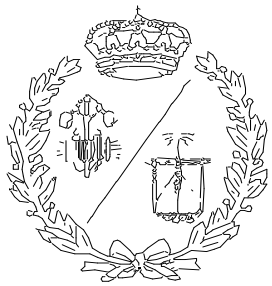
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

INCORPORACIÓN DE INTELIGENCIA ARTIFICIAL EN GEMELO VIRTUAL BÁSICO DE ROBOT INDUSTRIAL

(Incorporation of Artificial Intelligence in a virtual basic twin of an industrial robot)

Para acceder al Título de

GRADUADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Pablo Galán Saiz

Septiembre – 2019

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a:

- Todos los profesores que han puesto su granito de arena en este proyecto mostrando siempre predisposición para ayudar.

- A mis compañeros de clase, que han hecho estos cuatro años mucho más agradables.

- Pero sobre todo, a mi familia y a Carmen, porque ellos son los que han estado apoyándome todo este tiempo.

CONTENIDO

ILUSTRACIONES

ESQUEMAS

TABLAS

OBJETIVO DEL PROYECTO

En este proyecto, se unirán diferentes tecnologías que forman parte de la industria 4.0 como son las tecnologías de diseño 3D, las técnicas de ingeniería de sistemas y automática y la inteligencia artificial. Durante el proyecto, se utilizará un control dinámico sencillo desarrollado con anterioridad, y técnicas de identificación y modelado de sistemas dinámicos mediante el uso de Deep learning aplicado a través de redes neuronales.

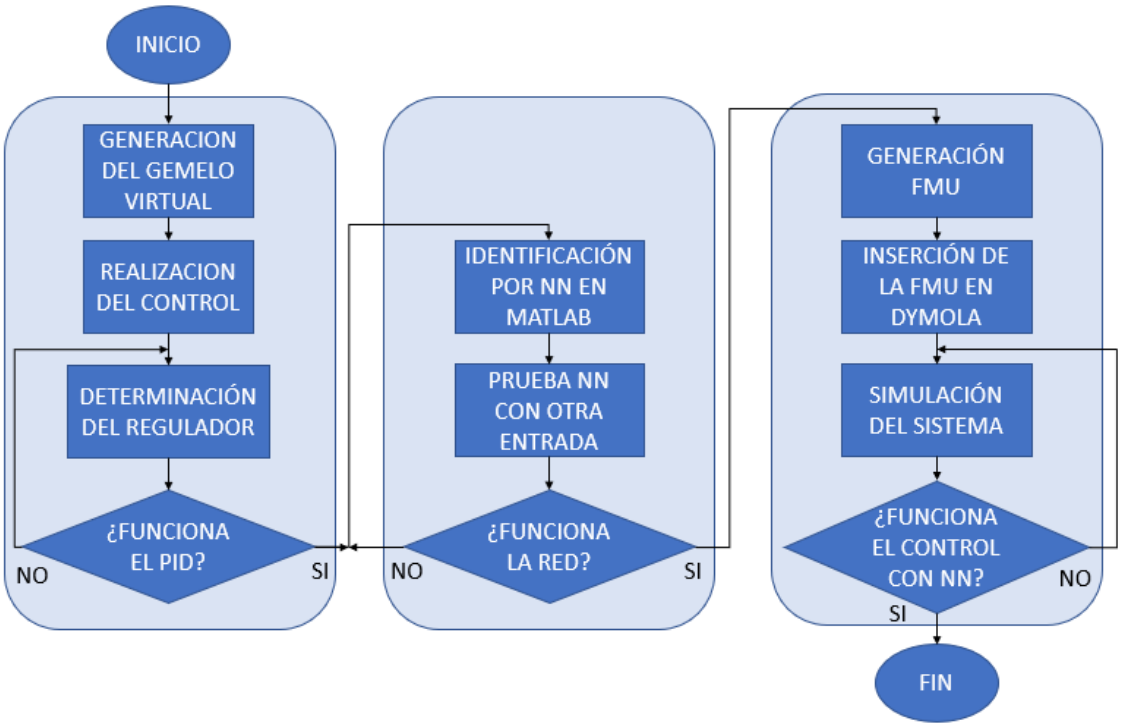
Entrando en detalle, el proyecto se llevará a cabo en diferentes fases. En primer lugar, se describirá la generación y posterior control dinámico del gemelo virtual del robot industrial. Una vez realizado el gemelo, se procederá con la implantación de técnicas de inteligencia artificial basadas en aprendizaje profundo (Deep learning). De esta forma, se tratará de conseguir un nuevo regulador que mantenga el comportamiento del anterior, pero que en este caso, se trate de un modelo neuronal del regulador previo. Posteriormente se confirmará que el funcionamiento del gemelo virtual sea el mismo o muy similar al previo, la única variación será que ahora en el control existirán técnicas de inteligencia artificial.

RESUMEN DEL PROYECTO

Durante el desarrollo del proyecto van a ser llevadas a cabo distintas fases con la finalidad de lograr el objetivo. El proyecto consta de tres fases principales presentadas en el esquema inferior como tres conjuntos. Estos a su vez se dividen en diferentes sub-fases que se desarrollarán para la consecución final de los objetivos planteados.

Las tres divisiones principales de las que consta el proyecto son las siguientes:

- Monitorización y control.
- Uso de IA mediante Deep Learning.
- Inserción del modelo neuronal en el control y comprobación de los resultados.



A continuación, se entra a comentar cada una de las fases del proyecto.

Durante la primera de ellas, el objetivo global será el análisis del funcionamiento del control del gemelo virtual del robot industrial, haciendo hincapié, en primer lugar, en cada una de las diferentes fases del control y, en segundo lugar, en el propio controlador en sí, al centrarse el resto de las fases del proyecto en este elemento del control. Por otro lado, se realizará un análisis de los resultados, simulando el control para determinar cómo será el funcionamiento

del gemelo ante determinadas referencias de posición y velocidad, con el objetivo de comparar estos resultados con los conseguidos en el futuro en el resto de las fases.

En segundo lugar, el proyecto se centrará en la inteligencia artificial. Antes de poder comenzar a desarrollar nuestros modelos, se deberá alcanzar un nivel de conocimiento suficiente para poder afrontar el desarrollo de las redes con la posibilidad de llegar a resultados positivos. Una vez adquirido el conocimiento base, se centrarán los esfuerzos, en la aplicación de las redes neuronales al modelado de sistemas dinámicos. En concreto, se tratará de conseguir un modelo del regulador existente en el controlador del gemelo virtual, mediante una correcta configuración de los pesos de una red neuronal.

Si tras un intenso análisis, los resultados del funcionamiento de la red neuronal son los deseados, se procederá ahora a la implantación de este nuevo control en el gemelo virtual. Para el desarrollo de esta tarea, serán necesarios interfaces que permitan conectar los programas utilizados. Finalmente, se comprobará el buen funcionamiento de este nuevo control, sobre el propio gemelo y se compararán los resultados obtenidos en esta fase, con los ya obtenidos durante la primera donde el control utilizado fue el ya existente.

1. INTRODUCCIÓN

1.1. ENTORNO

Para comenzar a hablar sobre el proyecto, se debe presentar en primer lugar la nueva revolución industrial que se lleva produciendo en los últimos años, la industria 4.0. En la actualidad, nos encontramos en la punta del iceberg de la transformación del modelo productivo que se llevaba produciendo durante las últimas décadas. Esta transformación, es originada por la inmersión de estos procesos productivos en un nuevo entorno donde nuevas tecnologías inteligentes emergentes, hacen que sean modificados mejorando de forma notoria la productividad de los procesos y generando con ello un tejido industrial mucho más estable, rentable y competente.

Con la nueva industria 4.0, se pretende llevar acabo la unión de los métodos de producción de la industria con las tecnologías de información y comunicación existentes en la sociedad, con el claro objetivo de, aparte de como se ha mencionado anteriormente, seguir la línea del resto de revoluciones industriales de mejora de productividad, diseñar y fabricar nuevos productos que aporten soluciones innovadoras a las demandas de la sociedad digitalizada. Pero la industria 4.0 no para ahí, también trata de integrar la información y el conocimiento en las distintas etapas de los productos realizados.

Las nuevas tecnologías inteligentes que han hecho posible la industria 4.0 son múltiples:

- Big Data
- Robótica
- Integración vertical y horizontal de sistemas
- Internet de las cosas
- Ciberseguridad
- La nube
- Simulaciones
- Realidad aumentada
- Fabricación por adición

FIG 1.1. Tecnologías que componen la industria 4.0 [2]

La industria 4.0 es continuista a la hora de requerir a los ingenieros la capacidad de aunar conocimientos de las diferentes ramas de la ingeniería. Durante este proyecto, diversas tecnologías que conforman parte de la industria 4.0, y que a su vez requieren la aplicación de ramas como la electrónica, mecánica o ingeniería de software van a ser aplicadas. Esto confirma la multidisciplinariedad necesaria para abordar proyectos de la nueva industria emergente.

2. GEMELO VIRTUAL

El primer paso sobre el que realizar los siguientes, es llevar acabo el gemelo virtual del robot. Aquí también es posible diferenciar diferentes fases que seguir:

2.1. SELECCIÓN DEL ROBOT INDUSTRIAL

En primer lugar y como base, se seleccionará el robot industrial del cual se va a llevar acabo el gemelo virtual. Este, será un ABB IRB 120, el cuál es el robot industrial multiusos más pequeño de la marca suiza.

FIG 2.1. Robot ABB IRB120

Se trata de un robot con 6 grados de libertad, por lo que existen 6 ejes de rotación diferentes. En los robots industriales de 6 grados de libertad, los 3 primeros ejes determinan la posición del TCP (Tool Center Point), mientras que, por otro lado, los 3 últimos determinan la orientación del eje correspondiente a la herramienta, que se encuentra en ese punto.

En cuanto a las especificaciones del robot, se puede hablar principalmente de las dimensiones y los rangos de movimiento de cada eje. Las imágenes que representan estas características han sido obtenidas de la hoja de especificaciones del robot facilitada por ABB.

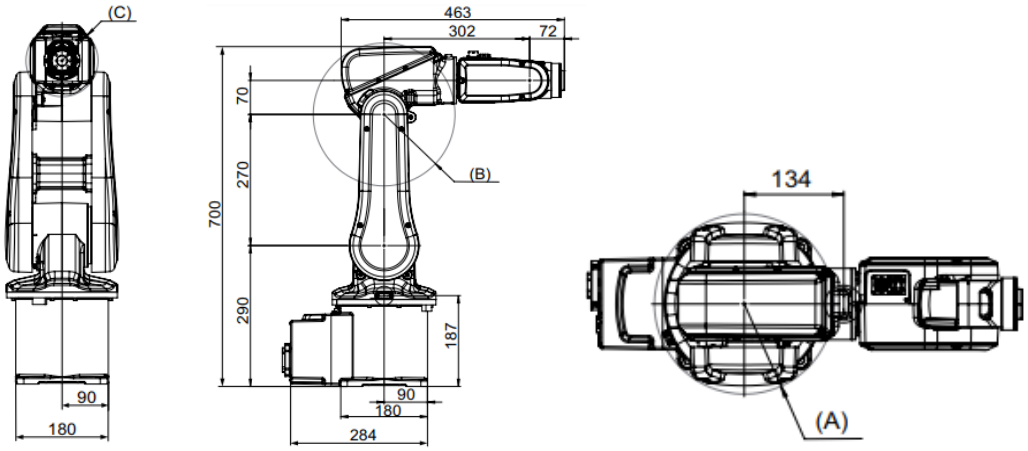


FIG 2.2. Dimensiones del robot industrial IRB120 [4]

La figura 3, extraída de la hoja de especificaciones del robot de ABB IRB120 [4], representa al propio robot junto las dimensiones de sus diferentes partes.

En la parte inferior izquierda, se encuentra una tabla cuyos valores fueron extraídos de [4] con los rangos de trabajo en grados, de cada uno de los ejes, así como las velocidades de giro para esos ejes. Mientras que, en la parte inferior derecha, se puede observar una ilustración con los ejes y su dirección de rotación, que también ha sido extraída de [4].

Eje	Rango de trabajo	Velocidad
1	[165º,-165º]	250º/s
2	[110º,-110º]	250º/s
3	[70º,-110º]	250º/s
4	[160º,-160º]	320º/s
5	[120º,-120º]	320º/s
6	[400º,-400º]	420º/s

Tabla 1. Rango movimientos robot IRB120 [4]

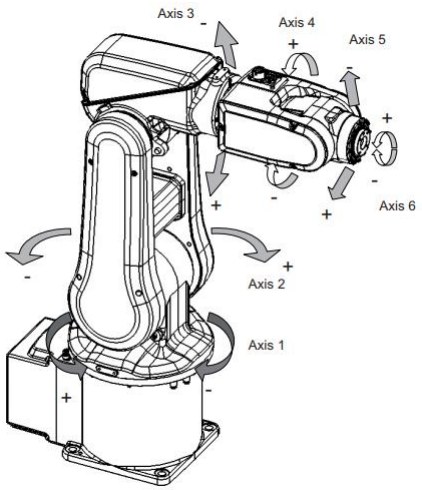


FIG 2.3. Sentido ejes de rotación de las articulaciones

2.2. ARQUITECTURA DEL SISTEMA

La mecatrónica es un concepto que surge en Japón mediados los sesenta. Es un campo multidisciplinario que engloba ramas partes de la ingeniería, como son la ingeniería mecánica, la ingeniería de sistemas, el control automático, la electrónica, o la ingeniería informática y de telecomunicaciones. Centrándose en el propio sistema, un sistema mecatrónico esta principalmente formado por el mecanismo en sí que genera el movimiento físico, los

detectores, sensores y transductores, para observar las variables físicas y transformarlas en eléctricas. Un actuador, que da potencia al servicio físico y el ordenador, este determina el comportamiento del sistema mediante señales de control.

Viéndolo desde el punto de vista de la robótica, todos los elementos que forman parte de la mecatrónica son visibles en un robot industrial [Tabla 2], por lo que es evidente que estos robots son sistemas mecatrónicos.

COMPARACION SISTEMA MECATRÓNICO – ROBOT INDUSTRIAL

Elementos De La Arquitectura Del Sistema Mecatrónico	Elemento Correspondiente En Un Robot Industrial
Mecanismo	Brazo
Detectores, Sensores y Transductores	Encoders
Actuadores	Motores eléctricos
Unidad de procesamiento	Controlador

Tabla 2: Relación sistema mecatrónico con robot industrial

A finales de los 70, Japón para la promoción de su industria de maquinaria, desarrolló una clasificación de los productos mecatrónicos en cuatro clases distintas, los robots industriales, se encuentran en la parte superior de esta clasificación en la clase IV, junto con el resto de los productos diseñados con tecnologías mecánicas y electrónicas integrados simultáneamente.

Como se ha explicado, el robot industrial es un sistema mecatrónico, así que, para realizar la estructura del sistema en el gemelo virtual, se utilizará como base, la arquitectura de un sistema mecatrónico.

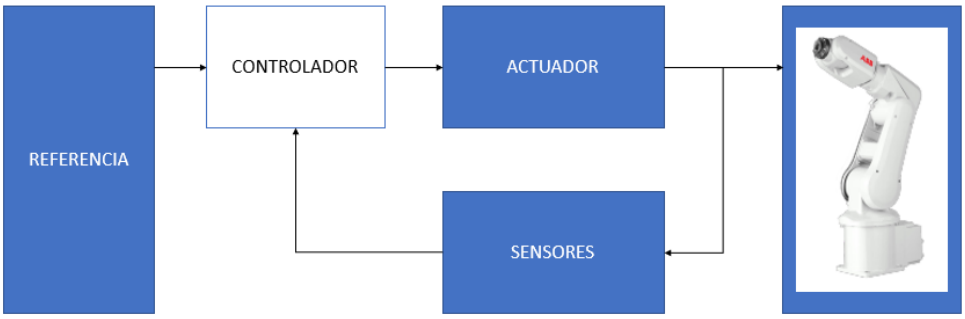


FIG 2.4. Arquitectura control una articulación del gemelo virtual

3. DESARROLLO DEL CONTROL DINÁMICO

3.1. ENTORNO DE TRABAJO

Una vez explicada superficialmente la arquitectura que tendrá el sistema, se procederá a detallar cada uno de los pasos a desarrollar para conseguir finalmente el control por realimentación del brazo robótico. El diseño de este mecanismo de control se desarrolló en la plataforma 3D EXPERIENCE de la compañía Dassault Systèmes, la cual proporciona soluciones de software para diferentes problemas del mundo de la empresa, así como en este caso, de la investigación. En el interior del software, se encuentran 4 grupos principales de aplicaciones, con diferentes enfoques:

- Aplicaciones sociales y corporativas
- Aplicaciones de inteligencia de información
- Aplicaciones de contenido y simulación
- Aplicaciones de modelización 3D

Para el desarrollo del control por realimentación del robot industrial, se utilizarán algunas de las múltiples funciones que permite el módulo de aplicaciones de contenido y simulación.

Principalmente la aplicación a usar para el diseño es “functional and logical design”. Esta aplicación se encuentra formada a su vez por cuatro grandes bloques, como se puede ver a continuación.

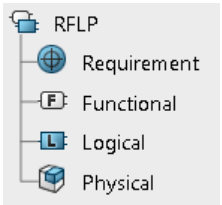


FIG 3.1. Árbol menú aplicación RFLP

En este proyecto, los bloques a desarrollar son principalmente el tercero y cuarto, que corresponden al diseño lógico y físico respectivamente.

3.2. MODELO FISICO

En cuanto al modelo físico, este se corresponde con la estructura física del robot. En este caso, para poder comenzar, únicamente se debe introducir en la aplicación una librería (Robotlib), donde se encuentra entre una gran cantidad de robots industriales, el robot IRB120, que se va a utilizar. En la imagen figura 2.1. se puede observar el modelo físico del robot industrial obtenido de la librería.

3.3. DESARROLLO DE LA ARQUITECTURA

En la ilustración 2.3, se muestra la arquitectura más clásica para un sistema mecatrónico. En ella, se puede observar que el sistema mecatrónico, está formado por el controlador propiamente dicho, actuadores, sensores y transductores. En el caso de estudio, el sistema mecatrónico, es un robot industrial IRB120, que, como ya se explicó, consta de seis grados de libertad, con seis articulaciones diferentes. Cada una de esas articulaciones, tiene sus propios elementos que forman parte de la arquitectura, por lo que, si tenemos en cuenta las seis articulaciones, nuestro sistema deberá controlar cada una de ellas, apareciendo pues un control por realimentación para cada una de las articulaciones.

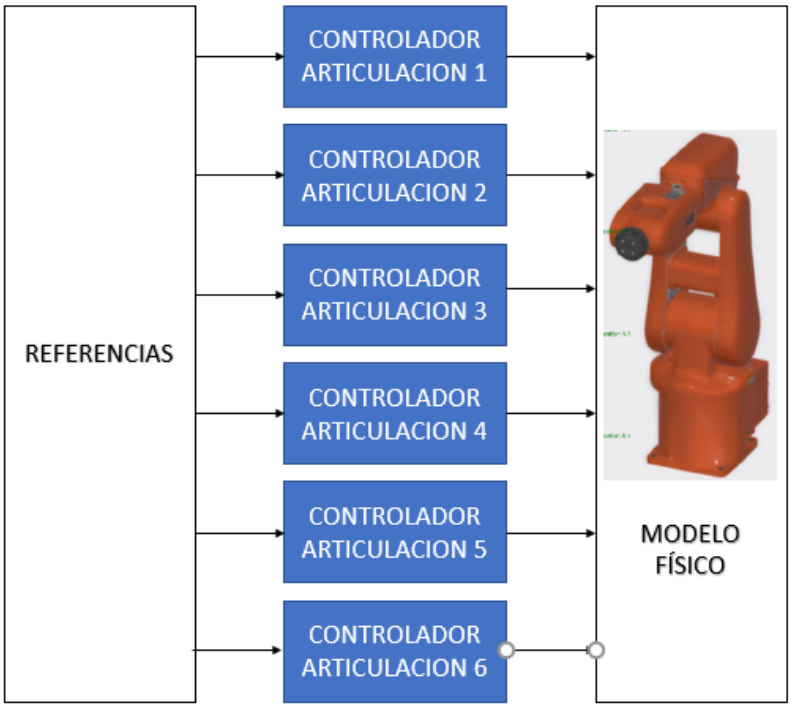


FIG 3.2. Control gemelo virtual del robot

Una vez la arquitectura del sistema ha sido comentada, se sabe que el control dinámico del sistema mecatrónico estará formado por seis diferentes servo controladores al existir seis articulaciones diferentes y que, a su vez, cada uno de estos controladores, tiene en su interior la arquitectura clásica de un sistema mecatrónico, es decir, están formados por el controlador, los actuadores y los sensores.

A continuación, se muestra el esquema general del control que se va a realizar sobre una de las articulaciones del robot a modo de ejemplo, teniendo en cuenta que este se va a aplicar por un lado sobre la velocidad y posición de los motores de las articulaciones, y por otro sobre la corriente que circula por cada uno de los motores para las propias articulaciones.

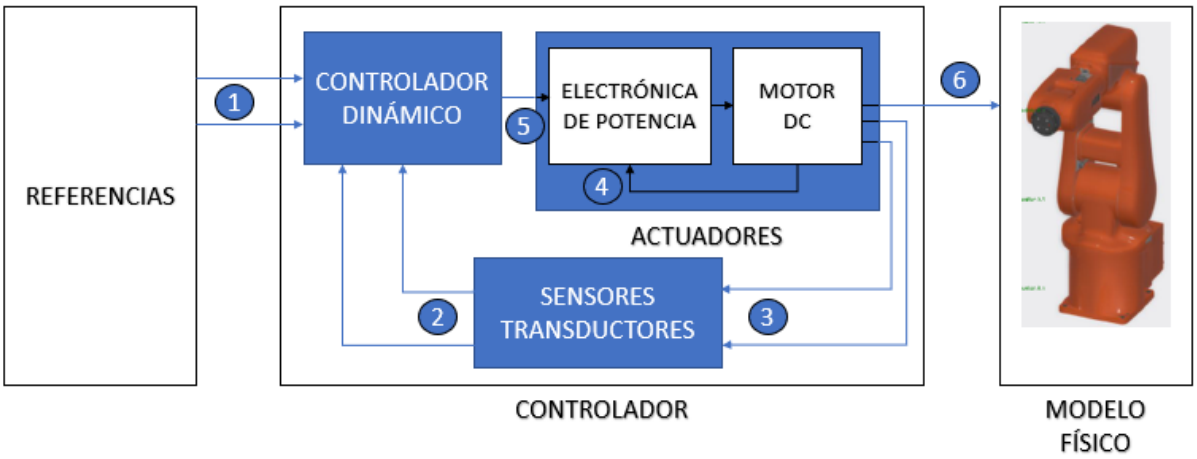


FIG 3.3. Esquema de control de una articulación.

1. Velocidad y posición angular de referencia
2. Velocidad y posición angular actuales
3. Medición de velocidad y posición angular
4. Corriente por el motor
5. Señal de control
6. Actuación mecánica

Sobre el esquema superior, se va a trabajar, explicando brevemente cada una de las partes que conforman el control de una articulación y su función.

3.3.1. SEÑALES DE REFERENCIA

En las señales de referencia, será el lugar donde se introduzcan los valores que se desea que alcancen cada una de las uniones, y la velocidad con la que realizarán el movimiento. Esto se realizará a través de un bloque generador de señales para sistemas cinemáticos.

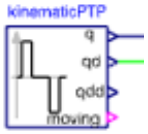


FIG 3.4. Generador señales cinemáticas

3.3.2. ACTUADORES

Una vez obtenida la señal de control, esta debe ser aplicada al actuador. Para ello previamente la señal de control es tratada con la finalidad de obtener la señal de potencia que controle el funcionamiento del motor.

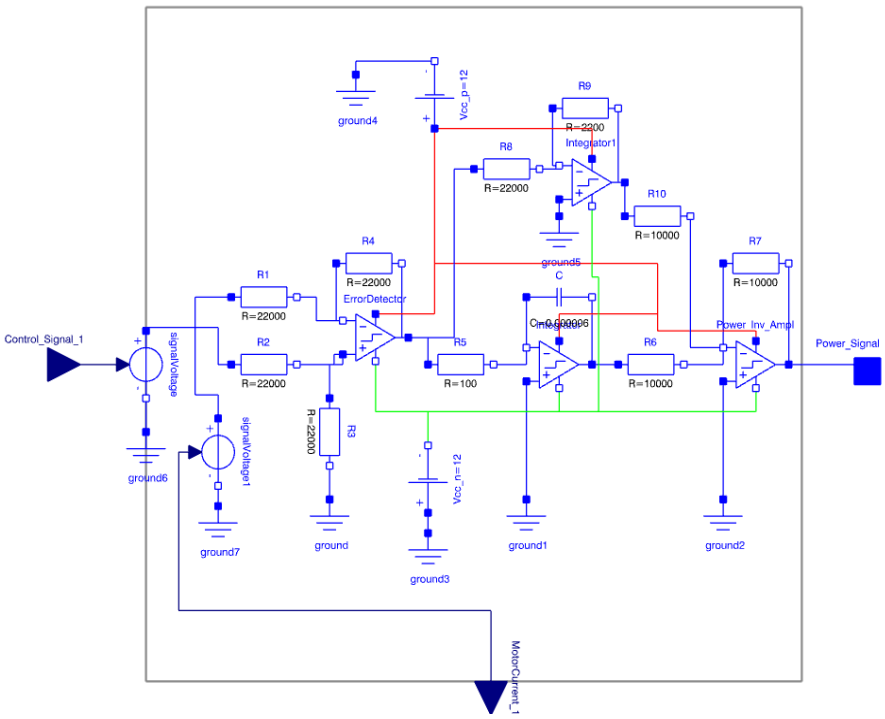


FIG 3.5. Electrónica de Potencia actuador

La señal es modificada en diferentes fases con el objetivo de conseguir una señal final idónea.

Una vez obtenida la señal de potencia, esta es transformada en corriente para posteriormente excitar al motor de la articulación. Por último, mediante el uso de sensores, la velocidad y posición angular del eje de salida del motor serán muestreadas y sus valores procesados para obtener la nueva señal de error. En la imagen inferior se observa el comportamiento del motor de la articulación implementado en DYMOLA.

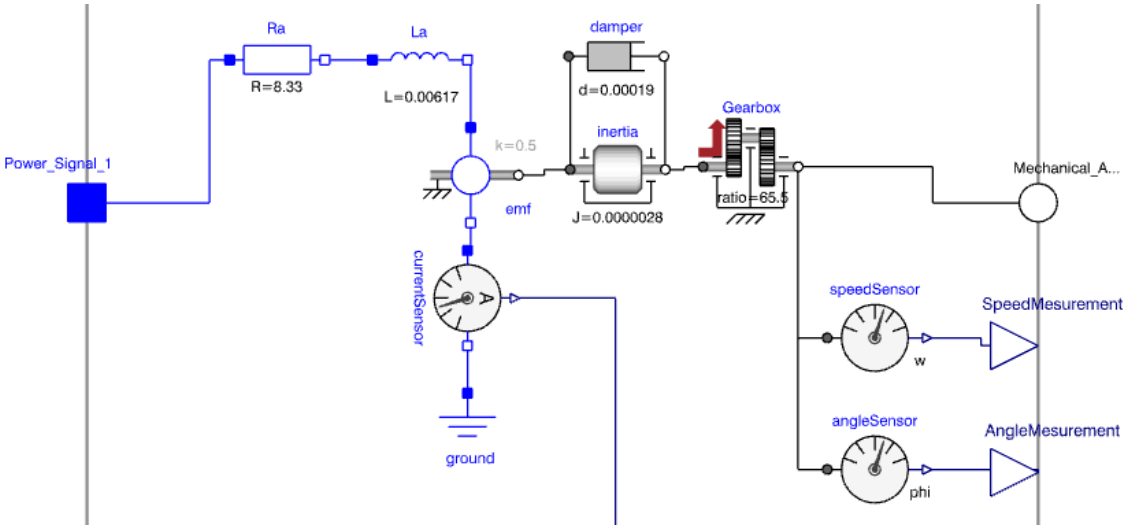


FIG 3.6. Motor configurado en DYMOLA

3.3.3. MODELO FÍSICO

Otro de los elementos importantes del control será el modelo físico del robot industrial. En este caso es fácil de conseguir debido a que el software lo genera automáticamente a partir del diseño en CAD del robot, teniendo en cuenta las diferentes partes del robot y el tipo de unión en cada articulación. Finalmente el sistema físico generado tendrá la forma de la siguiente figura para cada una de las uniones.

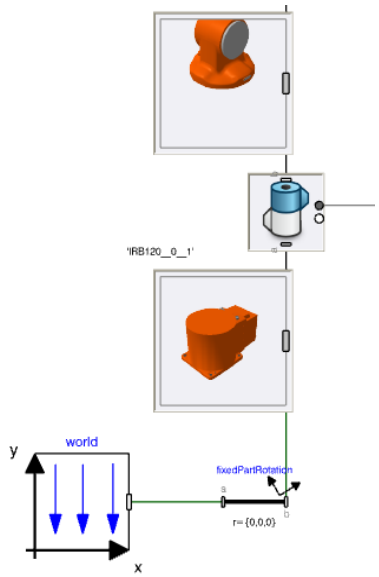


FIG 3.7. Modelo físico de la primera articulación

3.3.4. SENSORES Y TRANSDUCTORES

Los sensores captarán la velocidad y posiciones en cada instante de tiempo de las diferentes articulaciones y mediante las relaciones de conversión se transforma la señal real en el valor óptimo para ser comparado con la señal de referencia y obtener consigo los errores.

3.3.5. CONTROLADOR

Para el proyecto, el controlador del propio control dinámico es el bloque más importante debido a que será el bloque sobre el que se trabajará. A continuación, se va a explicar con detalle el funcionamiento del bloque controlador de una de las articulaciones del robot.

Como se puede observar en la figura 3.8, el bloque se encuentra formado de modo superficial por cuatro entradas y una única salida.

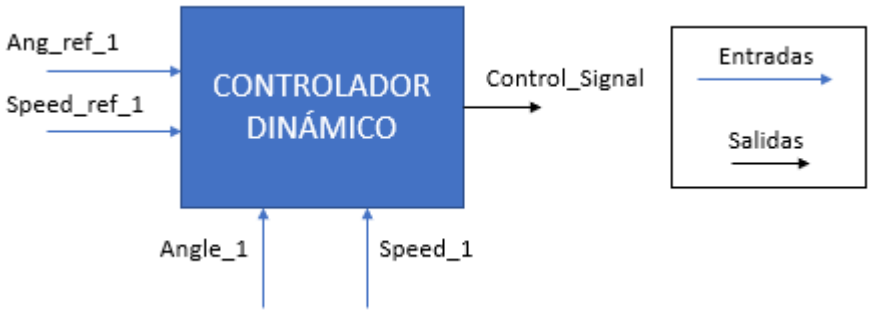


FIG 3.8. Bloque controlador con señales.

Entradas	Salidas
Ang_ref_1 Speed_ref_1 Angle_1 Speed_1	Control_Signal

Tabla 3. Señales de entrada y salida de cada bloque controlador

Las dos primeras señales de entrada corresponden a las señales de referencia introducidas manualmente en el bloque de referencias, mientras que las entradas restantes, corresponden a la velocidad y posición angular actuales del eje de salida del motor una vez estandarizadas.

El objetivo del proyecto es sustituir el regulador del controlador por su modelo neuronal. Por tanto, se considera importante comentar su funcionamiento.

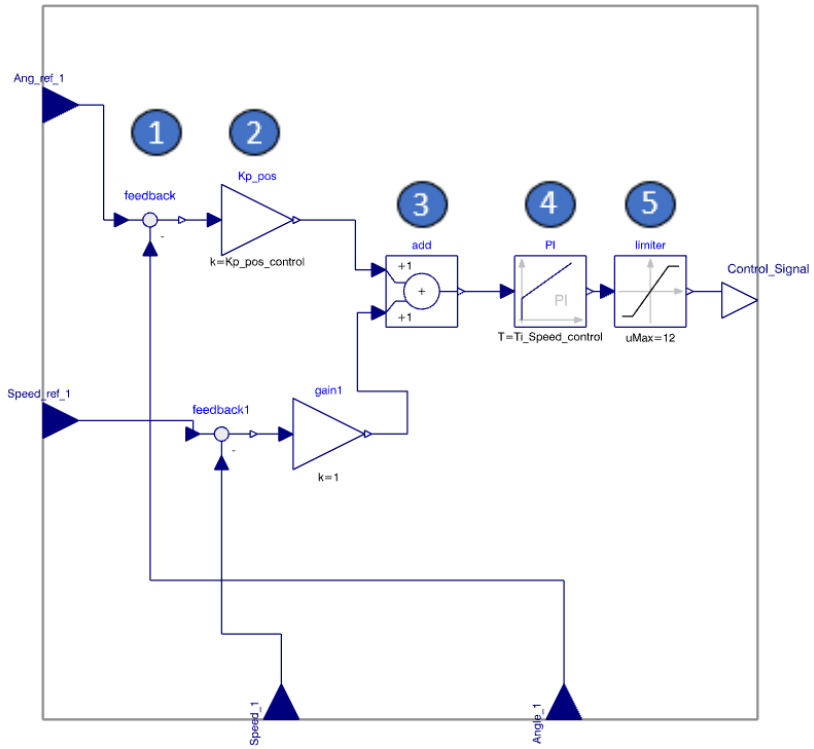


FIG 3.9. Esquema bloque controlador

Los elementos de numerados de la imagen superior pasarán a ser comentados.

- 1. En primer lugar, se realiza una comparación entre las señales de referencia y las señales reales en ese momento. Con ello, se obtiene el error, es decir, la diferencia entre la señal deseada y el punto en el que se encuentra el robot. En el caso de la velocidad, será la diferencia entre la velocidad a la que se pide que realice la función y la velocidad real con la cual lo está realizando, y por otro lado en el caso de la posición, será la diferencia entre la posición final y la posición en la que se encuentra, por lo que en este caso se obtiene la distancia angular que falta por realizar.

- 2, 3 y 4. Una vez realizada la comparación, ambas señales son multiplicadas por sendas ganancias, para después ser sumadas en un bloque sumador y finalmente, ser obtener la respuesta del regulador PI ante esa señal. El regulador tendrá la siguiente forma.

$$PI = 0{,}1\left(s + \frac{0{,}01}{s}\right) \qquad (1)$$

  Por lo que la ganancia del regulador será 0,1 y el tiempo de integración 100.

- 5. Finalmente, la señal una vez integrada, es limitada entre un máximo valor 12 y mínimo de menos 12.

Teniendo en cuenta todas las partes del control del gemelo virtual comentadas, la estructura final generada en la aplicación RFPL tendrá la siguiente forma.

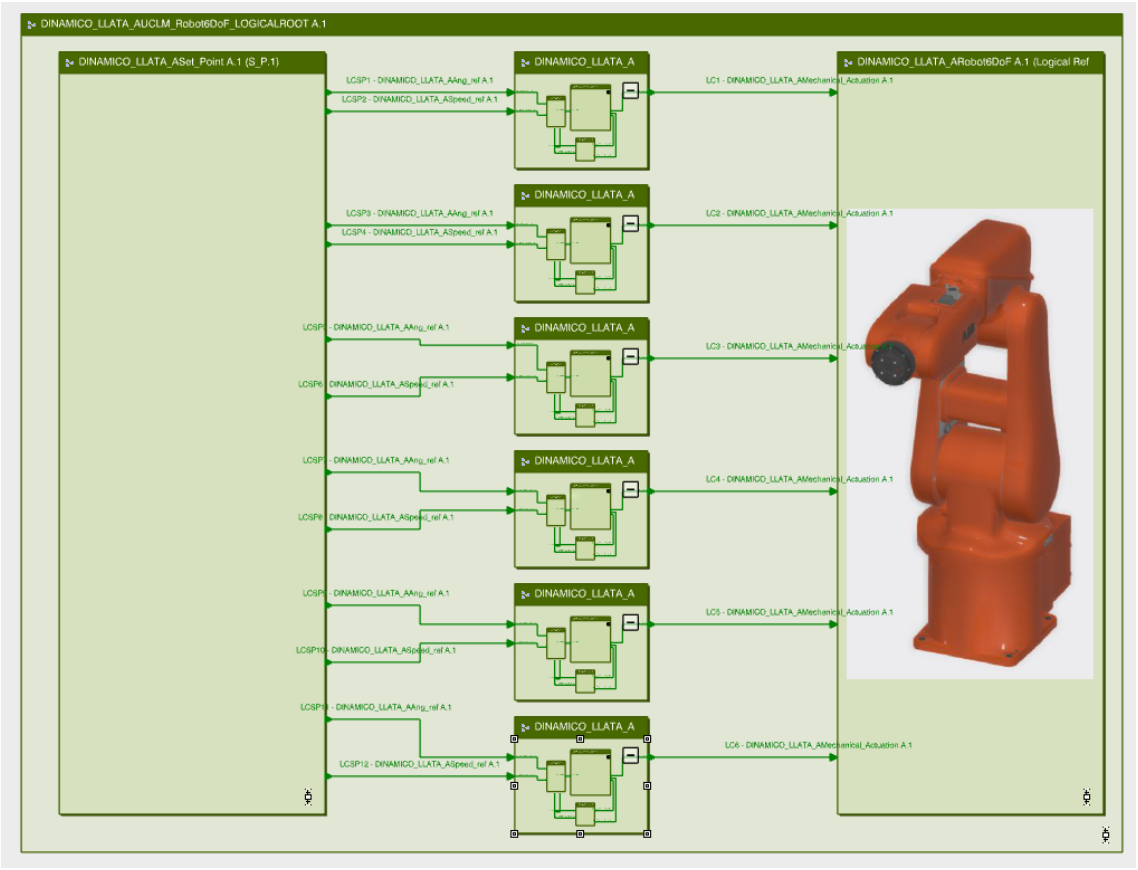


FIG 3.10. Gemelo virtual y control del robot industrial IRB120

4. REDES NEURONALES

El objetivo ya comentado del proyecto es introducir técnicas de inteligencia artificial basadas en aprendizaje profundo en el control comentado anteriormente. Para ello, se utilizan redes neuronales, de forma que estas aprendan el anterior funcionamiento del bloque que posteriormente sustituirán.

Las redes neuronales artificiales nacen en 1965 cuando a partir de la creación de Frank Rosenblatt, el perceptrón, se crea el multilayer perceptrón, el cual consiste en un conjunto de perceptrones unidos entre sí, de forma que el siguiente recogerá salidas de anteriores perceptrones y generará una nueva salida obteniéndose finalmente una nube de perceptrones que en un futuro dio lugar a las redes neuronales.

4.1. DEFINICIÓN DE RED NEURONAL

Una vez llegados al concepto actual de redes neuronales, existen distintas definiciones entre la que destaca: "*Una red neuronal es un modelo computacional, paralelo, compuesto de unidades procesadoras adaptativas con una alta interconexión entre ellas*". (Hassoun, 1995) [11].

Las redes neuronales, en su gran mayoría, tratan de simular el comportamiento biológico del cerebro. Son los elementos básicos donde se producen las acciones a realizar. Las neuronas en las redes neuronales se encuentran interconectadas mediante los pesos sinápticos, que varían en función de aprendizaje, de forma que el aprendizaje es el proceso que modifica las conexiones entre las neuronas de una red, para adaptarla a la tarea que debe desempeñar.

4.2. LA NEURONA ARTIFIAL

4.2.1. ELEMENTOS BÁSICOS DE UNA NEURONA ARTIFICIAL

Existen cuatro elementos básicos en todo modelo de neurona artificial:

- 1. Las conexiones denominadas pesos, que son las que determinan el comportamiento de la neurona. Estas conexiones pueden ser de excitadoras o inhibidoras, es decir si son positivas o negativas.

- 2. Un bloque sumador, que se encarga de sumar las entradas de la neurona, multiplicadas por su respectivo peso.

- 3. Una función en principio no lineal que elimina la linealidad de la red, haciendo que la red pueda funcionar con problemas no lineales además de limitar la amplitud de la salida de la neurona. Al aplicar las funciones de activación, se pueden modelar relaciones mucho más complicadas entre las entradas y los resultados predichos.

- 4. En algunos casos, existe un umbral que determina cuando se activa la neurona.

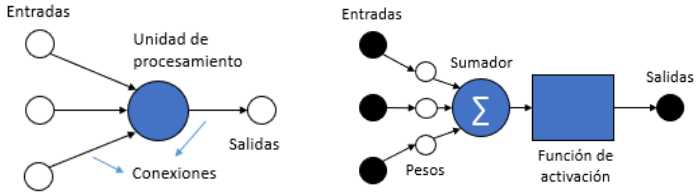


FIG 4.1. a) diagrama neurona artificial, b) arquitectura unidad procesamiento

4.2.2. FUNCIONAMIENTO DE UNA NEURONA ARTIFICIAL

El funcionamiento de una neurona sencillo. En primer lugar, se debe tener en cuenta la regla de propagación, que a su vez tiene en cuenta las entradas y los pesos de cada entrada. En este caso, se va a comentar el caso más más comúnmente utilizado, que combina las entradas y los pesos de la siguiente forma [17]:

$$h_i = \sum_{i=1}^{n} \omega_{ij} x_j \qquad (2)$$

Donde:

- ω_{ij} son los pesos sinápticos.
- x_j son las entradas.

Una vez tratadas las entradas, el valor resultante se debe pasar por la función de activación que devolverá la respuesta de la neurona artificial a esa entrada.

$$y_i = f_i(h_i) \qquad (3)$$

La función de activación variará dependiendo del uso que se le dé a la red neuronal.

4.3. ARQUITECTURAS NEURONALES

Según el criterio utilizado para la conexión de los elementos principales de una neurona artificial, se determina la arquitectura de la red neuronal. Por ello, existen tres tipos de arquitectura:

4.3.1. SEGÚN EL NUMERO DE CAPAS

- Redes neuronales monocapa: Una capa de neuronas, proyectan las entradas otra capa de neuronas donde se realizan los cálculos, esta de salida. Este tipo de arquitectura se utiliza principalmente para memorias asociativas.

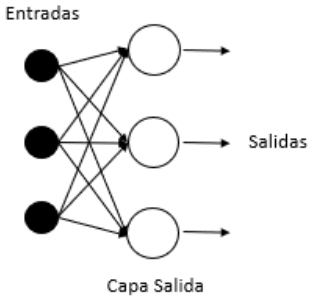


FIG 4.2. Arquitectura red neuronal monocapa

- Redes neuronales multicapa: Existen un conjunto de capas intermedias o capas ocultas, situadas entre la entrada y la salida. El término Deep Learning proviene de este tipo de redes debido a que al aumentar la complejidad de la red, se incrementa el potencial de computación de la red.

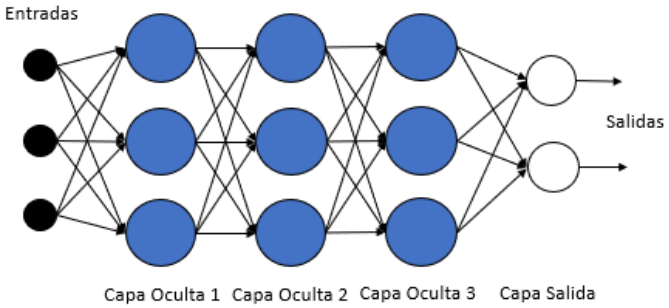


FIG 4.3. Arquitectura red neuronal multicapa

4.3.2. SEGÚN EL TIPO DE CONEXIONES

Las redes neuronales multicapa, pueden tener dos tipos de conexiones entre las diferentes neuronales que la conforman. Según este tipo de clasificación, estas pueden ser:

- No recurrentes: También denominadas feedforward, son redes donde las señales se transmiten únicamente en un sentido, es decir no tienen realimentaciones. Por ello, este tipo de red neuronal carece de memoria.

  Este tipo de red va a ser utilizado durante el proyecto, al ser más sencilla que sus antípodas y aun así lograr el resultado deseado.

  La ilustración 4.3. situada sobre este apartado, corresponde a una red feedforward.

- Recurrentes: Al contrario que las anteriores, estas permiten conexiones arbitrarias entre neuronas, por lo que se pueden hacer realimentaciones que dotan a la red de memoria. Existen diferentes tipos de redes recurrentes como las redes LSTM o las redes GRU.

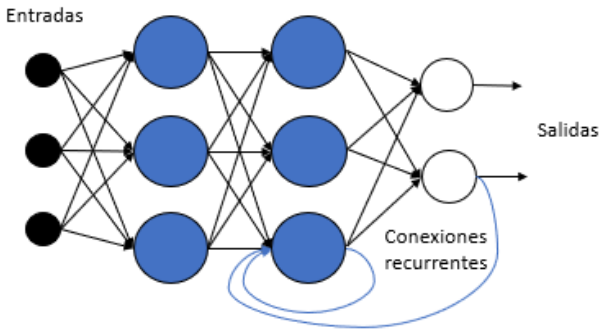


FIG 4.4. Red neuronal recurrente

4.3.3. SEGÚN EL GRADO DE CONEXIÓN

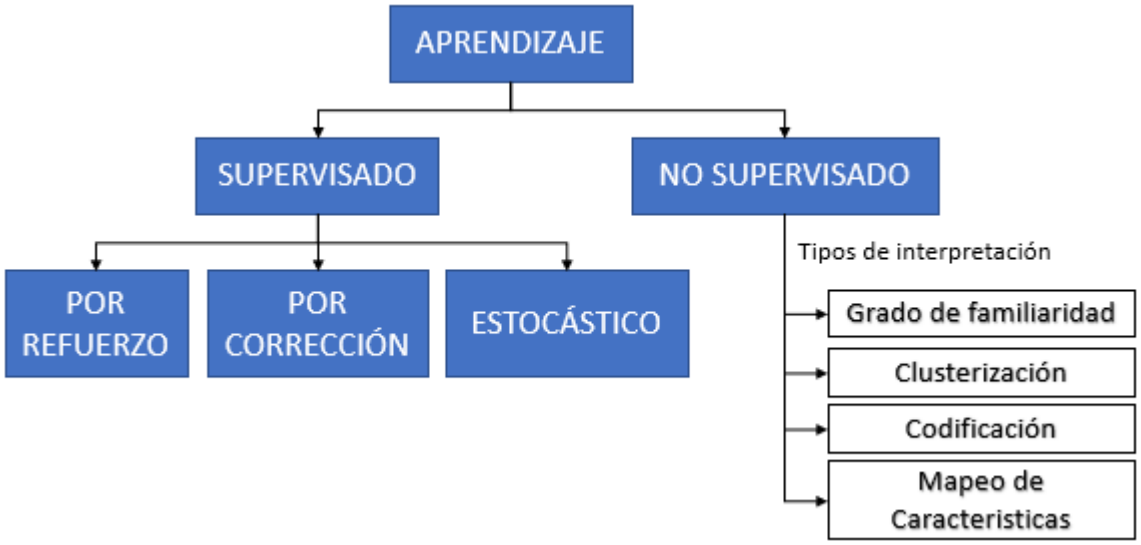
Según el grado de conexión, al igual que en el apartado anterior pueden ser de dos clases diferentes. Totalmente conectadas, es decir las neuronas de una capa se encuentran unidas a todas las neuronas de la siguiente capa, o por otro lado parcialmente conectadas, que como su nombre indica, en este caso existen neuronas que no se conectan a la capa posterior.

4.4. MÉTODOS DE APRENDIZAJE

El aprendizaje es un proceso por el cual los pesos de una red neuronal son reajustados continuamente en un proceso de estimulación por parte del entorno donde se sitúa el sistema. Por ello, el siguiente diagrama de flujo se lleva acabo para que se produzca el aprendizaje.

Esquema 1. Diagrama flujo aprendizaje red neuronal

En el esquema inferior, se puede observar la división de métodos de aprendizaje existentes:



Esquema 2. Tipos de aprendizaje de redes neuronales

4.4.1. APRENDIZAJE NO SUPERVISADO

En estos últimos, la señal de salida (deseada), es desconocida por lo que la red se debe organizar ajustando ella misma los pesos de las conexiones. Este aprendizaje, se basa en la búsqueda de diferentes características, regularidades o categorías, en los conjuntos de valores de entrada, para finalmente, aplicando una de las siguientes interpretaciones de la salida, se obtengan los resultados deseados. Los métodos de interpretación existentes son:

- Grado de familiaridad: Se basa en el reconocimiento de características entre la información de la entrada actual y otras anteriores.
- Clusterización: Se establecen diferentes categorías, de las cuales la red tratara de buscar características o propiedades en las entradas, para posteriormente ser clasificadas.
- Codificación: Se realiza una codificación de la entrada.
- Mapeo de características

4.4.2. APRENDIZAJE SUPERVISADO

En este otro caso, las salidas deseadas, son conocidas. En este tipo de aprendizaje, una vez la red ha obtenido un resultado, este es comparado con el deseado y finalmente los pesos de las conexiones son modificados dependiendo del error cometido. Hay tres variantes, donde las dos primeras comentadas adquieren peso en relación con la restante.

- Aprendizaje por Corrección de Error: Se presentan una entrada (datos de entrenamiento) y su salida deseada, una vez la red ha obtenido una primera respuesta a la entrada, esta se compara con la salida deseada y si se cumplen los criterios de error, se da por finalizado el entrenamiento. En caso contrario, se vuelven a presentar los datos de entrenamiento y se realizan de nuevo los mismos pasos.

- Aprendizaje por Refuerzo: En este tipo de aprendizaje, no se dispone de la salida deseada exacta, sino que se conoce como debería ser el comportamiento ante varias entradas. Para solucionarlo, en este caso, las redes neuronales realizan la tarea encomendada constantemente, solucionando los errores hasta llegar a un resultado lo más perfecto posible.

- Aprendizaje Estocástico: Este último tipo de aprendizaje consiste en variar el valor de los pesos, para estudiar el efecto que tiene el cambio en la respuesta. Si la respuesta se ajusta al objetivo es aceptada, caso contrario, se hace uso de una preestablecida distribución de probabilidades y se acepta en el caso de cumplirla.

4.5. ESTRUCTURAS NEURONALES

Para determinar la estructura que tendrá la red neuronal a utilizar, se debe tener en cuenta la aplicación para la cual va a ser utilizada. Existen diferentes tipos de disposiciones en función de la aplicación a desarrollar:

4.5.1. ESTRUCTURA DIRECTA

El objetivo de este tipo de estructura es conseguir el mínimo error cuando la salida del sistema desconocido y la red neuronal sea la misma. Para ello, como se puede observar en la figura 4.5. Ambas son excitadas con la misma entrada, posteriormente, las salidas son comparadas, y el error introducido de nuevo en la red neuronal.

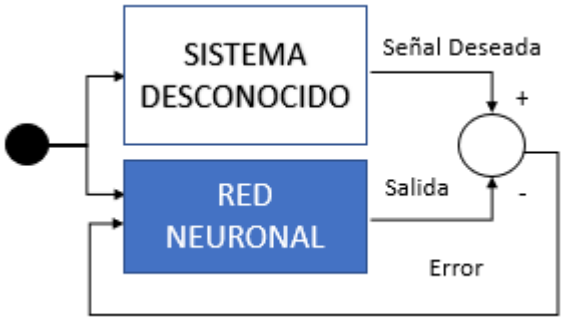


FIG 4.5. Estructura directa

Estas estructuras, se desenvuelven correctamente modelizando funciones de transferencia de sistemas los cuales conocemos su salida ante una determinada entrada, pero se desconoce cómo son internamente.

4.5.2. ESTRUCTURA INVERSA

En este segundo tipo de estructura, el sistema (que continúa siendo desconocido) es excitado con una entrada y posteriormente, esa salida obtenida, es introducida como entrada en la red neuronal, de forma que la salida a conseguir deberá ser igual que la señal de entrada introducida en el sistema. De conseguir esto, las funciones de transferencia del sistema y de la red neuronal serán las inversas, y el error el mínimo.

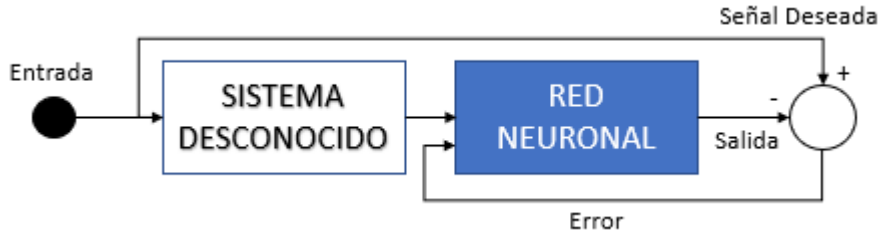


FIG 4.6. Esquema estructura inversa

Este tipo de estructura es clásica para la resolución de problemas de convolución (operador matemático que transforma dos funciones en otra que representa la magnitud donde se superponen una de las anteriores, y la otra trasladada e invertida).

4.5.3. ESTRUCTURA CON RETARDO

Este tipo de estructura tiende a minimizar la diferencia entre la señal deseada y la salida de la red neuronal. Con ello, se intenta modelizan la señal a partir de valores anteriores de esta mediante la inserción de retrasos.

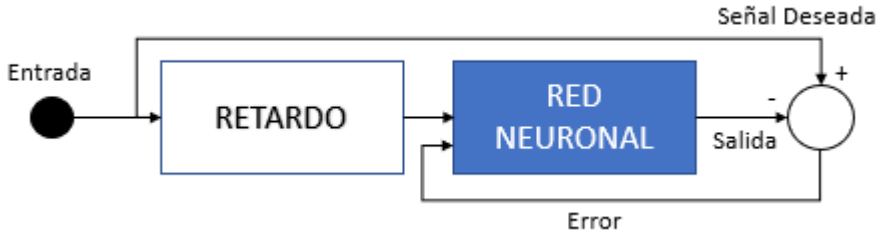


FIG 4.7. Estructura con retardo

Las principales aplicaciones de esta estructura son la predicción de señales, es decir estimar futuras muestras teniendo en cuenta las anteriores, y el control debido a la posibilidad de cambio en parámetros que modifiquen la evolución del sistema.

4.5.4. CANCELADOR DE RUIDO

Existe una última estructura, que corresponde al cancelador activo de ruido. En él, el papel que desarrolla la red neuronal, en ella, su algoritmo interno, trata de minimizar el error cuadrático entre la salida de la red neuronal y la señal de referencia.

Teniendo en cuenta la expresión del error cuadrático este será mínimo cuando la salida de la red neuronal sea igual que el ruido de la señal de referencia (deseada). En este momento, se obtendrá una señal completamente limpia de ruido.

5. REDES NEURONALES APLICADAS A SISTEMAS DINÁMICOS

Una vez tratado el tema teórico de las redes neuronales, se prosigue con su aplicación al proyecto. En este caso, el objetivo de esta parte del proyecto será la inserción de inteligencia artificial basada en redes neuronales, en el control del robot industrial. Control que al tratarse de un sistema dinámico, como es obvio, cambiará con el tiempo.

El objetivo ahora será realizar un modelo mediante el uso de redes neuronales que simule el funcionamiento del actual controlador existente. Para ello inicialmente se procederá a aplicar las redes neuronales a la identificación de sistemas dinámicos. De forma matemática, la identificación de sistemas es la obtención de una aproximación de su comportamiento expresado en forma de una o varias funciones no lineales. Como se explicará próximamente, las redes neuronales, son capaces de realizar esta acción debido a diferentes características que poseen.

Las redes neuronales, están formadas por cuatro elementos básicos como se describe en el apartado 4.2.1. Entre esos elementos se encuentra la función de activación, que dependiendo de cuál sea tiene como objetivo evitar o no la linealidad para a su vez, evadir problemas durante el entrenamiento. Por lo tanto, las redes neuronales pueden ser modelos lineales o no lineales haciéndolas muy útiles para el modelado de sistemas dinámicos. Por otro lado, las redes neuronales responden positivamente antes estas tareas, al tratarse de modelos dinámicos debido a su recurrencia. A su vez, se tratan de modelos paramétricos por el valor de los pesos de las neuronas, y adaptativos, debido a su facilidad de adaptación ante nuevos valores de entrada reentrenando la red.

Funcionalmente, las redes neuronales se basan en la medida del error, que trata de ser reducido durante el entrenamiento al ser aplicado el algoritmo de aprendizaje. A continuación se trata de dar una visión clarificada sobre las distintas partes del modelado de sistemas dinámicos con redes neuronales.

- El modelo, en este caso la red neuronal, es una función que estimas las salidas ante determinadas entradas en un instante de tiempo.
- El modelo, se encuentra definido por parámetros, en el caso de las redes neuronales, esos parámetros corresponden como se ha mencionado con anterioridad a los pesos de las diferentes neuronas.
- El error permite determinar cuan bueno es un modelo. Esta medida en cualquier identificación se suele realizar mediante la norma cuadrática, es decir, se utiliza el error cuadrático medio.
- Por último, para reducir el mencionado error cuadrático medio (MSE), se usan algoritmos iterativos, como es el caso del método de reducción de gradiente, en el cual se basa el algoritmo de Levenberg-Marquardt, el cual explicaremos próximamente al ser el utilizado en este proyecto para el entrenamiento del modelo.

5.1. REDES NEURONALES NARX

Para la realización del modelado de sistemas dinámicos y predicción de series temporales mediante redes neuronales, es necesario seleccionar dentro de la multitud de redes neuronales existentes, un determinado tipo que permita la obtención de un valor de salida para cada valor de la entrada en el mismo instante de tiempo. Existen varias configuraciones de redes neuronales diferentes que desarrollan esta tarea, una de ellas son las redes neuronales autorregresivas no lineales con exógena de entrada externa (NARX), que predicen valores futuros de la seria a partir de los valores anteriores, por lo que son el tipo de red neuronal que mejor que adapta al modelado de sistemas dinámicos.

Como se ha citado, existen otras configuraciones que se desenvuelven correctamente en problemas de series temporales, como las NAR, que se diferencian de las NARX en que únicamente utilizan los valores pasados de la serie para predecir futuros u otras como las redes FTDNN, donde no se conocen los valores pasados. Al ser la estructura más completa, las redes autorregresivas en este caso no lineales son las elegidas para el modelado.

Las redes neuronales NARX, se pueden utilizar para diferentes tipos de aplicaciones. Por un lado, se pueden usar como predictoras, ya que se puede obtener el siguiente valor a la

entrada, mientras que otra aplicación sería el filtrado no lineal, en la cual salida es la entrada libre de ruido.

5.1.1. ESTRUCTURA DE LAS REDES NARX

Las redes neuronales NARX, como se ha explicado anteriormente, combinan la estructura de las redes NAR y de las redes FTDNN (Redes con retraso en el tiempo), es decir, incluyen retrasos tanto en las entradas, como en las respuestas de la propia red, que funcionan a modo de realimentación. En la imagen inferior se observa la estructura de una red neuronal NARX.

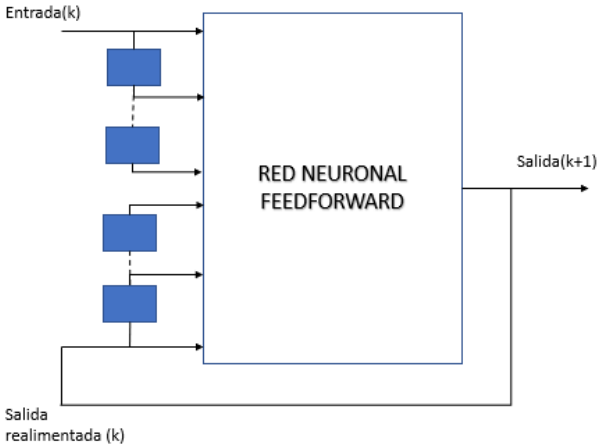


FIG 5.1. Esquema red neuronal NARX

Atendiendo a su estructura, se puede decir que la ecuación que define este modelo de red neuronal es la siguiente:

$$y(t) = f\big(y(t-1), y(t-2), \dots, y(t-n), u(t-1), u(t-2), \dots, u(t-n)\big) \qquad (4)$$

Donde los valores de la salida dependiente $y(t)$ dependen tanto de los valores anteriores de la salida como de los valores anteriores de las entradas exógenas.

Por otro lado, en las redes neuronales NARX, las conexiones suelen ser hacia delante, por lo que la red será feedforward.

La salida de las redes NARX, se puede considerar como estimación del sistema dinámico que se está intentando modelar, por ello, la salida se retroalimenta a la entrada de la red, siendo esta la estructura estándar de una red de este tipo (lazo cerrado), pero al disponerse de esta salida durante el propio entrenamiento, se puede crear otro tipo de estructura, la serie-paralelo o lazo abierto, donde se utiliza la salida real en vez de la estimada. Este tipo de

configuración confiere al entrenamiento de distintas ventajas, como son que la entrada a la red será más precisa, y que la orientación será hacia delante, es decir, puramente feedforward, por lo que una propagación inversa estática puede ser utilizada.

En la imagen inferior se observa una red neuronal NARX con el lazo abierto, mientras que la figura 5.1, representaba el caso de red neuronal NARX con lazo cerrado.

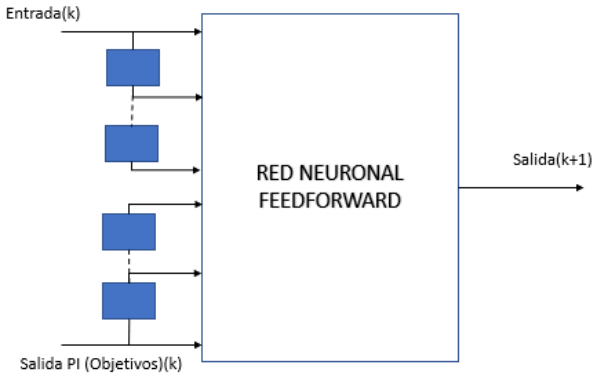
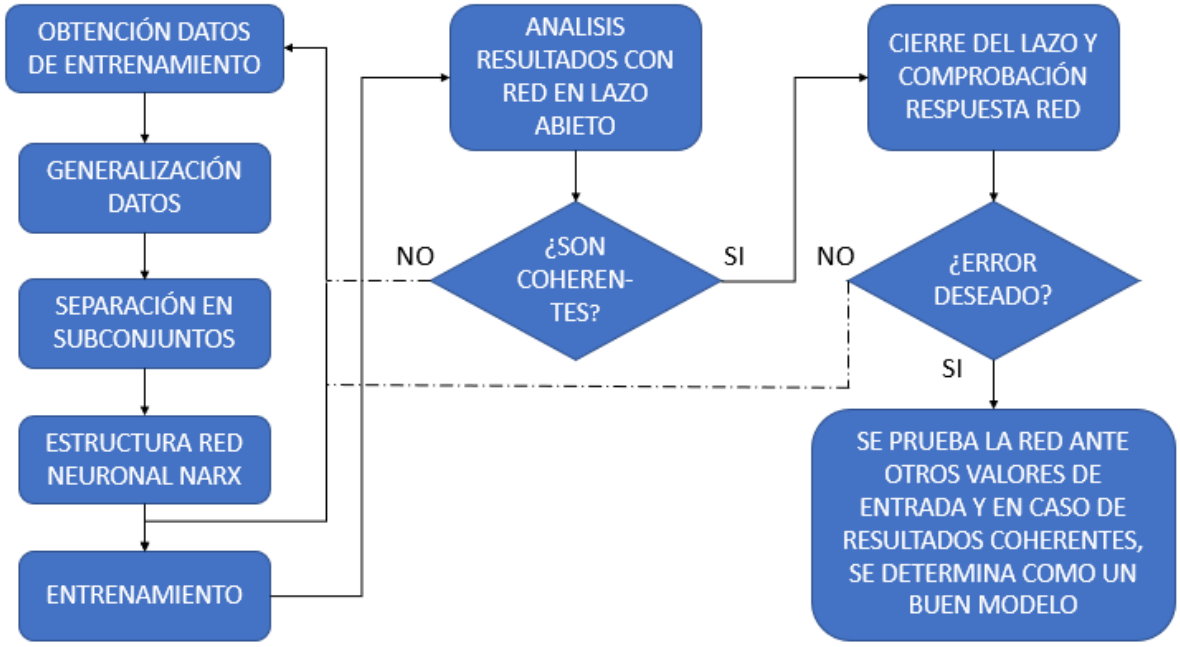


FIG 5.2. Red neuronal NARX lazo abierto

En este tipo de redes para el entrenamiento dinámico, los pesos tienen un doble efecto en la salida de la red ya que, en primer lugar, estos afectan de forma directa a la nueva salida de la red, como ocurre en las redes feedforward. Pero además de este efecto, existe un segundo efecto indirecto debido a los retrasos de la realimentación, ya que estos también serán pesos de la red. Para tener en cuenta este segundo efecto, el método utilizado para el cálculo de los gradientes, es la retro propagación dinámica (backpropagation) que conlleva una mayor complejidad, haciendo consigo el entrenamiento más lento.

5.2. GENERACIÓN DE REDES NARX EN MATLAB

Para la implementación de las redes neuronales en el problema propuesto, se van a utilizar las herramientas que brinda el software MATLAB. Esta aplicación permite el desarrollo y entrenamiento de redes mediante dos métodos, uno para principiantes, guiado con el cual únicamente se deben de seguir los pasos de la interfaz de la herramienta, y un segundo método, la generación de código, por el cual se tratará de obtener el modelo para el problema. A continuación, se muestran los pasos a seguir, si se quiere optar por alcanzar un resultado coherente.



Esquema 3. Flujograma entrenamiento red neuronal

ESTRUCTURA

Desarrollar la arquitectura, será el primer paso a cumplir con la red neuronal con la cual se va a trabajar. Los parámetros modificables en este caso son tanto el número de retrasos en la entrada como en realimentación de la salida, así como el número de capas ocultas y su respectivo número de neuronas por cada una. Las neuronas de la red se interconexionarán de forma feedforward. En la siguiente imagen se puede observar la estructura de una red neuronal compleja, que alejándose del modelo estándar de las redes NARX, está formada por varias capas ocultas.

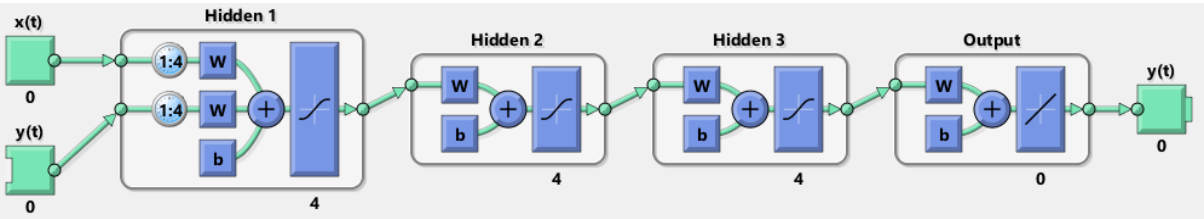


FIG 5.3. Red NARX con tres capas ocultas

Mediante la modificación del número de retrasos, se modifica la ecuación que define el modelo de una red NARX, haciendo que su salida en un instante t, dependa de un mayor número de entradas independientes anteriores en el caso de que se aumente el número de retrasos o por el contrario, de un menor número en el caso que se vean reducidos. A su vez,

la salida en ese instante también se verá afectada por la modificación del número de retardos en la salida realimentada. Teniendo esto en cuenta, al aumentar los retrasos, se estará confiriendo a la red una mayor capacidad de modelado de sistemas dinámicos al depender su salida de un mayor número de valores previos.

En cuanto al número de capas y neuronas utilizadas para el modelo, si se aumenta su número, aumentará consigo la complejidad de la red, obteniendo resultados más exactos a los de la planta a identificar, pero se deberá tener en cuenta uno de los mayores problemas del aprendizaje de las redes neuronales, el sobreajuste, ya que probablemente el valor final de los pesos tienda a ajustarse demasiado, sirviendo es red únicamente, para esos valores de entrada y no otros que difieran.

PREPARACIÓN DE LOS DATOS DE ENTRADA

El primero de los pasos por realizar con el conjunto de datos obtenidos del sistema a identificar, es tratarlos antes de realizar el entrenamiento, de forma que se hace por generalizar tanto los datos de entrada como los objetivos, para posteriormente evitar problemas durante el entrenamiento. El mayor de estos problemas es el sobreajuste durante el entrenamiento, que como se comentó anteriormente, se produce ante la existencia de modelos demasiado complejos, lo que hace que el conjunto de datos a tratar deba ser lo más sencillo posible. Para ello, en primer lugar, se elimina la temporalidad de los datos cambiando las series de tiempo a tantos valores como sean necesarios para completar los estados de retraso.

Una vez tratados los datos, se debe de tener en cuenta que el modelo se está desarrollando para realizar predicciones correctas con datos desconocidos. Para ello, el conjunto de datos es separado en tres volúmenes de datos diferentes.

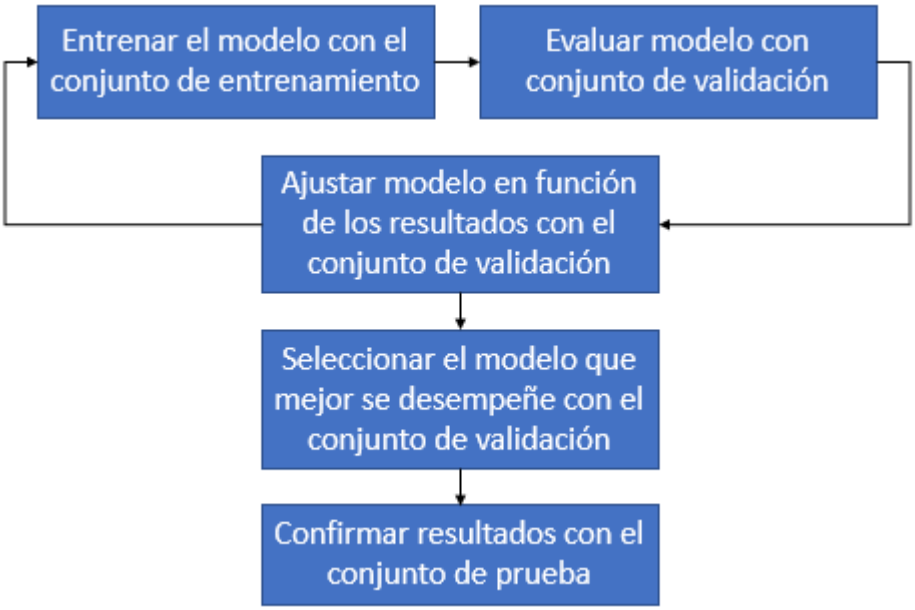
- Conjunto de datos de Entrenamiento: En este subconjunto de datos, se agruparán aquellos datos con los cuales se entrenará el modelo, es decir, se utilizará como un conjunto de datos ya conocido. Por ello, este grupo de datos en la práctica totalidad de los estudios es el de mayor tamaño, agrupando normalmente del 50 al 80% del total. El conjunto de datos de entrenamiento es sumamente importante ya que el modelo se tratará de ajustar a esos datos, si lo hace de forma demasiado estrecha, este se sobre ajustará y no generalizará correctamente sobre nuevos datos.

- Conjunto de datos de Validación: Esta partición de los datos, se lleva acabo para reducir la probabilidad de caer en el sobreajuste. Con este conjunto, se evaluarán los

resultados del conjunto de entrenamiento y se determinará cuando es el momento óptimo para detener el entrenamiento, al coincidir este con el error mínimo obtenido con el conjunto de validación, debido a que una vez sobrepasado este punto del entrenamiento, el error comenzará a aumentar debido al sobreajuste que se produce.

- Conjunto de datos de Prueba: Por último, el conjunto de datos de prueba constará de aquellos datos tratados como desconocidos, con los cuales el modelo una vez entrenado, será probado. El conjunto de datos de prueba debe ser los suficientemente grande debido a que, con su estudio, se determina si el modelo es bueno o no, esto se debe a que los datos de prueba son un indicador útil del buen rendimiento del modelo con nuevos datos.

Por lo tanto, teniendo en cuenta el método de validación más simple, conocido como “hold-out”, el flujo de trabajo de datos será como el del siguiente diagrama:



Esquema 4. Diagrama flujo del entrenamiento, GoogleDevelopers [25]

Con esta partición de los datos, se deberá tener en cuenta el rendimiento del modelo con el conjunto de datos de validación, seleccionando finalmente el modelo que mejores resultados muestre con el conjunto de validación.

ENTRENAMIENTO

En tercer lugar, se realizará el propio entrenamiento de la red neuronal. En la herramienta utilizada (MATLAB), existen de manera predeterminada tres algoritmos con los cuales

entrenar el modelo. Es recomendable utilizar alguno de ellos, debido a su optimización. En este ejemplo, la optimización de Levenberg-Marquardt será utilizada. Este tipo de entrenamiento actualiza los pesos de las entradas a las neuronas según su propio método.

ALGORTIMO LEVENBERG-MARQUARD

Fue diseñado de manera similar a los métodos de Gauss-Newton. Cuando la función de desempeño tiene la forma de una suma de cuadrados (Se suele dar en las redes feedforward), la matriz hessiana se puede aproximar a:

$$H = J^T J \qquad (5)$$

Y el gradiente como:

$$g = J^T e \qquad (6)$$

Donde J es la matriz jacobiana que contiene la primera derivada de los errores de la red respecto de los pesos. Por otro lado, e es el vector que contiene los errores. El método, utiliza esta aproximación a la matriz Hessiana en la siguiente versión del método de Newton:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \qquad (7)$$

Aquí cuando el escalar, μ es igual a cero, la expresión resultante es la misma que la del método de Newton usando la aproximación de la matriz Hessiana. En el caso de que μ sea grande, se convierte en un descenso de gradiente de tamaño de paso pequeño. El objetivo es llegar a la forma del método de Newton lo más rápido posible, debido a que este el más preciso y alcanza mejor el error mínimo. Por lo tanto, la función de rendimiento es siempre reducida por cada iteración del algoritmo.

ENTRENAMIENTO

Una vez explicado el funcionamiento del método de aprendizaje utilizado, se procede a entrenar el modelo con los conjuntos de datos utilizados.

Por cada iteración, el gradiente descenderá hasta que sea mínimo, y con ello el error cuadrático medio de los datos de validación sea también el menor. En la figura inferior, se puede observar la gráfica de la evolución del error cuadrático medio de los tres conjuntos de entrenamiento. En el caso de haber realizado un proceso adecuado hasta este punto, se debería observar una estrecha relación entre el error cuadrático medio (MSE) del conjunto de validación, y el propio del conjunto de prueba, mientras que, por otro lado, la gráfica debe arrojar un error del conjunto de entrenamiento inferior, debido a que esos son los datos con

los cuales se ha entrenado el modelo. En la figura 7.26 se observará un correcto comportamiento del error cuadrático medio.

VALIDACIÓN DE LOS RESULTADOS DEL ENTRENAMIENTO

Una vez entrenado el modelo, se procede al análisis tanto del entrenamiento, como de los resultados obtenidos, para este propósito la herramienta de redes neuronales de MATLAB proporciona un conjunto de gráficos, que harán más fácil el estudio de los resultados, para posteriormente tomar la decisión sobre si el modelo es o no el adecuado.

El primero de los métodos para estudiar, se trata del análisis de la regresión linear. Esta regresión, se obtiene entre la respuesta de la red y los objetivos que se marcaron para la respuesta. En las gráficas ploteadas por la aplicación, se pueden observar tres elementos principales: Los puntos de la salida obtenidos frente a los objetivos, el mejor ajuste lineal alcanzado, y el ajuste perfecto que corresponde a una recta de pendiente unidad.

Para determinar si las respuestas van en la buena dirección, se puede extraer la pendiente y la intersección en y de la mejor relación lineal. Si el resultado de estos valores fuese, uno y cero respectivamente, se corroborará que el ajuste es perfecto.

En algunos entrenamientos, se obtienen datos que distan en gran medida de la línea de pendiente unitaria, estos datos corresponderán a aquellos con un error mayor en la gráfica que muestra la respuesta temporal obtenida por el modelo.

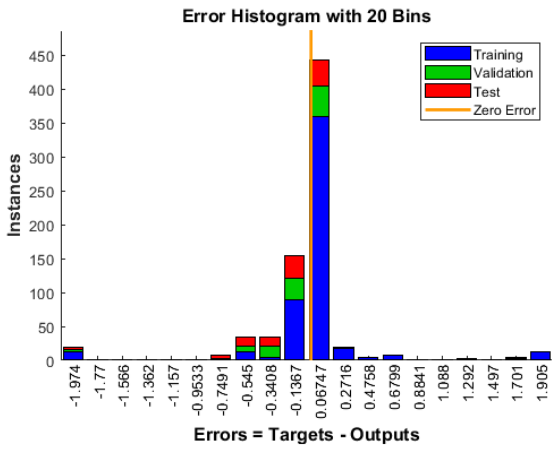


FIG 5.5. Ejemplo histograma

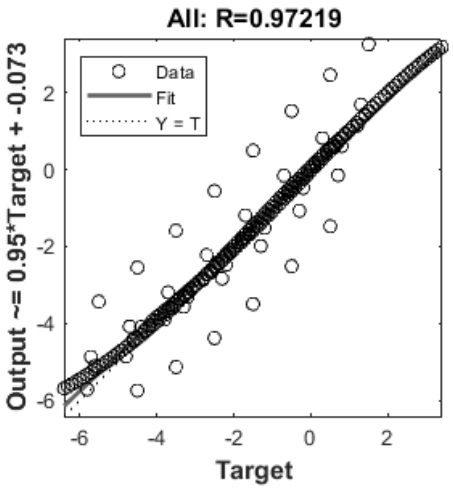


FIG 5.4. Ejemplo regresión lineal

Por otro lado, los errores mayores del histograma, es decir los que se distancian más del error nulo, también se corresponden con los puntos de datos con una menor regresión lineal.

Además, la herramienta, proporciona la información de las regresiones lineales de los diferentes conjuntos de datos, pudiendo, evaluar cuan buenas han sido las respuestas con cada conjunto por separado.

Los errores de estos conjuntos de datos también pueden ser evaluados mediante un histograma, en el cual se muestran diferenciados por colores los errores de cada uno de los conjuntos. En este histograma, a menudo se observan valores en los extremos, que como se comentó anteriormente, se encuentran relacionados con los datos lejanos a la línea de regresión ideal en la gráfica de regresión. En el caso de aparición de errores del conjunto de entrenamiento alejados de la línea de cero errores, estos probablemente corresponderán a fallos cometidos durante el entrenamiento ante cambios bruscos en la entrada. Por ejemplo, en las figuras de debajo, se observa un conjunto de errores de entrenamiento alejados del resto, los cuales como se aprecia en la respuesta temporal corresponden a las mayores diferencias entre la salida obtenida y los objetivos, cuando estos tienen cambios bruscos.

Finalmente, existe otro conjunto de tablas que permite conocer si un entrenamiento ha sido correcto, estas gráficas corresponden a correlaciones.

Existen dos diferentes. En la primera de ellas, se puede comprobar la correlación entre el error y el tiempo. En el centro de esta gráfica, se aprecia un valor de correlación correspondiente al error cuadrático medio del entrenamiento, que en caso de que este obtenga buenos resultados, la diferencia de tamaño de esta línea con el resto debe ser importante, encontrándose la mayor parte del resto dentro de los límites de confianza como se observa en el ejemplo inferior.

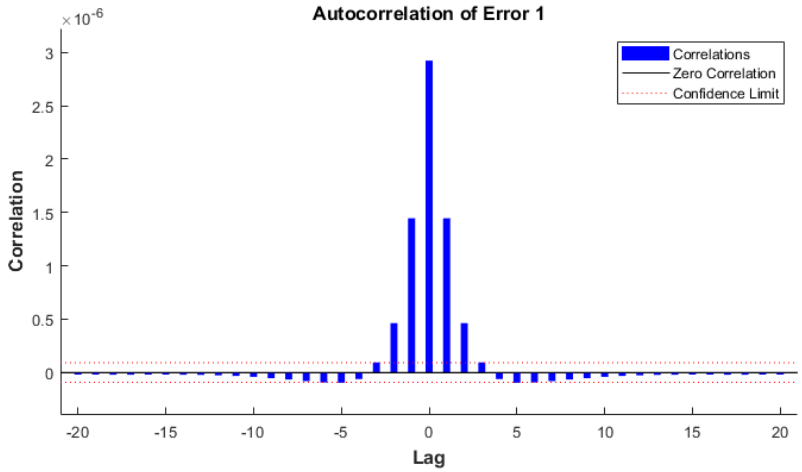


FIG 5.6. Ejemplo autocorrelación error [18]

En último lugar, la segunda de las tablas de correlación muestra la relación existente entre el error y las entradas, con diversos grados de retraso. En esta tabla, si los resultados del entrenamiento son correctos, la mayor parte de los datos se deben encontrar en los límites de confianza o cercanos a ellos, sin excepción alguna.

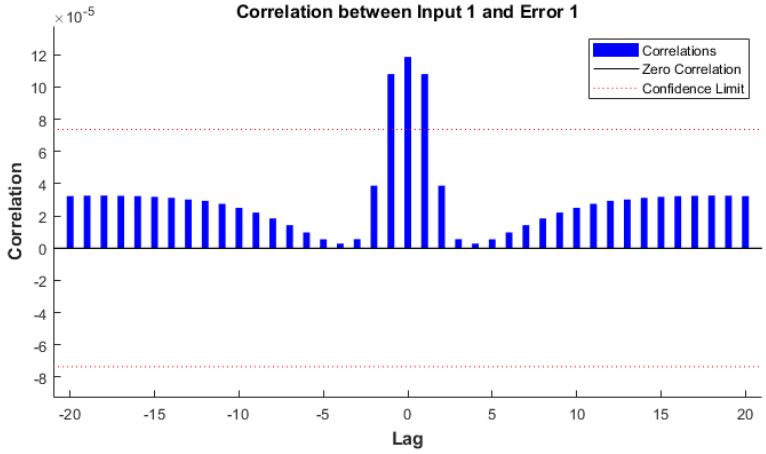


FIG 5.7. Ejemplo correlación entradas-error [18]

Una vez explicado cómo desarrollar el entrenamiento de una red neuronal para modelado de sistemas dinámicos y como analizar sus resultados, se procederán a realizar los diferentes casos de estudio del proyecto.

6. SEÑALES PRBS

Al tratar el estudio, del modelado de sistemas dinámicos mediante redes neuronales, estos variarán el valor de su salida, dependiendo de las entradas introducidas en ellos. Por ello, para llevar acabo la identificación del sistema, se debe utilizar un tipo de señal de entrada que permita una estimulación del sistema lo más completa posible. Por ello, se utilizarán señales PRBS

Una señal PRBS, es una señal binaria, entre dos valores, que además es periódica y determinista. Estas señales, como se explicará a continuación, tienen propiedades similares al ruido blanco, por lo que son idóneas para ser usadas como señal de prueba de entrada de un sistema, con el objetivo de obtener su respuesta a los impulsos.

6.1. CARACTERISTICAS DE UNA SEÑAL PRBS

Como se ha explicado anteriormente, una señal PRBS (Pseudo Random Binary Sequence), como su propio nombre indica, es una señal binaria, es decir, solo puede adoptar dos valores. Por otro lado, es una señal periódica, debido a que se pueden generar a partir del resto de la división de el vector de entradas pasadas

$$[u(n-1), \dots, u(n-k)] \qquad (8)$$

Por el coeficiente a_i dividido entre dos:

$$u(n) = resto\left(\frac{a_1 u(n-1) + \cdots + a_k u(n-k)}{2}\right) \qquad (9)$$

El vector de entradas pasadas únicamente podrá tomar un número determinado de valores, que corresponde con 2^k valores, siendo k el orden, por lo que se limitará el periodo máximo de la secuencia. Los coeficientes a_i, dan secuencias de periodo máximo, a las que se llama PRBS de máxima longitud, y que, bajo estas circunstancias, hace que se comporte de forma similar al ruido blanco.

Las señales PRBS se generan de forma hardware mediante registros de desplazamiento con realimentación de niveles lógicos mediante puertas lógicas OR exclusivas. En el caso de necesidad de retardos, se es posible introducir nuevos, mediante el uso de nuevo de otras puertas lógicas OR exclusivas. Por otro lado, uno de los niveles de estas señales puede asumir intervalos de tiempo, y las simulaciones realizadas podrán ser repetidas tantas veces se desee, sin que la señal de entrada, en este caso PRBS, sea modificada, debido a que como su nombre indica, es pseudoaleatoria por lo que los cambios entre ambos valores se producen de forma predeterminada mediante la modificación de la estructura del hardware de generación.

6.2. GENERACIÓN DE UNA SEÑAL PRBS

Existen dos métodos de generación de este tipo de señales, mediante Software o por el contrario mediante registros de desplazamiento, es decir Hardware.

6.2.1. GENERACIÓN MEDIANTE SOFTWARE

Es el método de generación de esta señal más sencillo, y el cual, por comodidad, ha sido utilizado para estimular durante el entrenamiento de los modelos. Para el desarrollo de las

señales PRBS durante el proyecto, la herramienta MATLAB dedicada a la identificación ha sido puesta a prueba.

Más concretamente, para generar la señal, se debe utilizar la función "*idinput*" que se encuentra dentro de la herramienta de identificación, y cuya forma es la siguiente.

$$idinput(numero\ de\ muestras, tipo\ de\ señal, banda, rango)$$

- El número de muestras será el número de valores escogidos de la secuencia completa de longitud 127 (PRBS de longitud máxima). Este número de muestras puede ser modificado, variando los parámetros del periodo, el número de periodos, es decir, cuantas veces se va a repetir la misma señal, y el número de canales en el caso de que se quieran generar varias señales PRBS con las mismas dimensiones. Esto será útil durante el proyecto, para poner a prueba los resultados ante diferentes entradas.

- El segundo de los parámetros a modificar será el tipo de señal a generar. Existen multitud de tipos diferentes, pero para la generación de este tipo de señales, se debe introducir el comando 'prbs'.

Por último, los valores restantes corresponderán a aquellos que tienen una mayor influencia en la forma de las señales PRBS.

- La banda, es el valor que la señal puede cambiar en cada paso de tiempo. Para introducirlo en MATLAB, se debe hacer de la siguiente forma:

  $$band = [0\ B]$$

  Siendo B la inversa del periodo del requerido del reloj.

- El rango, son los valores entre los cuales va a variar la señal. Para la determinación de estos valores, se ha tenido en cuenta la amplitud de las señales de entrada al bloque de control PI.

6.2.2. GENERACIÓN MEDIANTE HARDWARE

La segunda forma de generar la señal PRBS, es la cual se va a intentar aplicar para excitar el sistema. Para realizar este método de generación, es necesario utilizar registros de desplazamiento con retroalimentación lineal (LFSR).

A continuación, se va a detallar la estructura a desarrollar para la generación de señales PRBS en SIMULINK.

En primer lugar, los registros de desplazamiento se organizan en serie. En este caso, se van a utilizar flip-flops tipo D. Este tipo de flip-flop tiene cuatro entradas principales, que corresponden al reloj clk, la entrada D, y dos salidas Q y !Q opuestas.

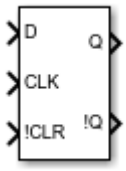


FIG 6.1. Flip-Flop D en SIMULINK

El funcionamiento de este elemento físico es el siguiente. El valor llega a la entrada D hasta que se produce la subida de nivel en el reloj, una vez ocurre esto, el valor pasa de la entrada a la salida Q y su valor opuesto a !Q. Entonces, no volverá a repetirse este proceso, hasta que el reloj haya regresado a su estado de reposo y vuelto de nuevo a su estado alto cortando el estado de memoria.

En segundo lugar, se deben generar las realimentaciones, que configurarán la forma de la señal. Estas se producen desde etapas intermedias y últimas, hasta la primera entrada mediante el uso de puertas lógica de tipo OR-Exclusiva (XOR). Este tipo de puerta lógica transmite la señal, únicamente si una de sus entradas tiene el valor alto. En la siguiente tabla, se observa el funcionamiento de una puerta XOR.

FIG 6.2. XOR

Entradas		Salidas
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 4. Lógica puerta XOR

Por último, es necesario introducir en el registro de estos el caso en el que todos los valores sean ceros, en este caso, para inicializar el sistema debido a que como se explicará a continuación, todos los valores iniciales de las salidas de los flip-flops son ceros. Esta inicialización, se realiza utilizando las entradas negadas, que tendrán unos en el momento de la inicialización, que serán introducidos en un operador AND que multiplica todas las entradas. El funcionamiento de una puerta lógica AND será en siguiente:

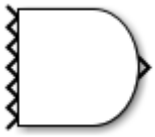


FIG 6.3. AND

Entradas		Salidas
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 5. Lógica puerta AND

Al inicializar todos los flip-flops a uno, se obtiene en la salida un uno, que es introducido en un último operador XOR, cuya otra entrada en el instante inicial es cero, por lo que la señal se transfiere y el generador se inicializa. En la imagen inferior se muestra la arquitectura final del registro de desplazamiento con la retroalimentación y la inicialización.

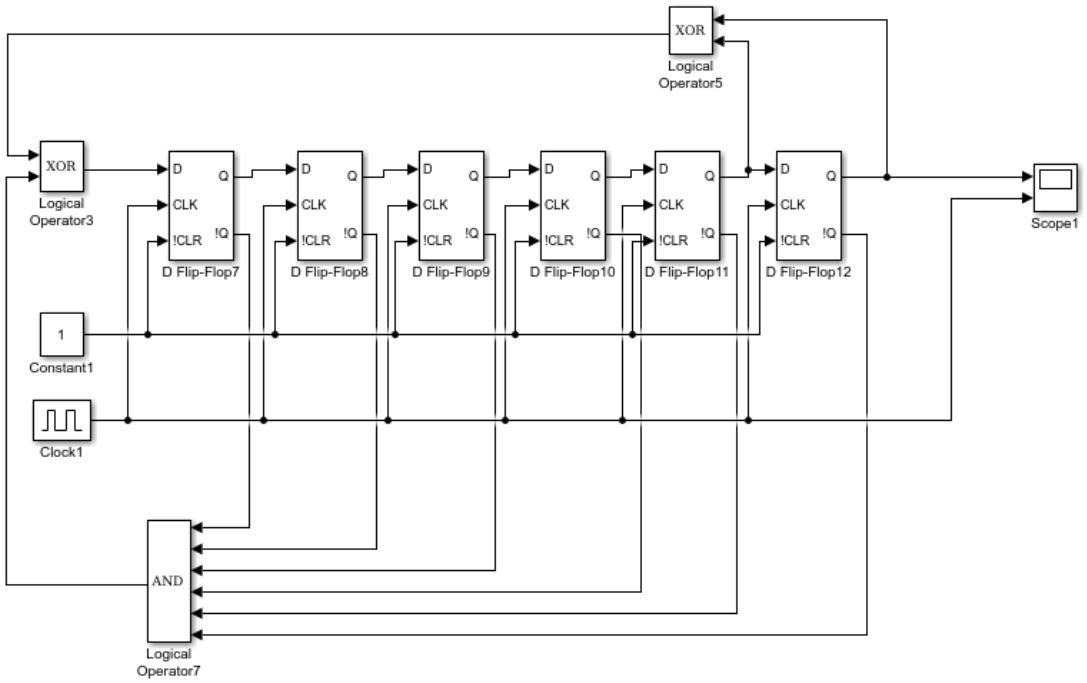


FIG 6.4. Generador señal PRBS SIMULINK

A continuación, se muestra un ejemplo de simulación donde se conseguirá la señal PRBS con la cual se va a trabajar a la hora de entrenar las redes neuronales.
En primer lugar, se muestra el estado de las señales internas del circuito en el momento de inicializarlo. Aquí se puede observar lo comentado anteriormente sobre la inicialización.

Finalmente, la salida del Flip-Flop D, que es la que se utilizará para ser introducida en las funciones de los controladores, será la siguiente:

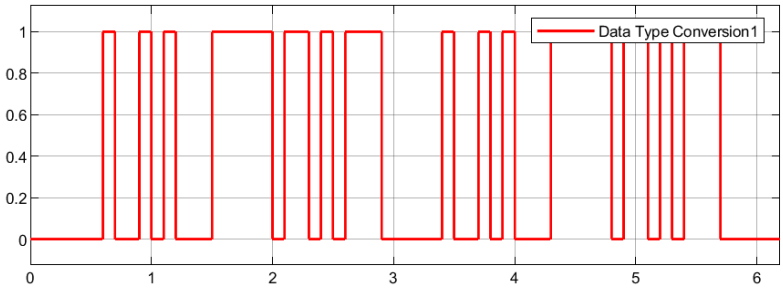


FIG 6.5. Señal PRBS generada

7. DESARROLLO MODELOS DEL REGULADOR PI DEL CONTROL CINEMÁTICO

7.1. CASO DE ESTUDIO 1, MODELADO REGULADOR PROPORCIONAL-INTEGRAL

Una vez se ha explicado el uso de las redes neuronales, principalmente el tipo que se va a utilizar a continuación, NARX, se procede a el uso de esta estructura para el modelado del sistema. Durante el siguiente apartado se tratará de conseguir una red neuronal tenga un comportamiento similar al del regulador proporcional integral de la primera de las uniones del robot. Este regulador, se encontrará también en el resto de las articulaciones del robot al ser general y no haberse personalizado cuando se hizo el control dinámico.

El regulador PI ya comentado durante el apartado 3.3.5. es el siguiente:

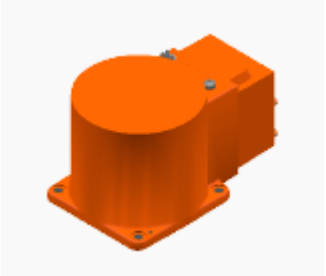


FIG 7.1. Base del robot

$$PI = \frac{0{,}1s + 0{,}001}{s}$$

FIG 7.2. Cuerpo del robot

Si se aplica una entrada escalón unitario al regulador, se obtendrá la siguiente respuesta.

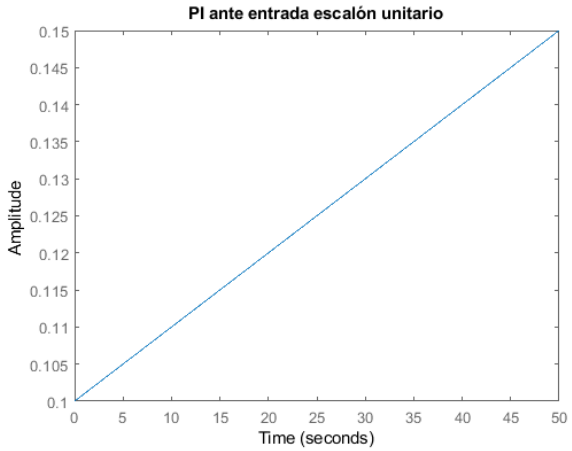


FIG 7.3. Respuesta PI ante escalón unitario

El objetivo final, será sustituir el bloque PI actual del gemelo virtual por un nuevo bloque que contenga una red neuronal NARX con los pesos configurados para obtener las mismas respuestas que obtendría el anterior controlador, por lo que la nueva red neuronal sería un modelo del anterior control.

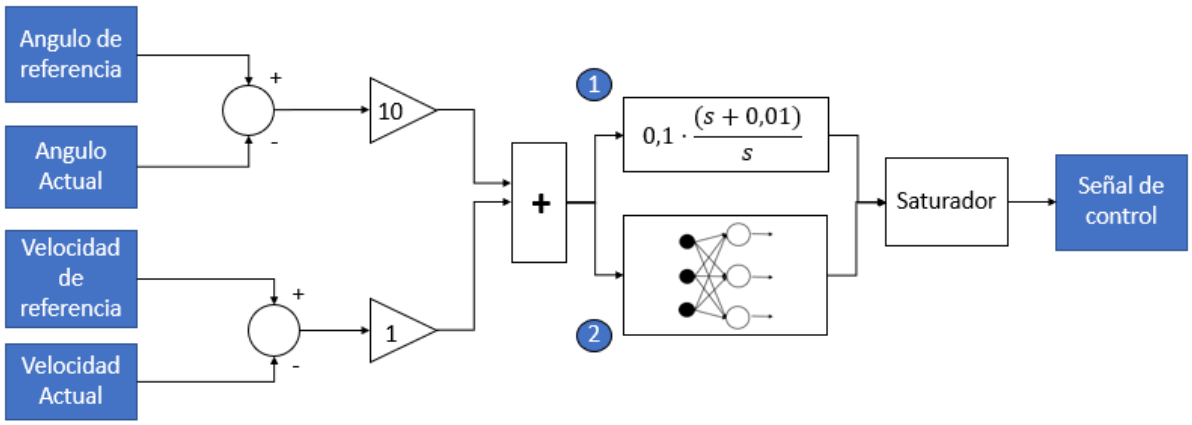


FIG 7.4. Esquema control con PI y con Red neuronal

La opción 1 de la figura 7.4. correspondería a el control antiguo con el regulador PI, mientras que la segunda de las opciones corresponde al controlador con el regulador ya sustituido por la red neuronal.

Con el entorno de trabajo expuesto, se procede a comentar los pasos a seguir para tratar de alcanzar un resultado idóneo.

7.1.1. OBTENCIÓN DE DATOS

En primer lugar, se determinará el sistema a identificar, es decir el regulador PID sobre el cual se quiere obtener su respuesta ante una determinada entrada, en este caso PRBS con el objetivo de conocer la respuesta del regulador al verse sometido durante diferentes tiempos a un valor de entrada.

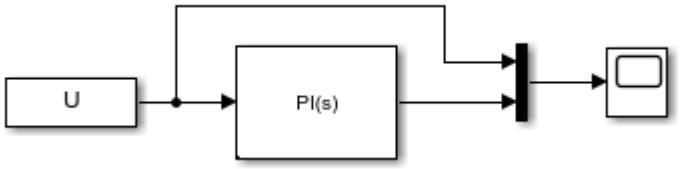


FIG 7.5. Sistema inicial con PI

El sistema, está configurado con una entrada U que se corresponde con la señal PRBS configurada próximamente. Para generar esta señal, no es necesario atender en gran medida al tiempo de muestreo, debido a que al tratarse de un integrador, la respuesta es una recta con la misma pendiente a lo largo del tiempo. Por ello el periodo escogido es el siguiente.

$$T_s = 1s$$

Y la configuración de la señal PRBS será:

$$Periodo = 150$$
$$Numero\ de\ Periodos = 5$$
$$Banda = 0.1s^{-1}$$
$$Amplitud = [-2,2]$$

Además se generarán dos señales con la misma configuración para comprobar una vez obtenido el modelo en la red neuronal la respuesta ante una entrada diferente, confirmando con ello un entrenamiento correcto. La primera de las señales generadas, con la cual se entrenará la red tendrá la siguiente forma.

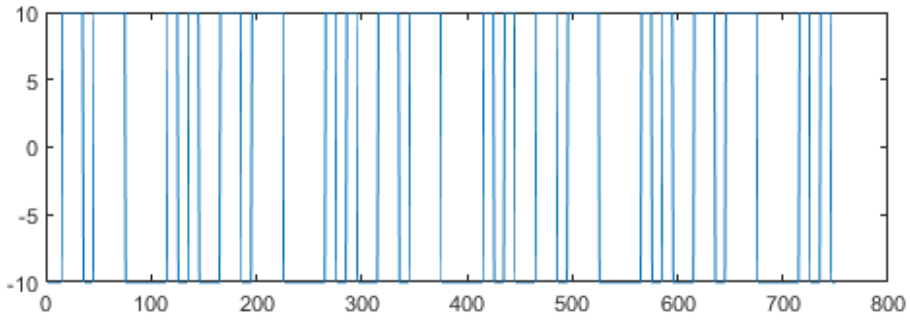


FIG 7.6. Señal entrada PRBS

Una vez configurada la entrada, únicamente será necesario configurar el tiempo de simulación para posteriormente estimular el regulador con esa entrada, con el objetivo de conseguir los datos que se quiere que replique la red neuronal. La respuesta del regulador ante esa entrada tendrá la siguiente forma.

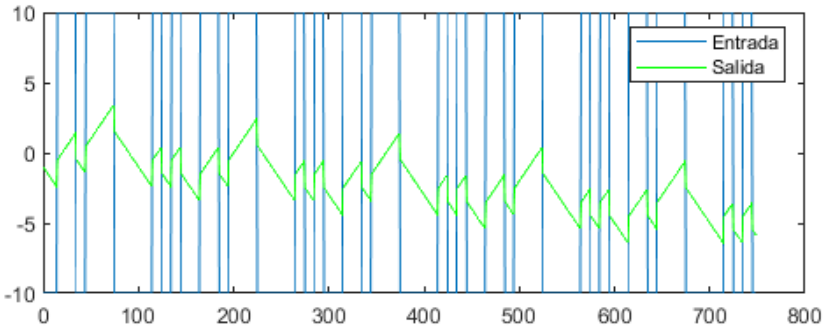


FIG 7.7. Entrada y su respectiva salida

Como se puede observar, la respuesta conseguida tiene las características clásicas de una acción integral. Cuando la entrada adquiere su valor positivo el regulador transforma esa señal constante en una rampa de pendiente positiva mientras que cuando el valor es negativo, la salida tiene la misma pendiente, pero negativa.

Una vez se tienen los datos, será necesario tratarlos, eliminando su temporalidad, para que se pueda hacer frente a los retrasos iniciales. Por último, los datos son divididos en los tres conjuntos citados en el apartado 5.2, entrenamiento, validación y prueba. Esta separación es realizada en tres bloques diferentes con datos seguidos en el tiempo, a diferencia de la opción predeterminada de la herramienta, que hace la separación de forma aleatoria. Este cambio, es realizado, con el objetivo final de que durante el entrenamiento, no existan saltos entre muestras de entrenamiento. En la imagen del siguiente apartado donde se muestra la respuesta temporal conseguida post-entrenamiento, se puede comprobar esta división de datos.

7.1.2. ESTRUCTURA RED NEURONAL Y ENTRENAMIENTO

Una vez obtenido el conjunto de datos, se configura la red neuronal que deberá imitar a esos valores. Como se comentó durante la explicación teórica de las redes NARX, estás tienen retrasos tanto en la propia entrada de la red, como en la entrada de realimentación. En este caso, se utilizará una red con cuatro retrasos en ambas entradas. A su vez, únicamente existirá una única capa oculta de diez neuronas, por lo que las interconexiones serán bastante simples y por consiguiente la red también lo será. En casos más complejos, esta red neuronal no sería suficiente para completar el problema, y en esos casos se añadirán un mayor número de capas ocultas, así como más neuronas por cada capa, pero mostrando atención a las

respuestas obtenidas, debido a que un aumento desmedido, provocará un sobreajuste en el entrenamiento, haciendo que el modelo solo responda correctamente ante esa entrada.

La red neuronal configurada para este ejemplo será la siguiente:

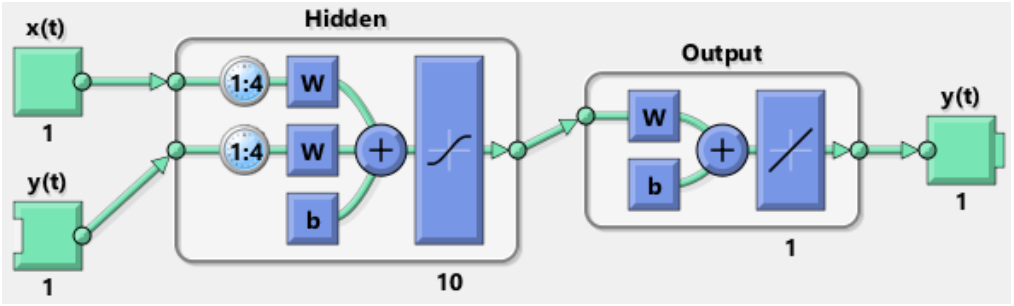


FIG 7.8. Red neuronal NARX con lazo abierto

Una vez generada la red neuronal con la que realizar el modelado, se procede al entrenamiento, mediante el método de Levenberg-Marquardt que utilizará únicamente nueve iteraciones para obtener el error cuadrático medio del conjunto de validación más bajo.

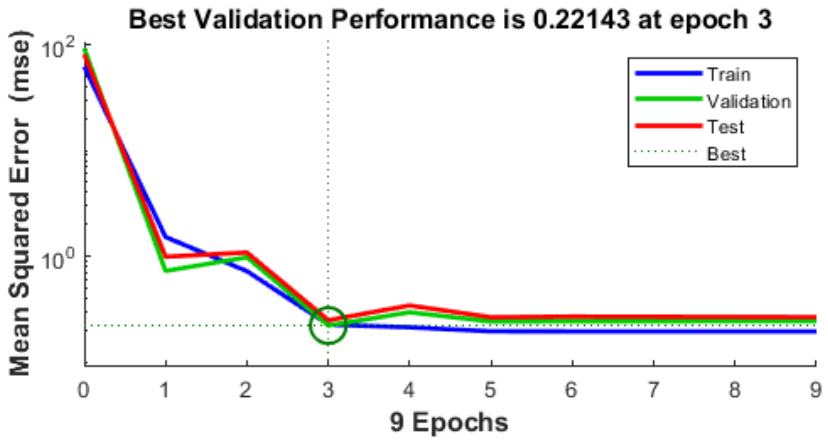


FIG 7.9. Error cuadrático medio de caja conjunto en cada conjunto de datos

En la gráfica superior, se observa como los errores de los conjuntos de validación y prueba siguen tendencias similares alcanzando finalmente un error MSE algo superior al del conjunto de entrenamiento.

En el histograma que muestra los errores de los conjuntos, FIG 7.8. se observa que en la línea central, es decir cuando el error es cero, la proporción de datos del conjunto de entrenamiento es mayor, pero a medida que estudiamos datos más alejados del error nulo, aumenta la proporción de datos de validación y prueba.

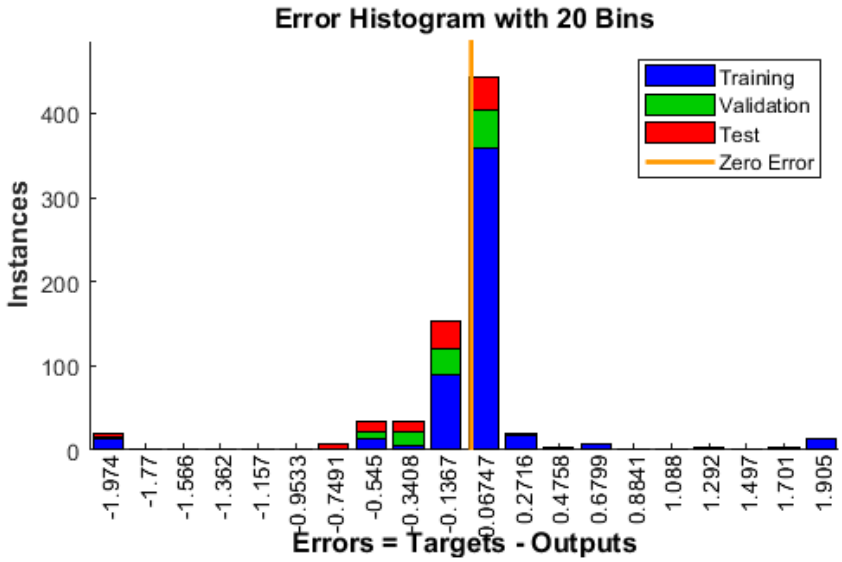


FIG 7.10. Histograma error

Otra forma de observar los datos con errores significativos, son las regresiones lineales de cada conjunto. Aquí se podrá observar como aquellos datos con mayor diferencia entre la salida obtenida y los objetivos, se encuentran distantes del ajuste lineal perfecto.

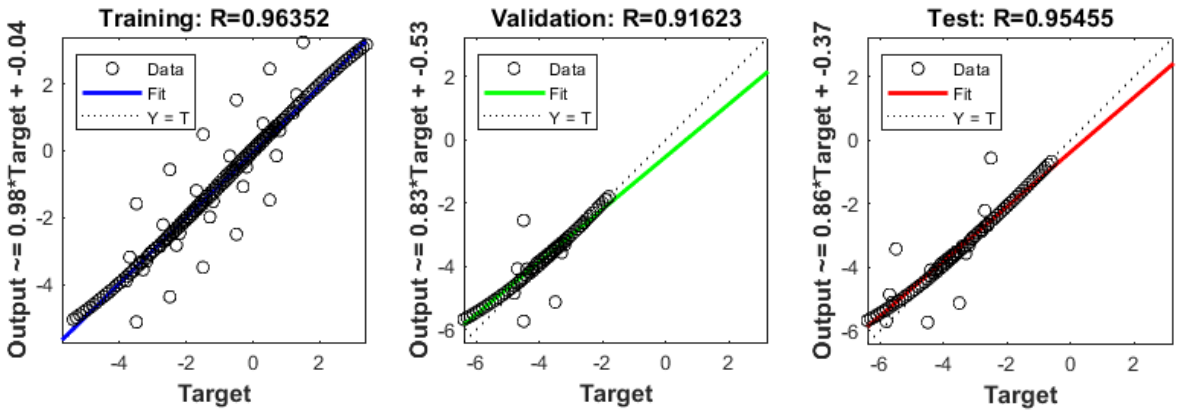


FIG 7.11. Regresiones lineales

El mayor volumen de datos erróneos se da en el conjunto de entrenamiento, pero al estar formado este por el 70% de los datos, las diferencias no serán tan significativas, y la regresión continuará teniendo valores mayores a los de los conjuntos de validación y prueba.

$$R_{Entrenamiento} = 0{,}9635$$

$$R_{Validación} = 0{,}9162$$

$$R_{Prueba} = 0{,}9545$$

$$R_{Total} = 0{,}97219$$

A continuación se evaluará la respuesta temporal obtenida, donde se podrán observar aquellos comentarios realizados mediante el uso del resto de gráficas.

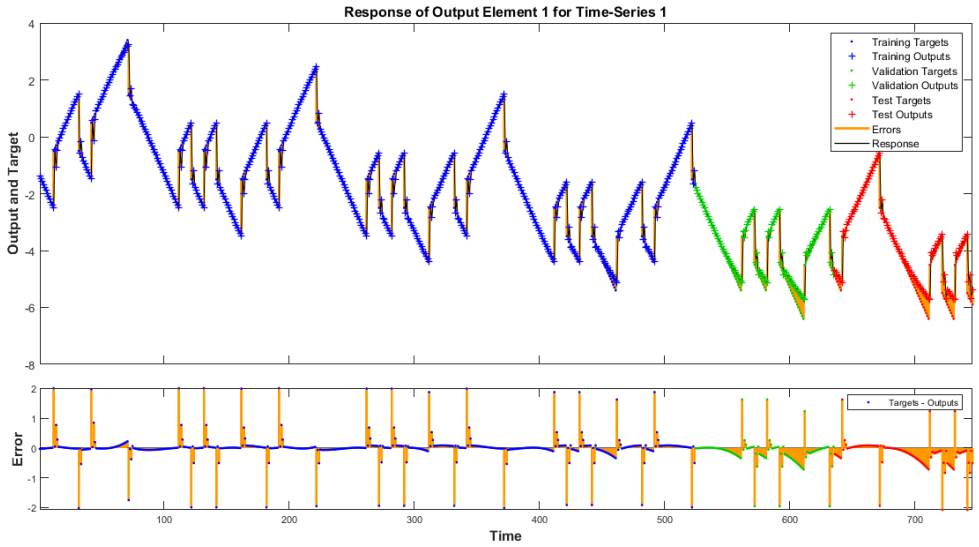
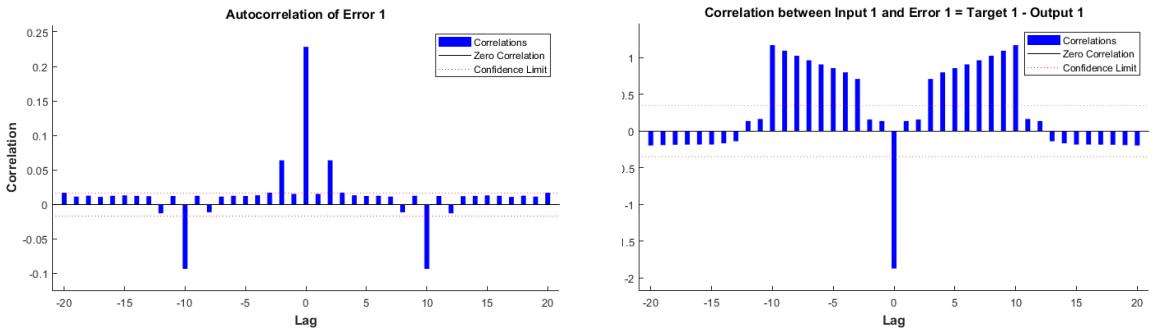


FIG 7.12. Respuesta temporal obtenida y error respecto objetivo

Entre las características de esta respuesta se encuentra, la diferencia existente entre la salida del PI original y la respuesta obtenida por la red neuronal cuando existe un cambio brusco en los objetivos. Por otro lado, es observable en la gráfica correspondiente a los errores, lo comentado en el histograma, donde se observa que las diferencias entre los valores de los objetivos cuando la entrada PRBS se encuentra en su nivel inferior, con respecto a la salida obtenida en los conjuntos de validación y prueba son mayores.

Por último, se muestran las gráficas de correlación de los errores con respecto del tiempo y las entradas.



a) b)

FIG 7.13. Autocorrelaciones. a) error-tiempo b) error-entrada

En la gráfica de la izquierda, correlación error tiempo, se puede ver como el entrenamiento es medianamente correcto, debido a que la correlación central (MSE) es la mayor, mientras

que la mayor parte del resto se encuentran dentro de los límites de confianza. A pesar de ello, existen otras dos que prácticamente doblan esos límites. Por otro lado, en la gráfica situada a la derecha, hay una gran cantidad de correlaciones entre las entradas y el error, que superan estos límites de seguridad, por lo que el entrenamiento en este aspecto no tendrá un rendimiento óptimo. Este problema, se verá reflejado a continuación a la hora de confirmar el buen funcionamiento del modelo red neuronal.

7.1.3. COMPROBACIÓN BUEN FUNCIONAMIENTO MODELO OBTENIDO

Con los resultados de la simulación en la mano, es hora de comprobar si estos son extrapolables a otras entradas, y por lo tanto la red neuronal es un buen modelo del regulador PI del controlador.

En primer lugar, se cambiará la estructura de la red neuronal a lazo cerrado, de forma que a partir de las entradas objetivo, usará sus propias predicciones, para calcular las siguientes. Si se realiza este cambio estructural, la nueva forma de la red neuronal será la siguiente.

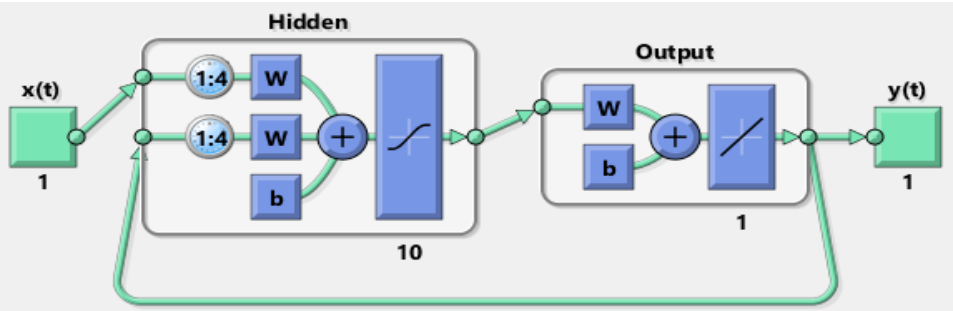


FIG 7.14. Red neuronal lazo cerrado

Con esta forma de red, se conseguirán resultados que difieren de la respuesta temporal de la figura 7.12, y que dependerán de cuan bueno haya sido el entrenamiento. Ahora, los valores objetivo serán desconocidos y la respuesta temporal obtenida será:

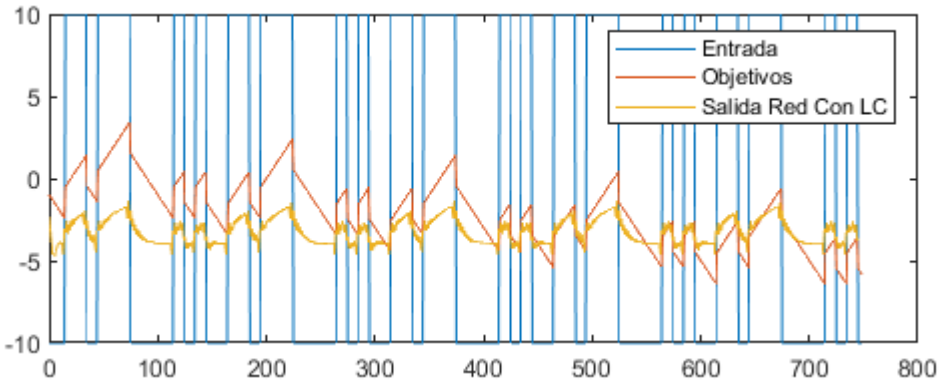


FIG 7.15. Comparación objetivos con respuesta temporal lazo cerrado

Como se observa, la diferencia entre ambas salidas, la del regulador y la de la red neuronal, ha aumentado. Esto es observable también en el error cuadrático medio, que ha aumentado exponencialmente su valor.

	Lazo Abierto	**Lazo Cerrado**
MSE	0,2285	3,8682

Tabla 6. MSE Caso de estudio 1

Relacionando esta respuesta con la obtenida con el entrenamiento 7.12, se observa que en este caso la respuesta es idéntica por cada repetición de la entrada y no va acumulando error como si ocurría durante el entrenamiento. Esto se debe a que en este caso la salida en cada instante de tiempo depende de las anteriores salidas y por contrario en lazo abierto dependía de la respuesta obtenida por el regulador PI.

Finalmente, el nuevo sistema con la red neuronal colocada en paralelo con el regulador para comparar las salidas de ambos es generado, y se comparan sus respuestas ante la entrada PRBS.

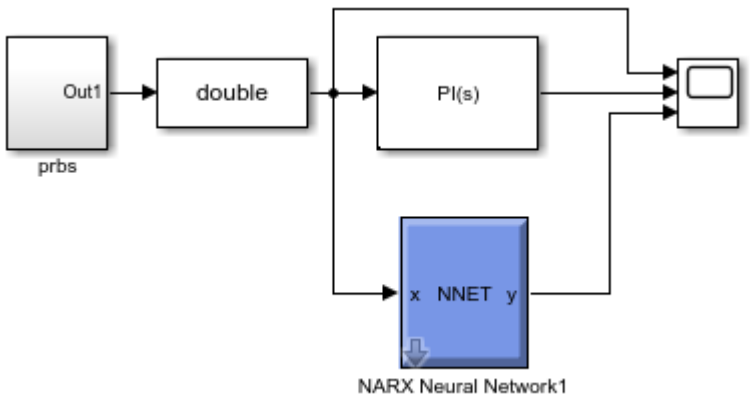


FIG 7.16. Nuevo sistema para comprobar funcionamiento red neuronal

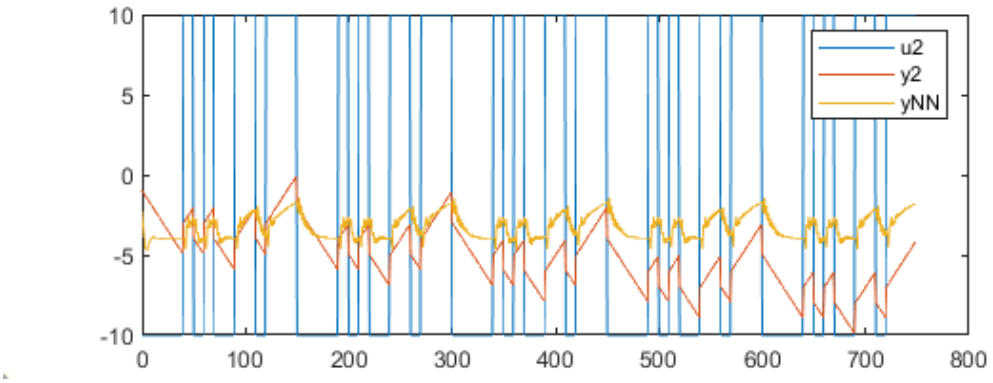


FIG 7.17. Respuestas ante nueva entrada PRBS

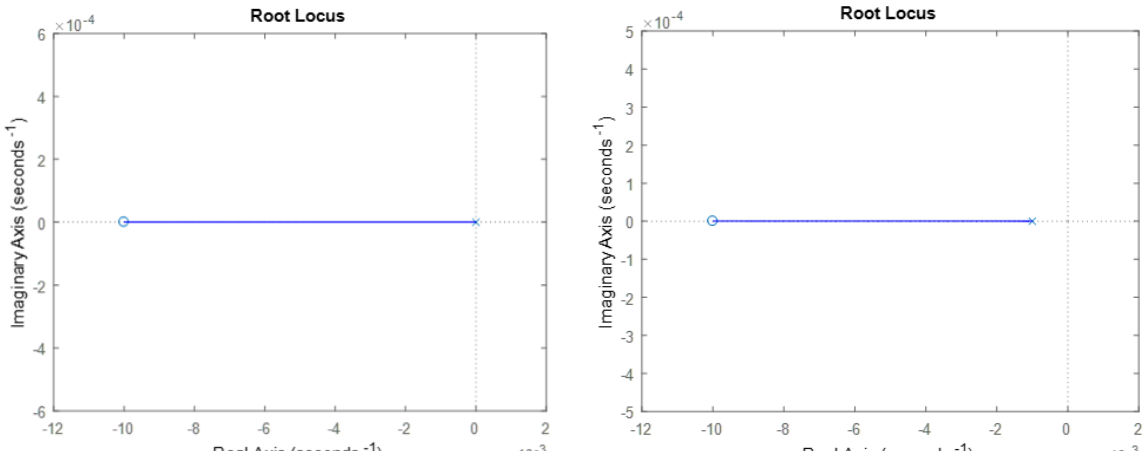
En este caso, al igual que la anterior, ambas salidas distan, por lo que el modelo obtenido con los pesos de la red neuronal no es correcto para identificar el PI tratado. Estos resultados son los mejores obtenidos durante el proceso de aprendizaje.

Debido a la complejidad de encontrar una red neuronal que realice una función similar a la del regulador, se decide modificar el sistema a identificar. En este caso, se expone la posibilidad de configurar una red de retardo que simule el funcionamiento del regulador PI.

7.2. CASO DE ESTUDIO 2, MODELADO RED DE RETRASO DE FASE

Una vez se ha determinado que un método para la resolución del problema puede el realizar una red de retardo de fase con un comportamiento similar al regulador proporcional integral, se procede a evaluare el antiguo PI para posteriormente generar la red.

El PI existente tiene la forma de la ecuación (1) , por lo que como se puede observar, está formado por un cero real en -0.01 y un polo situado en el origen. En el siguiente lugar de las raíces de la parte inferior izquierda, se puede observar donde están situados el polo y el cero.



a) b)

FIG 7.18. Lugar de las raíces. a) PI. b) Red de Retardo

Ahora se procederá a la sintonización de la red de retardo. Esta como su propia definición indica, deberá estar formada por un polo y cero, que deben cumplir, que el cero sea mayor que el polo, y que β debe ser mayor que la unidad.

$$R_{retardo}(s) = \frac{1}{\beta}\left(\frac{s+\frac{1}{T}}{s+\frac{1}{\beta T}}\right) \qquad (10)$$

El cero elegido corresponderá con el cero del regulador PI, mientras que el polo debe ser lo más similar posible a polo del origen de regulador, pero teniendo en cuenta que cuanto menor sea el polo, más rápida será la respuesta, por lo que se debe prestar atención a ese factor a la hora de elegir el polo para la red.

Por todo ello, y después de haber comprobado las respuestas con diferentes valores, el polo elegido se sitúa en 0,001. Esta selección se realiza debido a que las respuestas de ambos reguladores son parecidas en un principio, variando únicamente en que como integrador el PI continúa integrando constantemente, mientras que la respuesta del regulador de frecuencia se estabiliza a los 5000 segundos aproximadamente. Esta situación es altamente improbable que se de en una simulación del gemelo virtual, por lo que la selección del polo es correcta.

$$R_{retardo}(s) = 0.1\left(\frac{s+0.01}{s+0.001}\right) \qquad (11)$$

Ahora para confirmar esa posición, se deberá comprobar que las condiciones de ser un compensador en retraso de fase se cumplen.

	Condición	**Valor del caso**
β	$\beta > 1$	$\frac{1}{\beta} = 0.1 \rightarrow \beta = 10$
z y p	$cero > polo$	$0{,}01 > 0{,}001$

Ambas condiciones se cumplen, por lo que la red de retraso (11) es en principio un buen caso sobre el que trabajar. En la figura 7.18.b) se puede ver el lugar de las raíces de esta red.

Al modificar el regulador del sistema, las respuestas del control del gemelo virtual variarán. Debido a esto, se deben comprobar con el objetivo de confirmar que la red de retardo de fases escogida es un buen ejemplo.

Por ello, se comparan las respuestas del control utilizando las herramientas que nos brinda 3DEXPERIENCE. Todos los ejemplos, serán aplicados sobre la primera de las articulaciones del robot.

Regulador PI	**Red de retardo**

Velocidad de referencia y velocidad de giro del motor.

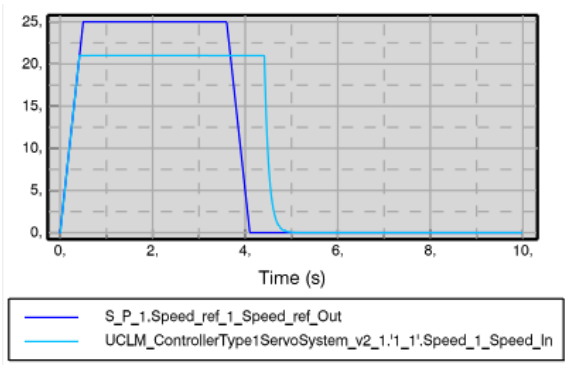


FIG 7.19. Comparación velocidades con PI

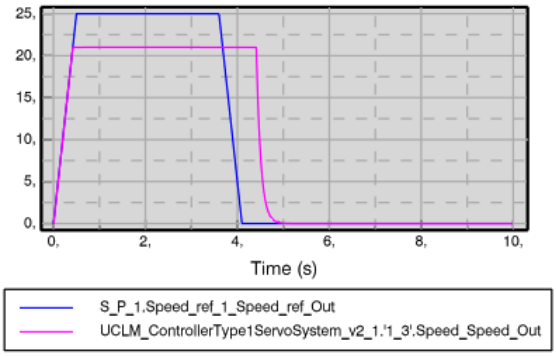


FIG 7.20. Comparación velocidades con red retardo

Posición angular de la articulación

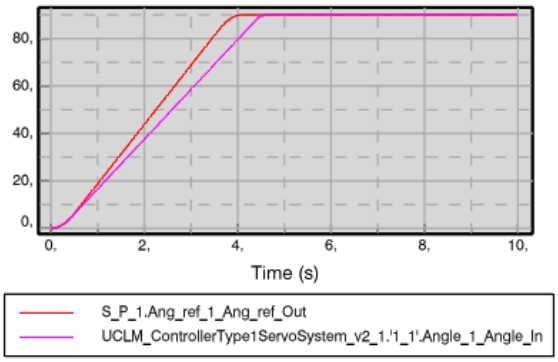


FIG 7.21. Comparación posición angular PI

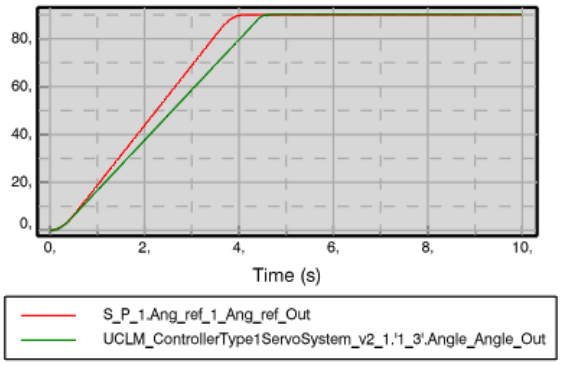


FIG 7.22. Comparación posición angular red de retardo

Al encontrarse ambas respuestas reguladas por el controlador, la sustitución del anterior por la nueva red de retardo, no afectará en gran medida al sistema. Esto se debe al pequeño efecto que tenía el PI de por si.

A continuación se muestra un ejemplo de como varía la señal de respuesta del regulador previo paso del limitador.

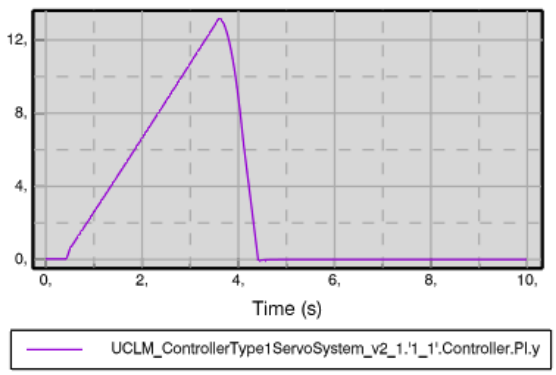


FIG 7.23. Simulación respuesta PI

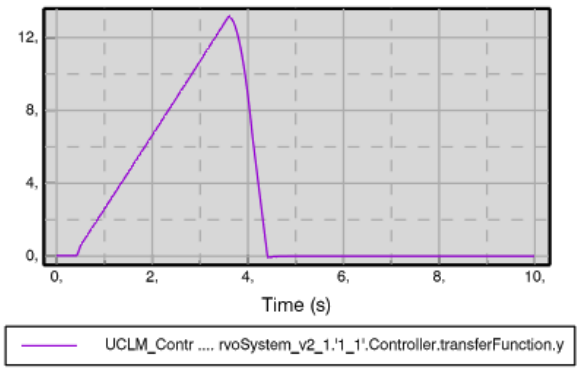


FIG 7.24. Simulación respuesta red retardo

Como se observa, la diferencia entre ambas respuestas tambien son realmente similares. La única diferencia tangible se encuentra en su valor máximo, ya que con la nueva red, la señal tendrá un valor levemente superior.

Finalmente, se puede determinar que el nuevo control tendrá un efecto similiar al existente. Por ello, se procede a continuación a realizar el entrenamiento del modelo neuronal de la red de retardo.

7.2.1. ENTRENAMIENTO MODELO RED NEURONAL

A continuación será necesario generar una entrada adecuada para poder comprobar la respuesta de nuevo sistema. Por ello el periodo escogido tienen un valor de 50 segundos ya que así se tomarán valores significativos de la respuesta.

$$T_s = 50s$$

Y la configuración de la señal PRBS será:

$$Periodo = 150$$
$$Numero\ de\ Periodos = 5$$
$$Banda = 0.02s^{-1}$$
$$Amplitud = [-2,2]$$

Donde la banda ha sido configurada como la inversa del periodo de muestreo escogido para que se pueda ver cómo responde el sistema ante la entrada. Con esa configuración la entrada generada y la respuesta de la red de retraso ante esa entrada serán.

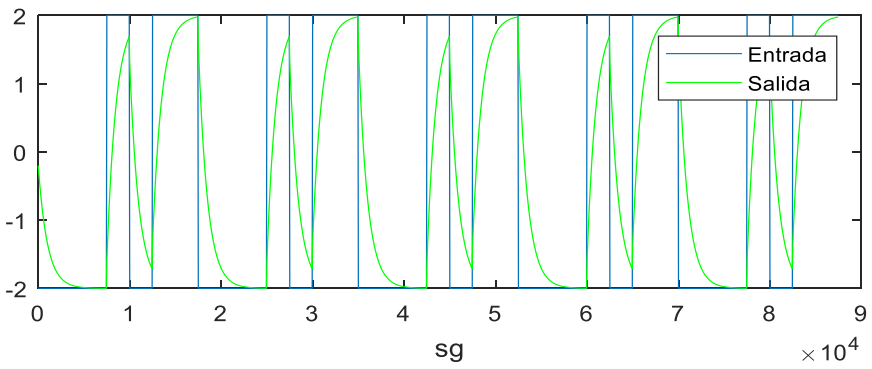


FIG 7.25. Señal PRBS y respuesta de la red de retardo

Una vez configurada la entrada, obtenida la entrada, y tratados los datos para por un lado poder afrontar los retrasos, y por otro eliminar posibilidades de sobreajuste mediante la separación de estos datos en los tres conjuntos de entrenamiento, validación y prueba, se configura la estructura de la nueva red neuronal NARX que tratará de modelar este sistema dinámico. A diferencia con la red del estudio anterior, esta constará de dos retrasos en las entradas, tanto independientes como la realimentación de la salida, y por otro lado una capa oculta de 10 neuronas.

Una vez tratados los datos para el entrenamiento, y configurada la red, se procede al igual que en casos anteriores, a entrenar el modelo para ajustar los pesos de las neuronas. En este caso el entrenamiento también es muy rápido, únicamente son necesarias 16 iteraciones para llegar al MSE mínimo al realizar las comprobaciones con el conjunto de validación. En este caso, la evolución de los errores cuadráticos medios de los tres conjuntos, se mueven en parámetros bastante más pequeños que en el anterior caso. Por lo que el entrenamiento probablemente muestra mejores resultados.

	Regulador PI	**Red de retraso**
MSE (Lazo abierto)	0,2285	0,0017
MSE (Lazo cerrado)	3,8682	0,0105

Tabla 7. Relación MSE ambos casos de estudio

Por otro lado, los errores de los conjuntos de entrenamiento, prueba y validación continúan mostrando comportamientos normales entre sí, al seguir siendo el de entrenamiento mayor.

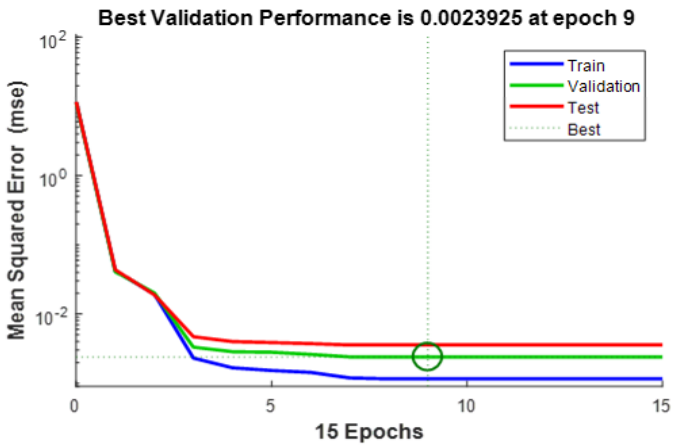


FIG 7.26. Entrenamiento caso 2, MSE conjuntos de datos

Si se atiende a algún otro método de análisis, se observará como las regresiones de los conjuntos son también muy cercanas a la unidad, obteniendo el total un valor de 0,9962, por lo que la regresión será prácticamente perfecta. Por otro lado, la autocorrelación del error y la correlación entre las entradas y las salidas, muestran valores en su mayoría cumpliendo los límites de correlación.

En cuanto a la respuesta temporal obtenida, observando la gráfica se puede confirmar el pequeño error existente. De hecho gran parte de ese error vuelve a aparecer cuando la entrada PRBS cambia de su valor mínimo a máximo y viceversa. Pero en este caso no existen diferencias tan grandes entre los valores deseados y los obtenidos, y las diferencias existentes, se deben a que las nuevas salidas dependen de los valores anteriores, por lo que se observa una continuidad en esos puntos que no debería darse. En la imagen inferior derecha, se observa ese pequeño error de cálculo.

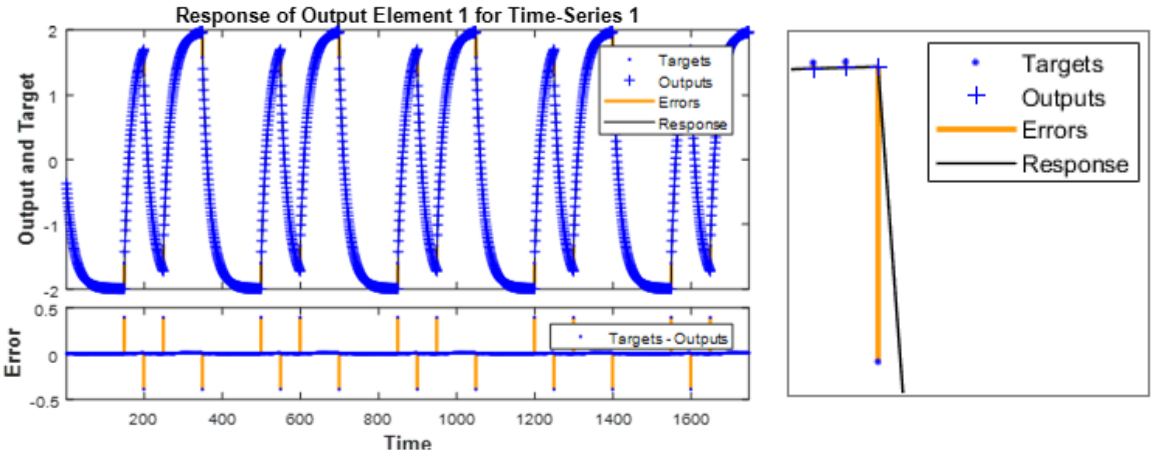


FIG 7.27. Respuesta temporal red neuronal, y caso de error significativo

7.2.2. COMPROBACIÓN DEL FUNCIONAMIENTO DEL MODELO

Para comprobar si la red neuronal imita las respuestas de la red de retraso, al igual que en casos anteriores, se debe cerrar la realimentación de la salida, para comprobar cómo responde, cuando sus entradas dependen también de sus anteriores salidas. En este caso, la respuesta temporal obtenida tendrá un mayor error al no tratarse encontrarse la red neuronal con la estructura utilizada para el entrenamiento.

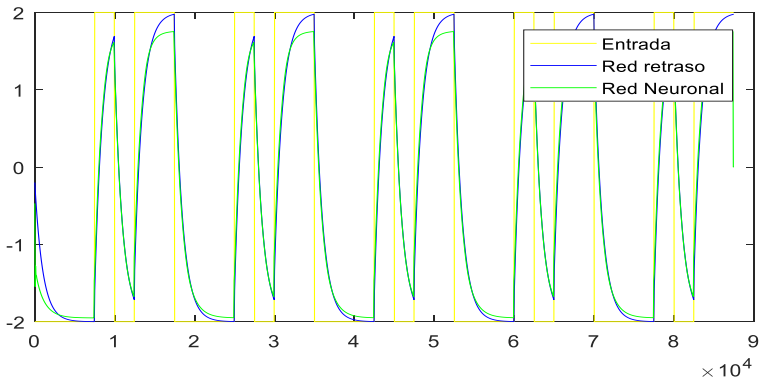


FIG 7.28. Respuesta temporal lazo cerrado

Si la red neuronal, es estimulada con la misma entrada con la que fue entrenada, en este caso la respuesta temporal obtenida distará en mayor medida de la respuesta de la red de retraso, pero los resultados muestran que el comportamiento continúa siendo muy parecido al deseado. En la tabla 7. donde se muestran los errores cuadráticos medios de este caso de estudio respecto del anterior, se alberga una diferencia crucial a la hora determinar esta red neuronal como un buen modelo del sistema dinámico expuesto.

Para confirmar la suposición de que sea un buen modelo, la red neuronal es excitada ahora con una nueva entrada PRBS diferente para comprobar si su respuesta continúa siendo la correcta.

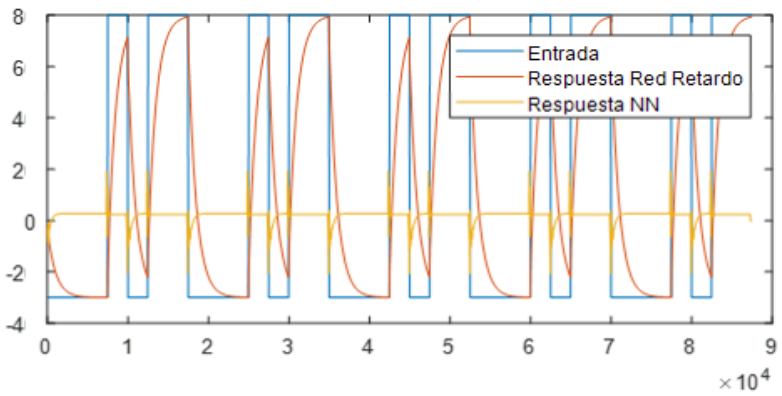


FIG 7.29. Respuesta red ante entrada diferente amplitud

Como se observa, el modelo ahora no se adapta correctamente a la nueva entrada, sino que tiende a mantener un comportamiento similar al mostrado con los datos con los que fue entrenado. Esto, se puede deber a dos problemas, el primer es, que se produzca sobreajuste en el entrenamiento, y por lo tanto la salida obtenida de la red neuronal, únicamente validez si su entrada es la señal libre con la que fue entrenada. La otra posibilidad, es que al tratarse de un modelo lineal que se está tratando de modelar con uno no lineal, la red NARX, el sistema obtenido en la red será no lineal. Esto hará que la red responda de forma diferente para diferentes amplitudes de entrada.

Para resolver el problema existente, y debido a que en posteriores pruebas del modelo obtenido se consiguieron respuestas muy variadas, se procede modificar la red neuronal, tratando de eliminar su no linealidad. Por ello, se sustituye la función de activación previa, por una nueva que convierta el modelo neuronal en un sistema lineal al igual que el objetivo.

La nueva función lineal pura, hará que la red neuronal pase a tener la siguiente estructura.

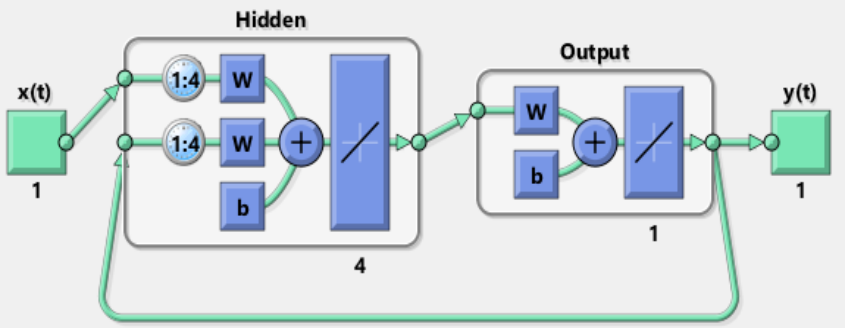


FIG 7.30. Red neuronal ARX (función de activación linear)

Una vez configurada de nuevo la red, se procederá igual que en los anteriores casos al entrenamiento y posterior análisis de los resultados obtenidos. En este caso, la entrada y los objetivos del entrenamiento continuarán siendo los mismos, únicamente variará la nueva entrada PRBS con la cual se comprobarán los resultados.

Una vez finalizado el entrenamiento repitiendo de nuevo los pasos del diagrama de flujo del esquema 3, se obtiene una respuesta temporal realmente similar a la conseguida mediante la red neuronal no linear.

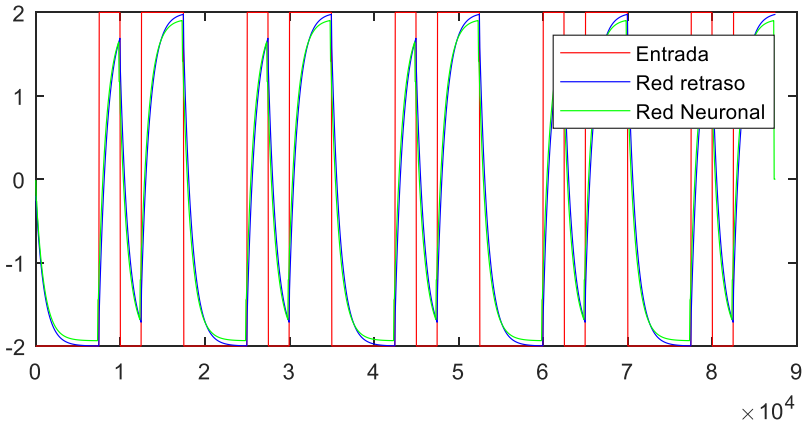


FIG 7.31. Respuesta red lineal ante entrada original

De hecho el error cuadrático medio en ambas pequeñas es similar y a la vez pequeño.

	Red Linear	**Red No Linear**
MSE (Lazo abierto)	0,0017	0,0017
MSE (Lazo cerrado)	0,0059	0,0105

Tabla 8: Comparación MSE redes linear y no linear

Pero si ahora se modifica la entrada, variando tanto la configuración del periodo, como la amplitud, se observarán diferencias significativas en las respuestas mostradas por ambos modelos.

$$Periodo = 750\ s$$
$$Numero\ de\ Periodos = 5$$
$$Banda = 0.02 s^{-1}$$
$$Amplitud = [-8,3]$$

La respuesta de esos modelos ante la nueva PRBS será:

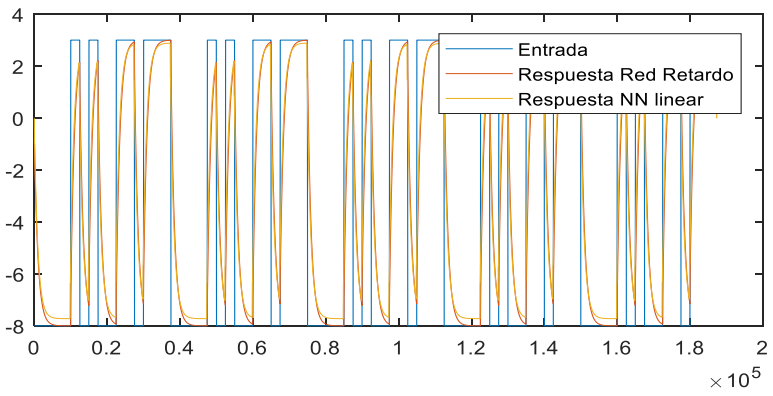


FIG 7.32. Respuesta red neuronal linear ante nueva entrada

Como se observa en la gráfica anterior, la nueva red neuronal linear, consigue respuestas adecuadas para cualquier amplitud de la entrada, por lo que será un buen modelo lineal del sistema dinámico, por lo que la segunda fase del proyecto se habría completado con éxito a la espera del confirmar los resultados al ser introducido el nuevo modelo en el control dinámico.

8. FMU

Una vez conseguido el modelo del sistema, se debe generar en SIMULINK el bloque de la red neuronal desarrollada. La red se generará en este software con un periodo de muestreo igual al determinado previamente para conseguir una señal de respuesta del sistema con la red de retardo significativa. Además, los pesos de las diferentes neuronas se trasladarán también a la nueva red neuronal en SIMULINK, pudiendo ser probado el modelo de nuevo en esta aplicación.

El siguiente paso después de transferir el modelo a SIMULINK, es que este sea introducido en el software de 3D EXPERIENCE, para proseguir con la sustitución del regulador proporcional-integral original. Para llevar a cabo este paso del proyecto, se hace uso de la herramienta FMU (Functional Mock-up Unit).

8.1. ¿QUÉ ES UNA FMU?

Una FMU es un modelo de simulación que permite la implementación de la interfaz de simulación funcional (FMI). Una FMI es una herramienta independiente estándar para el intercambio de modelos entre diferentes softwares o para la co-simulación [19]. En este caso, su aplicación será el intercambio de modelos, ya que una vez configurada la red neuronal en MATLAB y SIMULINK, esta deberá ser introducida en el control del gemelo virtual desarrollado en DYMOLA (3DEXPERIENCE), de modo que el antiguo bloque regulador proporcional, quede sustituido por su modelo neuronal. Para que el software donde se introducirá el modelo pueda usarlo, se debe generar en este caso en SIMULINK código con las mismas características del modelo, pero en este caso en lenguaje C. Para la generación de código, se utilizarán las herramientas que nos brinda tanto MATLAB como los compiladores Visual Studio y Cmake.

8.2. GENERACIÓN FMU MODELO NEURONAL

Para generar la FMU con el modelo neuronal en su interior, se debe tomar el bloque SIMULINK del modelo neuronal configurado con realimentación cerrada. Una vez separado del resto del sistema, se colocarán puertos numéricos correspondientes a la entrada y salida, por donde llegarán los datos de la señal a controlar una vez procesada, y saldrá la propia señal de control que será posteriormente limitada y aplicada al actuador o motor.

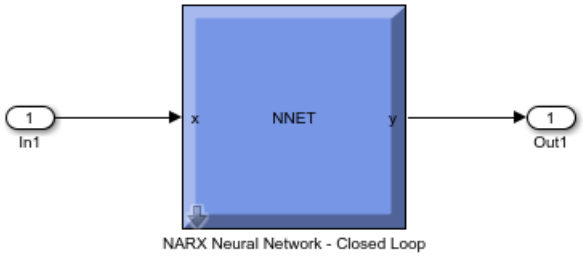


FIG 8.1. Fichero SIMULINK para FMU

Una vez generada la FMU, esta debe ser introducida en 3DEXPERIENCE para su posterior simulación. Para ello, se debe abrir el comportamiento del bloque controlador desarrollado en DYMOLA, una vez en el control, en la barra inferior del propio software, existe la posibilidad de importar una FMU.

Una vez importado, este nuevo bloque podrá ser utilizado como cualquier otro, pudiendo ser sustituido el bloque PI por el modelo neuronal. Una vez sustituido, la arquitectura final del controlador quedará de la siguiente forma:

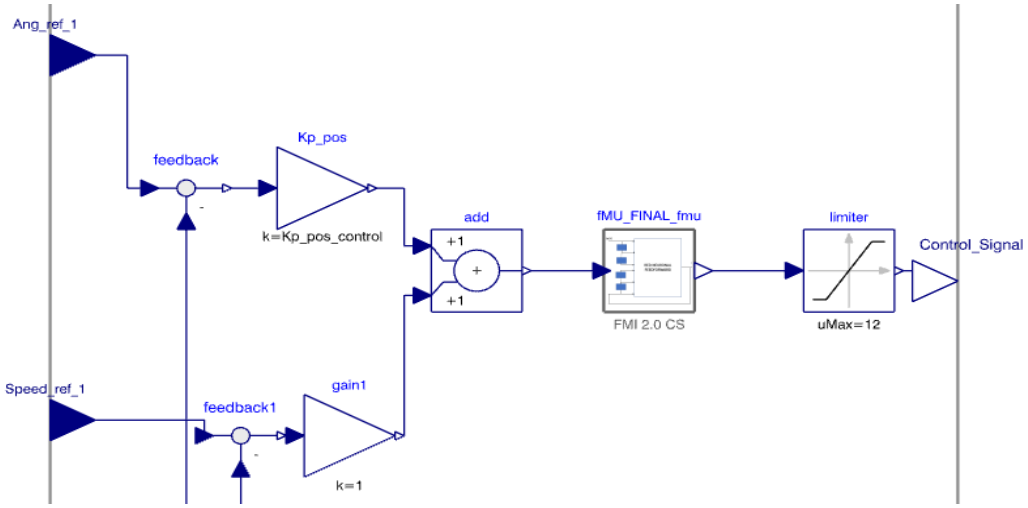


FIG 8.2. Bloque controlador con modelo neuronal incluido

Por último, únicamente faltará realizar la nueva simulación del gemelo virtual, para comprobar su nueva respuesta y ser comparada con las previas con el control original.

9. FUTURAS LÍNEAS DE INVESTIGACIÓN

Durante este proyecto de investigación, el esfuerzo se centra en sustituir en el controlador del gemelo virtual del robot industrial un nuevo regulador utilizando redes neuronales, elaborado a partir del ya existente. Pero existen otras líneas similares, por las cuales el proyecto puede ser ampliado en un futuro.

La primera y la más simple, se trata de generar un nuevo control utilizando técnicas de aprendizaje inteligente, y como con el control realizado, orientadas a la identificación de sistemas dinámicos, pero en este caso sustituyendo el bloque controlador al completo, no únicamente el propio regulador. Para realizar este nuevo control, el tipo de red neuronal debe ser el mismo, una red neuronal autorregresiva no lineal, pero en este caso con múltiples entradas independientes, una por cada entrada del bloque controlador que se muestra en la figura 3.8. En este caso, el entrenamiento seguirá las mismas fases que las realizadas durante el proyecto, y los resultados deberían ser similares.

Una segunda expansión del proyecto trataría no de obtener un nuevo controlador que sustituya al actual, sino la obtención del punto que el robot industrial debe alcanzar para recoger un objeto cuya posición en un momento inicial es desconocida.

Para esta expansión del proyecto, se podrían utilizar además de las ya utilizadas, nuevas tecnologías de la industria 4.0, para solucionar los problemas existentes.

- Gemelo virtual. Tanto el robot industrial utilizado, como el control, será el mismo que el utilizado para el proyecto ya desarrollado.
- Técnicas de visión artificial. La herramienta del software MATLAB para la visión por computador, podría ser utilizada para conseguir la posición del objeto a alcanzar en el puesto de trabajo.
- Técnicas de aprendizaje inteligente. Una vez obtenida la posición del objeto y sabidas sus dimensiones, se podrá aplicar esta técnica para obtener el punto exacto por donde debe ser agarrado por la herramienta del robot. La técnica de Deep Learning, puede ser aplicada en este caso por diferentes softwares como el ya utilizado MATLAB, u otros diferentes más centrados en redes neuronales como es TensorFlow. Esta herramienta, es una biblioteca de código abierto para aprendizaje automático (machine learning), desarrollada por Google y que utiliza lenguaje Python.

Para el desarrollo de esta tarea, se realizará un tipo de aprendizaje diferente, en este caso será un aprendizaje supervisado por refuerzo, donde el robot ejecutará la tarea de alcanzar

el punto deseado durante operaciones consecutivas de forma que el error se vaya reduciendo progresivamente, hasta alcanzar el punto deseado.

Esta fase, fue comenzada al inicio del proyecto, al aprender el uso tanto del lenguaje Python en primer lugar, como del software TensorFlow en segundo. Esta herramienta de Google es idónea para desarrollar este tipo de tareas, al tratarse de una aplicación con un gran rendimiento a la hora del aplicar el aprendizaje automático, y una gran potencia a la hora de la generación de gráficas para evaluar resultados al ser Python un lenguaje con una gran cantidad de recursos de este tipo.

10. CONCLUSIONES

Durante este proyecto, la complejidad de desarrollo de aplicaciones de este tipo se ha puesto en escena. Para la generación de control por aprendizaje automático aplicando técnicas de Deep Learning, como las redes neuronales, es necesario un conocimiento previo del tema, por lo cual el inicio del proyecto consistió en la adquisición de los conocimientos necesarios para la obtención final de resultados concluyentes. Una vez obtenidos los conocimientos, se procedió a realizar distintas fases del proyecto donde se ponen en conjunto conocimientos de las diferentes ramas del grado, por lo que se pone manifiesta la importancia de la multidisciplinariedad a la hora de desarrollar proyectos ingenieriles.

La obtención de respuestas concluyentes sobre el proyecto ha sido conseguida después de un largo proceso de prueba y error donde se ha realizado un elevado número de entrenamientos para la obtención del modelo, además los resultados de cada uno de los entrenamientos fueron sometidos a diferentes técnicas de análisis que hicieron la obtención de los resultados deseados sea más sencilla.

Por todo ello, se concluye por un lado, que el desarrollo de un proyecto de este tipo requiere una gran cantidad de tiempo dedicado tanto para abordar cada una de las fases, desde el aprendizaje teórico previo de las redes neuronales, pasando por su aplicación a las herramientas utilizadas como Tensor Flow o Matlab, hasta finalmente llegar a la obtención de los resultados deseados habiendo solucionado todos los problemas que han aparecido. Mientras que, por otro lado se cree, que la aplicación de las nuevas técnicas de inteligencia artificial y aprendizaje automático, tienen una gran proyección en cualquier rama de la ingeniería, y en especial durante la monitorización, automatización y control de procesos al solucionar de manera sencilla y eficaz muchos de los problemas que hoy en día requieren un gran gasto tanto de tiempo como de dinero para las empresas.

11. ANEXOS

MODELADO NEURONAL PI

```
% Autor: Pablo Galán Saiz
% Descripción: Modelado de sistemas dinámicos. Identificación PI mediante
% uso de redes neuronales NARX.

clc
clear all
close all

% Se introduce el PI del controlador del robot
s = tf('s');
PI = 0.1*((s+0.01)/s)
% Para evaluar la respuesta el PI es estimulado con una entrada escalón
% unitario
figure; step(PI);
title('PI ante entrada escalón unitario')

% Configuración de la señal PRBS
P = 200;    %Periodo de la señal
Chn = 2;    %Numero de señales a generar
NP = 2;     %Repeticiones de la señal
B = 1/50;    %Inversa del reloj
%Se genera con la toolbox de identificación la señal PRBS
u=(idinput([P,Chn,NP],'prbs',[0,B],[-1,1]));   % La señal tendrá las
%                                               características
%                                               anteriores a parte de
%                                               estar limitada entre
%                                               +2 y -2

% Se separan las entradas, una para realizar el entrenamiento y otra para
% comprobar resultados ante una entrada diferente
u1 = u(:,1);
u2 = u(:,2);
figure; plot(u1);
figure; plot(u2);

% Determinación del periodo de muestreo
Ts = 1;
% A partir de la señal de entrada, se genera el vector de tiempos
t = (0:Ts:Ts*length(u1))';
t = t(1:end-1);
figure; plot(t,u1); hold on;
% Con el vector de tiempos y la entrada, se genera la respuesta del
% regulador ante esa entrada
y = lsim(PI,u1,t); plot(t,y,'g'); legend('Entrada','Salida'); hold off;


% Se introducen las caracteristicas de la red
inputDelays = 1:4;          % 4 retrasos en la entrada independiente
feedbackDelays = 1:4;       % 4 retrasos en la realimentación de la salida
hiddenLayerSize = 10;       % Una capa de 10 neuronas
```

```
% Se genera la estructura de la red neuronal a utilizar, en este caso en
% lazo abierto
netPI = narxnet(inputDelays,feedbackDelays,hiddenLayerSize);

% Definición de la entrada-salida de la red
inPI = num2cell([u1']);
outPI = num2cell([y']);
% Una vez obtenidos los datos, estos se generalizan para reducir el riesgo
% de sufrir problemas como es sobreajuste
[inputs,inputStates,layerStates,targets] =...
    preparets(netPI,inPI,{},outPI);
% Se determina el método a utilizar para dividir los datos
netPI.divideFcn = 'divideblock';
% Division del conjunto de datos
netPI.divideParam.trainRatio = 70/100;    % Entreno
netPI.divideParam.valRatio = 15/100;      % Validación
netPI.divideParam.testRatio = 15/100;     % Prueba
% Entreno de la red neuronal y obtengo la red con sus nuevos pesos y los
% valores del entrenamiento
[netPI,tr] = train(netPI,inputs,targets,inputStates,layerStates);
% Se muestra la estructura de la red
view(netPI)

% Se muestran gráficos con los resultados del entrenamiento
% figure, plotperform(tr)
% figure, plottrainstate(tr)
% figure, plotregression(targets,outputs)
% figure, plotresponse(targets,outputs)
% figure, ploterrcorr(errors)
% figure, plotinerrcorr(inputs,errors)

% Se obtiene la salida de la red ante las entradas y estados
% predeterminados
yPI = netPI(inputs,inputStates,layerStates);
% Se calcula la diferencia entre los resultados de la red y la respuesta
% original de PI
errorsPI = gsubtract(targets,yPI);
figure; plot(t,errorsPI);
meanePI = mean(errorsPI);
% Por último se obtiene el valor del error cuadrático medio
performancePI = perform(netPI,targets,yPI);


% Se crea la red con el lazo cerrado a partir de la red con lazo abierto
netPIc = closeloop(netPI);
% La nueva estructura de la red es mostrada
view(netPIc)
% Se vuelven a preparar los datos, en este caso para la nueva estructura de
% red
[xc,xic,aic,tc] = preparets(netPIc,inPI,{},outPI);
% Se calcula la salida con la red lazo cerrado y el error
yPIc = netPIc(xc,xic,aic);
closedLoopPerformance = perform(netPIc,tc,yPIc);
```

```
% Lazo abierto
% En primer lugar, se observan los resultados en lazo abierto, es decir
% como si fuese un sistema estático.
%yPInet = netPI(inputs);
yPInet = netPI(inputs);
yPInet = cell2mat(yPInet)';
yPInet = [yPInet; 0; 0; 0; 0];
yPI = [yPI; 0; 0; 0; 0];
figure; plot(t,u1); hold on; plot(t,y)
plot(t,yPInet); hold off
legend('PRBS','yPI','con red (yPInet)')

% Lazo cerrado
yPInetc = netPIc(xc);
yPInetc = cell2mat(yPInetc)';
yPInetc = [yPInetc; 0; 0; 0; 0]';
figure; plot(t,u); hold on; plot(t,y)
plot(t,yPInetc); hold off;
legend('u','y','con red (yPInetc)')


% Si se quiere simular y comprobar en simulink
% Se exportan las señales PRBS
U = [t,u];
% Se exporta el modelo neuronal
gensim(necPIc,Ts);
```

MODELADO NEURONAL RED DE RETARDO

```
% Autor: Pablo Galán Saiz
% Descripción: Generación modelo neuronal de red de retraso

clc
clear all
close all

% El regulador del controlador es
s = tf('s');
PI = 0.1*((s+0.01)/s);

% Determino la red de retraso que se tratará de modelar.
% Para ello en primer lugar se muestras el lugar de las raíces del PI
figure; rlocus(PI);
% Ahora se determinan los ceros y polos de la red obteniendo
Rr = 0.1*tf([1 0.01],[1 0.001]);
% Y se muestra el nuevo lugar de las raíces
figure; rlocus(Rr);

% Una vez determinada la red ya se pueden obtener los datos de entrenamiento
% Se determina un periodo con el que se observe la evolución de la respuesta
figure; step(Rr);
T = 50;

% Se genera la señal de excitación PRBS
```

```
P = 150;          % Periodo de la señal
Ncha = 1;         % Numero de canales
NP = 5;           % Numero de periodos
B = 0.02;         % Banda, inversa del tiempo para una muestra
u = idinput([P,Ncha,NP],'prbs',[0,B],[-2,2]);

% Con la entrada generada, se determina el tiempo de la señal
t = 0:T:T*length(u); t = t(1:end-1);
% Respuesta de la red ante la entrada PRBS
y = lsim(Rr,u,t);

figure; plot(t,u); hold on; plot(t,y);
legend('Entrada PRBS','Respuesta Red Retardo'); hold off;

% Los datos de entrenamiento de la red serán
in = num2cell([u']); out = num2cell([y']);


% A continuación se define la estructura de la red NARX
inputDelays = 1:4;        % Retrasos entrada exógena
feedbackDelays = 1:4;     % Retrasos realimentación
HiddenLayerSize = 10;      % Numero de neuronas por capa oculta
net=narxnet(inputDelays,feedbackDelays,HiddenLayerSize);

% Se preparan los datos para el entrenamiento
% En primer lugar se elimina su temporalidad para afrontar los retrasos
[inputs,inputStates,layerStates,targets] =...
    preparets(net,in,{},out);

% Posteriormente se separan los datos en los conjuntos de entrenamiento,
validacion y prueba
% Esta división se hará de forma predeterminada en bloques
net.divideFcn = 'divideblock';
net.divideParam.trainRatio = 70/100;  % 70% Entrenamiento
net.divideParam.valRatio = 15/100;    % 15% Validación
net.divideParam.testRatio = 15/100;   % 15% Prueba
% Se realiza ahora el entrenamiento de la red neuronal

[net,tr] = train(net,inputs,targets,inputStates,layerStates);

% Una vez entrenada se analizan los resultados del entrenamiento
% La salida conseguida será
outputs = net(inputs,inputStates,layerStates);
% El error respecto del objetivo
errors = gsubtract(targets,outputs);
% El error cuadrático medio MSE
performance = perform(net,targets,outputs);

% La red entrenada tendrá la siguiente estructura
view(net)


% El entrenamiento puede ser analizado mediante las siguientes graficas
% Evolución del error mse en los tres conjuntos
```

```
% figure, plotperform(tr)
% Evolución del entrenamiento
% figure, plottrainstate(tr)
% Regresión lineal
% figure, plotregression(targets,outputs)
% Respuesta temporal conseguida
% figure, plotresponse(targets,outputs)
% Autocorrelación de los errores
% figure, ploterrcorr(errors)
% Correlación entradas-errores
% figure, plotinerrcorr(inputs,errors)


% A continuación se cierra el lazo de forma que ahora la entrada realimentada
% dependa no de los objetivos sino de las salidas previas conseguidas
netc = closeloop(net);
view(netc)

% De nuevo se vuelven a preparar los datos de entrenamiento
[xc,xic,aic,tc] = preparets(netc,in,{},out);
% Y posteriormente se calcula la nueva respuesta de la red ante esos datos
yc = netc(xc,xic,aic);
% Para analizar los resultados, se calcula el mse
closedLoopPerformance = perform(netc,tc,yc);


% Una vez obtenida la respuesta de la red ante esa entrada, esta se muestra en
% comparación con la respuesta de la red original.
ynetc = netc(xc);
yg = lsim(Rr,u,t');figure; plot(t,u); hold on; plot(t,yg);
aux = cell2mat(ynetc); aux = [aux 0 0 0 0];
plot(t,aux); hold off; legend('Entrada','Red retraso','Red Neuronal')
% Los resultados obtenidos son correctos.


% Se comprueba ahora la respuesta de la red ante una entrada de diferente
amplitud
u1 = idinput([P,Ncha,NP],'prbs',[0,B],[-3,8]);
% La respuesta de la red de retardo será
y1 = lsim(Rr,u1,t);
% Se vuelven a configurar los datos
in1 = num2cell([u1']); out1 = num2cell(y1');
[xc1,xic1,aic1,tc1]=preparets(netc,in1,{},out1);

% Se calcula la respuesta de la red neuronal ante esa misma entrada
y1netc = netc(xc1);
aux1 = cell2mat(y1netc); aux1 = [aux1 0 0 0 0];
figure; plot(t,u1); hold on; plot(t,y1); plot(t,aux1); hold off;
legend('Entrada','Red retraso','Red Neuronal')

% Como se observa en la respuesta, la red no se adapta bien, esto se puede
% deber a la no linealidad de la red.
% Por ello, se modifica ahora la red neuronal convirtiéndola en una lineal
netLi=narxnet(inputDelays,feedbackDelays,HiddenLayerSize);
```

```
netLi.layers{1}.transferFcn = 'purelin';

% Se vuelve a entrenar con los datos de la entrada -2,2
[netLi,trLi] = train(netLi,inputs,targets,inputStates,layerStates);

outputsLi = net(inputs,inputStates,layerStates);
errorsLi = gsubtract(targets,outputsLi);
performanceLi = perform(netLi,targets,outputsLi);

% Al igual que con la red, se comprueba su funcionamiento en lazo cerrado
netLic = closeloop(netLi);
view(netLic)

[xcLi,xicLi,aicLi,tcLi] = preparets(netLic,in,{},out);
ycLi = netc(xcLi,xicLi,aicLi);
closedLoopPerformanceLi = perform(netLic,tcLi,ycLi);

% Los resultados en este caso serán.
ynetLic = netLic(xcLi);
auxLi = cell2mat(ynetLic); auxLi = [auxLi 0 0 0 0];
figure; plot(t,u); hold on; plot(t,yg);
plot(t,auxLi); hold off; legend('Entrada','Red retraso','Red Neuronal')

% Y ante la entrada de diferente amplitud.
[xcLi1,xicLi1,aicLi1,tcLi1]=preparets(netLic,in1,{},out1);
ynetLi1c = netLic(xcLi1);
auxLi1 = cell2mat(ynetLi1c); auxLi1 = [auxLi1 0 0 0 0];
figure; plot(t,u1); hold on; plot(t,y1); plot(t,auxLi1); hold off;
legend('Entrada','Red retraso','Red Neuronal')

% Se muestra por último la comparación de respuestas ante esa entrada
figure; plot(t,y1); hold on; plot(t,aux1); plot(t,auxLi1);
title('Comparación')
legend('Respueta Red Retraso','Respuesta No lineal','Respuesta Lineal')

% Finalmente se observa como la red lineal consigue adaptarse a la perfección
% Debido a ello el modelo en simulink es conseguido.
NetNoLinear = gensim(netLic,T)
```

12. BIBLIOGRAFÍA

[1] Llorenç Guilera, Antoni Garrell. 2019. La industria 4.0 en la sociedad digital. ISBN 978-84-17313-86-9.

[2] Prieto, Mario. 2017. Cómo aprovechar la industrial 4.0 para mejorar la eficiencia de los equipos industriales. *Smart-lighting.*

[3] Samaniego, Juan F. 2018. Industria 4.0: ¿Qué tecnologías están transformando las fábricas? *Hablemosdeempresas*.

[4] ABB Robotics. Especificaciones del producto IRB120. 3HAC035960-005 Revisión: S

[5] Aquino Robles, Jose Antonio; Germán Corona, Leonel; Trujillo C., Juan Carlos. 2013. Tendencias en la enseñanza de la Ingeniería Mecatrónica y su campo disciplinar. Instituto Politécnico Nacional, Universitat Politécnica de Catalunya.

[6] Dassault Systèmes. 2018. FMI Kit for Simulink versión 2.4.2.

[7] Dassault Systèmes. 2019. Dymola (Dynamic Modeling Laboratory): FMI Support in Dymola.

[8] Blochwitz, T.; Otter, M.; Akesson, J.; Arnold, M.; ClauB, C.; Elmqvist, H.; Friedrich, M.; Junghanns, A.; Mauss, J; Neumerkel, D.; Olsson, H.; Viel, A. 2012. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *9th International Modelica Conference.*

[9] 2. Redes neuronales para el modelado de Sistemas dinámicos: los módulos neuronales. [material didáctico].
Disponible en:
https://www.tdx.cat/bitstream/handle/10803/6179/05Cap2.pdf?sequence=5&isAllowed=y

[10] Dingyü Xue; YangQuan Chen. 2014. System Simulation Techniques with MATLAB and Simulink. ISBN: 978-1-118-64792-9 o 978-1-118-69435-0.

[11] Serrano, Antonio J.; Soria, Emilio; Martín, José D. 2010. Redes Neuronales Artificiales. [Material Didáctico]. En *OCW Universitat de Valencia.*

[12] González Sarabia, Esther. 2012. Técnicas avanzadas de identificación y reconstrucción de objetos mediante ultrasonidos. Pérez Oria, Juan Mª (dir); Llata García, José Ramón. (dir). Tesis doctoral, *Universidad de Cantabria.*

[13] Zouhour Neji; F-Mouria Beji. 2000. Neural Network ans Time Series Identification and Prediction. *IEEE.*

[14] J.R. Llata, E.G. Sarabia, D. Fernandez, J. Arce, J.P. Oria. 2000. Aplicación de Inteligencia Artificial en Sistemas Automatizados de Producción. En *Revista Iberoameticana de Inteligencia Artificial No.10*.

[15] Gómez Rojas, G. Alonso; Henao López, J. Carlos; Salazar Isaza, Harold. 2004. Entrenamiento de una red neuronal artificial usando el algoritmo simulated annealing. En *Scientia et Technica Año X, No 24.*

[16] Pineda Palacios, J. Guillermo. 1993. Construcción de un módulo generador de PRBS. [Trabajo fin de Grado]. *Escuela Politécnica Nacional.*

[17] Pedro Larrañaha, Iñaki Inza, Abdelmaik Moujahid. Redes Neuronales [Material didáctico]. En *Departamento de Ciencias de la Computación e Ingeniería Artificial, Universidad del País Vasco.*

[18] Hudson Beale, Mark; T.Hagan, Martin; B.Demuth, Howard. 2019. Matlab Deep Learning Toolbox User's Guide. *MATHWORKS*

[19] Simon Cöster. 2014. Interfacing functional mock-up units in modelica. [Master's Thesis]. *Lund University.*

[20] Jesus Lopez; Eduardo F. Calcedo Bravo. 2007. Identificación de sistemas usando Redes Neuronales Entrenadas con Aprendizaje Bayesiano. *Universidad del Valle.*

[21] Vázquez Rey, Daniel; Huerta Sánchez, José Maria (dir). 2017. Estudio y aplicación de métodos de identificación paramétrica en sistemas de tiempo discreto y su validación en equipos didácticos de laboratorio. [Trabajo fin de Grado]. *Universitat Politécnica de Catalunya.*

[22] Monserrat, Mª del Carmen; Crespo, José Luis (dir). 2013. Sistemas inteligentes para el ajuste de modelos hidrológicos. [Tesis doctoral]. *Universidad de Cantabria.*

[23] J.R. Llata, E.G. Sarabia, and J.P. Oria. Pattern recognition with ultrasonic sensors: a neural networks evaluation*. Sensor Review, Vol. 21 Iss: 1:45 – 51, 2001*.

[24] J.R. Llata. Feedback control for Mechatronic Fundamentals. [Curso online]. *Peer Learning Experience.*

[25] Curso intensive de aprendizaje automático con API de TensorFlow. [Curso online]. *Google Developers.*