations);

    [x,Fval,exitFlag,Output] = gamultiobj (FitnessFunction,...
                               numberOfVariables,A,b,Aeq,beq,lb,ub,options);

    % registro de frentes de pareto y soluciones no inferiores
    RegParetoSet(k,1) = {Fval};
    RegNonInfSol(k,1) = {x};
```

```
    % Decision Maker

    % Criterio de mínima distancia al origen

    d_min = 1000000;

    for i=1:length(Fval(:,1))

        d = Fval(i,1)^2 + Fval(i,2)^2;

        if d < d_min

           d_min = d;
           index = i;

        end
    end

%     Criterio Mínimo Error

%     d_min = 1000000;
%
%     for i=1:length(Fval(:,1))
%
%         d = Fval(i,1)^2;
%
%         if d < d_min
%
%             d_min = d;
%             index = i;
%
%         end
%
%     end

    % selección acción de control

    uk(1,k) = x(index,1);
    uk(2,k) = x(index,1+Hc);

    %              SIMULACIÓN PLANTA - CALCULO DE ERRORES
    % Salida real planta

    xk(:,k) = modelo_TR_Discreto(xk(:,k-1), uk(:,k-1));

    yk(1,k) = xk(1,k);
    yk(2,k) = xk(3,k);

    % Salida predicha de la planta - Matemática

    ypk(1,k) = xk(1,k);
    ypk(2,k) = xk(3,k);

    % Salida predicha de la planta - Neuronal

    OutNNpk(:,k) = Paso_a_Paso_NARX_MIMO_con([uk(1,k-1); uk(2,k-1)],[xk(1,k-1), xk(3,k-1), xk(1,k-2),
xk(3,k-2)]);

    % error de predicción

    epred(1,1) = yk(1,k) - ypk(1,k);
    epred(2,1) = yk(2,k) - ypk(2,k);

    epredNN(1,1) = yk(1,k) - OutNNpk(1,k);
    epredNN(2,1) = yk(2,k) - OutNNpk(2,k);

end
```