MÉTODOS PARAMÉTRICOS VS REDES NEURONALES PARA EL MODELADO DE LA UNIDAD DE TRACCIÓN DE UN AGV INDUSTRIAL

TRABAJO FIN DE MÁSTER
CURSO 2021-2022

AUTOR
RAFAEL RODRÍGUEZ VILCHES

DIRECTORES:
MATILDE SANTOS PEÑAS
JESÚS ENRIQUE SIERRA GARCÍA

MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL

UNIVERSIDAD COMPLUTENSE DE MADRID-UNED

MÉTODOS PARAMÉTRICOS VS REDES NEURONALES PARA EL MODELADO DE LA UNIDAD DE TRACCIÓN DE UN AGV INDUSTRIAL

TRABAJO DE FIN DE MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL

AUTOR
RAFAEL RODRÍGUEZ VILCHES

DIRECTORES
MATILDE SANTOS PEÑAS
JESÚS ENRIQUE SIERRA GARCÍA

CONVOCATORIA: SEPTIEMBRE 2022

MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL

UNIVERSIDAD COMPLUTENSE DE MADRID-UNED

19 DE SEPTIEMBRE DE 2022

AGRADECIMIENTOS

A María por su apoyo.

A mis directores por haberme guiado en la realización de este trabajo.

RESUMEN

En el presente proyecto fin de máster se ha procedido a la identificación y comparación de los modelos obtenidos mediante técnicas paramétricas y redes neuronales de un vehículo de guiado automático AGV (Automatic Guided Vehicle). Se ha seleccionado un sistema basado en un AGV de tipo tractor-triciclo. La identificación de sistemas con alta complejidad, como son el caso de estos vehículos industriales, es generalmente una tarea difícil debido a las no linealidades asociadas a su comportamiento, estructura híbrida, y a la acción de fuerzas externas durante el movimiento y transporte de la carga.

Como datos de origen se han empleado los valores de velocidad de los pulsos proporcionados por los encoder de la rueda izquierda y derecha de un vehículo real. Se ha llevado a cabo una filtración y depuración de estos datos, eliminándose sus *outliers* y el ruido asociado a las medidas, para posteriormente transformar los datos de velocidad filtrados y trabajar de esta forma con una velocidad lineal sin ruido ni distorsiones.

Analizando los resultados obtenidos en el dominio del tiempo, se muestra como las redes neuronales obtienen un mejor ajuste a cambio de un mayor coste computacional frente a los métodos paramétricos polinómicos, los cuales aunque precisan de polinomios con órdenes muy altos para conseguir ajustes, tienen un coste computacional menor.

Se ha observado en el caso de las redes neuronales, cómo se alcanza un punto en el que en el que redes con mayor número de neuronas en la capa oculta no consiguen un mejor ajuste frente a redes más simples, incrementándose solo el coste de computación y apareciendo fenómenos de overfitting. De igual forma, se ha podido comprobar como el fenómeno de overfitting aparece con mayor frecuencia en las redes tipo NAR frente a las NARX para el mismo tamaño de neuronas y capas ocultas.

Palabras clave

Vehículo de Guiado Automático (AGV), identificación, modelado, filtrado, técnicas paramétricas polinómicas, redes neuronales, overfitting.

ABSTRACT

In this final master thesis, it has been identified and compared the models obtained by parametric techniques and neural networks from an automatic guided vehicle (AGV). A system based on a tractor-tricycle type AGV has been selected. The identification of highly complex systems such as these AGV is generally a challenging task due to non-linearities associated with their behaviour, structure and action of exogenous forces during movement and load transportation.

As source data, has been employed the speed values of pulses (pulses/s) provided by the left and right wheel encoders from a real AGV. These data have been filtered and debugged, removing outliers and noise related to measurements in order to subsequently transform the filtered velocity data in pulses/s to cm/s and being able to work with this velocity without noise and distortions.

Analysing the results obtained in time domain, it is shown how neural networks obtain a better fit than polynomial parametric methods but with a higher computational cost. Although the polynomial methods require polynomials with high order to obtain a fit, the computational cost is lower.

In the example of neural networks, it has been noticed how a point is reached in which networks with a greater number of neurons in hidden layer do not achieve a better fit than smaller networks, only increasing the computational cost and appearing overfitting phenomena. Similarly, it has been proven that overfitting phenomena appears more frequently in NAR type networks than in NARX networks for the same number of neurons and hidden layers.

Keywords

Automated Guided Vehicle (AGV), identification, modelling, filtering, outliers, polynomial parametric techniques, neural networks, overfitting.

ÍNDICE DE CONTENIDOS

Contenido

Agradecimientos 2

Resumen 3

Abstract 4

Índice de contenidos 5

Índice de figuras 7

Índice de tablas 10

Capítulo 1 - Introducción 11

1.1 Importancia de los AGVs en la industria 11

1.2 Motivación: Importancia de modelos computacionales para probar algoritmos de control 12

1.3 Objetivo general y objetivos específicos 13

1.4 Estructura de la memoria 14

Capítulo 2 - Estado del arte 15

2.1 Tipos de AGV y características 15

2.2 Tipo de AGV empleado para la obtención de los datos 17

2.2.1 Dinámica del AGV tipo triciclo 17

2.3 Identificación de sistemas mediante métodos paramétricos polinómicos 19

2.3.1 Descripción de los modelos 22

2.3.2 Complejidad computacional en modelos paramétricos polinómicos 26

2.4 Identificación de sistemas mediante redes neuronales 30

2.4.1 Concepto de neurona artificial 30

2.4.2 Descripción de una red neuronal 31

2.4.3 Aprendizaje en una red neuronal 33
2.4.4 Complejidad computacional en redes neuronales 35
2.5 Artículos científicos relacionados 37
Capítulo 3 - Materiales y datos 39
3.1 Filtrado y procesado inicial de los datos 39
Capítulo 4 - Resultados de la identificación del AGV 51
4.1 Aplicación de técnicas paramétricas 51
4.1.1 Métodos de estimación de los coeficientes de los polinomios 51
4.1.2 Estimación de los órdenes de los polinomios 54
4.1.3 Parámetros empleados para realizar la comparación del ajuste entre las diferentes técnicas 54
4.1.4 Mejores ajustes obtenidos para los datos del encoder derecho 56
4.1.5 Mejores ajustes obtenidos para los datos del encoder izquierdo 59
4.1.6 Cálculo del coste computacional. Modelos paramétricos encoder derecho e izquierdo. 62
4.2 Identificación con redes neuronales 63
4.2.1 Parámetros empleados para realizar la comparación entre los ajustes obtenidos mediante las diferentes redes neuronales 65
4.2.2 Mejores ajustes obtenidos para los datos del encoder derecho 66
4.2.3 Mejores ajustes obtenidos para los datos del encoder Izquierdo 77
4.2.4 Cálculo del coste computacional. Redes neuronales. Encoder derecho e izquierdo 87
4.3 Comparativa de la mejor técnica paramétrica vs mejor red neuronal para cada encoder 89
Capítulo 5 - Conclusiones y trabajos futuros 91
Capítulo 6 - Bibliografía 93

ÍNDICE DE FIGURAS

Ilustración 1: Evolución de la inversión en AGV en Norte América, APAC (Asia-Pacífico), Europa y ROW (Rest of World) [11]. 11

Ilustración 2: Esquema AGV tipo triciclo.[12] 18

Ilustración 3: Esquema de bloques general para métodos polinómicos.[4] 21

Ilustración 4: Modelo ARX.[4] 22

Ilustración 5: Modelo ARMAX.[4] 24

Ilustración 6: Modelo OE.[4] 25

Ilustración 7: Modelo BJ.[4] 26

Ilustración 8: Modelo neuronal-Perceptrón de McCulloch-Pitts (1943). [22] 30

Ilustración 9: Esquema de una red neuronal. [6] 32

Ilustración 10: Datos de origen. 39

Ilustración 11: Datos de partida ED y EI. 40

Ilustración 12: Diagrama de flujo para el filtrado y procesado de los datos. 41

Ilustración 13: Representación datos Setpoint vs Velocidad (pulsos/s). Encoder derecho. 43

Ilustración 14: Representación datos Setpoint vs Velocidad (pulsos/s). Encoder Izquierdo. 44

Ilustración 15: Velocidad Encoder derecho con outliers vs sin outliers para 50 y 100ms. 47

Ilustración 16: Velocidad Encoder izquierdo con outliers vs sin outliers para 50 y 100ms.48

Ilustración 17: Representación de los valores de velocidad (cm/s) a la salida del filtro de paso bajo para el encoder derecho. 49

Ilustración 18: Representación de los valores de velocidad (cm/s) a la salida del filtro de paso bajo para el encoder izquierdo. 50

Ilustración 19: Método de estimación modelo ARX. 52

Ilustración 20: Ejemplo selección método de estimación para el modelo ARMAX. 53

Ilustración 21: Ajuste mediante métodos paramétricos. Encoder derecho 56

Ilustración 22: Respuesta en frecuencia métodos paramétricos. Diagrama de Bode. Encoder derecho.....58

Ilustración 23: Respuesta frente a escalón. Encoder derecho. 58

Ilustración 24: Ajuste mediante métodos paramétricos. Encoder Izquierdo.....59

Ilustración 25: Respuesta en frecuencia métodos paramétricos. Diagrama de Bode. Encoder izquierdo. 61

Ilustración 26: Respuesta frente a escalón. Encoder Izquierdo. 61

Ilustración 27: Ejemplo red NAR. 64

Ilustración 28: Ejemplo red NARX. 64

Ilustración 29: Porcentaje del conjunto de datos destinados a entrenamiento, validación y test. 65

Ilustración 30: Respuesta red NARX. Nº neuronas capa oculta: 25. Tiempo retardo:2. Encoder derecho. 67

Ilustración 31: Regresión en cada fase. Red NARX. Tamaño capa oculta 25. Tiempo retardo:2. Encoder derecho. 68

Ilustración 32: Mejor desempeño red NARX. 25 neuronas. Tiempo de retardo 2. Encoder derecho. 69

Ilustración 33: Sobreajuste en red NARX. 100 neuronas. Tiempo de retardo2. Encoder derecho. 70

Ilustración 34: Respuesta red NAR. Tamaño capa oculta:25. Tiempo retardo:2. Encoder derecho. 72

Ilustración 35: Regresión en cada fase. Red NAR. Tamaño capa oculta 25. Tiempo retardo:2. Encoder derecho 73

Ilustración 36: Mejor desempeño red NAR. 25 neuronas. Tiempo de retardo 2. Encoder derecho. 74

Ilustración 37: Sobreajuste en red NAR. 100 neuronas. Tiempo de retardo 2. Encoder derecho. 75

Ilustración 38: Sobreajuste en red NAR.200 neuronas. Tiempo de retardo 1. Encoder derecho. 75

Ilustración 39: Sobreajuste en red NAR.200 neuronas. Tiempo de retardo 2. Encoder derecho. 76

Ilustración 40: Respuesta red NARX. Tamaño capa oculta:25. Tiempo retardo:2. Encoder izquierdo. 78

Ilustración 41: Regresión en cada fase. Red NARX. Tamaño capa oculta 25. Tiempo retardo:2. Encoder izquierdo. 79

Ilustración 42: Mejor desempeño red NARX. 25 neuronas. Tiempo de retardo 2. Encoder izquierdo 80

Ilustración 43: Sobreajuste en red NARX. 200 neuronas. Tiempo de retardo 2. Encoder izquierdo. 81

Ilustración 44: Respuesta red NAR. Tamaño capa oculta :25. Tiempo retardo:2. Encoder izquierdo. 83

Ilustración 45: Regresión en cada fase. Red NAR. Tamaño capa oculta 25. Tiempo retardo:2. Encoder izquierdo. 84

Ilustración 46: Mejor ajuste red NAR. 25 neuronas. Tiempo de retardo 2. Encoder izquierdo. 85

Ilustración 47: Sobreajuste en red NAR. 200 neuronas. Tiempo de retardo 1. Encoder izquierdo. 86

Ilustración 48: Sobreajuste en red NAR. 200 neuronas. Tiempo de retardo 2. Encoder izquierdo. 86

ÍNDICE DE TABLAS

Tabla 1: Parámetros del modelo dinámico de la unidad tractora del AGV. [12] 19

Tabla 2: Resultados ajuste métodos paramétricos. Encoder derecho.....57

Tabla 3: Resultados ajuste métodos paramétricos. Encoder izquierdo. 60

Tabla 4: Comparativa entre los costes computacionales de la mejor técnica para el encoder derecho e izquierdo. 62

Tabla 5: Resumen valores de ajuste con red NARX. Encoder derecho.....66

Tabla 6: Listado ajustes con red NAR. Encoder derecho.....71

Tabla 7: Listado ajustes con red NARX. Encoder izquierdo. 77

Tabla 8: Listado ajustes con red NAR. Encoder izquierdo. 82

Tabla 9: Coste computacional NN. Encoder derecho. 88

Tabla 10: Coste computacional NN. Encoder izquierdo.....88

Tabla 11: Comparación mejor técnica paramétrica vs mejor red neuronal. Encoder derecho. 89

Tabla 12: Comparación mejor técnica paramétrica vs mejor red neuronal. Encoder Izquierdo. 89

Tabla 13: Ajuste técnicas paramétricas. Encoder derecho/izquierdo. 89

Tabla 14: Ajuste red neuronal NARX. Encoder derecho/izquierdo. 89

Capítulo 1 - Introducción

1.1 Importancia de los AGVs en la industria

Dentro de un proceso productivo, los subprocesos de transporte y manipulación de cargas ya sea de materias primas, productos intermedios o acabados, representan un volumen importante de las tareas, recursos y tiempo del proceso global.

Si se observan los desplazamientos de las mercancías por la instalación, generalmente siguen rutas bien definidas y repetitivas, por lo que se hace posible un amplio abanico de métodos de automatización de estos procesos de transporte y búsqueda de nuevas formas de operación. Por esto, desde hace años existe un elevado interés en el desarrollo y mejora de vehículos que permitan realizar este transporte de forma más flexible y autónoma frente a los habituales sistemas de instalación fija, con una disminución tanto de los errores humanos como de los costes operativos. Estos motivos han sido los que han ido originando que año tras año las ventas globales de AGVs (Automated Guided Vehicle) e inversión por parte de las compañías haya aumentado progresivamente hasta alcanzar una tasa estimada de crecimiento anual compuesto (CAGR) del 7,7%, tal y como se muestra en la siguiente ilustración.

Ilustración 1: Evolución de la inversión en AGV en Norte América, APAC (Asia-Pacífico), Europa y ROW (Rest of World) [11].

1.2 Motivación: Importancia de modelos computacionales para probar algoritmos de control

El mercado cada vez exige AGVs con un rendimiento y precisión superiores, una contención en los costes y plazos de entrega ajustados. Para los ingenieros encargados del desarrollo, el flujo de trabajo tradicional en donde las mejoras basadas en la prueba y validación de los sistemas de control tienen lugar en una etapa muy avanzada del ciclo de desarrollo del AGV (cuando las diferentes piezas del AGV están disponibles) está quedando obsoleto. Hasta ahora ha sido habitual en todo el ciclo de desarrollo de un producto la aparición de problemas en los sistemas de control o sistemas mecánicos en las fases finales del desarrollo del producto, acarreando graves inconvenientes y dificultades a la hora de resolverlo al requerir cambios complejos y generalmente costosos. Es por esto, por lo que los fabricantes han dado cada vez más importancia al desarrollo y diseño basados en modelos computacionales que permitan simular los diferentes estados del producto y probar los algoritmos de control y la lógica de funcionamiento para detectar los errores cuando aún son fáciles de corregir. Entre las principales **ventajas** que ofrecen los modelos computacionales en el desarrollo de un producto se encuentran:

- Capacidad de poner a prueba todos los sistemas de funcionamiento del AGV mediante simulación. Forma más económica frente a las pruebas tradicionales experimentales.
- Evaluar diversas estrategias de control y optimizar el sistema.
- Rápida detección de errores en el control antes de tener disponibles físicamente todas las piezas.
- Capacidad de reutilizar modelos para desarrollar pruebas en tiempo real.

No obstante, no todo son ventajas. Como **principal inconveniente** se encuentra la dificultad en encontrar modelos realistas que obtengan un ajuste correcto del sistema, siendo esto consecuencia de la alta complejidad que se observa en la dinámica de los AGVs.

Esta importancia de la obtención y mejora de modelos de simulación e identificación se alinea con el proceso de revolución en el que se encuentra sumergida la industria ahora mismo, la denominada revolución industrial 4.0. Esta ha sido posible gracias a la mejora, desarrollo e implantación de la mecatrónica y de las tecnologías de la información en los procesos productivos de las compañías. Uno de los agentes que ha servido de apoyo y ha ido adquiriendo protagonismo son los vehículos de guiado automático o AGVs. De ahí el interés en desarrollar un trabajo en la temática de estos vehículos, en donde se pueda describir la identificación de modelos computacionales de estos sistemas en su movimiento por el espacio de trabajo.

1.3 Objetivo general y objetivos específicos

El objetivo principal del presente trabajo es comprobar cómo, aun siendo una tarea compleja, es posible la identificación modelos computacionales de la dinámica de un AGV mediante modelos neuronales y paramétricos.

Los objetivos específicos del presente trabajo se basan en la obtención y optimización de los parámetros que describen cada modelo de ajuste mediante **métodos paramétricos** y la comparación de sus resultados frente a los obtenidos mediante **redes neuronales** (NN).

Como datos de partida se emplearán la velocidad de pulsos/s de los encoder de las ruedas izquierda y derecha del AGV, se filtrarán y eliminarán los outliers , el ruido y se transformará la velocidad de pulsos/s a cm/s.

Una vez realizado esto, se llevará a cabo la identificación del sistema mediante los dos métodos diferentes**.**

Una vez obtenidos los modelos con ambos métodos, se compararán y optimizarán los ajustes mediante la selección del modelo que presente una mejor relación entre el ajuste, complejidad y coste computacional. La descripción de cada método de identificación, los modelos utilizados y los resultados obtenidos se explicarán en los correspondientes capítulos del presente trabajo.

1.4 Estructura de la memoria

A continuación se indican los apartados que forman parte de esta memoria así como una breve explicación de lo que se trata en cada uno:

Capítulo 1: Introducción. Se pone en relieve el protagonismo de los AGV en la industria 4.0 así como la importancia de tener modelos computacionales que permitan testear sus algoritmos de control. En este mismo apartado se define la motivación, el objetivo general y los objetivos específicos de este Proyecto Fin de Máster.

Capítulo 2: Estado del arte. Se muestran los tipos de AGV, explicándose el modelo empleado para extraer los datos en este trabajo. Se indica la dinámica de este AGV y se hace una introducción teórica a los distintos tipos de métodos para la identificación de los sistemas.

Capítulo 3: Materiales y datos. Se explica los procedimientos realizados en la manipulación de los datos brutos obtenidos del AGV, como es el filtrado, eliminación de los outliers mediante el empleo de un ventanado móvil, y la aplicación de un filtro Hampel para la eliminación del ruido.

Capítulo 4: Resultados de la identificación del AGV. Se comentan los resultados obtenidos después de la identificación mediante técnicas paramétricas polinómicas y redes neuronales, llevándose a cabo un análisis y comparación entre los métodos.

Capítulo 5: Conclusiones y trabajos futuros. Se describe las principales ideas y conclusiones obtenidas después de la realización del trabajo. Se proponen futuras líneas que se podrían abordar con mayor nivel de detalle en futuros trabajos o investigaciones.

La memoria termina con la bibliografía en donde se citan las fuentes consultadas.

Capítulo 2 - Estado del arte

2.1 Tipos de AGV y características

Los vehículos de guiado automático se pueden clasificar en función de la forma en la que transportan su carga y en función de su sistema de guiado [1]:

- *Configuración según el transporte de la carga:*

  - AGV con remolque: el AGV conforma la unidad tractora de una serie o tren de remolques.

  - AGV con carga unitaria: la carga se transporta encima del propio AGV.

  - AGV con palas elevadoras: el AGV elevará mediante horquillas la carga que deben transportar.

- *Configuración según el sistema de guiado:*

  - AGV filoguiados/inductivos: es uno de los más comunes. Se traza el recorrido que debe seguir el AGV a lo largo del espacio de trabajo mediante un hilo enterrado. Es un sistema poco flexible, pero de gran fiabilidad.

  - AGV con guiado óptico: el guiado del AGV se realiza mediante el seguimiento de un espejo o banda magnética en el suelo que es detectada mediante los sensores ópticos o magnéticos de aproximación del AGV. Este sistema es más flexible que el filoguiado ya que al no depender de cables enterrados, permite una modificación de la ruta más fácilmente.

  - AGV con guiado inercial: el AGV emplea un giroscopio para averiguar su dirección de desplazamiento. Es común la colocación de pequeños dispositivos magnéticos en el suelo, los cuales llevan codificada su posición en la instalación con el objetivo de ayudar a los AGV. Es un sistema más flexible que los filoguiados pero menos que los de tipo laser u ópticos. De igual forma, este sistema es menos preciso que los basados en guiado por cable o laser.

  - AGV con guiado laser: el guiado se realiza mediante un esquema de posiciones en el interior del AGV y un láser giratorio que al ir realizando barridos será el encargado

de ir controlando la distancia existente en cada momento respecto al conjunto de receptores de la ruta. Es necesario la instalación de espejos catadióptricos para que el vehículo pueda posicionarse correctamente. Este sistema es más flexible frente a otro que precise de una instalación fija.

- AGV con guiado basado en encoders: este sistema no suele emplearse de forma aislada, sino que suele encontrarse en combinación con otro sistema de guiado en el AGV. Su principal inconveniente es su sensibilidad al deslizamiento de ruedas, al patinaje del AGV así como, en el caso de que la carga sea muy pesada, a la deformación de las ruedas. Este sistema experimenta acumulación de errores, por lo que si se emplease como único modo de guiado, al cabo de un tiempo el AGV se encontraría en una posición errónea.

- AGV con guiado basado en faros: funcionamiento similar al método de guiado láser, con la diferencia de que en lugar de emplear retro-reflectores se instalan a lo largo del espacio de trabajo focos luminosos, capturando el AGV haces de luz en lugar de emitirlos. Al igual que en el caso laser, este sistema presenta un alta flexibilidad para cambios en la ruta con el inconveniente que los focos emisores de luz son más caros que los retro-reflectores.

- AGV guiado mediante visión artificial: este sistema es el más moderno. Consiste en el empleo de cámaras 360º de visión artificial de tipo industrial, mediante las cuales el AGV recibirá imágenes digitales que la unidad de control procesará, y en función de esta entrada, el vehículo obtendrá información de su posición mediante el reconocimiento de marcas establecidas. El sistema operativo será el encargado de procesar toda la información recogida por los sensores a través de un sistema de comunicación vía radio. Dentro de los sistemas más habituales mediante los cuales el AGV obtiene las órdenes de trabajo se encuentra la banda ancha (433 MHz/868MHz), empleada en los espacios en donde la recepción de la señal se hace más difícil y el Wireless 802.11g.

A medida que el software y la mecatrónica avanzan, aparece en el mercado tecnologías que proporcionan sistemas más flexibles y adaptados a las necesidades y contexto de cada cliente. Se está experimentando una creciente demanda de AGVs robustos, polivalentes y de gran precisión sin necesidad de llevar a cabo la instalación

de ningún elemento externo, permitiendo la reconfiguración de su ruta de forma rápida y sencilla. Mediante el empleo de cámaras, ultrasonidos y sistema LIDAR, los nuevos AGV son capaces de realizar un mapeo 2D y 3D del escenario de trabajo e ir adaptando de forma continua su trayectoria.

2.2 Tipo de AGV empleado para la obtención de los datos

Los datos que se emplearán en el presente trabajo se corresponden a los encoders de la rueda izquierda y derecha de un AGV industrial de tipo triciclo.

Este modelo de AGV puede llevar a cabo el transporte de la carga ubicada en un remolque.

2.2.1 Dinámica del AGV tipo triciclo

EL AGV de tipo triciclo se compone de una rueda delantera y dos ruedas traseras. La unidad tractora puede ser la rueda delantera o las dos traseras, siendo la más habitual la rueda tractora delantera.

La unidad de tracción del AGV se verá influenciada por la distribución de carga del propio AGV, apareciendo en las ruedas un vector de fuerza normal. La rueda tractora generará una fuerza de tracción aplicada en un punto del AGV, la cual deberá ser de magnitud suficiente para permitir el transporte del peso de la carga remolcada más el peso del propio AGV.

A continuación se describe las ecuaciones que rigen el modelo dinámico para un AGV de tipo triciclo [16]:

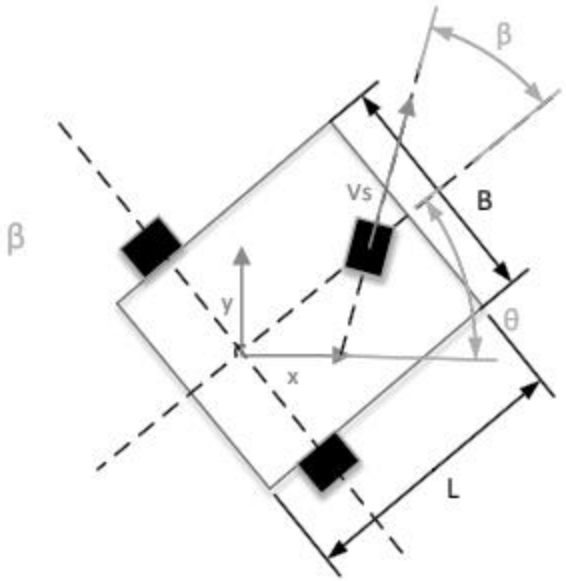

Ilustración 2: Esquema AGV tipo triciclo.[12]

$$\dot{x} = v * \cos(\theta) \tag{2.1}$$

$$\dot{y} = v * \sin(\theta) \tag{2.2}$$

$$\dot{\theta} = \omega \tag{2.3}$$

$$v = v_s * \cos(\beta) \tag{2.4}$$

$$\omega = \frac{v_s}{L} * \sin(\beta) \tag{2.5}$$

$$R_{curva} = L * \cot(\beta) \tag{2.6}$$

En donde los diferentes parámetros del modelo son (Tabla 1):

$\dot{x}$	Posición del centro del eje trasero del AGV en el eje x
$\dot{y}$	Posición del centro del eje trasero del AGV en el eje y
ω	Velocidad de rotación (rad/s)
θ	Ángulo de rotación del cuerpo del AGV respecto al sistema de referencia inercial
v_s	Velocidad lineal de la rueda directora (m/s)
L	Distancia entre la rueda directriz y las ruedas traseras (m)
R_{curva}	Distancia desde el centro del eje trasero hasta el centro instantáneo de rotación (CIR)

Tabla 1: Parámetros del modelo dinámico de la unidad tractora del AGV. [12]

2.3 Identificación de sistemas mediante métodos paramétricos polinómicos

El objetivo de la identificación de sistemas es la obtención de modelos que permitan describir y relacionar las variables medidas de entrada y salida en el AGV.

Dentro de los métodos paramétricos, en el presente trabajo se van a emplear los de tipo polinómico para llevar a cabo la identificación de la unidad de tracción del AGV. Los métodos paramétricos concentran toda la información en la propia estructura del modelo con un número limitado de parámetros [3]. Estos métodos se basan en modelos auto regresivos (AR) empleados para la modelización de series temporales sin entradas [4, 5]:

$$y_n = a_0 + a_1.y_{n-1} + a_2.y_{n-2} + a_3.y_{n-2} + \cdots + a_{na}.y_{na-n} \quad (2.7)$$

Los órdenes *n* y los parámetros *a, b, c, d* y *f* serán ajustados para obtener la serie $\{ yk = y_0, y_1 \ldots . y_n, \ldots \}$.

Dependiendo del método de selección elegido, los polinomios presentan la siguiente forma generalizada:

$$A\,(z^{-1}).\,y_n = \frac{B(z^{-1})}{F(z^{-1})}U_{n-nk} + \frac{C(z^{-1})}{D(z^{-1})}e_n$$

(2.8)

Donde:

y_n = vector de salida muestreado.

U_n = vector de entrada muestreado.

e = perturbaciones o ruidos estimados.

$$A(z^{-1}) = 1 + a_1.z^{-1} + +a_2.z^{-2} + a_3.z^{-3} + \cdots + a_{na}.z^{-na}$$

(2.9)

$$B(z^{-1}) = b_0 + b_1.z^{-1} + +b_2.z^{-2} + b_3.z^{-3} + \cdots + b_{nb}.z^{-nb}$$

(2.10)

$$C(z^{-1}) = c_0 + c_1.z^{-1} + +c_2.z^{-2} + c_3.z^{-3} + \cdots + c_{nc}.z^{-nc}$$

(2.11)

$$D(z^{-1}) = d_0 + d_1.z^{-1} + +d_2.z^{-2} + d_3.z^{-3} + \cdots + d_{nd}.z^{-nd}$$

(2.12)

$$F(z^{-1}) = f_0 + f_1.z^{-1} + +f_2.z^{-2} + f_3.z^{-3} + \cdots + f_{nf}.z^{-nf}$$

(2.13)

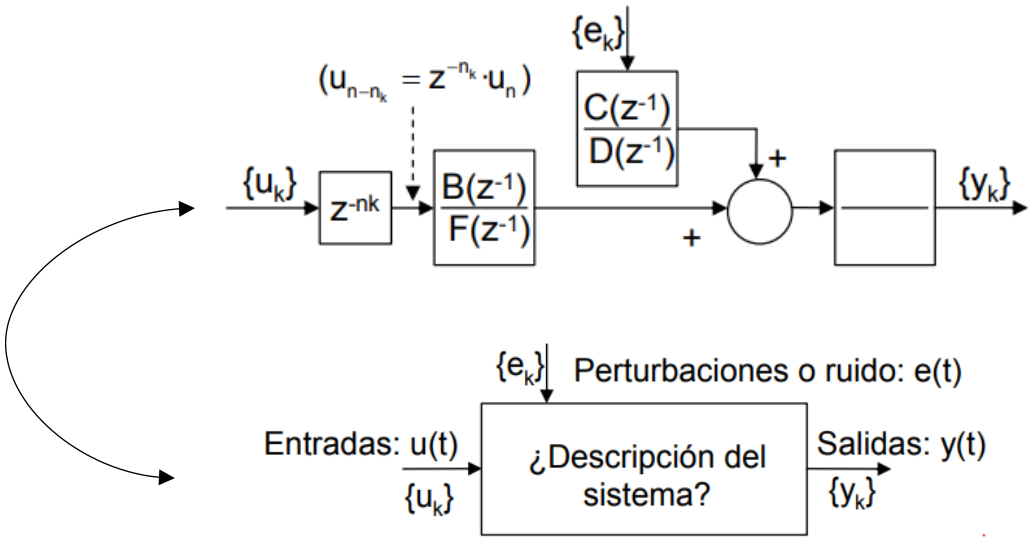

Ilustración 3: Esquema de bloques general para métodos polinómicos.[4]

En la ilustración 3 se puede observar cómo se representa de forma general la estructura de los modelos paramétricos, donde se puede comprobar la secuencia de los vectores de entradas (U_k) y salidas (y_K) muestreadas (T_m: tiempo de muestreo), y las perturbaciones o ruidos a los que se encuentra sometido el sistema (e_K).

Dentro de este sistema de identificación, las principales dificultades se encuentran en la definición y estructura del modelo y en el desarrollo del algoritmo para la estimación de los parámetros del modelo.

Para la determinación de la estructura de estos modelos, se debe seleccionar el orden de los polinomios n_a , n_b, n_c, n_d, n_f y el retardo del sistema n_k. En la práctica totalidad de los sistemas, n_a será igual al número de polos y n_b al número de ceros [4].

Los coeficientes de cada polinomio a, b, c, d ... serán los parámetros que se deberán ajustar mediante alguno de los siguientes métodos de ajuste:

- Mínimos cuadrados: minimiza la suma de los cuadrados de uno y otro lado de la expresión.
- Variable instrumental: se determina a_i y b_i para que el error entre la izquierda y derecha de la expresión no tenga correlación con ciertas combinaciones lineales de las entradas.

- Máxima verosimilitud/ error de predicción: se basa en una minimización iterativa de un criterio o reducción iterativa del error e_k de la expresión.
- Basado en subespacios: este método es empleado para el espacio de estados.

En el presente trabajo se ha optado por la utilización del método de mínimos cuadrados ya que es comúnmente utilizado en los modelos de ajuste de regresión polinomial al minimizar la varianza de los estimadores no sesgados de los coeficientes [17].

A continuación se procede a la descripción de cada método paramétrico polinómico.

2.3.1 Descripción de los modelos

2.3.1.1 ARX (Auto Regressive model with eXternal input)

Para este método sólo se seleccionan los siguientes órdenes de los polinomios n_a, n_b ($n_c = n_d = n_f = 0$) y n_k.

En donde:

n_a = orden del polinomio A (ec 2.9)

n_b = orden del polinomio B (ec 2.10)

n_c = orden del polinomio C (ec 2.11)

n_d = orden del polinomio D (ec 2.12)

n_f = orden del polinomio F (ec 2.13)

n_k = retardo del sistema

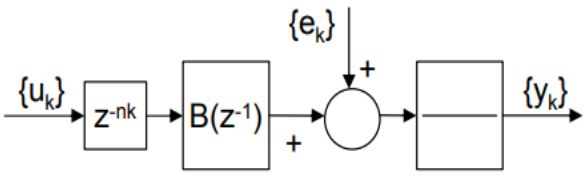

Ilustración 4: Modelo ARX.[4]

$$A(z^{-1}).y_n = \sum_{i=1}^{n} B(Z^{-1}).U_{n-nk} + e_n$$

(2.14)

e_n= perturbaciones o ruido.

Desarrollando más la ecuación 2.14 se obtiene:

$$y_n + a_1.y_{n-1} + a_2.y_{n-2} + \cdots + a_{na}.y_{n-na} = b_0.u_{n-nk} + b_1.u_{n-nk-1} + \cdots + b_{nb}.u_{n-nk-na} + e_n \quad (2.15)$$

Para la obtención de los polinomios A y B será necesario la resolución de un problema de regresión lineal. Si se consideran valores altos de los coeficientes n_a, n_b y n_k, las estimas de estos coeficientes son más consistentes pero puede ocasionar problemas de varianza en presencia de una perturbación significativa. Por otro lado, si son considerados valores de los coeficientes n_a y n_b muy pequeños, en presencia de una perturbación importante o cuando la estructura del modelo sea incorrecta aparecerán problemas a la hora de estimar los valores de estos coeficientes.

2.3.1.2 ARMAX (Auto Regressive Moving Average with eXternal inputs)

Para este método solo se selecciona n_a , n_b , n_c y n_k $(n_d = n_f = 0)$

En donde:

n_a = orden del polinomio A (ec 2.9)

n_b = orden del polinomio B (ec 2.10)

n_c = orden del polinomio C (ec 2.11)

n_d = orden del polinomio D (ec 2.12)

n_f = orden del polinomio F (ec 2.13)

n_k = retardo del sistema

$$A(Z^{-1}).y_n = B(Z^{-1}).U_{n-nk} + C(Z^{-1}).e_n \quad (2.16)$$

Desarrollando más la ecuación anterior se obtiene:

$$y_n + a_1.y_{n-1} + a_2.y_{n-2} + \cdots + a_{na}.y_{n-na} = b_0.u_{n-nk} + b_1.u_{n-nk-1} + \cdots + b_{nb}.u_{n-nk-na} + c_0.e_n + c_1.e_{n-1} + \cdots + c_{nc}.e_{n-nc} \quad (2.17)$$

e_n= perturbaciones o ruido.

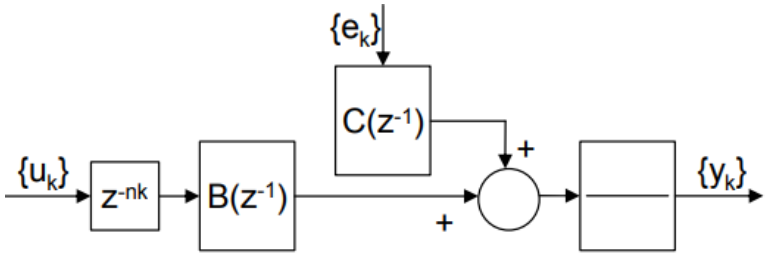

Ilustración 5: Modelo ARMAX.[4]

Generalmente, se seleccionan valores pequeños de los coeficientes n_a, n_b y n_c. La presencia de un polinomio autorregresivo $A(Z^{-1})$ puede originar problemas de sesgo en caso de presencia de un ruido significativo y/o en el desajuste del modelo. No obstante, el polinomio de media móvil $C(Z^{-1})$ en ocasiones puede contrarrestar este efecto.

2.3.1.3 OE (Output Error model)

En este método la fuente de ruido del modelo (e_n) coincide con la perturbación. Es decir, será el error entre la salida actual y la salida libre de ruido. Al igual que en modelos previos, la estimación de los polinomios $B(Z^{-1})$ y $F(Z^{-1})$ precisa resolver un problema de regresión no lineal.

Para este método se selecciona n_b , n_f y n_k ($n_a = n_c = n_d = 0$).

En donde:

n_a = orden del polinomio A (ec 2.9)

n_b = orden del polinomio B (ec 2.10)

n_c = orden del polinomio C (ec 2.11)

n_d = orden del polinomio D (ec 2.12)

n_f = orden del polinomio F (ec 2.13)

n_k = retardo del sistema

Normalmente, se opta por la selección de valores pequeños para n_b y n_f.

$$y_n = \frac{B(Z^{-1})}{F(Z^{-1})}.u_{n-nk} + e_n \qquad (2.18)$$

e_n= perturbaciones o ruido.

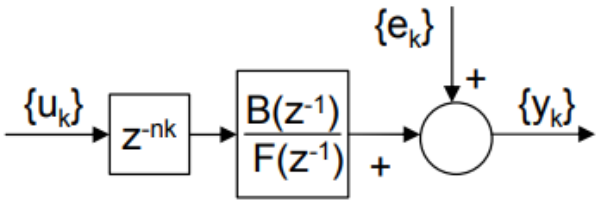

Ilustración 6: Modelo OE.[4]

2.3.1.4 BJ (Box-Jenkins model)

Para este método se selecciona n_b , n_c , n_d , n_f y n_k $(n_a = 0)$

En donde:

n_a = orden del polinomio A (ec 2.9)

n_b = orden del polinomio B (ec 2.10)

n_c = orden del polinomio C (ec 2.11)

n_d = orden del polinomio D (ec 2.12)

n_f = orden del polinomio F (ec 2.13)

n_k = retardo del sistema

$$y_n = \frac{B(Z^{-1})}{F(Z^{-1})}.u_{n-nk} + \frac{C(Z^{-1})}{D(Z^{-1})}.e_n \qquad (2.19)$$

e_n= perturbaciones o ruido.

Nuevamente, la estimación de los valores de los coeficientes de los polinomios $B(Z^{-1})$, $C(Z^{-1})$ y $D(Z^{-1})$ requiere resolver un problema de regresión no lineal. El modelo BJ, proporciona funciones de transferencia independientes para la parte determinista y estocástica del modelo. Este modelo es difícil de estimar ya que precisa de la realización de muchas iteraciones computacionalmente costosas.

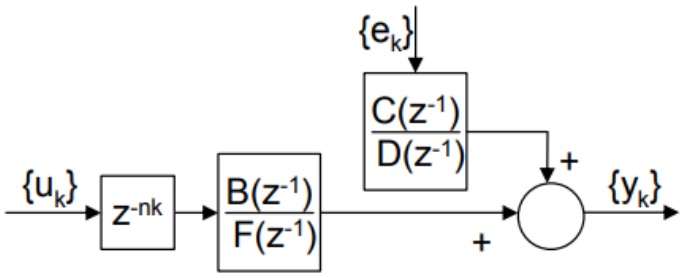

Ilustración 7: Modelo BJ.[4]

2.3.2 Complejidad computacional en modelos paramétricos polinómicos

El coste computacional permite comparar la complejidad de cada método paramétrico en función del número de operaciones elementales que realice durante su tiempo de ejecución [15]. Para llevar a cabo el cálculo de este coste computacional, se descompondrá cada algoritmo como un conjunto de operaciones elementales, en donde:

N_A = número de sumas/restas.

N_M = número de multiplicaciones/divisiones.

n_a, n_b, n_c, n_f = es el orden de cada polinomio.

Para cada técnica paramétrica se define el cálculo del número de operaciones de la siguiente forma:

ARX:

$$N_A = n_a + n_b - 1 \quad (2.20)$$

$$N_M = n_a + n_b \quad (2.21)$$

ARMAX:

$$N_A = n_a + n_b + n_C - 1 \quad (2.22)$$

$$N_M = n_a + n_b + n_C \quad (2.23)$$

OE:

$$N_A = n_b + n_f - 1 \quad (2.24)$$

$$N_M = n_b + n_f \quad (2.25)$$

BJ:

$$N_A = n_b + n_c + n_d + n_f - 1 \quad (2.26)$$

$$N_M = n_b + n_c + n_d + n_f \quad (2.27)$$

Para poder obtener un valor que permita realizar la comparación de la complejidad computacional entre las diferentes técnicas, y que por tanto se encuentre relacionado con número de operaciones elementales de cada una, se va a establecer una regla, en donde:

- Sumas y restas tendrán un coste (T_{CA}) igual a 1.
- Multiplicación y división: la relación (K_M) entre el coste de una operación de multiplicación/división (T_{CM}) frente a una suma/resta dependerá de la arquitectura del elemento electrónico empleado para la ejecución del código y realización de las operaciones [15].

  La relación entre el coste entre ambas operaciones se encontrará definida por la constante K_M, la cual será siempre mayor a 1.
- Función exponencial y operación exponencial:

  En la relación para la función exponencial ($T_{C\theta}$) y relación para la operación exponencial (K_E), a diferencia del caso de la multiplicación y división, los costes de la función exponencial ($T_{C\theta}$) y de la operación exponencial (T_{CE}) solo vienen definidas por el algoritmo empleado y no por la arquitectura del tipo de dispositivo utilizado.

A continuación se resume el coste de cada operación:

- Coste suma/resta:

$$T_{CA} = 1$$

(2.28)

- Coste multiplicación/división:

$$T_{CM} = K_M * T_{CA}$$

(2.29)

- Coste función exponencial:

$$T_{C\theta} = K_\theta * T_{C\theta} = K_\theta$$

(2.30)

- Coste operación exponencial:

$$T_{CE} = K_E * T_{CE} = K_E$$

(2.31)

En donde a su vez:

$$K_M = \frac{n^{1.465}}{n} = n^{0,465}$$

(2.32)

$$K_\theta = n^{1/2} * n^{0.465} = n^{0,965}$$

(2.33)

$$K_E = k * n^{0,465}$$

(2.34)

Asumiendo un algoritmo Toom-Cook o Toom-3 [20], se definen n=3 y k =3. Habiendo fijado los valores de n y K, ya se podrá obtener el valor de las diferentes constantes:

$$K_M = 3^{0.465} = 1,667$$

(2.35)

$$K_\theta = 3^{0.965} = 2,886$$

(2.36)

$$K_E = 3 * 3^{0,465} = 5$$

(2.37)

2.4 Identificación de sistemas mediante redes neuronales

2.4.1 Concepto de neurona artificial

Una neurona artificial es un modelo matemático basado en el comportamiento de una neurona biológica y en cómo forman una red para componer la estructura del cerebro.

Un modelo neuronal (habitualmente llamado nodo o unidad) se compondrá principalmente de los siguientes elementos [22]:

- Conjunto de entradas (x_1, x_2 ,…., x_n): se corresponden con el estímulo que recibe la neurona.

- Pesos sinápticos asociados a cada entrada (w_1, w_2,…w_n) y que se irán actualizando en el aprendizaje.

- La función de agregación (∑)

- La función de activación f (lineal, sigmoidal…etc): seleccionada en función del trabajo a realizar por la neurona.

- Salida (Y): Respuesta al estímulo.

En donde:

$$Y = f(\sum_{i=1}^{n} wi * xi)$$

(2.38)

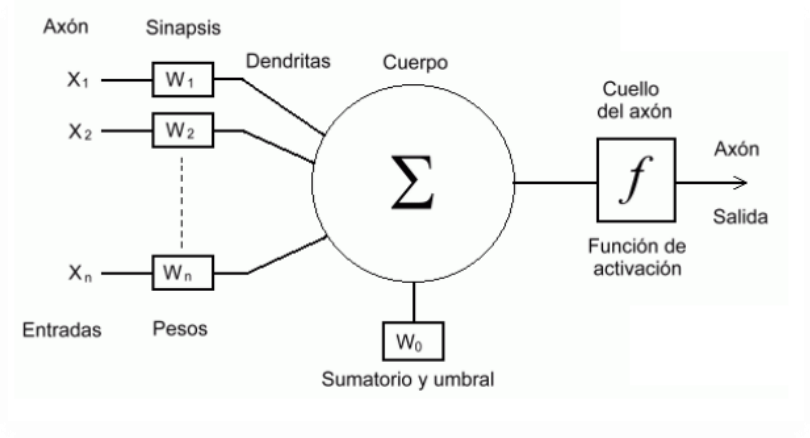

Ilustración 8: Modelo neuronal-Perceptrón de McCulloch-Pitts (1943). [22]

El estudio del perceptrón rápidamente dio a conocer sus limitaciones en cuanto a su incapacidad de separar datos linealmente no separables. Para llevar a cabo esto, es necesario el acoplamiento de más neuronas, iniciándose así el origen de las redes neuronales multicapas.

2.4.2 Descripción de una red neuronal

Una red neuronal se puede definir como un modelo computacional basado en un conjunto de unidades neuronales simples, donde cada una de las cuales puede disponer de una regla o condición diferente. Dentro de esta red neuronal, se distribuye un número *n* de neuronas a lo largo de una serie de capas o niveles. La clasificación de estas capas se hace con relación a la situación de éstas dentro de la propia red, pudiéndose distinguir [6]:

- Capa de entrada: recibe de forma directa la información proveniente de una fuente externa a la propia red.

- Capa de salida: la información de salida de la red hacia el exterior.

- Capa(s) oculta(s): son capas internas que no se encuentran en contacto con el exterior. El número de ellas no es fijo, por lo que puede ir desde 0 a un número elevado, existiendo diferentes tipos de conexión entre ellas. Estos tipos de conexión entre las capas así como el número de éstas formarán las diferentes topologías de la red neuronal. Generalmente redes con mayor profundidad (mayor número de capas ocultas) disponen de mayor flexibilidad a la vez que se minimiza la aparición del sobreajuste u overfitting [7].

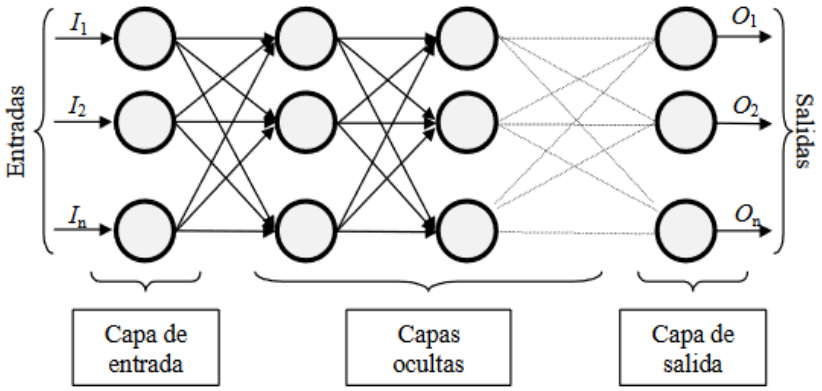

Ilustración 9: Esquema de una red neuronal. [6]

Las neuronas pueden tomar valores reales o binarios.

En función de la organización y disposición de las neuronas hacen que se puedan distinguir diferentes topologías de la red.

- Redes monocapa: son redes con una única capa de neuronas con conexiones entre ellas.

- Redes multicapa: se dispone de un conjunto de neuronas agrupadas en varios niveles o capas. Una neurona pertenecerá a una capa concreta en función del origen de la señal de entrada y el destino de la señal de salida. A cada neurona se sumarán las señales que le llegan multiplicada por los pesos, y será a dicha suma a la que se le aplicará la función de activación. Generalmente una neurona recibirá la señal de entrada desde la capa anterior y transmitirá la señal de salida a la capa siguiente (conexión feedforward). No obstante, también es posible conectar la señal de salida a las neuronas de capas anteriores formando las conexiones feedback.

La neurona tratará el conjunto de valores de entrada como uno solo, es decir, como una entrada global. Esta función calcula el sumatorio del producto de cada entrada simple por el peso (parámetros).

$$Input_{Global} = (in_{i1}.w_{i1}) + (in_{i2}.w_{i2}) + \cdots + (in_{in}.w_{in}) \quad (2.39)$$

En donde:

in_i = Entrada simple *n*

w_i = Peso neurona *n*

La función de activación será la encargada de calcular el estado de actividad de la neurona, transformando la entrada global a un valor de activación. Es decir, una neurona puede encontrarse activa o inactiva. Esta función de activación será obtenida transformando la entrada global (gin_i) menos el umbral (Θ_i)) en un estado de activación: inactivo (0 o -1) o estado activo (1). Algunas de las funciones de activación más comúnmente empleadas [6] son la lineal, la sigmoide, ReLU, etc. El valor resultante de esta función se corresponderá con la salida de la neurona.

En la aplicación de una red neuronal existen dos fases, la fase de entrenamiento para enseñarle la función que se quiere que implemente y la de validación.

- *Fase de entrenamiento*: en esta fase se emplean un conjunto de datos de entrenamiento o patrones para determinar los pesos (parámetros) que definirán el modelo de red neuronal. Estos pesos se calcularán de forma iterativa en función de los valores de los datos de entrenamiento con la intención de minimizar el error cometido en la diferencia entre la salida real obtenida y la salida deseada.

- *Fase de prueba:* durante la fase de entrenamiento se corre el riesgo de ajustar demasiado el modelo a las particularidades concretas del conjunto de datos de entrenamiento, originándose una pérdida de flexibilidad y generalización de aprendizaje para modelos nuevos (sobreajuste), por lo que se debe emplear en esta fase de prueba otro conjunto de datos diferente a los del entrenamiento, denominado como *datos de validación.*

2.4.3 Aprendizaje en una red neuronal

El aprendizaje de una red neuronal se basa en el proceso por el cual modifica sus pesos a consecuencia de una información de entrada, traduciéndose este aprendizaje

en la modificación, eliminación o creación de nuevas conexiones entre las diferentes neuronas. En función del criterio que es empelado para la modificación de los pesos, se puede clasificar el aprendizaje en [6, 21]:

- 1) Aprendizaje supervisado: los datos para el entrenamiento están formados por varios patrones de entrenamiento entrada-salida. La salida es conocida por lo que el entrenamiento se ve influenciado por la supervisión de un maestro. Los pesos se adaptan de la siguiente forma:

$$w_{ij}^{m+1} = w_{ij}^{m} + \Delta w_{ij}^{m}$$

(2.40)

Dentro de este tipo den entrenamiento hay tres formas de llevarlo a cabo, siendo estas:

1.1 Aprendizaje por corrección del error: Se ajustan los pesos de las conexiones entre las neuronas en función de la diferencia entre los valores deseados y los de salida en la red.

1.2 Aprendizaje por refuerzo: Se basa en no proporcionar a la red el valor del conjunto completo de las salidas, es decir, no indicar de forma exacta la salida que se desea ante una determinada entrada durante el entrenamiento. En este caso, el maestro o supervisor indicará mediante una señal de refuerzo si la salida obtenida se ajusta a la deseada (-1: fracaso, +1: éxito) y en función de esta señal se irán ajustándose los pesos mediante un mecanismo basado en probabilidades.

1.3 Aprendizaje estocástico: consiste en la realización de cambios aleatorios en los valores de los pesos e ir evaluando su efecto a partir de la salida deseada y la distribución de probabilidad.

- 2) Aprendizaje no supervisado: para este aprendizaje, el conjunto de datos de entrenamiento consiste solamente en patrones de entrada mediante los cuales la red es entrenada, es decir, la red realiza su aprendizaje únicamente en función de las experiencias recogidas de patrones de entrenamiento anteriores. Ejemplos de este tipo de aprendizaje son:

2.1 Aprendizaje de Hebb: se refuerza el peso que conectan los dos nodos excitados simultáneamente.

2.2 Regla de aprendizaje competitivo: en esta regla si un patrón pertenece a una clase ya conocida, la inclusión de este nuevo patrón matizará la representación de esta clase. Por el contrario, si no pertenece a una clase previamente conocida, se llevará a cabo el ajuste de los pesos y estructura de la red neuronal para llevar a cabo el reconocimiento de esta nueva clase.

Para ambos procesos de aprendizaje, tanto supervisado como no supervisado, una vez que los pesos se mantienen estables, se puede decir que la red ha finalizado este aprendizaje.

2.4.4 Complejidad computacional en redes neuronales

Al igual que para el caso de las técnicas paramétricas polinómicas, será calculado el coste computacional para la red neuronal con la que mejor ajuste se obtenga en cada encoder.

Como la red neuronal estará formada por un número b de neuronas en la capa oculta, en el cálculo de la complejidad computacional se deberá realizar de forma aislada para una neurona y multiplicar su valor por las b neuronas que conforman la capa oculta y por el número de capas ocultas c, para de esta forma, calcular el coste computacional global de la red.

Las operaciones que se realiza en una neurona de la capa oculta son las siguientes:

$$Y = \sum (w_i * i_i) + bias$$

(2.41)

En donde:

w_i = peso asociado a cada valor de entrada

i_i = entrada

Ahora, una vez calculado el sumatorio de los pesos asociados a las entradas, se introducirá el valor en la función de activación para que en función del resultado esta neurona se "dispare" o no.

La función de activación que se empelará en la capa oculta es la función sigmoidal, la cual posee la siguiente expresión:

$$\frac{1}{1+e^{-Y}}$$

(2.42)

Ahora mediante un proceso simple de sustitución, se obtendrá la expresión a la salida de la neurona:

$$\frac{1}{1+e^{-\sum(w_i*i_i)+bias}}$$

(2.43)

Observando la expresión anterior en términos de coste computacional, se puede comprobar como el mayor coste vendrá asociado al cálculo del sumatorio de la exponencial, por esto se aproximará el valor del coste como el obtenido para esta operación.

Una vez indicado lo anterior y conociendo el coste asociado a las sumas y funciones exponenciales indicados en el apartado 2.3.2, se podrá obtener el valor del coste por neurona:

$$T_c\, neurona = (a * K_M + a * T_{Ca} + K_M) + K_\theta$$

(2.44)

Ahora se obtendrá el cote total sabiendo que el número de operaciones totales efectuadas serán:

nº operaciones totales = nº operaciones en una neurona x nº neuronas en capa oculta x nº capas ocultas

$$T_c\, red = \big((a * K_M + a * T_{Ca} + K_M) + K_\theta\big) * b * c$$

(2.45)

En donde para las dos expresiones anteriores:

- a= nº de entradas.
- b= nº neuronas en la capa oculta.
- c= nº capas.
- T_{CA} = coste de la suma =1 (definido en la ec 2.28).
- K_M = coste de la función suma =1,667 (definido en la ec 2.35).
- K_θ = coste de la función exponencial= 2.886 (definido en la ec 2.36).

2.5 Artículos científicos relacionados

Fruto del interés y desarrollo que está generando el campo de la robótica y control, son numerosos los autores que han realizado estudios relacionados con la identificación y modelado de estos sistemas.

El modelado de un AGV mediante ecuaciones es una tarea compleja debido a su alta dinámica. No obstante, autores como *Sánchez et al.* [12] abordan en su trabajo este aspecto en donde obtienen modelos orientados al control y llevan a cabo su validación mediante el empleo de la simulación. Otros autores, como Sierra-García y Santos [16] proceden a describir y modelar matemáticamente diferentes subsistemas de un AGV triciclo-diferencial, como el sistema de control de la carga, sistema de conducción, sistema de seguridad y sistema de guiado, identificando además las interacciones y representándolas en la arquitectura que se propone para ese AGV, obteniéndose unos rendimientos esperados para los modelos.

Autores como *Constaín, at al.* [23] han llevado a cabo la identificación paramétrica de un robot obteniendo bajos errores en el seguimiento de su trayectoria.

Otros autores, *Torres, et al.* [24], han procedido a la identificación mediante una red neuronal dinámica de un sistema robótico (pierna con dos articulaciones de revolución) estudiando los aspectos de entrenamiento e inicialización de la red para obtener una mejor identificación del sistema.

Existen estudios más novedosos como el realizado por Sierra-García *et al. [15]*, en donde se propone la combinación de ambos mundos, es decir, la mezcla de técnicas paramétricas con redes neuronales y neuro-difusas para la identificación de un sistema autónomo aéreo (UAV). De esta forma, se probó que se puede combinar lo mejor de cada método: la simplicidad proporcionada por las técnicas paramétricas y la flexibilidad de las técnicas de soft computing.

Capítulo 3 - Materiales y datos

3.1 Filtrado y procesado inicial de los datos

En un inicio se parte de un conjunto de datos facilitados en una hoja Excel en la que aparecen distribuidas en diversas columnas a lo largo del tiempo los valores de señal del encoder derecho, señal del encoder izquierdo, el setpoint de la rueda derecha y el setpoint de la rueda izquierda.

	A	B	C	D	E
1	t	ED	EI	SPED	SPEI
2	0				
3	0,001009				
4	0,001974	0	0		
5	0,002938	0	0		
6	0,003501	0	0		
7	0,004015	0	0		
8	0,004034	0	0		
9	0,006027	0	0		
10	0,006991	0	0		
11	0,008035	0	0		
12	0,00901	0	0		
13	0,009965	0	0		
14	0,013312	0	0		
15	0,013344	0	0		
16	0,01342	0	0	16384	16384
17	0,013568	0	0	16384	16384
18	0,013714	0	0	16384	16384
19	0,013984	0	0	16384	16384
20	0,015037	0	0	16384	16384
21	0,016	0	0	16384	16384
22	0,017045	0	0	16384	16384

Ilustración 10: Datos de origen.

Donde:

- t: valores del tiempo (s).
- EDp: valores velocidad del encoder derecho (pulsos/s).
- Elp: valores velocidad del encoder izquierdo (pulsos/s).
- SPED: valores del setpoint del encoder derecho (valor consigna sin unidades).
- SPEI: valores del setpoint del encoder izquierdo (valor consigna sin unidades).

A continuación se muestran representados de forma acumulativa los valores de pulsos proporcionados en el encoder izquierdo y derecho a medida que se avanza en

el tiempo. Estos datos son los proporcionados originalmente y no se encuentran filtrados ni procesados, pudiéndose comprobar rápidamente al observar las gráficas la aparición de valores que se van de escala debido a algún error en la adquisición de estos datos.

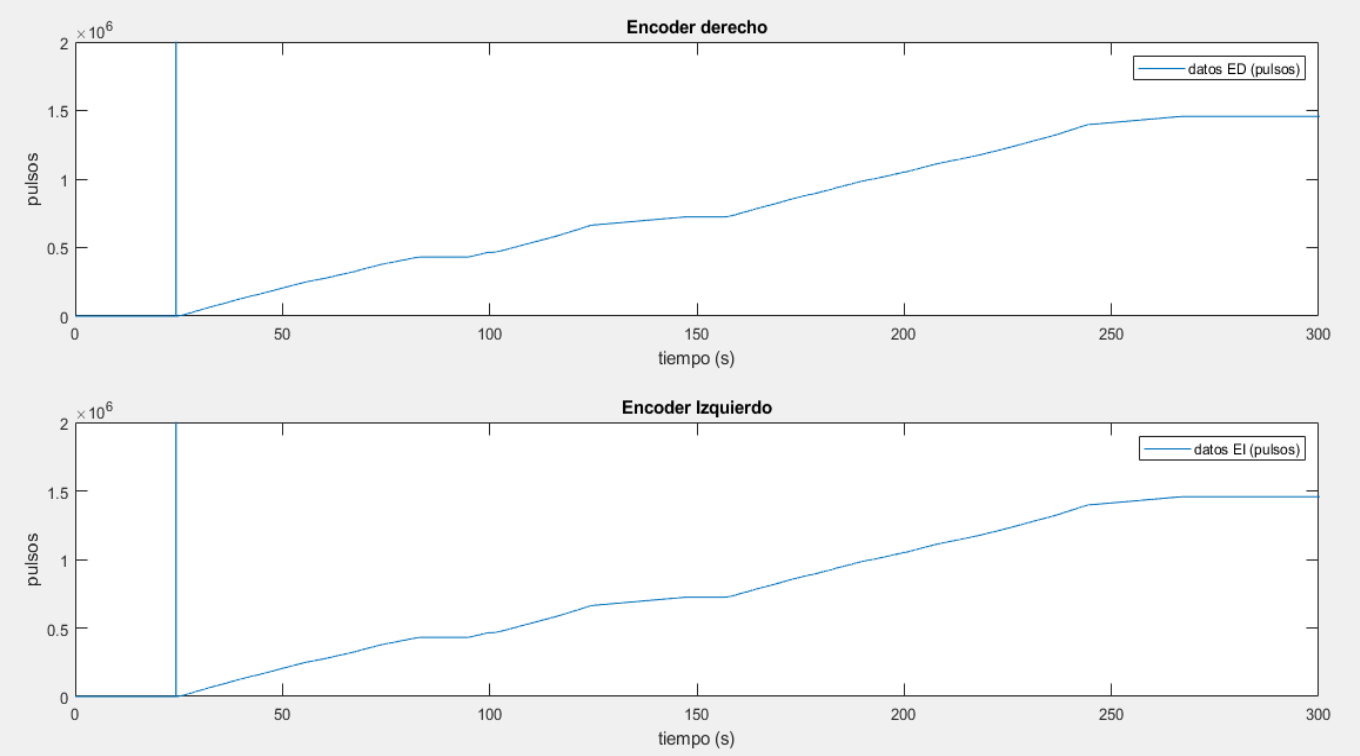

Ilustración 11: Datos de partida ED y EI.

Observando más detalladamente los datos raíz suministrados, se comprueba la existencia de un número elevado de outliers y valores de tiempo que se repiten a lo largo de las celdas, siendo necesario por tanto el filtrado y posterior eliminación de estos datos erróneos y duplicados. Para esta operación, se seguirá el siguiente diagrama de flujo:

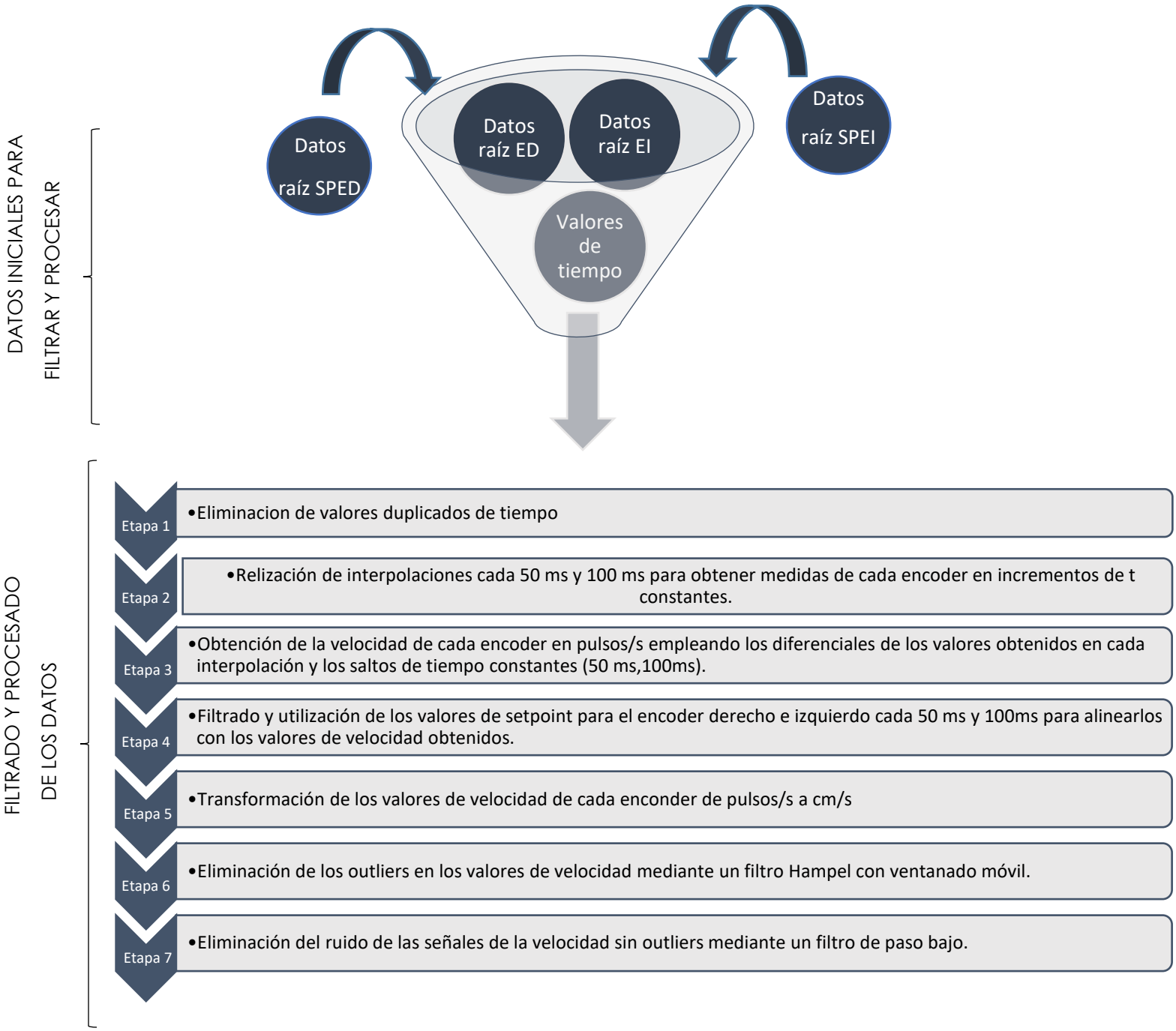

Ilustración 12: Diagrama de flujo para el filtrado y procesado de los datos.

A continuación, se procede a la explicación de cada etapa del diagrama de flujo con mayor nivel de detalle, a la vez que se va comentando los valores que se van obteniendo en cada etapa:

Etapa 1:

Para poder trabajar con los datos iniciales en Matlab, se debe realizar un primer paso en el que se eliminen los valores de una fila de tiempo repetido evitando así duplicidades.

Etapa 2:

Con el objetivo de obtener medidas de cada encoder en incrementos de tiempo constantes, se procede a la obtención de los valores mediante la realización de dos interpolaciones, la primera cada 50 ms y la segunda cada 100 ms.

Etapa 3:

A continuación se obtiene la velocidad de cada encoder (pulsos/s), con la fórmula:

$$V\left(\frac{pulsos}{s}\right) = \frac{(Ei-Ej)}{(tj-ti)} \qquad (3.1)$$

En donde: Ei = Valor de la señal de encoder en el instante i, Ej = valor de la señal del encoder en el instante j, ti= valor del tiempo en el instante i, tj= valor del tiempo en el instante j.

Se calculará el diferencial ente los valores del encoder obtenidos mediante las interpolaciones a 50 ms y 100 ms.

Una vez obtenidos los vectores fila con los valores diferenciales de (i50j-i50i) y (i100j-i100i), y conociendo que los saltos de tiempo de interpolación son constantes, se podrá calcular la velocidad con la ec 3.1, generándose dos vectores fila por encoder, denominados VEDp50, VEIp50, VEDp100 y VEIp100.

Etapa 4:

Para poder representar de forma correcta la comparativa entre los valores del SPED vs VED50t/VED100t y SPEI vs VEI50t/VEI100t a lo largo del tiempo, se deben emplear sólo los valores de SPED y SPEI en los mismos instantes de tiempo en el que se han calculado los valores de la velocidad, es decir, cada 50ms y cada 100ms (disponiendo en ambos casos del mismo número de valores que el vector velocidad).

Encoder derecho

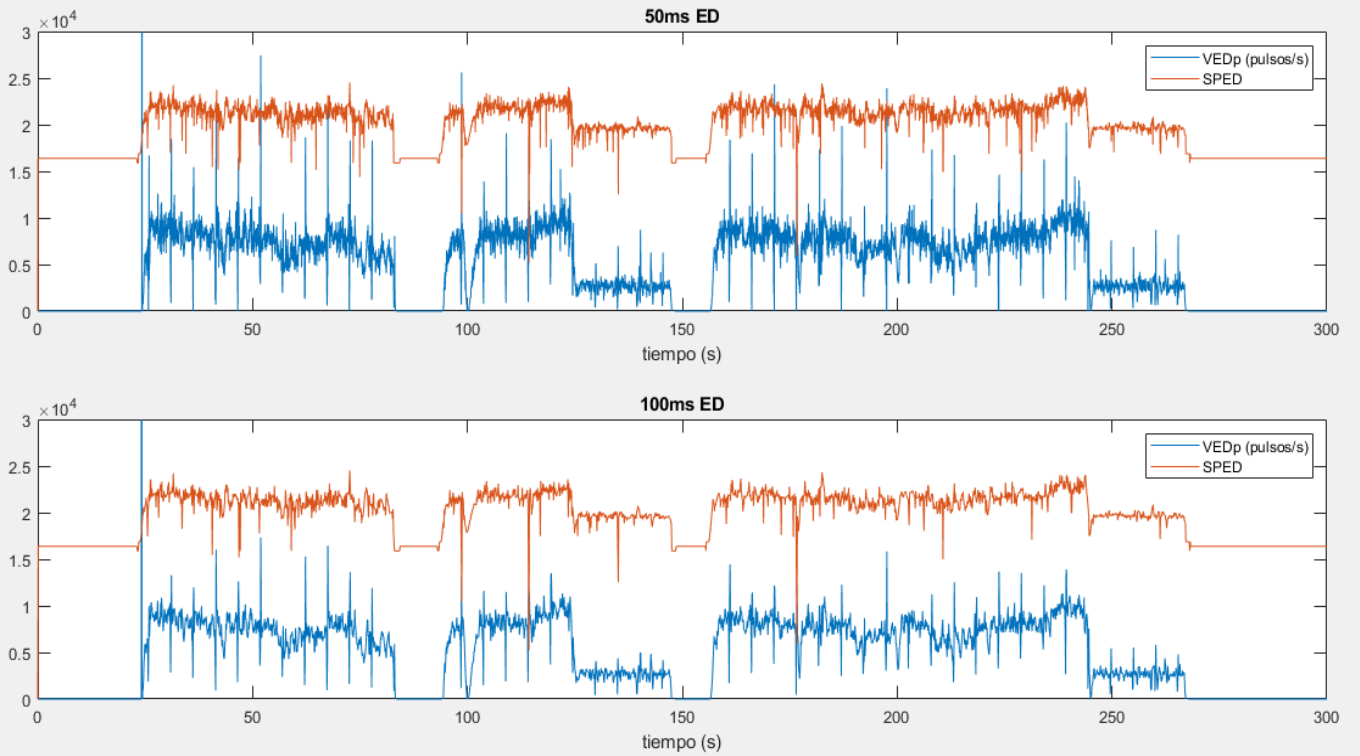

Ilustración 13: Representación datos Setpoint vs Velocidad (pulsos/s). Encoder derecho.

En la anterior ilustración, se puede observar cómo se encuentran representados los valores obtenidos para el encoder derecho con las interpolaciones para 50 ms y 100 ms del setpoint (líneas en rojo SPED: Set Point Encoder Derecho) vs los valores de las lecturas de la velocidad (líneas en azul VEDp: Velocidad Encoder Derecho [pulsos/s]). Se puede comprobar una alta presencia de ruido y outliers en las gráficas azules de la velocidad.

Encoder izquierdo

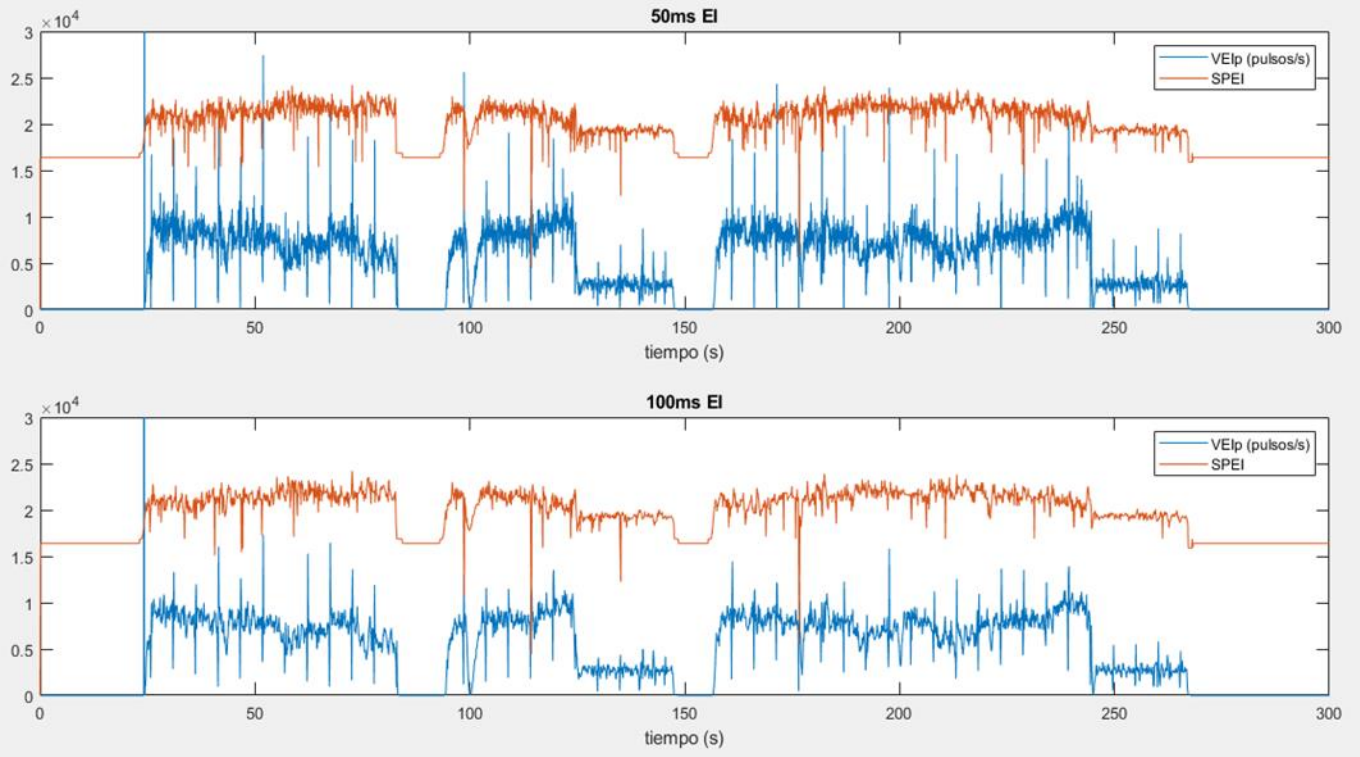

Ilustración 14: Representación datos Setpoint vs Velocidad (pulsos/s). Encoder Izquierdo.

Al igual que se ha realizado previamente con el encoder derecho, se realiza el mismo procedimiento para el encoder izquierdo, mostrándose los resultados en la ilustración 14. Se representan los valores obtenidos para el encoder izquierdo con las interpolaciones para 50 ms y 100 ms del setpoint (líneas en rojo SPEI: Set Point Encoder Izquierdo) vs los valores de las lecturas de la velocidad (líneas en azul VElp: Velocidad Encoder Izquierdo [pulsos/s]). Se comprueba nuevamente una alta presencia de ruido y outliers en las gráficas de la velocidad.

Etapa 5:

Para facilitar la manipulación de los datos evitando escalas de órdenes elevados en los valores de la velocidad, y conociendo que **la relación entre pulsos en cada encoder a cm recorridos por la rueda es 3,7625 x10^{-3}, se realizará la transformación de la velocidad en pulsos/s a cm/s** mediante la siguiente expresión:

$$V\left(\frac{cm}{s}\right) = V\left(\frac{pulsos}{s}\right) * 3{,}7625\ \text{x}10^{-3}\ (cm/pulsos) \quad (3.2)$$

Etapa 6:

Es esta etapa se procederá a la **eliminación de los outliers** previamente observados. Para ello, el primer paso ha sido identificar estos puntos discordantes dentro de los vectores de valores de la velocidad. Se ha empleado un filtro Hampel con un ventanado móvil que permite la detección de estos valores atípicos. Este filtro marcará un valor como outlier si se encuentra a más de tres desviaciones absolutas medias de la mediana local resultante en función de la longitud especificada en la ventana.

Este filtro consiste en una variación de la regla de "tres-sigma" estadística, la cual es robusta frente a valores atípicos. Dada una secuencia $x_1, x_2 \ldots x_n$ y una ventana deslizante de longitud K, define punto a punto las desviaciones estándar y mediana empleando para ello las siguientes expresiones [13]:

- *Mediana*

$$m\ (mediana) = L_i + \frac{\frac{N}{2} - F_{i-1}}{fi} a_i \quad (3.3)$$

En donde:

L_i = límite inferior de la clase donde se encuentra la mediana.

$\frac{N}{2}$= semisuma de las frecuencias absolutas.

F_{i-1} = frecuencia acumulada previa a la clase mediana.

fi= frecuencia absoluta de la clase mediana.

a_i= amplitud de la clase.

- *Mediana local*

$$m_{local} = mediana(x_{i-k}, x_{i-k+1}, x_{i-k+2}, \dots, x_i, \dots x_{i+k-2}, x_{i+k-1}, x_{i+k})$$

(3.4)

- *Desviación estándar*

$$\sigma_i = k\ mediana(|x_{i-k} - m_i|, \dots, |x_{i+k} - m_i|)$$

En donde k:

$$k \approx 1{,}4826 = \frac{1}{\sqrt{2}\ erfc^{-1}(\frac{1}{2})} \qquad (3.5)$$

La desviación absoluta media será definida como el coeficiente entre σ_i/k.

Dada una muestra x_i, si se cumple que:

$$|x_s - m_i| > [n_\sigma\ x\ \sigma_i]$$

El filtro Hampel declarará al valor x_s como outlier y lo reemplazará por m_i (mediana local).

Por otro lado, si $\sigma_i = 0$, el filtro Hampel se comportará como un filtro de mediana normal.

En este filtrado se ha empleado como ventana móvil [25, 10], es decir, los 25 valores anteriores de la serie del valor actual y los 10 siguientes, haciendo la mediana con un total de 36 valores. Se ha empleado este intervalo por observarse después de numerosas pruebas que era la ventana que mejor resultados proporcionaba.

A continuación, se representan para cada encoder los valores obtenidos de la velocidad en cm/s con el filtrado y eliminación de outliers vs los mismos valores sin el filtrado.

Encoder derecho

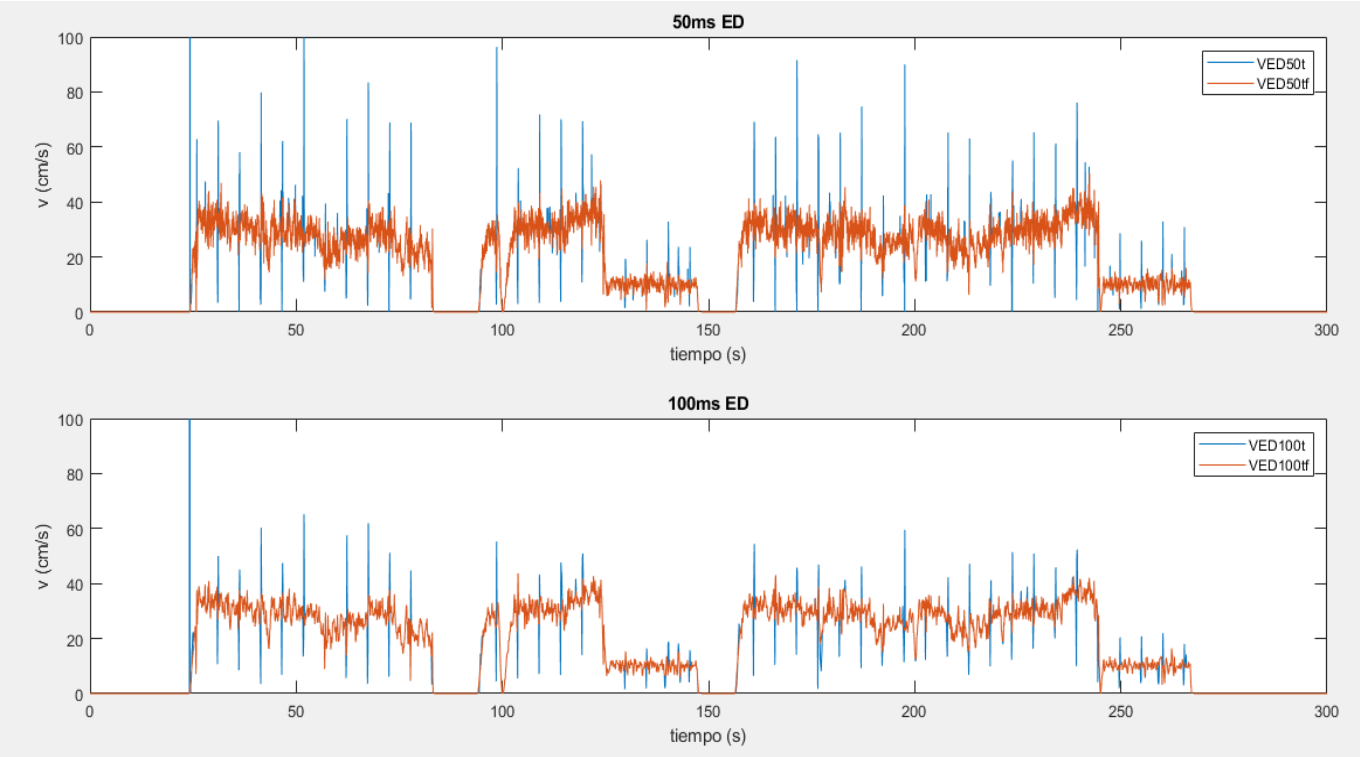

Ilustración 15: Velocidad Encoder derecho con outliers vs sin outliers para 50 y 100ms.

Se puede observar la comparación entre los valores de la velocidad (cm/s) obtenidos mediante las lecturas del encoder derecho con los outliers eliminados (líneas rojas VED50tf, VED100tf) vs sin los outliers eliminados (líneas azules VED50t, VED100t). Se puede comprobar como la eliminación de los outliers ha sido muy efectiva en ambas gráficas.

Se procede de la misma forma con los valores para el encoder izquierdo:

Encoder Izquierdo

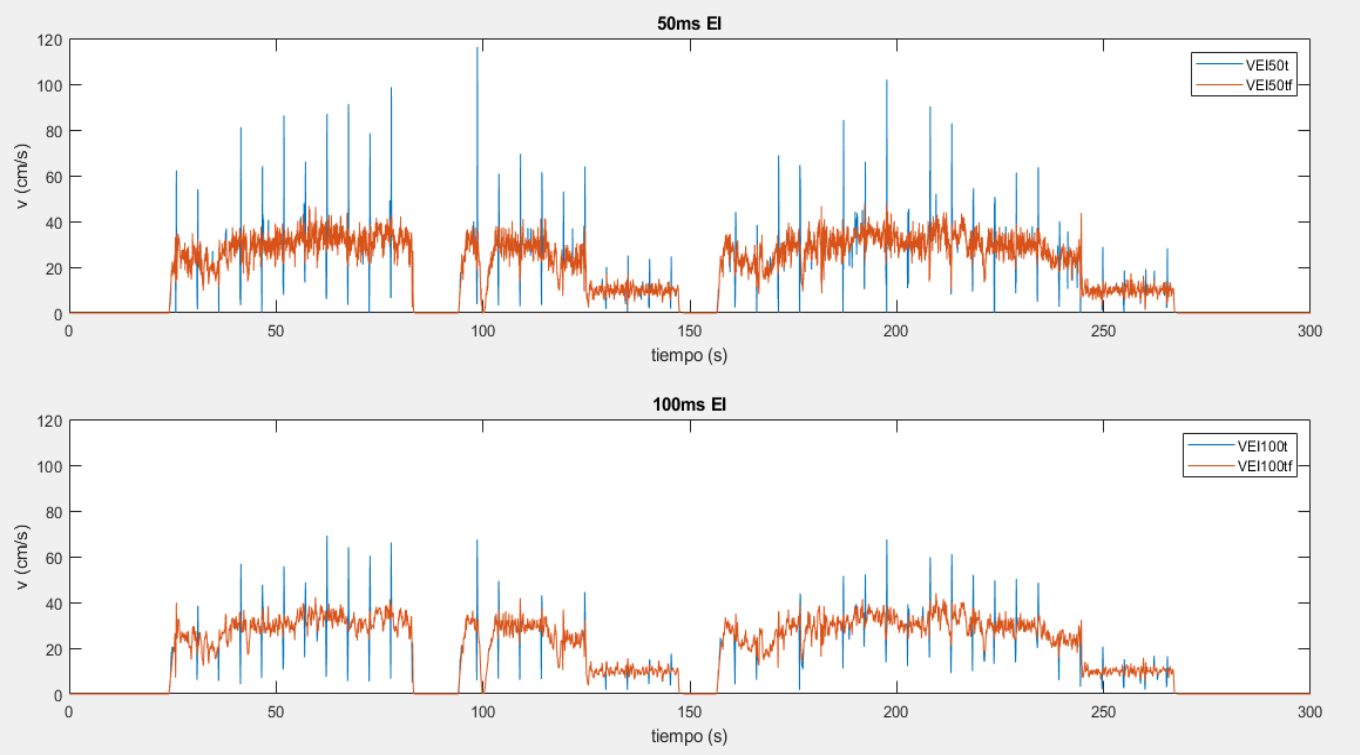

Ilustración 16: Velocidad Encoder izquierdo con outliers vs sin outliers para 50 y 100ms.

Al igual que lo explicado en la anterior ilustración con los valores del encoder derecho, en esta gráfica se representa nuevamente los valores de la velocidad en cm/s con los oultiers eliminados en las líneas rojas (VEI50tf, VEI100tf) vs la velocidad sin la eliminación de los outliers en las líneas azules (VEI50t, VEI100t).

Nuevamente, se puede comprobar como los outliers han sido eliminados de forma muy efectiva.

Etapa 7:

Aunque la eliminación de los outliers ha sido muy satisfactoria, se observa la permanencia de un elevado ruido en las gráficas resultantes. Por este motivo se decide minimizarlo mediante la realización de un filtrado de las señales sin outliers con un filtro de paso bajo. Este filtro utilizará el promedio móvil con coeficientes del filtro iguales al

recíproco del paso del tiempo, obteniendo de esta forma a su salida un suavizado de la señal de entrada [14].

$$y_s(i) = \frac{1}{2N+1}(y(i+N) + y(i+N-1) + \cdots + y(i-N) \qquad (3.6)$$

En donde:

$y_s(i)$ = valor suavizado para el punto de datos i-ésimo.

N = número de puntos de datos vecinos en ambos lados de $y_s(i)$.

$2N + 1$ = intervalo de valores.

A continuación se muestra las señales filtradas obtenidas:

Encoder derecho

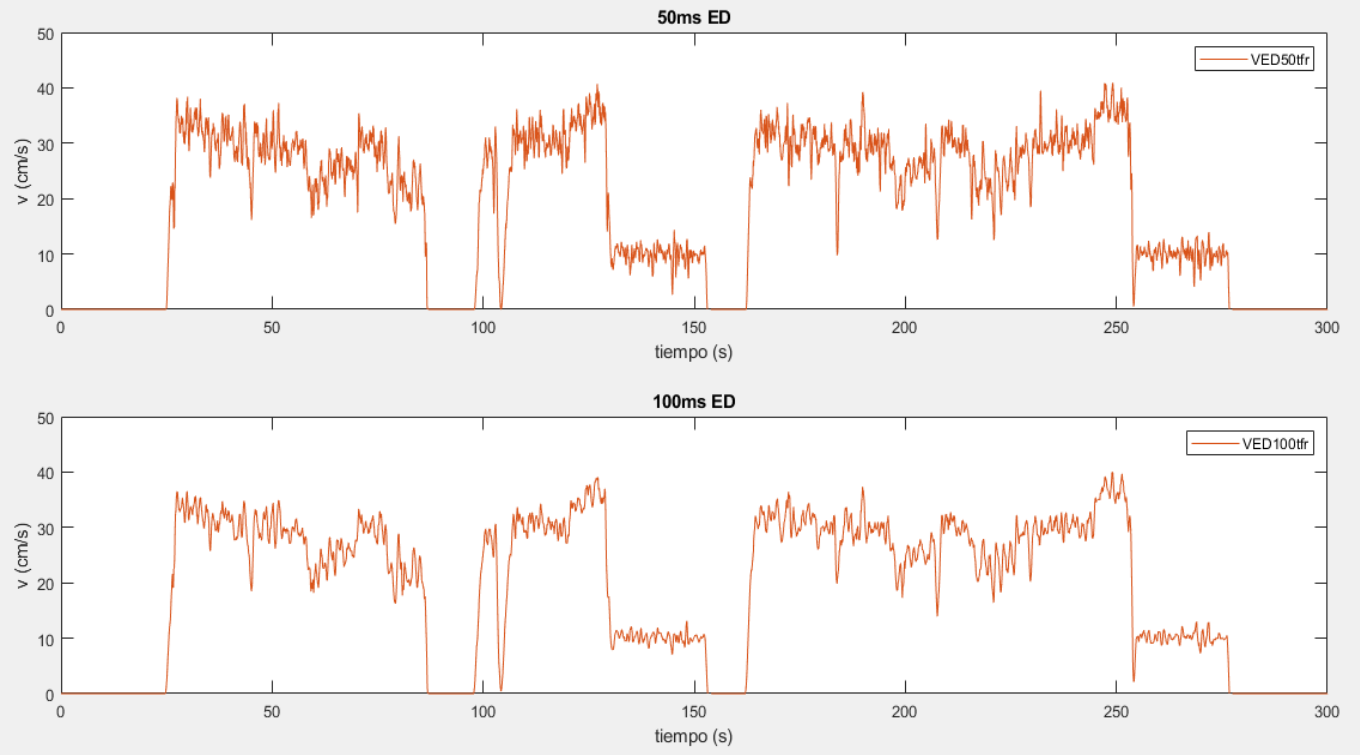

Ilustración 17: Representación de los valores de velocidad (cm/s) a la salida del filtro de paso bajo para el encoder derecho.

Se comprueba como las señales de la velocidad a la salida del filtro de paso bajo para el encoder derecho (VED50tfr, VED100tfr) han sido suavizadas respecto a la señal de entrada de la velocidad sin outliers que se muestra en la ilustración 15 (gráfica roja).

Encoder izquierdo

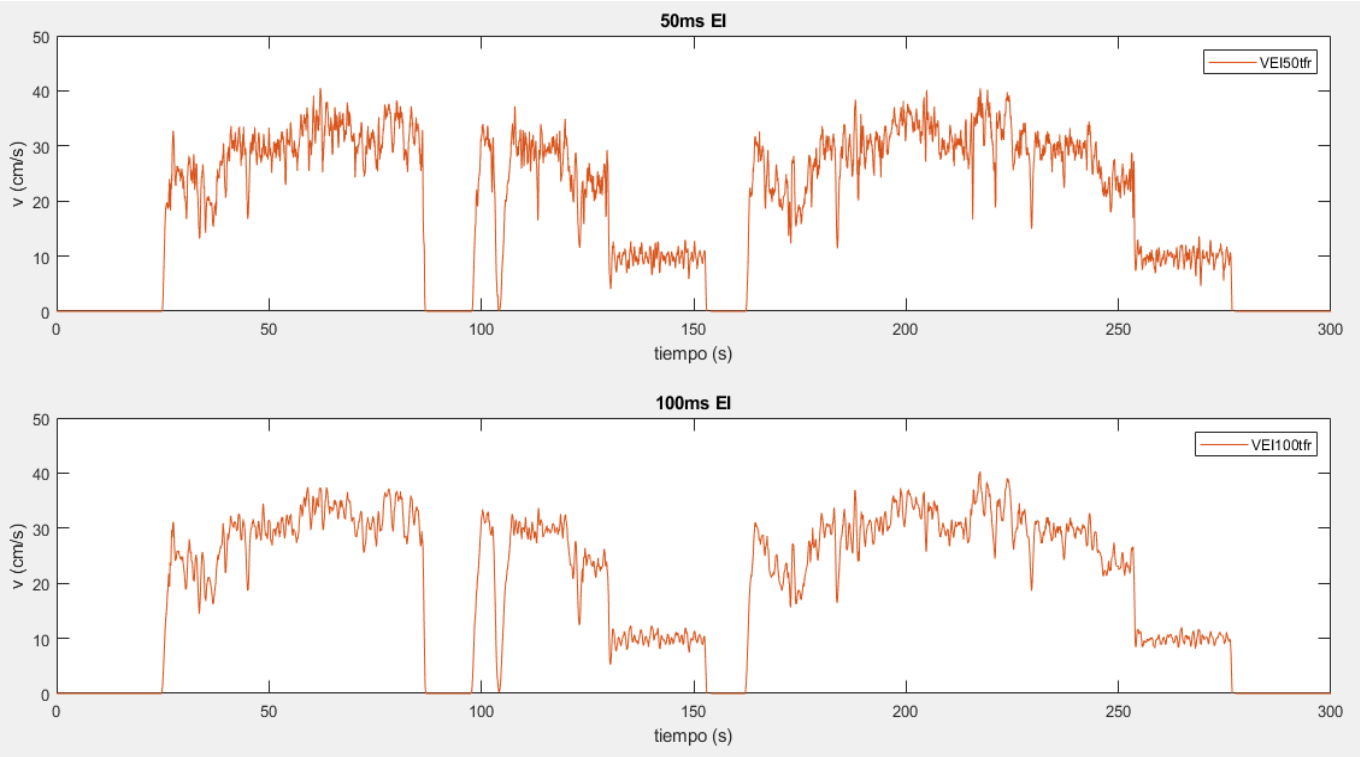

Ilustración 18: Representación de los valores de velocidad (cm/s) a la salida del filtro de paso bajo para el encoder izquierdo.

Al igual que ocurre con el encoder derecho, se comprueba como las señales de la velocidad a la salida del filtro de paso bajo para el encoder izquierdo (VEI50tfr, VEI100tfr) han sido suavizadas frente a la señal de entrada de la velocidad sin outliers que se muestra en la ilustración 16 (gráfica roja).

Una vez finalizado todas las etapas de procesado y filtrado de los datos, serán los valores de velocidad interpolados cada 100 ms y obtenidos a la salida del filtro de paso bajo (tanto para el encoder derecho como izquierdo), VED100tfr y VEI100tfr respectivamente, los que se utilizarán de aquí en adelante como velocidad (cm/s).

Al igual que con los datos de la velocidad, los valores del set point también se pasan por el filtro para eliminar el ruido de la señal, obteniéndose SPED100fsm y SPEI100fsm. Estos valores a la salida del filtro serán los empleados de aquí en adelante como valores del setpoint.

Capítulo 4 - Resultados de la identificación del AGV

4.1 Aplicación de técnicas paramétricas

Como métodos paramétricos se han empelado los métodos de tipo polinómico para conseguir la identificación del sistema.

Para llevar a cabo esta identificación se ha empleado la app de Matlab ® System Identification Tool.

Las técnicas de identificación se han llevado a cabo en el dominio del tiempo. Se han realizado numerosas pruebas con los diferentes modelos paramétricos (ARX, ARMAX, OE y BJ), buscando la técnica que proporcionase el menor error de predicción y del error de salida entre el sistema y el modelo.

4.1.1 Métodos de estimación de los coeficientes de los polinomios

Para el ajuste de los parámetros se ha empleado los siguientes métodos en función del modelo paramétrico:

Modelo ARX

- ARX: seleccionando el método ARX, se selecciona el método de mínimos cuadrados para la estimación de los parámetros. Este método minimiza la suma de los cuadrados del lado derecho menos el lado izquierdo de las expresiones que definen este modelo (ver aparatado 2.3.1).

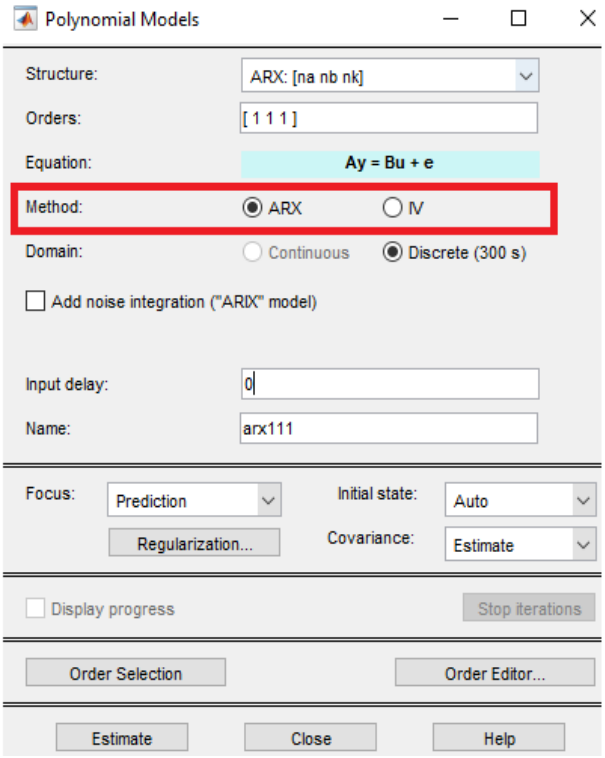

Ilustración 19: Método de estimación modelo ARX.

Modelos ARMAX, OE (Output-Error) y BJ (Box-Jenkins)

- Para los métodos ARMAX, OE y BJ, los coeficientes de los polinomios se estiman mediante un error de predicción/método de máxima verosimilitud minimizando el tamaño del error "e" en la ecuación general del modelo descrita en el apartado 2.3.1. Dentro del menú desplegable de iteración, hay varias opciones para el procedimiento de minimización del error. Se ha optado por elegir el método de selección automática para que sea el propio programa el que elija el método más efectivo.

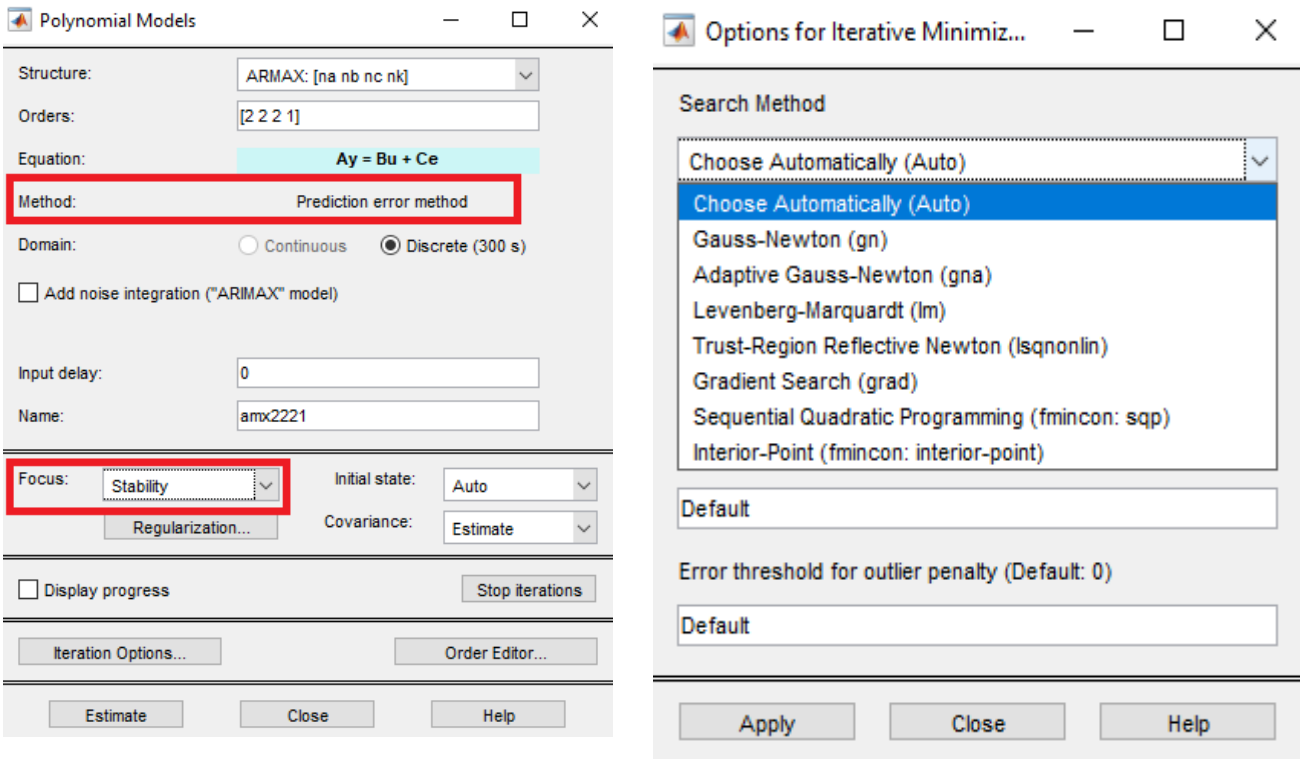

Ilustración 20: Ejemplo selección método de estimación para el modelo ARMAX.

En cuanto a las propiedades del método de estimación seleccionado, se ha empleado los siguientes para cada técnica:

ARX, ARMAX y BJ:

Se ha empleado la predicción junto con la estabilidad como valor para el enfoque o focus. Explicando con más detalle esto, se puede indicar que el focus hace referencia a la propiedad que afecta a la ponderación aplicada al ajuste entre el modelo y los datos. Este se emplea para asegurar que el modelo se aproxima correctamente al sistema real en ciertos intervalos de frecuencia.

Por su parte, para el focus se ha seleccionado la opción de predicción con estabilidad reforzada (Stability), es decir, el algoritmo se modifica para que se garantice un modelo estable, pero la ponderación sigue correspondiendo a la predicción. Esta opción influye en que el modelo se determinará minimizando los errores en esta etapa de predicción.

OE:

Se ha utilizado como opción del focus la simulación. Esto significa que la ponderación de la frecuencia de ajuste de la función de transferencia viene definida por el espectro de entrada. Los rangos de frecuencia en los que la entrada tiene una potencia considerable serán mejor descritos por el modelo. Es decir, la aproximación del modelo es tal que éste producirá las mejores simulaciones posibles cuando las entradas tengan los mismos espectros que los empleados para la estimación. El modelo resultante se garantiza que sea estable.

4.1.2 Estimación de los órdenes de los polinomios

La obtención de los órdenes de los polinomios de los distintos métodos paramétricos así como su comparativa se ha realizado manualmente con la ayuda de la herramienta de Matlab®. Para ello, se han ido probando multitud de métodos paramétricos con diferentes órdenes y se ha ido observando con cuales se obtenía un mejor ajuste.

4.1.3 Parámetros empleados para realizar la comparación del ajuste entre las diferentes técnicas

Para realizar una comparativa entre la precisión de cada modelo se va a emplear el valor de 2 parámetros:

1) El ajuste del modelo respecto a la gráfica de los valores de v (cm/s) correspondiente a cada encoder. A mayor valor, el modelo es más preciso. El valor 100 indica que el modelo ajusta perfectamente y 0 que el modelo tiene un ajuste muy pobre, siendo éste el mismo ajuste que proporcionaría la media de la señal de salida [25].

2) El valor de MSE (Mean-Squared-Error). El error cuadrático medio mide el promedio de los errores al cuadrado entre dos conjuntos de datos, o lo que es lo mismo, la diferencia entre un valor predicho y un valor observado o conocido [19]. Este valor es una medida de la calidad de un predictor o de un estimador, pudiéndose expresar su fórmula de forma diferente en función si hace referencia a uno u otro:

Para un predictor:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Y_i} - Y_i)$$

(4.1)

En donde:

$\widehat{Y_i}$ = vector de predicciones.

Y_i = vector con los valores reales.

Para un estimador:

$$MSE\ (\hat{\theta}) = E_\theta\left[(\hat{\theta} - \theta)^2\right]$$

(4.2)

En donde:

$\hat{\theta}$ = estimador.

θ = parámetro desconocido.

4.1.4 Mejores ajustes obtenidos para los datos del encoder derecho

A continuación se muestran los mejores ajustes obtenidos para el valor de velocidad del encoder derecho entre los diferentes métodos paramétricos probados (ARX, ARMA, OB, BJ):

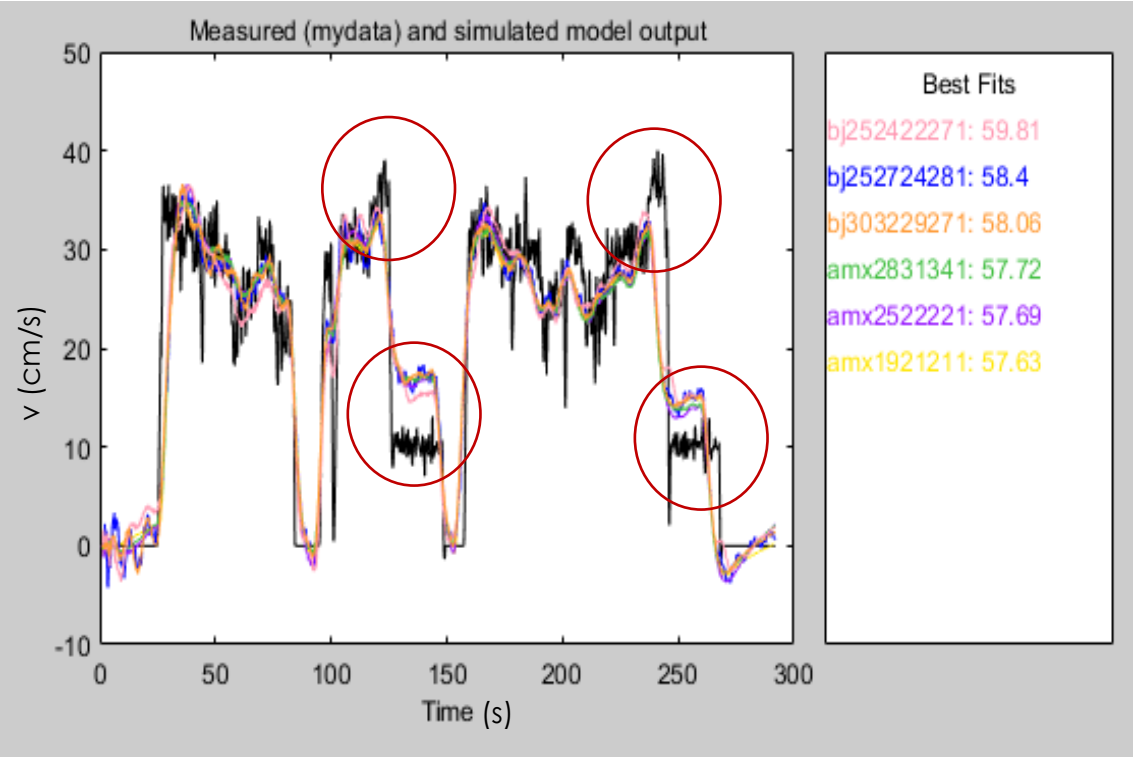

Ilustración 21: Ajuste mediante métodos paramétricos. Encoder derecho

Al observar la ilustración 21, se puede comprobar cómo, aunque el ajuste de las técnicas paramétricas sigue la tendencia de la señal de entrada, no es todo lo preciso que se desea, apareciendo underfitting. Este underfitting generará que el modelo no presente una generalización adecuada, generando unas predicciones imprecisas.

Las técnicas paramétricas no son capaces de ajustarse en las zonas con caídas o subidas más abruptas. Este fenómeno se traduce en que, dependiendo de la técnica empleada, se obtiene unos valores de ajuste que van de 57.63 a 59.81.

A continuación, se especifica una tabla resumen con los datos obtenidos para el ajuste y el error cuadrático medio de los modelos:

	Modelo Paramétrico	% Ajuste	MSE*
Encoder Derecho	bj252422271	59,81	43,63
	bj252724281	58,4	44,21
	bj303229271	58,06	45,26
	amx2831341	57,72	45,84
	amx2522221	57,69	46,16
	amx1921211	57,63	46,21

Tabla 2: Resultados ajuste métodos paramétricos. Encoder derecho.

** MSE: Mean-Squared-Error*

Los modelos que mejores resultados han proporcionado se han marcado con un rectángulo rojo. Ambos aportan unos valores muy semejantes. Más adelante se procederá a realizar el cálculo del coste computacional de cada uno para conocer en cuál de ellos es menor.

Se ha optado por seleccionar como mejor método de Box Jenkins el bj303229271, aunque no sea éste el que mejor ajuste ofrezca entre los métodos del mismo tipo. Esta decisión se ha tomado basándose en que presenta menos armónicos en el diagrama de Bode de la siguiente ilustración 22 frente a los otros dos modelos BJ.

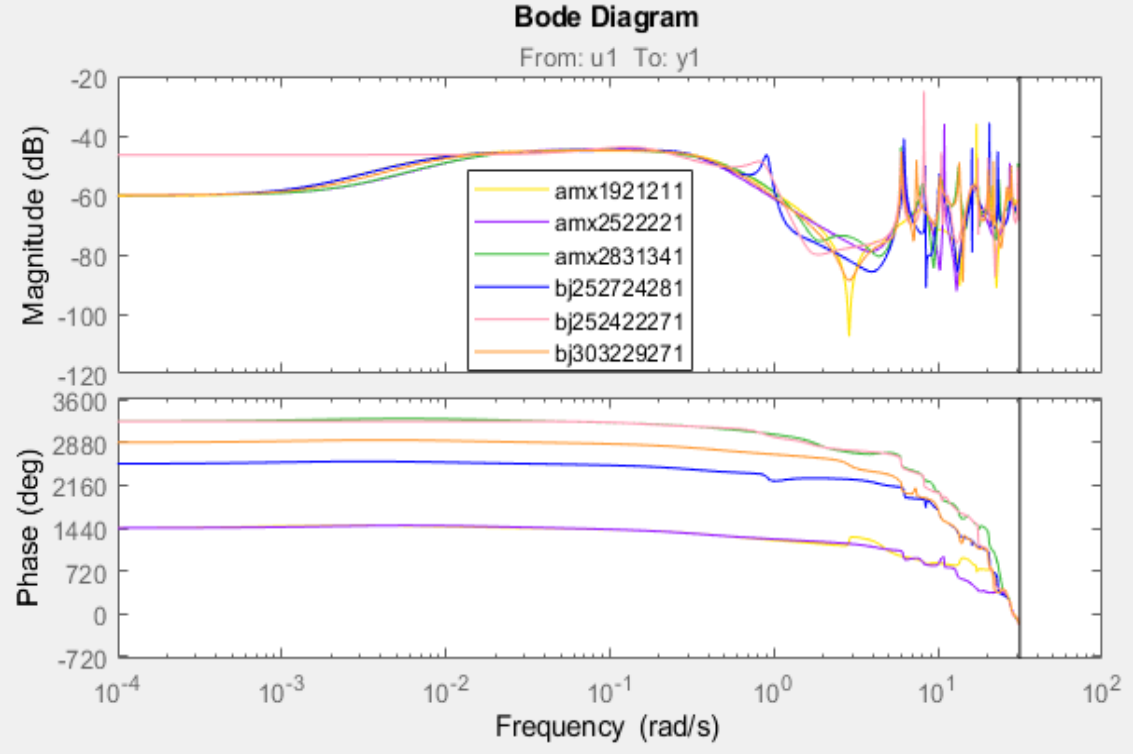

Ilustración 22: Respuesta en frecuencia métodos paramétricos. Diagrama de Bode. Encoder derecho.

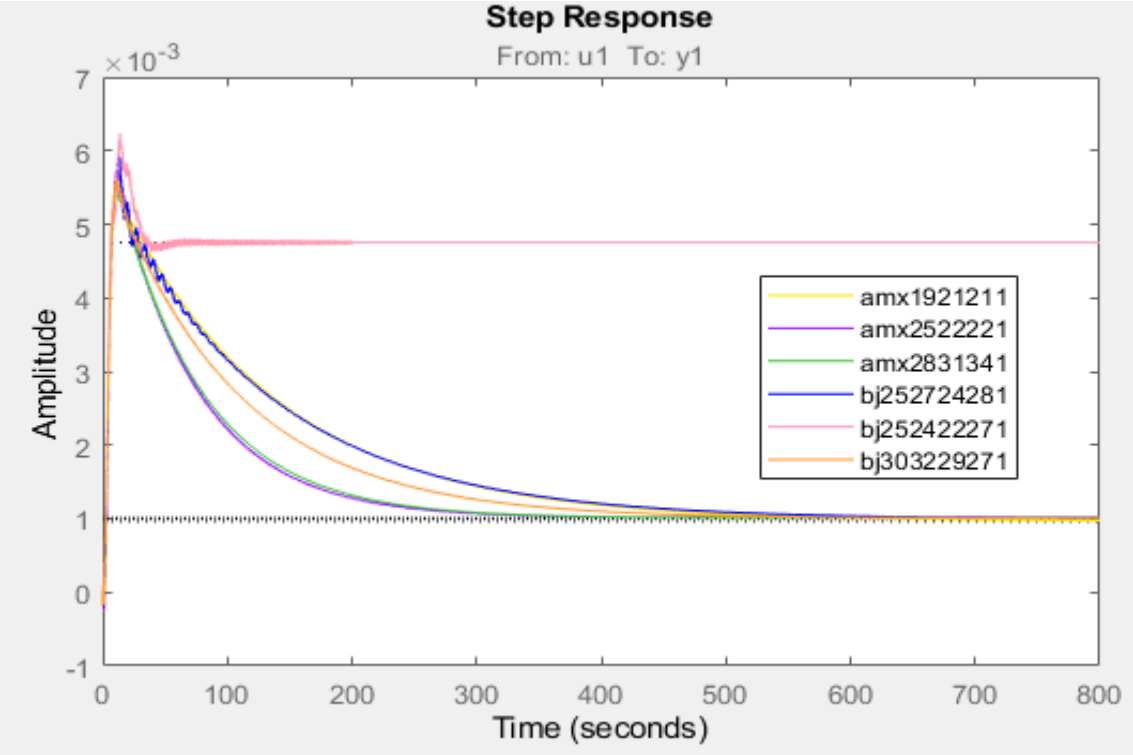

Ilustración 23: Respuesta frente a escalón. Encoder derecho.

4.1.5 Mejores ajustes obtenidos para los datos del encoder izquierdo

A continuación se muestran los mejores ajustes obtenidos para el valor de velocidad del encoder izquierdo entre los diferentes métodos paramétricos probados (ARX, ARMA, OB, BJ):

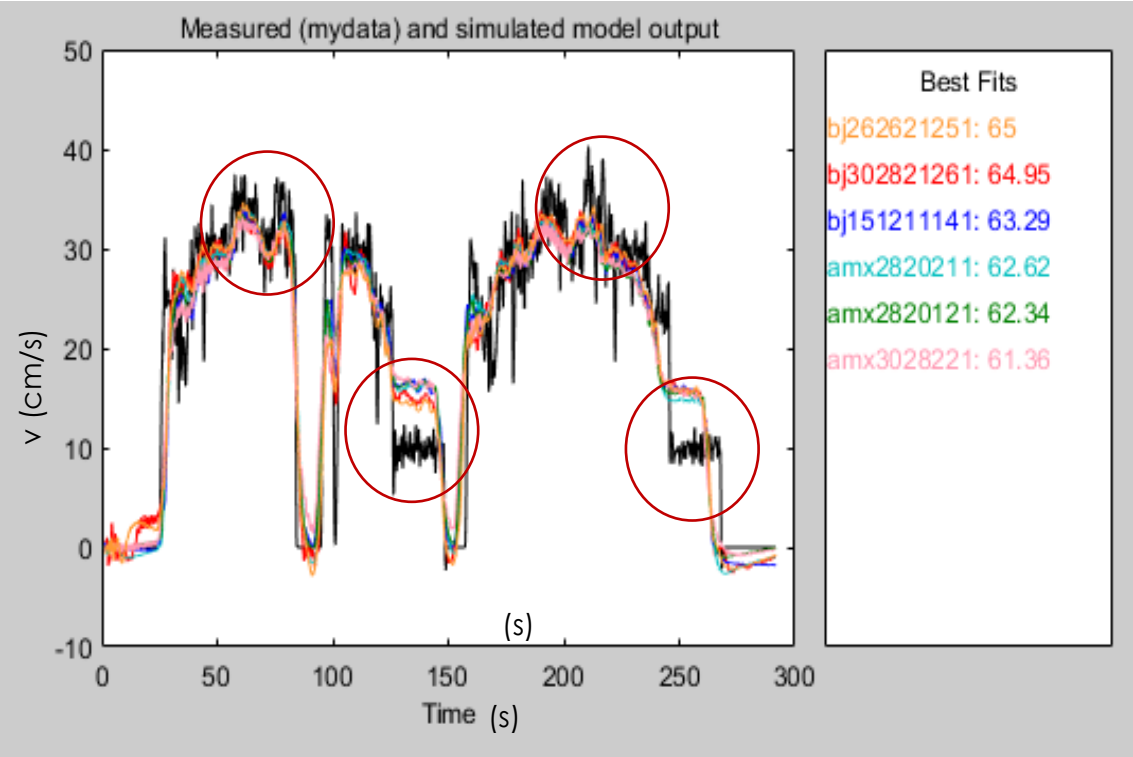

Ilustración 24: Ajuste mediante métodos paramétricos. Encoder Izquierdo.

Al igual que ha ocurrido con los valores del encoder derecho, se observa cómo no se ha conseguido un ajuste bueno en zonas concretas de la gráfica (underfitting), apareciendo una generalización deficiente y por tanto, no generándose predicciones correctas. Se comprueba nuevamente como los valores correspondientes a zonas con subidas o bajadas más abruptas respecto a la tendencia del resto de datos no han conseguido ser ajustados bien por los modelos paramétricos. Se ha obtenido un ajuste ligeramente superior frente al encoder derecho, llegándose a un rango situado entre el 65-61.36 dependiendo de la técnica empleada.

- Tabla resumen de las técnicas paramétricas que mejor ajuste han proporcionado para el encoder izquierdo:

	Modelo Paramétrico	% Ajuste	MSE*
Encoder Izquierdo	bj262621251	65,00	31,47
	bj302821261	64, 95	32,22
	bj151211141	63,29	33,06
	amx2820211	62,62	34,21
	amx2820121	62,34	34,58
	amx3028221	61,36	36,10

Tabla 3: Resultados ajuste métodos paramétricos. Encoder izquierdo.

**MSE: Mean-Squared-Error*

Nuevamente, los modelos de cada técnica que mejores resultados han proporcionado se indican mediante el rectángulo rojo. Ambos aportan unos valores muy semejantes. Más adelante se procederá a realizar el cálculo del coste computacional de cada uno para conocer en cuál de ellos es menor.

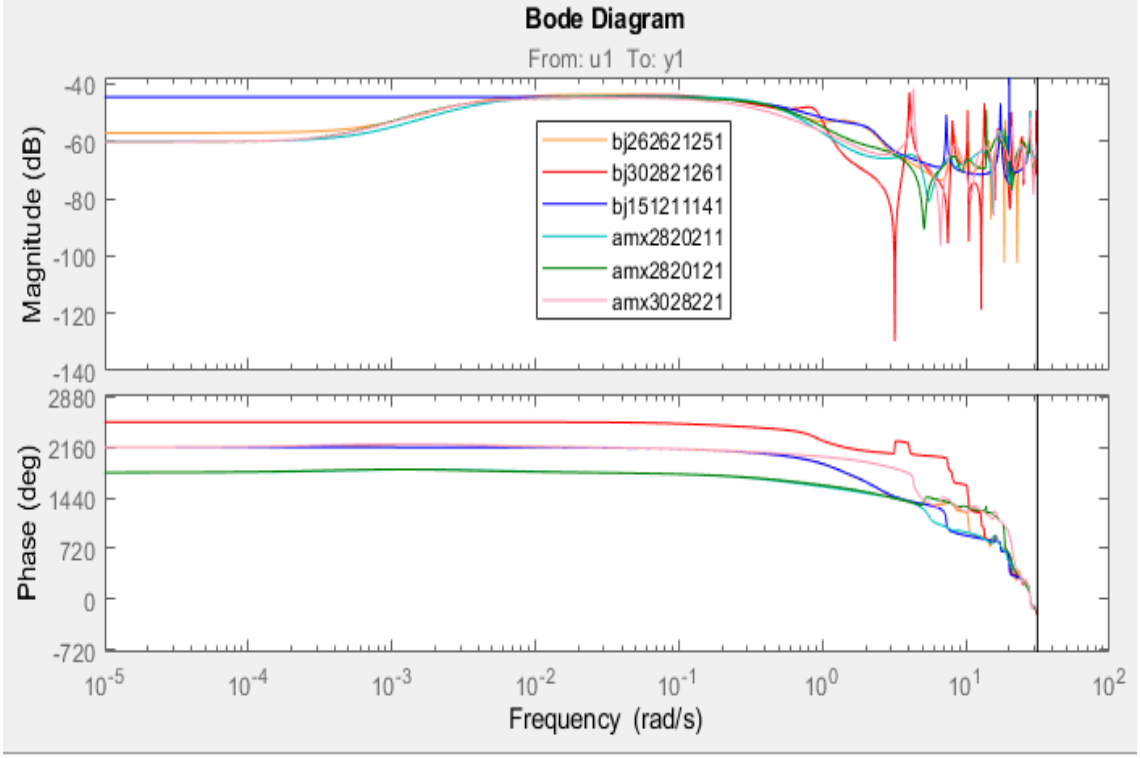

Ilustración 25: Respuesta en frecuencia métodos paramétricos. Diagrama de Bode. Encoder izquierdo.

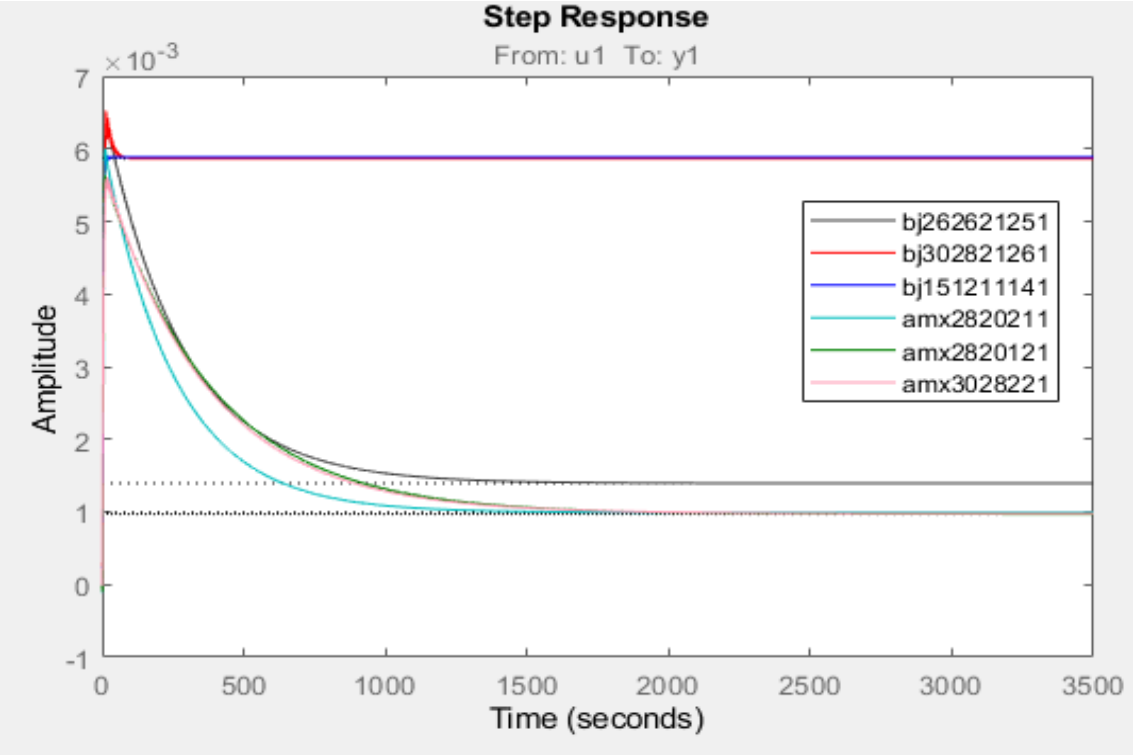

Ilustración 26: Respuesta frente a escalón. Encoder Izquierdo.

Después de haber trabajado con los datos procedentes de ambos encoders, se observa como las técnicas paramétricas no consiguen una identificación con altos porcentajes de ajuste para las señales de estos encoders. Por ello se propone el empleo de redes neuronales que sean capaces de "aprender" e ir mejorando el ajuste en cada etapa.

4.1.6 Cálculo del coste computacional. Modelos paramétricos encoder derecho e izquierdo.

A continuación, se indican los valores de los costes computacionales (definidos en unidades de coste) calculados siguiendo el procedimiento explicado en el apartado 2.3.2 para la mejor técnica BJ y ARMAX de cada encoder:

Encoder	Técnica paramétrica	Coste computacional (Tc)
Derecho	bj303229271	**321,592**
	amx2831341	**254,917**
Izquierdo	bj26262125	**268,252**
	amx2820211	**190,909**

Tabla 4: Comparativa entre los costes computacionales de la mejor técnica para el encoder derecho e izquierdo.

Tal y como era de esperar, el coste computacional aumenta conforme se incrementan los órdenes de los polinomios y su cantidad en el modelo. El modelo BJ tiene un mayor coste computacional frente al modelo ARMAX a consecuencia de que necesita 4 polinomios (nb, nc, nd, nf) frente a los 3 de modelo ARMAX (na, nb, nc).

Debido a que el ajuste obtenido por los mejores modelos BJ y ARMAX de cada encoder son muy parecidos, y junto con que se ha comprobado que el coste computacional del modelo ARMAX es menor en ambos casos, se procede a seleccionar los modelos amx2831341 y amx2820211 como los mejores modelos para el encoder derecho e izquierdo respectivamente.

4.2 Identificación con redes neuronales

El empleo de redes neuronales en la identificación de sistemas presenta numerosas ventajas, entre las que se puede destacar su capacidad de generalizar y continuo aprendizaje. No obstante, también aparecen algunas complicaciones como son la selección del modelo apropiado y el ajuste de sus parámetros de acuerdo con una ley adaptativa que permita que la respuesta del modelo a una señal/es de entrada se aproxime a la respuesta del sistema real frente a esa misma entrada [9].

En el presente trabajo se ha empleado la toolbox "*Neural Net Time Series*" del software Matlab® para la identificación de nuestro AGV. Mediante el uso de este programa, ha sido posible la creación y definición de redes neuronales y llevar a cabo el entrenamiento, validación y prueba de las redes diseñadas estableciendo una comparación entre ellas. De igual forma, ha sido posible evaluar el rendimiento de cada red mediante el error cuadrático medio y un análisis de regresión.

Par el estudio se han empleado los siguientes tipos de redes neuronales:

- **NAR** (no lineal autorregresiva): predice el valor de una serie y(t) mediante valores pasados de y(t) (ilustración 27).

- **NARX** (no lineal autorregresiva con entrada externa): predice el valor de una serie y(t) mediante valores pasados de y(t) y de otra serie x(t) (ilustración 28).

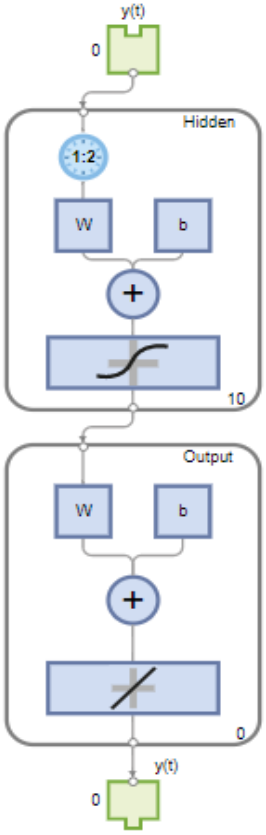

Ilustración 27: Ejemplo red NAR.

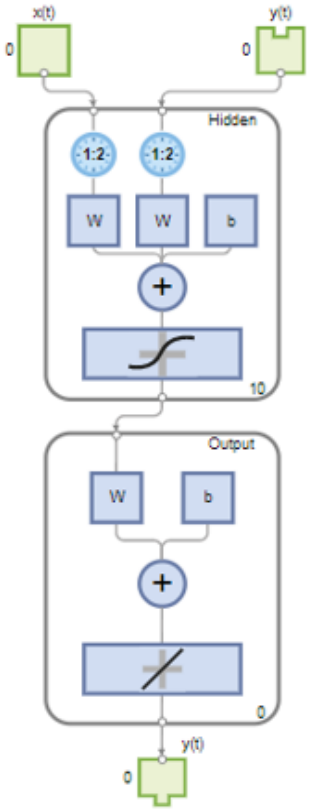

Ilustración 28: Ejemplo red NARX.

Como algoritmo de entrenamiento se ha empleado Levenberg-Marquardt (método de mínimos cuadrados amortiguados) por ser más rápido que el bayesiano y con una alta precisión.

Se ha empleado una validación cruzada, definiéndose un 70% del conjunto de valores como valores de entrenamiento, un 15% como valores de validación y otro 15% como valores para la realización del test (ilustración 29). Esta validación cruzada se ha realizado con la intención de minimizar la aparición del fenómeno de overfitting.

Ilustración 29: Porcentaje del conjunto de datos destinados a entrenamiento, validación y test.

4.2.1 Parámetros empleados para realizar la comparación entre los ajustes obtenidos mediante las diferentes redes neuronales

Para llevar a cabo la comparación entre los ajustes obtenidos por los diferentes tipos de redes neuronales se va a emplear 2 parámetros:

- *R (regression R-Value)*: este parámetro refleja la correlación existente entre las salidas y las respuestas. Su valor se corresponde a una medida estadística de la aproximación de las predicciones de la regresión con respecto a los puntos de datos reales. El rango de valores posibles en este caso va de 0 a 1, indicando este último valor que las predicciones de la regresión se ajustan perfectamente a los datos y el modelo es bueno para predecir resultados.

- *MSE (Mean-Squared-Error)-Error cuadrático medio*: este parámetro ya ha sido explicado previamente en el apartado de las técnicas paramétricas.

A continuación se muestran los ajustes obtenidos para los datos de velocidad de cada encoder (pulsos/s) vs tiempo.

4.2.2 Mejores ajustes obtenidos para los datos del encoder derecho

4.2.2.1 Red tipo NARX

	RED	Nº NEURONAS (CAPA OCULTA)	TIEMPO RETARDO	FASE	R*	MSE*	ITERACIÓN CON MAYOR RENDIMIENTO EN EL AJUSTE
ENCODER DERECHO	NARX	25	1	Training	0.9977	0.7723	4
				Validation	0.9959	1.4723	
				Test	0.9976	0.7839	
			2	Training	0.9987	0.4583	9
				Validation	0.9986	0.4201	
				Test	0.9986	0.4554	
		50	1	Training	0.9979	0.7068	4
				Validation	0.9978	0.7560	
				Test	0.9975	0.8518	
			2	Training	0.9988	0.4288	5
				Validation	0.9985	0.4912	
				Test	0.9984	0.5469	
		100	1	Training	0.9980	0.6779	8
				Validation	0.9974	0.9197	
				Test	0.9976	0.8419	
			2	Training	0.9988	0.4062	6
				Validation	0.9984	0.5850	
				Test	0.9983	0.5527	
		200	1	Training	0.9981	0.6526	4
				Validation	0.9971	1.0297	
				Test	0.9949	1.6625	
			2	Training	0.9989	0.3983	10
				Validation	0.9974	0.8019	
				Test	0.9984	0.5341	

Tabla 5: Resumen valores de ajuste con red NARX. Encoder derecho.

**R: Regression R-value*

**MSE: Mean-Squared-Error*

En el recuadro rojo se muestra la red que mejores resultados ha proporcionado en términos de error cuadrático medio y desempeño.

A continuación se muestran con detalle los resultados obtenidos para esta red NARX.

- ***Tamaño de la capa oculta (Layer size): 25 neuronas. Tiempo de retardo (Time delay): 2. NARX encoder derecho.***

- Respuesta:

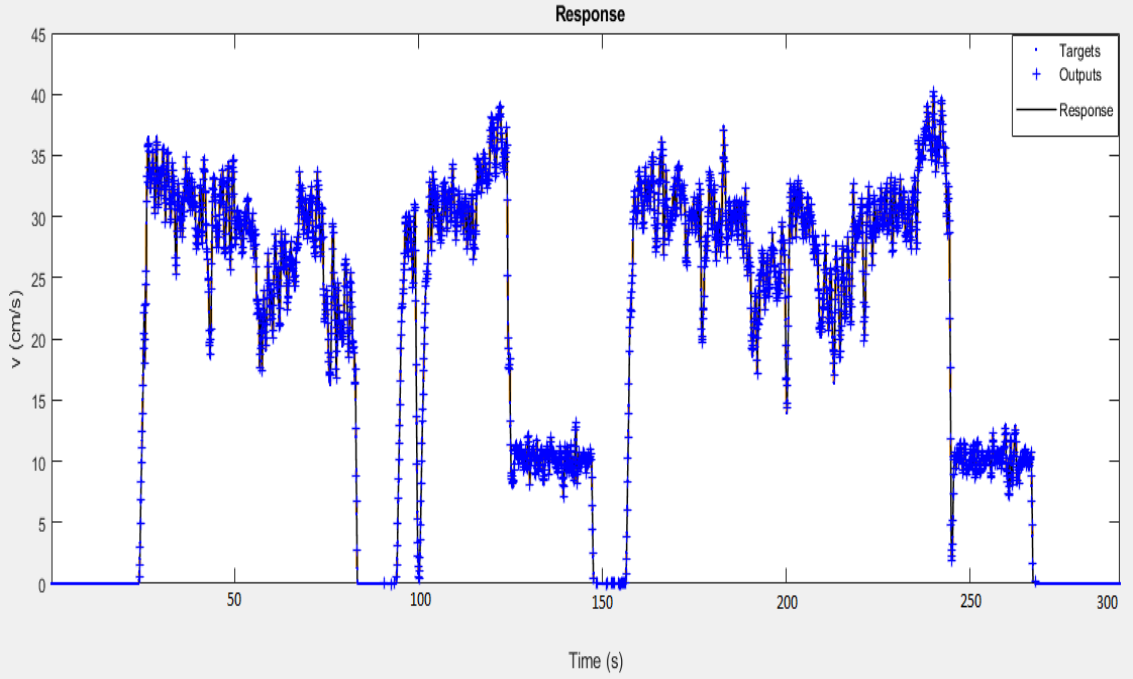

Ilustración 30: Respuesta red NARX. Nº neuronas capa oculta: 25. Tiempo retardo:2. Encoder derecho.

- Regresión en cada fase:

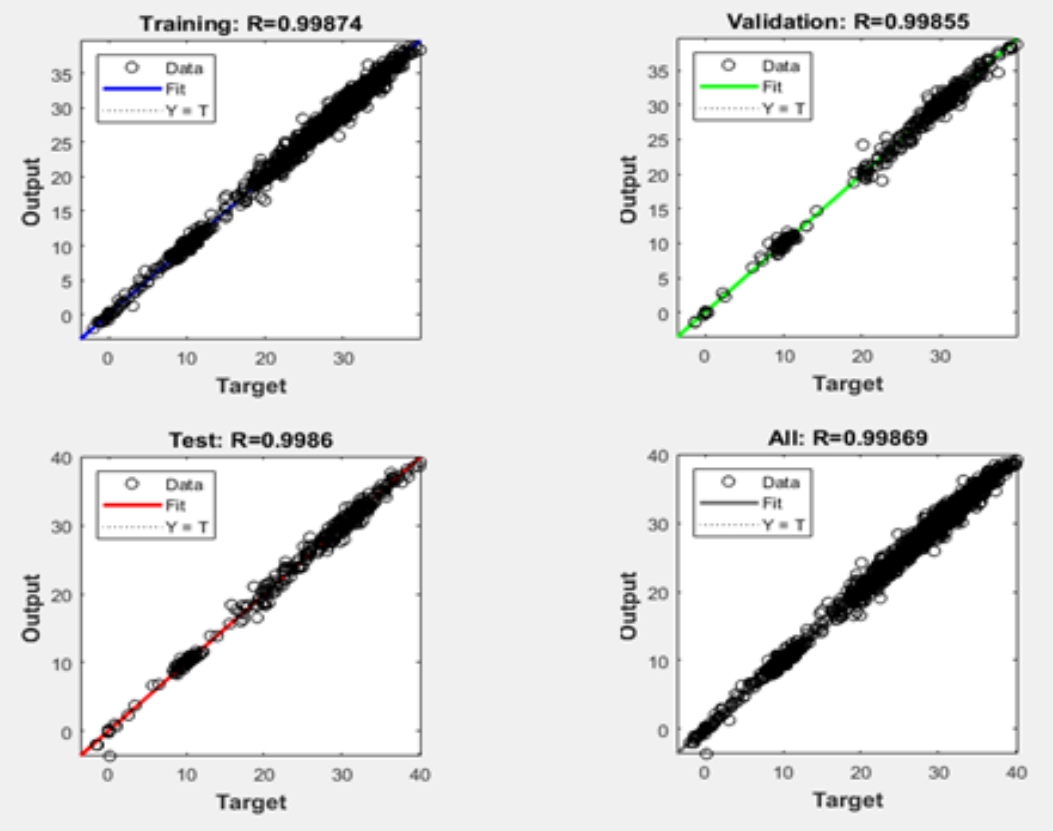

Ilustración 31: Regresión en cada fase. Red NARX. Tamaño capa oculta 25. Tiempo retardo:2. Encoder derecho.

Se puede comprobar como la red consigue unos ajustes excelentes respecto a los datos empelados para cada fase y el global.

- Desempeño

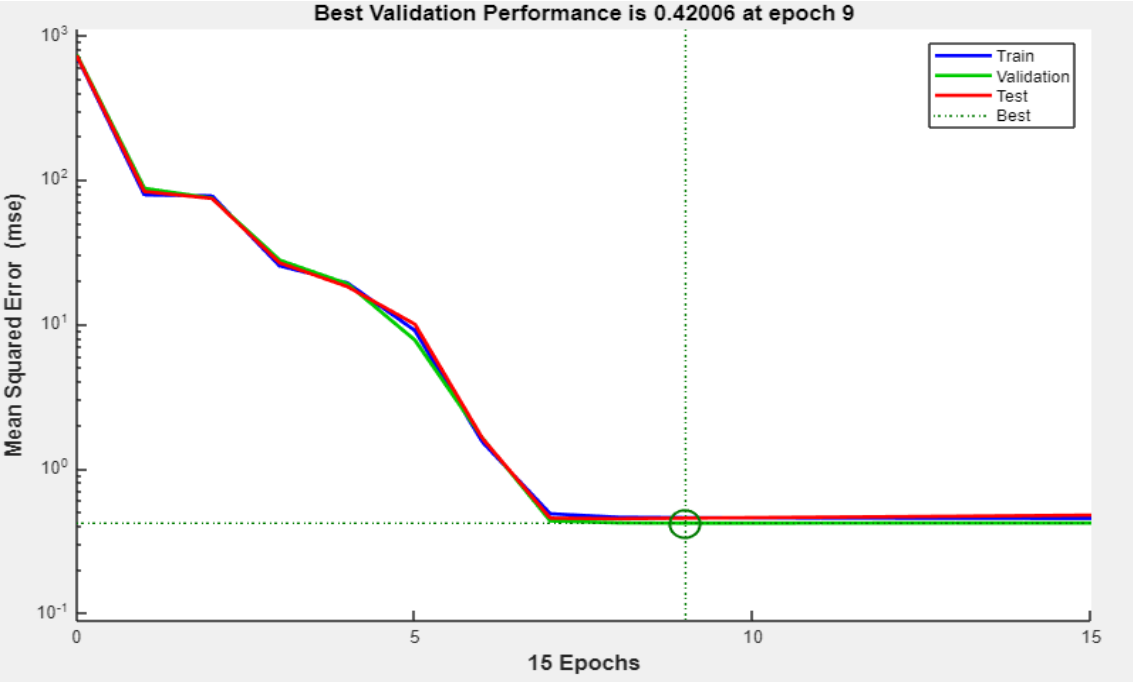

Ilustración 32: Mejor desempeño red NARX. 25 neuronas. Tiempo de retardo 2. Encoder derecho.

Después de numerosas pruebas, se ha podido comprobar como para redes neuronales NARX con un tamaño de capa oculta mayor a 25 neuronas comienza la aparición del fenómeno de **sobreajuste u overfitting**, incrementándose este sobreajuste a medida que se aumenta el tamaño de esta capa oculta. Un ejemplo de este fenómeno se puede observar para la misma red pero con un tamaño de la capa oculta de 100 neuronas:

Red NARX: Número de neuronas: 100. Tiempo de retardo :2

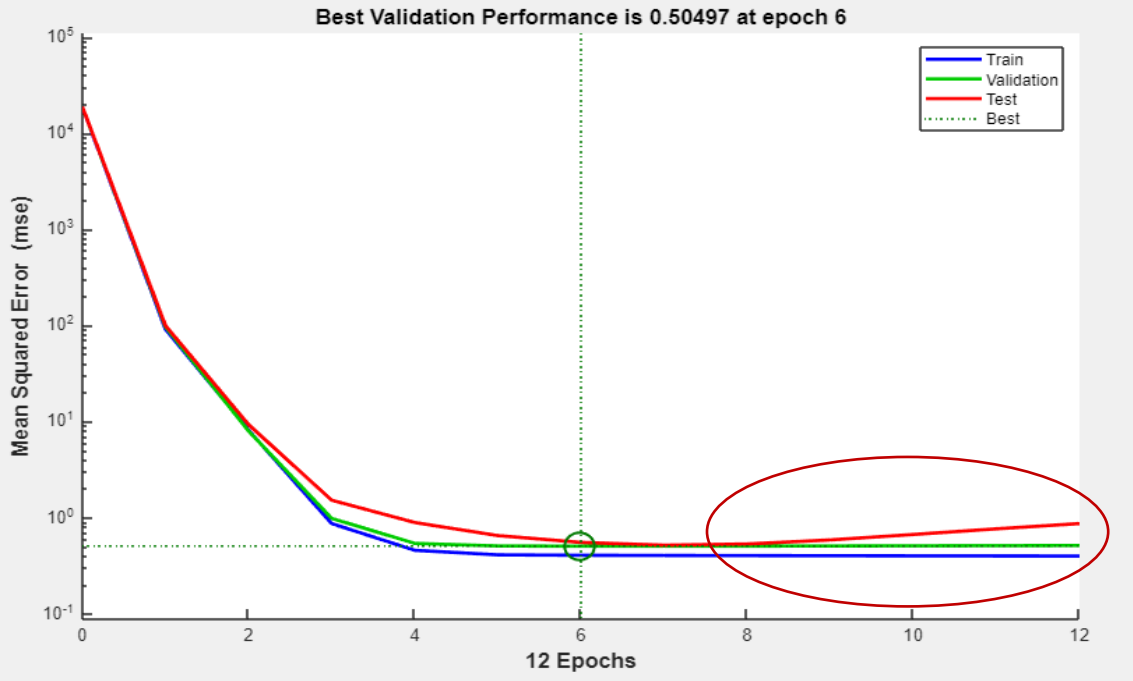

Ilustración 33: Sobreajuste en red NARX. 100 neuronas. Tiempo de retardo2. Encoder derecho.

Se observa en la anterior ilustración la aparición del fenómeno de overfitting a partir de la época 9. Este sobreajuste se manifiesta como una **separación cada vez mayor a medida que se avanza de época entre las gráficas que muestran el error cuadrático medio en la etapa de entrenamiento y test.** Este fenómeno demuestra que la red pierde capacidad de generalización con diferentes datos a los de entrenamiento y comienza un proceso de memorización y no de aprendizaje

4.2.2.2 Red tipo NAR

	RED	Nº NEURONAS	TIEMPO RETARDO	FASE	R*	MSE*	ITERACIÓN CON MAYOR RENDIMIENTO EN EL AJUSTE
ENCODER DERECHO	NAR	25	1	Training	0.9978	0.7478	7
				Validation	0.9973	0.9197	
				Test	0.9977	0.8074	
			2	Training	0.9987	0.4812	5
				Validation	0.9982	0.5277	
				Test	0.9985	0.5231	
		50	1	Training	0.9979	0.7221	10
				Validation	0.9978	0.7825	
				Test	0.9973	0.9502	
			2	Training	0.9987	0.4295	13
				Validation	0.9984	0.5352	
				Test	0.9988	0.4340	
		100	1	Training	0.9980	0.6871	2
				Validation	0.9967	1.1614	
				Test	0.9959	1.3807	
			2	Training	0.9987	0.4460	5
				Validation	0.9983	0.5596	
				Test	0.9982	0.6085	
		200	1	Training	0.9979	0.7148	2
				Validation	0.9956	1.3876	
				Test	0.9792	6.8984	
			2	Training	0.9987	0.4569	4
				Validation	0.9969	1.0582	
				Test	0.9977	0.7829	

Tabla 6: Listado ajustes con red NAR. Encoder derecho.

**R: regression R-value*

**MSE: Mean-Squared-Error*

En el recuadro rojo se muestra la red que mejores resultados ha proporcionado en términos de error cuadrático medio y desempeño.

- ***Tamaño de la capa oculta (Layer size): 25 neuronas. Tiempo de retardo (Time delay): 2. NAR encoder derecho.***

A continuación se muestran con detalle los resultados obtenidos para esta red:

- Respuesta:

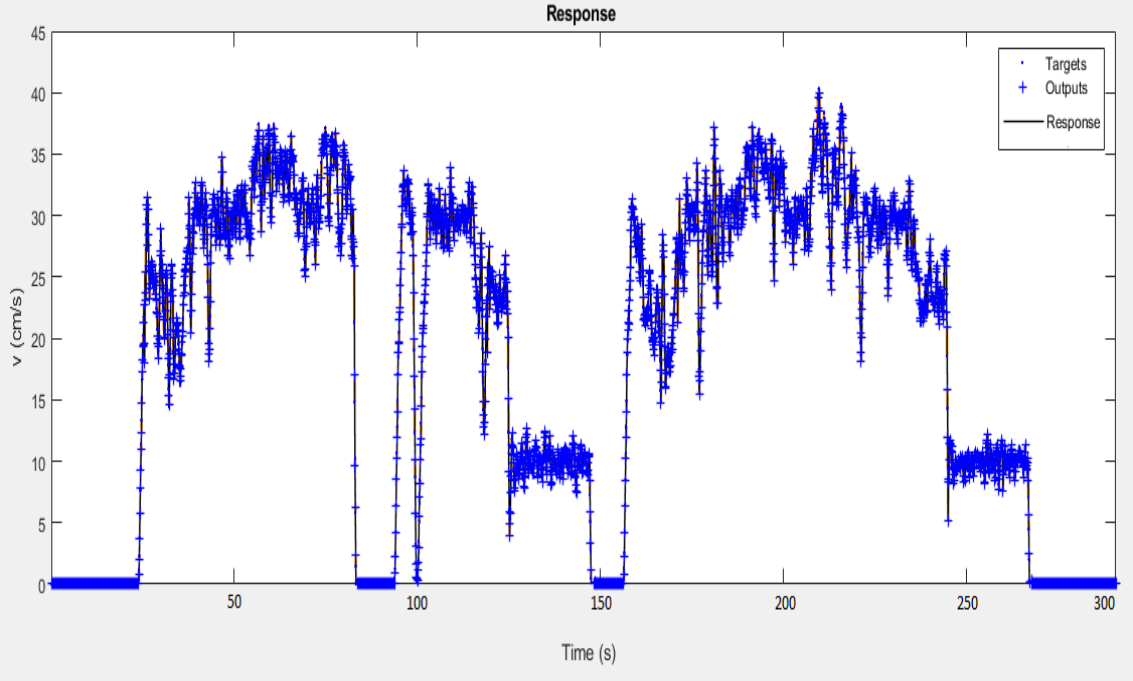

Ilustración 34: Respuesta red NAR. Tamaño capa oculta:25. Tiempo retardo:2. Encoder derecho.

- Regresión en cada fase:

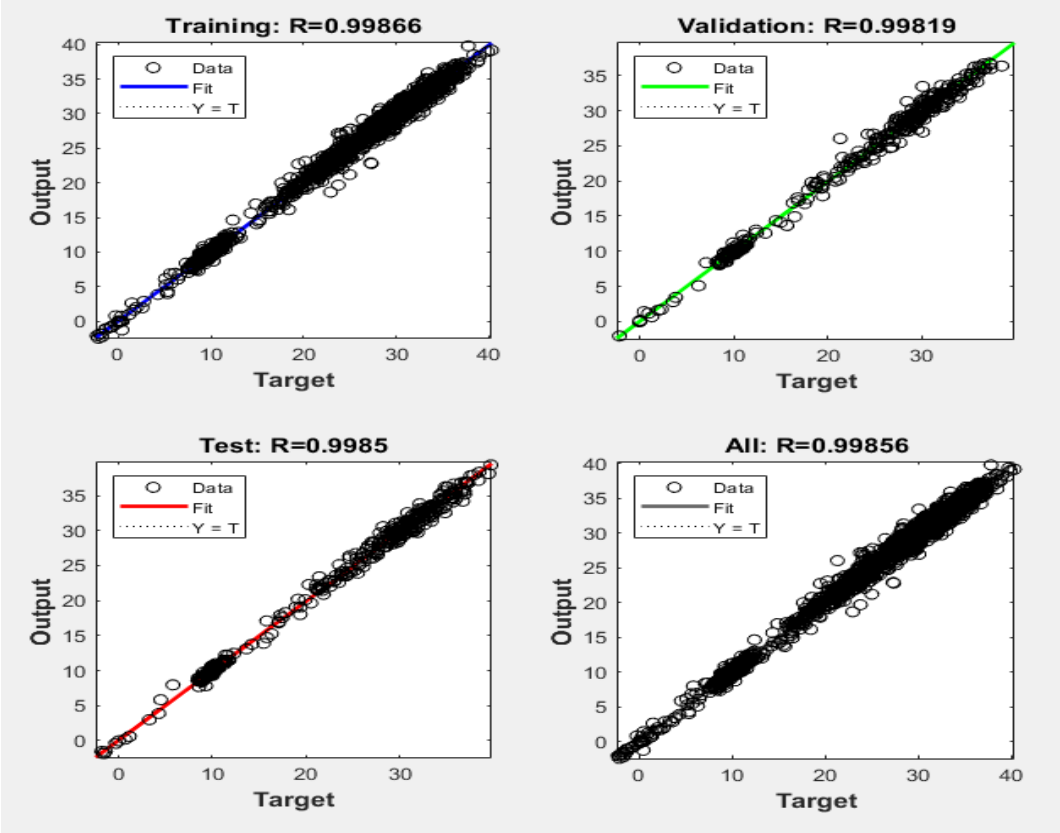

Ilustración 35: Regresión en cada fase. Red NAR. Tamaño capa oculta 25. Tiempo retardo:2. Encoder derecho

Se observa como el ajuste de la red para cada fase y el global es muy bueno.

- Desempeño:

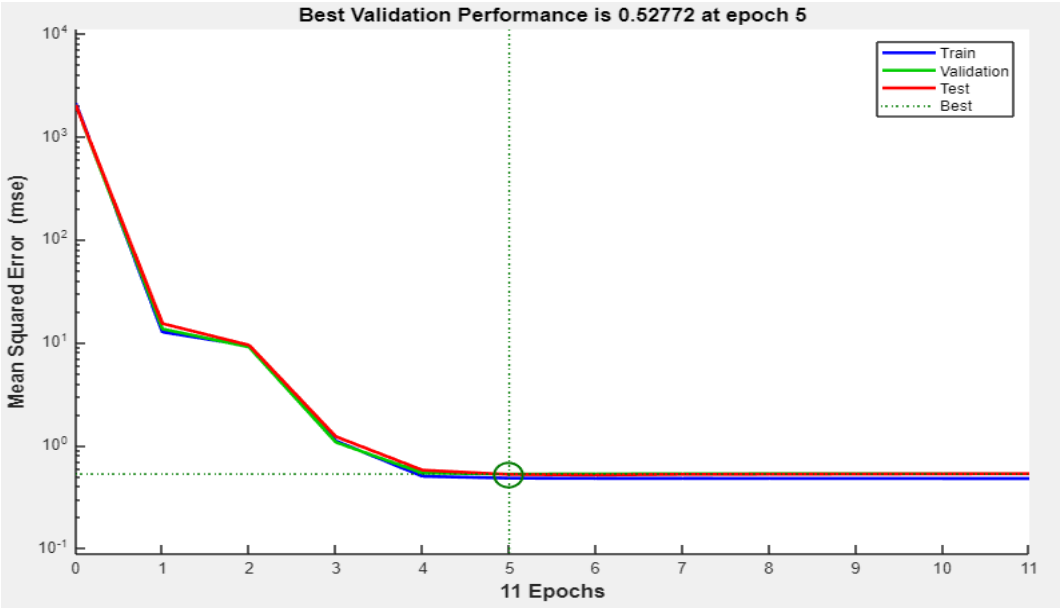

Ilustración 36: Mejor desempeño red NAR. 25 neuronas. Tiempo de retardo 2. Encoder derecho.

Se observa como el desempeño de esta red es muy satisfactorio para cada una de las fases de entrenamiento, validación y test.

Al incrementarse el tamaño de la capa oculta comienza la aparición de sobreajustes tal y como se puede comprobar en las siguientes gráficas de desempeño obtenidas a modo de ejemplo para tres redes NAR: la primera con un tamaño de capa oculta de 100 neuronas y tiempo de retardo igual a 2 , la segunda con tamaño de capa oculta de 200 neuronas y tiempo de retardo igual a 1, y la tercera con un tamaño de capa oculta de 200 neuronas y tiempo de retardo igual a 2.

A continuación se muestran las gráficas:

Tamaño de la capa oculta (Layer size) :100 neuronas. Tiempo de retardo (Time delay): 2

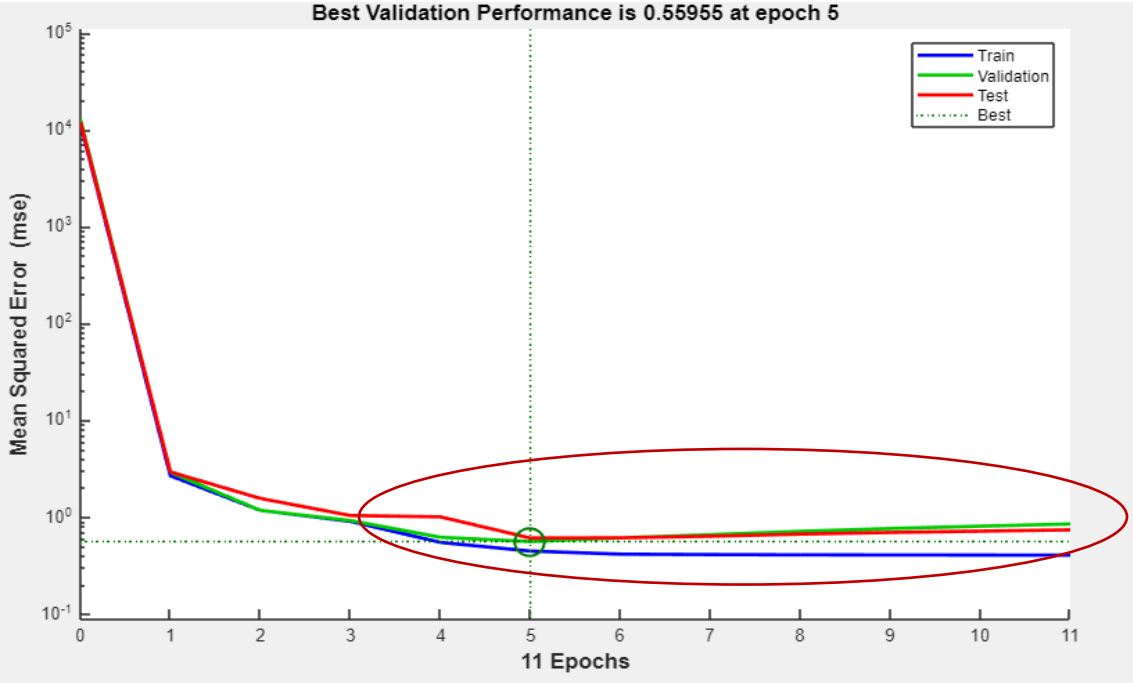

Ilustración 37: Sobreajuste en red NAR. 100 neuronas. Tiempo de retardo 2. Encoder derecho.

Tamaño de la capa oculta (Layer size) :200 neuronas. Tiempo de retardo (Time delay):1

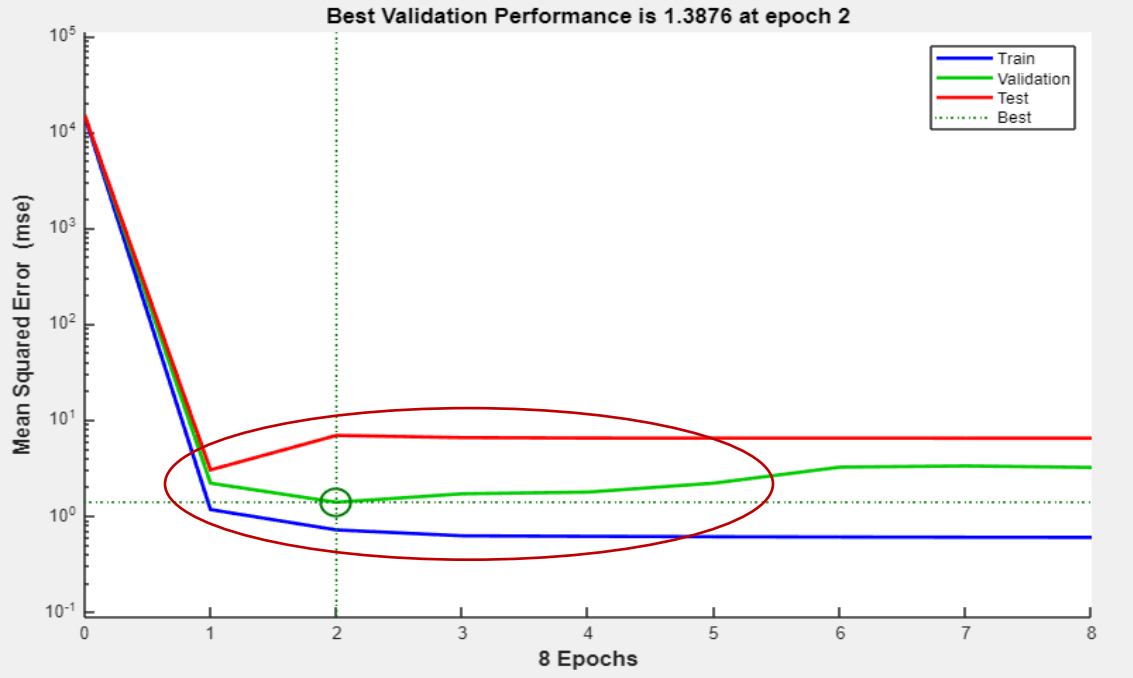

Ilustración 38: Sobreajuste en red NAR.200 neuronas. Tiempo de retardo 1. Encoder derecho.

Tamaño de la capa oculta (Layer size) :200 neuronas. Tiempo de retardo (Time delay):2

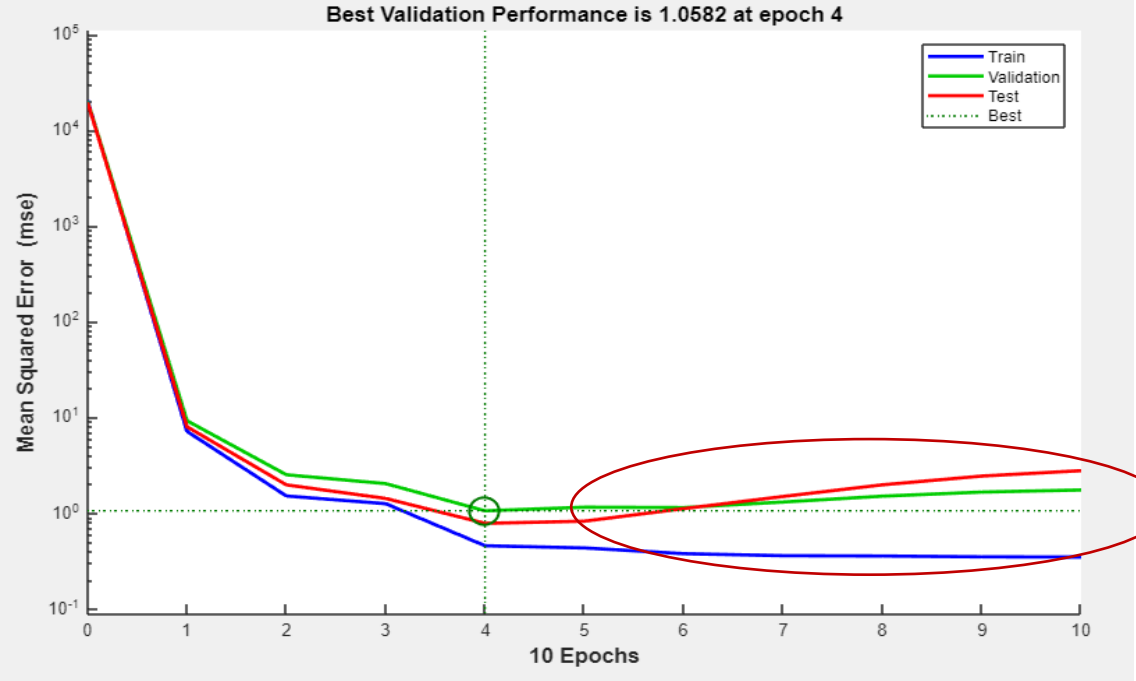

Ilustración 39: Sobreajuste en red NAR.200 neuronas. Tiempo de retardo 2. Encoder derecho.

Se puede observar en las ilustraciones 37, 38 y 39 la aparición del fenómeno de sobreajuste para los datos del encoder derecho. Este sobreajuste se manifiesta en la gráfica de desempeño como una separación del error de la fase de test frente a la fase de entrenamiento a medida que se avanza en el número de iteraciones. Se comprueba en las zonas de overfitting marcadas mediante el círculo rojo como el error de la fase de entrenamiento va disminuyendo mientras el error en la fase de test sube. En estas zonas la red pierde capacidad de generalización para diferentes datos y comienza un proceso de memorización de los datos de entrenamiento y no de aprendizaje.

Después de haber trabajado con ambos tipos de redes para el encoder derecho, se ha observado como la red NARX tiene una menor tendencia a la aparición de este sobreajuste, originándose en redes con mayor número de neuronas en la capa oculta frente a las necesarias para su aparición en la red NAR. Este fenómeno puede ser explicado a consecuencia de que la red NARX tiene como entrada dos series de datos, x(t) e y(t), frente a sólo una serie en la red NAR, y(t). Esto permite a la red NARX una entrada mayor de valores para su entrenamiento, y por tanto la capacidad de

entrenarse con un número mayor de valores, obteniéndose así una red más robusta con una capacidad de generalización mejor.

4.2.3 Mejores ajustes obtenidos para los datos del encoder Izquierdo

4.2.3.1 Red tipo NARX

	RED	Nº NEURONAS	TIEMPO RETARDO	FASE	R*	MSE*	ITERACIÓN CON MAYOR RENDIMIENTO EN EL AJUSTE
ENCODER IZQUIERDO	NARX	25	1	Training	0.9979	0.7140	4
				Validation	0.9978	0.7210	
				Test	0.9973	0.9032	
		25	2	Training	0.9986	0.4838	5
				Validation	0.9987	0.4704	
				Test	0.9985	0.5003	
		50	1	Training	0.9980	0.6877	8
				Validation	0.9976	0.7940	
				Test	0.9974	0.8745	
		50	2	Training	0.9986	0.4588	5
				Validation	0.9984	0.5815	
				Test	0.9984	0.5175	
		100	1	Training	0.9980	0.6741	6
				Validation	0.9906	3.0873	
				Test	0.9976	0.8052	
		100	2	Training	0.9987	0.4570	4
				Validation	0.9955	1.5364	
				Test	0.9972	1.0183	
		200	1	Training	0.9982	0.6178	8
				Validation	0.9964	1.1760	
				Test	0.9989	3.8593	
		200	2	Training	0.9988	0.3983	6
				Validation	0.9986	0.4965	
				Test	0.9956	1.5568	

Tabla 7: Listado ajustes con red NARX. Encoder izquierdo.

**R: regression R-value*

**MSE: Mean-Squared-Error*

En el recuadro rojo se muestra la red que mejores resultados ha proporcionado en términos de error cuadrático medio y desempeño.

Tamaño de la capa oculta (Layer size): 25 neuronas. Tiempo de retardo (Time delay):2. NARX encoder izquierdo

- Respuesta

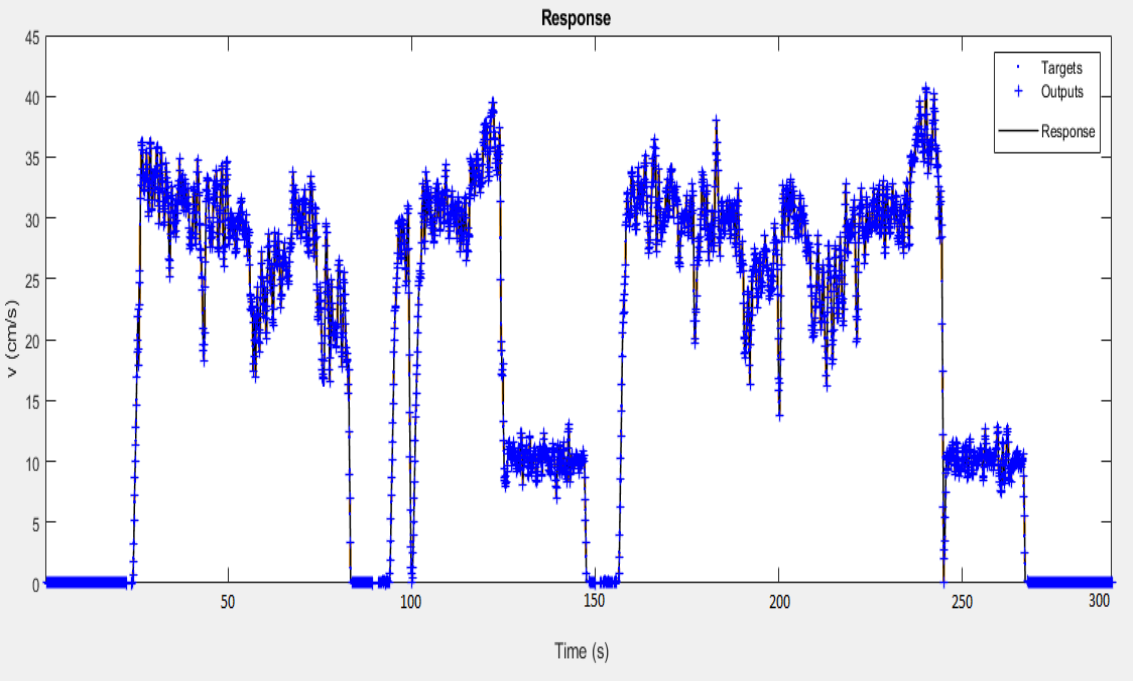

Ilustración 40: Respuesta red NARX. Tamaño capa oculta:25. Tiempo retardo:2. Encoder izquierdo.

- Regresión para cada fase:

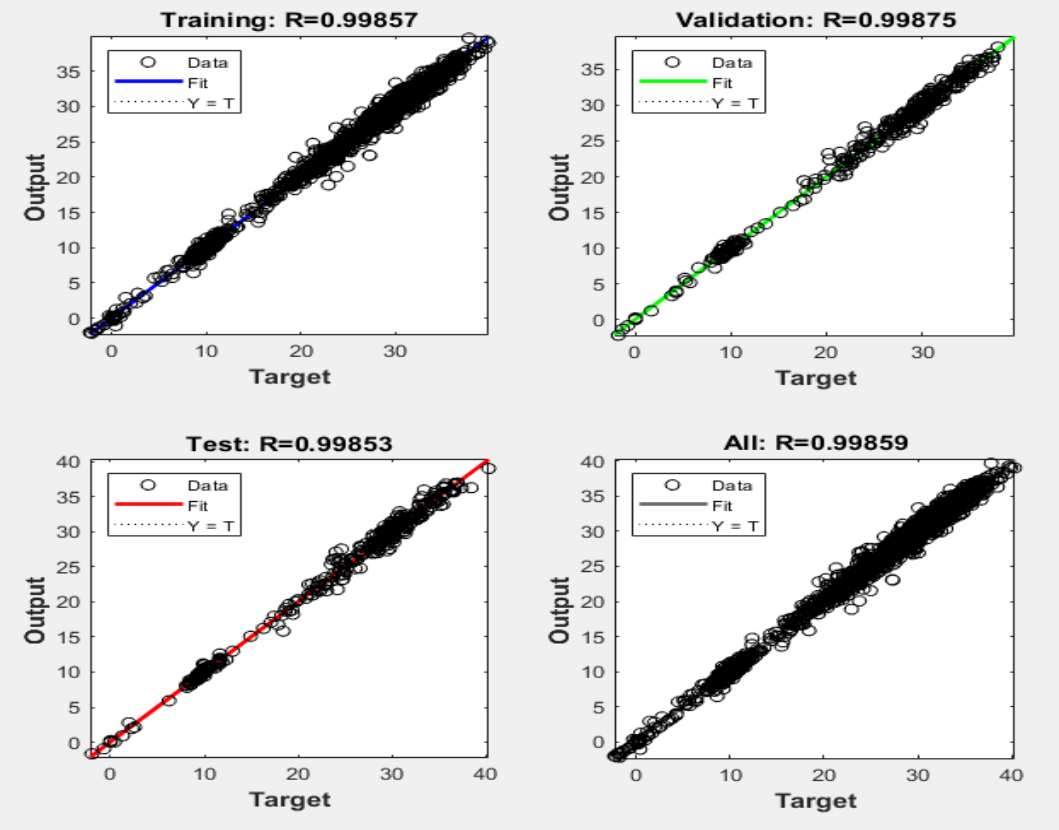

Ilustración 41: Regresión en cada fase. Red NARX. Tamaño capa oculta 25. Tiempo retardo:2. Encoder izquierdo.

Al igual que ocurría con los valores del encoder derecho, para el encoder izquierdo los ajustes siguen siendo muy buenos.

- Desempeño:

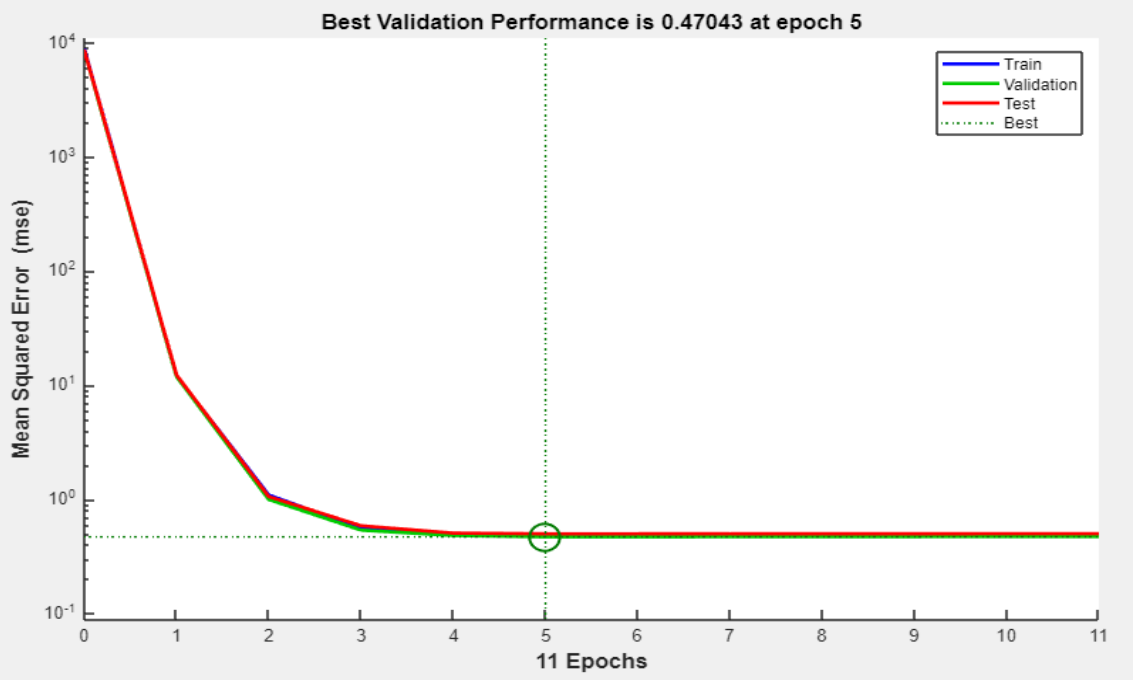

Ilustración 42: Mejor desempeño red NARX. 25 neuronas. Tiempo de retardo 2. Encoder izquierdo

Se observa un buen desempeño para esta red NARX (25 neuronas en la capa oculta y tiempo de retardo 2), no observándose ningún fenómeno de underfitting ni overfitting.

Al igual que para el caso del encoder derecho, se ha estudiado la evolución de este desempeño a medida que aumentamos el tamaño de la capa oculta y se ha observado la aparición de sobreajustes tal y como se muestra en la siguiente gráfica:

Tamaño de la capa oculta (Layer size) :200 neuronas. Tiempo de retardo (Time delay):2

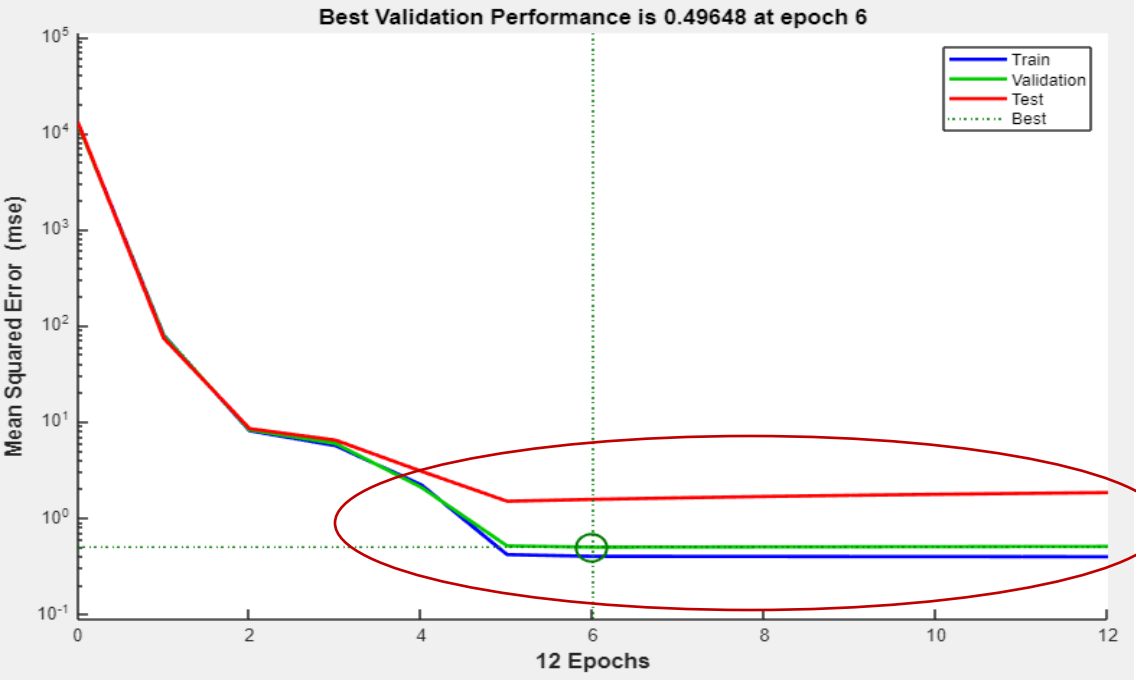

Ilustración 43: Sobreajuste en red NARX. 200 neuronas. Tiempo de retardo 2. Encoder izquierdo.

En la anterior ilustración, el inicio de la aparición de overfitting se muestra marcado mediante el círculo rojo. En esta zona se observa como el error cuadrático medio de la fase de test va incrementándose mientras el de la fase de entrenamiento se mantiene constante.

4.2.3.2 Red tipo NAR

En el recuadro rojo se muestra la red que mejores resultados ha proporcionado en términos de error cuadrático medio y desempeño.

	RED	Nº NEURONAS	TIEMPO RETARDO	FASE	R	MSE	ITERACIÓN CON MAYOR RENDIMIENTO EN EL AJUSTE
ENCODER IZQUIERDO	NAR	25	1	Training	0.9975	0.8585	3
				Validation	0.9979	0.7417	
				Test	0.9981	0.6768	
		25	2	Training	0.9987	0.4700	8
				Validation	0.9985	0.5076	
				Test	0.9983	0.5531	
		50	1	Training	0.9973	0.9314	2
				Validation	0.9980	0.6832	
				Test	0.9972	0.9739	
		50	2	Training	0.9987	0.4361	9
				Validation	0.9982	0.6148	
				Test	0.9981	0.6426	
		100	1	Training	0.9979	0.7222	4
				Validation	0.9975	0.8407	
				Test	0.9974	0.8723	
		100	2	Training	0.9978	1.0049	3
				Validation	0.9961	1.5568	
				Test	0.9974	1.1805	
		200	1	Training	0.9968	1.1201	1
				Validation	0.9934	2.3859	
				Test	0.9684	10.7099	
		200	2	Training	0.9987	0.4595	6
				Validation	0.9956	1.4730	
				Test	0.9936	2.1955	

Tabla 8: Listado ajustes con red NAR. Encoder izquierdo.

**R: regression R-value*

**MSE: Mean-Squared-Error*

Tamaño de la capa oculta (Layer size): 25 neuronas. Tiempo de retardo (Time delay):2. NAR encoder izquierdo

- *Respuesta:*

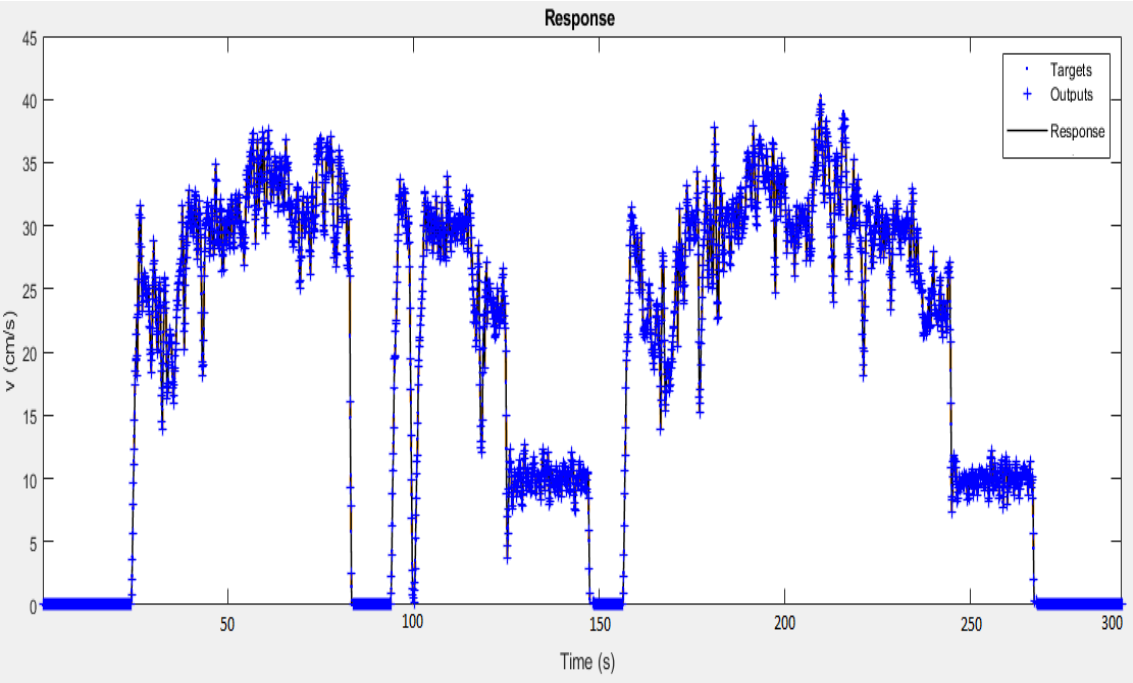

Ilustración 44: Respuesta red NAR. Tamaño capa oculta :25. Tiempo retardo:2. Encoder izquierdo.

- *Regresión en cada fase:*

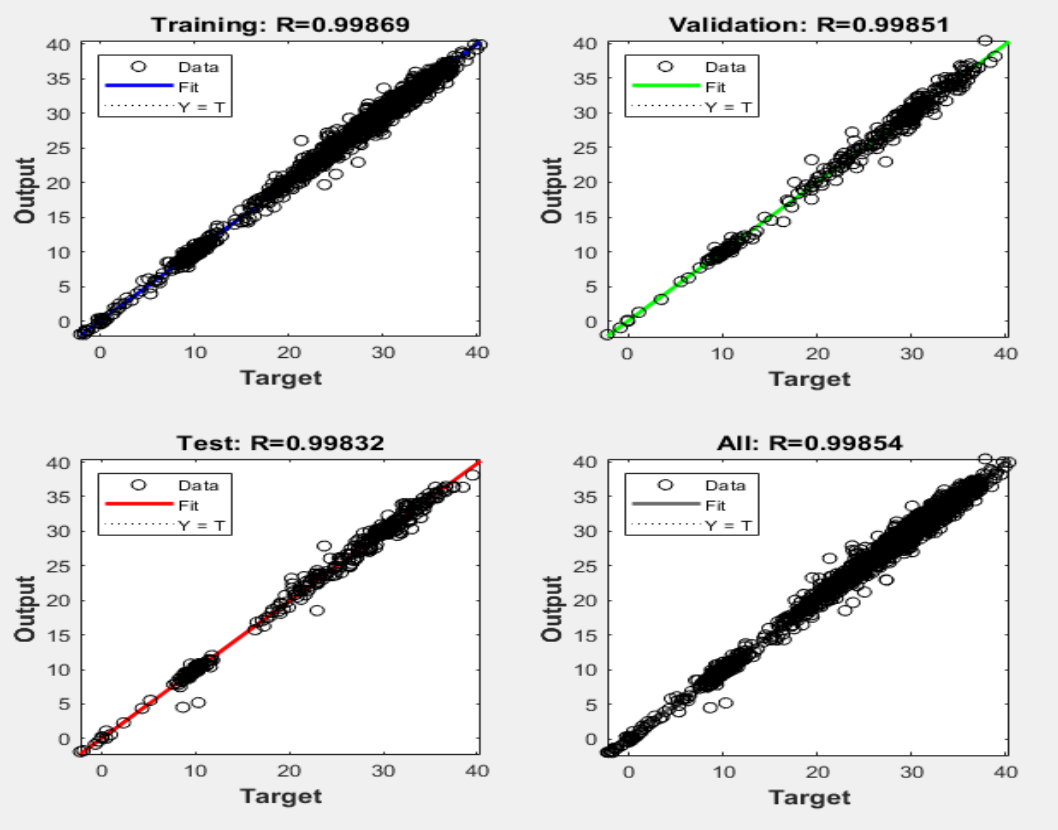

Ilustración 45: Regresión en cada fase. Red NAR. Tamaño capa oculta 25. Tiempo retardo:2. Encoder izquierdo.

Al igual que en todos los casos anteriores, se obtienen unos ajustes muy elevados.

- *Desempeño:*

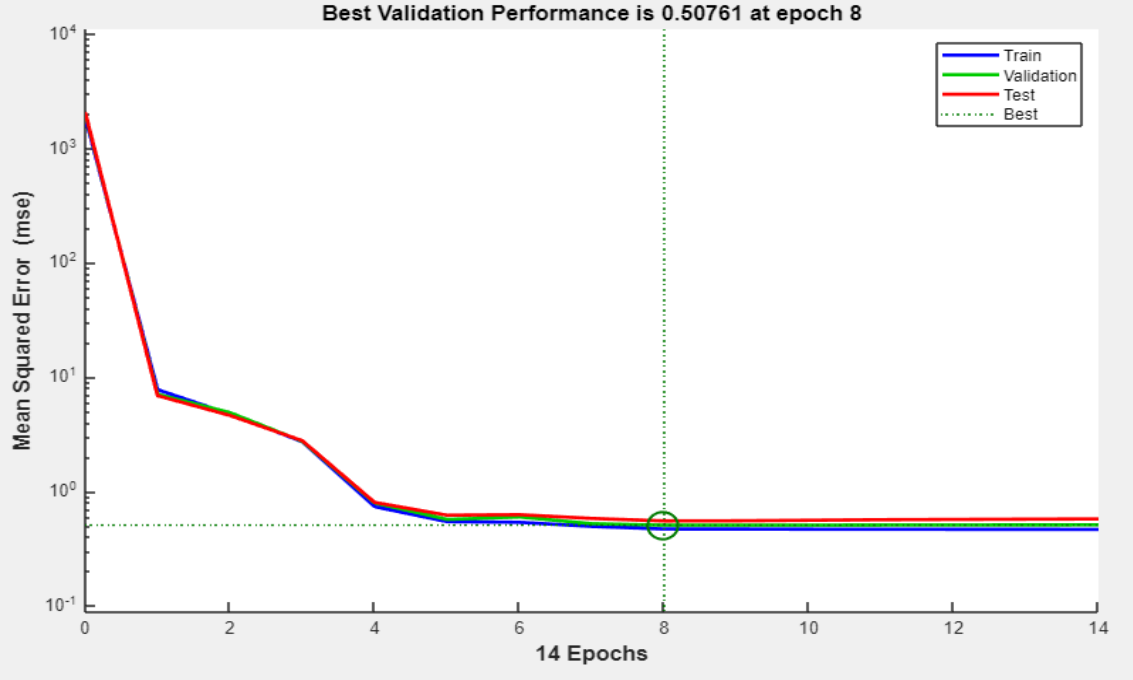

Ilustración 46: Mejor ajuste red NAR. 25 neuronas. Tiempo de retardo 2. Encoder izquierdo.

Se observa un desempeño muy bueno para esta red NAR (25 neuronas en la capa oculta y tiempo de retardo 2), no observándose ningún fenómeno de underfitting ni overfitting.

Al igual que se ha realizado con las redes anteriores, se procede a evaluar el desempeño de las redes NAR con un número mayor de neuronas en la capa oculta.

Tamaño de la capa oculta (Layer size): 200 neuronas. Tiempo de retardo (Time delay):1

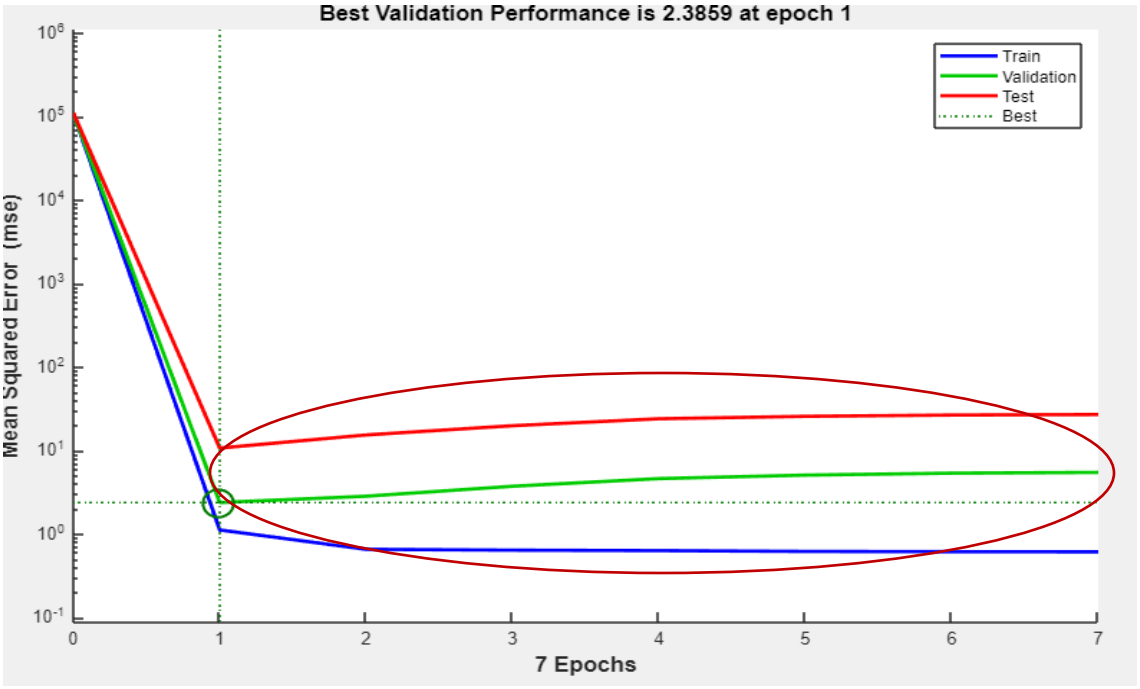

Ilustración 47: Sobreajuste en red NAR. 200 neuronas. Tiempo de retardo 1. Encoder izquierdo.

Tamaño de la capa oculta (Layer size): 200 neuronas. Tiempo de retardo (Time delay):2

Ilustración 48: Sobreajuste en red NAR. 200 neuronas. Tiempo de retardo 2. Encoder izquierdo.

Se puede observar en las ilustraciones 47 y 48 como el fenómeno de sobreajuste se manifiesta de forma visible, observándose la separación cada vez más pronunciada entre las gráficas que muestran el error cuadrático medio entre la fase de entrenamiento y test. Mientras el error de la fase de entrenamiento disminuye ligeramente a medida que avanzamos de iteración, el error de la fase de test va incrementándose.

Después de haber trabajado con ambos tipos redes para el encoder izquierdo, se ha observado como la red NARX tiene una menor tendencia a la aparición de este sobreajuste, originándose en redes con mayor número de neuronas en la capa oculta frente a las necesarias para su aparición en la red NAR. Este fenómeno puede ser explicado a consecuencia de que la red NARX tiene como entrada dos series de datos x(t) e y(t) frente a sólo una serie en la red NAR, y(t). Esto permite a la red NARX una entrada mayor de valores para su entrenamiento, y por tanto la capacidad de entrenarse con un número mayor de valores, obteniéndose así una red más robusta con una capacidad de generalización mejor.

4.2.4 Cálculo del coste computacional. Redes neuronales. Encoder derecho e izquierdo

Al igual que se realizó para los modelos polinómicos, a continuación se procede a calcular el coste computacional de la red NAR y NARX con 25 neuronas en la capa oculta, por ser éstas las que mejor ajuste han proporcionado para cada encoder. El cálculo del coste computacional se realiza siguiendo el procedimiento explicado en el apartado 2.4.4, en donde se obtuvo la siguiente expresión

$$T_c\, red = \left((a * K_M + a * T_{Ca} + K_M) + K_\theta\right) * b * c$$

- Tc= Complejidad computacional en unidades de coste.
- a= nº de entradas.
- b= nº neuronas en la capa oculta.

- c= nº capas.
- T_{CA} = coste de la suma =1 (definido en la ec 2.28).
- K_M = coste de la función suma =1,667 (definido en la ec 2.35).
- K_θ = coste de la función exponencial= 2.886 (definido en la ec 2.36).

Para el caso del número de entradas (a), la red NARX dispondrá del doble de valores frente a la red NAR. La red NARX dispone de dos series, la serie y(t) con los valores de la velocidad y otra serie x(t) con los valores del setpoint, tratándose ambos de un array de 2909x1 valores.

A continuación se muestran los resultados obtenidos:

ENCODER DERECHO							
NAR 25_2				**NARX 25_2**			
Parámetros red			TC	Parámetros red			TC
a	b	c		a	b	c	
2909	25	1	**194071,4**	5818	25	1	**388028,98**

Tabla 9: Coste computacional NN. Encoder derecho.

ENCODER IZQUIERDO							
NAR 25_2				**NARX 25_2**			
Parámetros red			TC	Parámetros red			TC
a	b	c		a	b	c	
2909	25	1	**194071,4**	5818	25	1	**388028,98**

Tabla 10: Coste computacional NN. Encoder izquierdo

** Red NAR y NARX con nº neuronas en la capa oculta=25 y tiempo de retardo=2.*

Se puede comprobar como la red NARX, al disponer de un mayor número de entradas, tiene un coste computacional mucho mayor frente a la red NAR. No obstante, aunque el coste computacional sea mayor, se ha observado como la red NARX se

comporta mejor frente al overfitting que la red NAR, siendo este motivo por el que se seleccionará como mejor red para el ajuste.

4.3 Comparativa de la mejor técnica paramétrica vs mejor red neuronal para cada encoder

Una vez efectuadas las identificaciones mediante los modelos paramétricos y redes neuronales, se procede a realizar una comparación entre la técnica y red con la que mejores valores se han obtenido para cada encoder.

A continuación se indican, a modo de resumen, las tablas con los valores de complejidad y MSE obtenidos para cada técnica y red, así como sus ajustes para ambos encoders.

ENCODER DERECHO	TÉCNICA PARAMÉTRICA	RED NEURONAL
	amx2831341	NARX 25_2*
MSE	45,84	0,4554 (FASE TEST)
COMPLEJIDAD COMPUTACIONAL	254,92	388028,98

Tabla 11: Comparación mejor técnica paramétrica vs mejor red neuronal. Encoder derecho.

ENCODER IZQUIERDO	TÉCNICA PARAMÉTRICA	RED NEURONAL
	amx2820211	NARX 25_2*
MSE	34,21	0,5003 (FASE TEST)
COMPLEJIDAD COMPUTACIONAL	190,91	388028,98

Tabla 12: Comparación mejor técnica paramétrica vs mejor red neuronal. Encoder Izquierdo.

	ENCODER DERECHO	ENCODER IZQUIERDO
TÉCNICA PARAMÉTRICA	**amx2831341**	**amx2820211**
AJUSTE RESPECTO DATOS ENCODER	57,72	62,62

Tabla 13: Ajuste técnicas paramétricas. Encoder derecho/izquierdo.

RED NEURONAL NARX 25_2*	ENCODER DERECHO	ENCODER IZQUIERDO
	R	**R**
FASE ENTRENAMIENTO	**0,9987**	**0,9986**
FASE VALIDACIÓN	**0,9986**	**0,9987**
FASE TEST	**0,9986**	**0,9985**

Tabla 14: Ajuste red neuronal NARX. Encoder derecho/izquierdo.

** nº neuronas en la capa oculta=25, tiempo de retardo=2.*

MSE: Mean-Squared-Error.

R: regression R-value.

Analizando los valores anteriores, se puede comprobar como las redes neuronales consiguen un ajuste mucho mejor frente a las técnicas paramétricas a cambio de un alto coste computacional. Este mayor coste es consecuencia de todas las iteraciones y reajustes realizados por la red durante su proceso de aprendizaje.

Para los valores de los encoders, su identificación mediante técnicas paramétricas sufre de underfitting, poniéndose de manifiesto la incapacidad de estimar fielmente sistemas tan dinámicos mediante el empleo únicamente de técnicas polinómicas. La utilización de estas técnicas será más apropiada para sistemas con valores de datos más lineales y progresivos.

Por su parte, las redes neuronales NARX 25_2 (tamaño capa oculta: 25 neuronas, tiempo retardo: 2) consiguen en ambos encoders unos ajustes (R) y errores cuadráticos medios muy próximos entre la etapa de entrenamiento, validación y test. No se observa la aparición de fenómenos de overfitting en su desempeño, comprobándose como se consiguen unos ajustes muy buenos. El elevado coste computacional de estas redes frente a las técnicas paramétricas queda compensado por la obtención de un modelo robusto y con una capacidad de estimación y generalización mucho mejores.

Capítulo 5 - Conclusiones y trabajos futuros

En el presente trabajo se han empleado los datos de velocidad de pulsos proporcionados por los encoder de la rueda derecha e izquierda de un AGV industrial para la obtención de modelos que reflejen el comportamiento de este sistema. Previamente se han procesado los datos, eliminando valores repetidos, outliers y ruido, para posteriormente, realizar una comparación entre el ajuste obtenido mediante diferentes técnicas paramétricas polinómicas y modelos basados en redes neuronales. De igual forma, se ha procedido a la comparación de las salidas obtenidas de los modelos respecto a la salida real del sistema.

Se ha podido comprobar como el movimiento de un AGV es muy complejo de modelar debido a su dinámica no lineal y al verse influido por fuerzas externas que afectan a los datos de trabajo.

Se ha observado como los métodos paramétricos polinómicos no han sido capaces de generar estimaciones que consigan una representación fiel de estos sistemas, no consiguiendo un buen ajuste y alta generalización del modelo. Con las redes neuronales se han conseguido buenos ajustes mediante sucesivos entrenamientos de la red. Igualmente, se ha comprobado cómo, una vez alcanzado una buena configuración de la red, el empleo de redes con mayor número de neuronas en la capa oculta no repercutía en un mejor ajuste ni en ventajas computacionales, más bien todo lo contrario, observándose la aparición de fenómenos de overfitting. En estos casos, se observaba como la red comenzaba un proceso de memorización y no de aprendizaje, perdiéndose la capacidad de generalización del modelo para diferentes datos, a la vez que se incrementaba el coste computacional por tratarse de una red de mayor tamaño.

Después de haber trabajado con dos tipos de redes neuronales, NAR y NARX, se ha observado como esta última red es más resistente a la aparición de sobreajuste al disponer de dos series de datos diferentes como entrada para realizar su entrenamiento, frente a una sola serie disponible en el caso de la red NAR. Se ha observado como la aparición de overfitting se originaba en redes NARX con mayor tamaño de la capa oculta frente a la necesaria para su aparición en la redes NAR.

Como posibles líneas de continuación del presente trabajo se plantea:

- Probar y comparar técnicas de regularización diferentes a la disminución del número de neuronas de la capa oculta, como pueden ser la regularización de pesos y dropout y observar su influencia en la aparición de overfitting.

- Programar el entrenamiento para que lleve a cabo su parada en el punto óptimo donde se minimice el error y se evite la aparición de overfitting.

- Realizar una optimización de la profundidad de la red atendiendo a la disminución del error y aparición de sobreajuste.

- Implementar un controlador generado con ayuda de la identificación y observar cómo se comporta el AGV en la realidad.

- Probar diferentes algoritmos de entrenamiento y comparar sus resultados.

Capítulo 6 - Bibliografía

[1] Carpio, J. B., & Martí, J. M. S. (2006). Sistema industrial de múltiples vehículos autónomos guiados por láser. Publicacions de la Universitat Jaume I.

[2] https://revistaderobots.com/robots-y-robotica/robot-agv-aiv-los-vehiculos-de-guiado-automatico-inteligentes/

[3] Departamento de informática de Sistemas y Computadores UPV. http://www.disca.upv.es/penalver/Investigaci%F3n/Identificaci%F3n/IdentiPara.PDF

[4]. Universidad de Oviedo. ISA, "Diseño de sistemas de control por computador". Octubre 2008

[5] Diaz, Martinez, M y Muñoz, Mansilla, R. (Julio 2012). 3ª Edición. Apuntes de identificación de sistemas. Departamento de Informática y Automática. ETSI Informática. UNED

[6] Matich, D. J. (2001). Redes Neuronales: Conceptos básicos y aplicaciones. Universidad Tecnológica Nacional, México, 41, 12-16.

[7] Berzal, F. Inteligencia Artificial. Haykin (2009). Redes neuronales artificiales. neural network and learning machines. Prentice hall.

https://elvex.ugr.es/decsai/deep-learning/slides/NN3%20Backpropagation.pdf

[8] http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm

[9] Gil, R. V., & Páez, D. G. (2007). Identificación de sistemas dinámicos utilizando redes neuronales RBF. Revista iberoamericana de automática e informática industrial RIAI, 4(2), 32-42.

[10] Neural net time series - Mathworks [En línea].

Available: https://es.mathworks.com/help/deeplearning/ref/neuralnettimeseries-app.html;jsessionid=96a167dd590c92cee9baaebf41c6

[11] Automated Guided Vehicle (AGV) Market with COVID-19 Impact Analysis, by Type (Tow Vehicles, Unit Load Carriers, Forklift Trucks, Assembly Line Vehicles, Pallet

Trucks), Navigation Technology, Industry, and Region - Global Forecast to 2026. [En línea].

Available: https://www.marketsandmarkets.com/Market-Reports/automated-guided-vehicle-market-27462395.html

[12] Sánchez, R., Sierra-García, J. E., & Santos, M. (2022). Modelado de un AGV híbrido triciclo-diferencial. Revista Iberoamericana de Automática e Informática Industrial, 19(1), 84-95.

[13] Filter outliers using Hampel identifier - Simulink - MathWorks España. [En línea]. Available: https://es.mathworks.com/help/dsp/ref/hampelfilter.html

[14] Filtrar y suavizar datos - MATLAB & Simulink - MathWorks España [En línea].

Available: https://es.mathworks.com/help/curvefit/smoothing-data.html

[15] Sierra (2014). Comparative study of unmanned aerial vehicle models obtained by parametric and soft computing techniques. Proyecto final de Máster. Director: Santos, Peñas, Matilde.

[16] Sierra-García, J. E., & Santos, M. (2020). Mechatronic modelling of industrial AGVs: a complex system architecture. Complexity, 2020.

[17] Chang, Y. W., Hsieh, C. J., Chang, K. W., Ringgaard, M., & Lin, C. J. (2010). Training and testing low-degree polynomial data mappings via linear SVM. Journal of Machine Learning Research, 11(4).

[18] fpe. Akaike's Final Prediction Error for estimated model [En línea]

Available: https://es.mathworks.com/help/ident/ref/idmodel.fpe.html

[19] Lehmann, E. L., & Casella, G. (2006). Theory of point estimation. Springer Science & Business Media.

[20] Knuth, D. E. (2014). Art of computer programming, volume 2: Seminumerical algorithms. Addison-Wesley Professional.

[21] Marín, Diazaraque, J.M. (2012). Introducción a las redes neuronales aplicadas. CURSO DE EXPERTOS DE U.C.M. Universidad Carlos III de Madrid.

[22] Sancho, Caparrini, F. Redes Neuronales: una visión superficial (us.es).

[23] Constaín, A., Torres, K., Arango, J., & Vivas, A. (2009). Modelado, Identificación Paramétrica y Control del Robot SCORBOT-ER 5 PLUS. In Presentado en el VIII Congreso de la Asociación Colombiana de Automática, Cartagena, Colombia.

[24] Torres, S., Becerra, V. M., Méndez, J. A., Acosta, L., Marichal, G. N., Hamilton, A. F., & Sigut, M. Identificación de un sistema robótico mediante redes neuronales dinámicas.

[25] System Identification Overview - MATLAB & Simulink - MathWorks España [En línea]

Available:https://es.mathworks.com/help/ident/gs/about-system-identification.html