



Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias

IDENTIFICACIÓN NO LINEAL DE PÉNDULO INVERTIDO SOBRE CARRO MÓVIL
TESIS

Que como parte de los requisitos para obtener el grado de
Maestro en Ciencias

Presenta:


Ing. Luis Mario Lizárraga Orozco

Dirigido por:

M. en C. Guillermo Ronquillo Lomeli

SINODALES

M. en C. Guillermo Ronquillo Lomeli
Presidente



Firma

Dr. Mario Trejo Perea
Secretario



Firma

Dr. Juan Carlos Jáuregui Correa
Vocal



Firma

Dr. Marco Tulio Angulo Ballesteros
Suplente

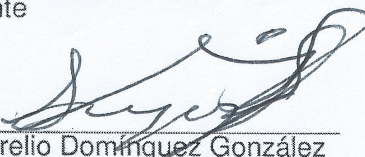


Firma

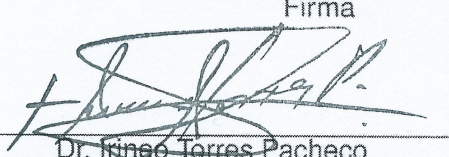
Dr. José Gabriel Ríos Moreno
Suplente



Firma



Dr. Aurelio Domínguez González
Director de la Facultad



Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Septiembre de 2013
México

RESUMEN

Esta investigación concierne con la aplicación de diferentes tipos de redes neuronales artificiales (RNAs) para la identificación del modelo no lineal de un péndulo invertido simple sobre un carro móvil (PISC). Se muestra que una red neuronal polinomial es más eficiente en cuestiones de aproximación que una red neuronal prealimentada con estructura de perceptrón multicapa (MLP). Para soportar la afirmación anterior se identificó el modelo no lineal del PISC tanto en simulación como en un prototipo de laboratorio. Las redes que se utilizaron en este trabajo fueron: una red no lineal autoregresiva con entrada externa (NARX) prealimentada con estructura de MLP y con algoritmo de retropropagación (BP) para la actualización de pesos, una red con polinomio de Volterra como función base (VPBF) y una red con polinomio de Chebyshev como función base (CBF). Se desarrolló un algoritmo compacto para implementar una red neuronal en el cuál se puede utilizar cualquier polinomio como función base sin necesidad de adecuar, adaptar o crear un algoritmo específico para cada uno. Los tres tipos de red neuronal se entrenaron durante 60 segundos utilizando el error entre la salida del modelo y la salida estimada por la red. Las redes se validaron fijando los pesos finales del entrenamiento y aplicándose a otros 15 segundos de datos diferentes a los de entrenamiento. Las redes polinomiales mostraron un mejor desempeño en comparación con la red multicapa prealimentada, tanto con los datos de simulación como con los datos del prototipo experimental. Mediante los resultados de un diseño experimental se muestra el desempeño superior de las redes neuronales polinomiales.

(Palabras clave: Identificación, Sistema no lineal, Red neuronal artificial, Función base polinomial, Polinomio de Volterra, Polinomio de Chebyshev)

SUMMARY

This research is concerned with the application of different kinds of artificial neural networks (ANNs) for identification of the nonlinear model of a single inverted pendulum on a moving cart (SIPC). It is shown that a polynomial neural network is more efficient on approximation issues than a feedforward neural network with multilayer perceptron (MLP) structure. To support the latter, the nonlinear model of a SIPC, on simulation and on a laboratory prototype, was identified. The neural networks used in this work were: a nonlinear autoregressive with exogenous input (NARX) feedforward with MLP structure and backpropagation (BP) algorithm for weights update network, a Volterra polynomial as basis function (VPBF) network and a Chebyshev polynomial as basis function (CBF) network. A compact algorithm for neural network implementation was developed, in this algorithm every polynomial as basis function can be used without the necessity of adjusting, adapting or creating a specific one for each one. The three kinds of neural network were trained over 60 seconds using the error between the model's actual output and the network's estimated output. The networks were validated by fixing the final training weights and applying to them (the networks) to another 15 seconds of data, different from the training data. The polynomial networks showed a better performance in comparison with the feedforward multilayer network, with both simulation and prototype data. The efficiency of polynomial networks was proved through the results of a design of experiments.

(Key words: Identification, Nonlinear system, Artificial neural network, Polynomial basis function, Volterra Polynomial, Chebyshev polynomial)

A mis padres: Humberto y Tere
– Quienes nunca pierden la fe

AGRADECIMIENTOS

- A mi familia, que no falla, que siempre ha estado ahí para mí.
- Al M. en C. Guillermo Ronquillo Lomeli, gracias por todo el apoyo, por los consejos y la confianza, sin usted este trabajo no habría sido posible.
- A los Dres.: Gabriel, Juan Carlos, Marco y Mario; gracias por sus comentarios y consejos para hacer a éste un mejor trabajo.
- A todos los profesores que durante la Maestría me guiaron y compartieron sus conocimientos y experiencia.
- A mis compañeros de estudio, en especial a mis “hermanos” de Maestría: Adrián, Basurto, Molano y Pedro, gracias por su amistad y haber hecho estos dos años tan divertidos.
- Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por haberme becado durante éstos estudios.
- A la Universidad Autónoma de Querétaro (UAQ) por dejarme formar parte de su comunidad estudiantil.
- A Denisse, por sus amigables consejos y por no dejar que me volviera loco en estos dos años.
- A Marlen, por su apoyo y sus comentarios sinceros, siempre directos y siempre correctos.
- A todas esas buenas personas de Querétaro que me han aceptado como uno más de los suyos, que me han abierto las puertas de sus casas, que me han dado su cariño y confianza... Muchas gracias.

ÍNDICE GENERAL

RESUMEN	I
SUMMARY	II
DEDICATORIA	III
AGRADECIMIENTOS	IV
1. INTRODUCCIÓN	1
1.1. Motivación	1
1.2. Objeto de estudio	1
1.3. Descripción del problema	4
1.4. Objetivo	5
1.5. Hipótesis	5
1.6. Contribución	5
1.7. Distribución de capítulos	6
2. REVISIÓN DE LITERATURA	7
3. DESCRIPCIÓN MATEMÁTICA DEL SISTEMA	12
3.1. Introducción	12
3.2. Modelo matemático del sistema	13
3.2.1. Modelado por segunda ley de Newton	13
3.2.2. Ecuación de estado no lineal del PISC	18
3.2.3. Aproximación lineal	19
3.2.4. Controlador por retroalimentación completa del estado	22
3.3. Modelado del sistema de movimiento lineal	23
3.3.1. Modelado del servomotor de C.D.	23
3.3.2. Modelado completo del sistema de movimiento	24
4. IDENTIFICACIÓN CON RED NEURONAL MULTICAPA	25
4.1. Introducción	25
4.2. Modelado con red pre-alimentada	26
5. IDENTIFICACIÓN CON RED NEURONAL POLINOMIAL	32
5.1. Introducción	32
5.2. Modelo NARX	33

5.3.	Modelado no lineal por redes con polinomio de Volterra (VPBF)	34
5.4.	Modelado no lineal por redes con polinomio de Chebyshev (CBF)	36
5.5.	Algoritmo de actualización de pesos	39
5.5.1.	Mínimos cuadrados ortogonales (MCO)	39
5.5.2.	Mínimos cuadrados recursivos (MCR)	41
5.6.	Reducción por método de Gram-Schmidt	42
5.7.	Notas del capítulo	44
6.	METODOLOGÍA	46
6.1.	Prototipo de laboratorio	46
6.2.	Algoritmo compacto para red polinomial	47
6.3.	Simulación	49
6.4.	Experimento	53
7.	RESULTADOS Y DISCUSIÓN	56
7.1.	Introducción	56
7.2.	Red NARX prealimentada	58
7.3.	Red con polinomio de Volterra como función base	70
7.4.	Red con polinomio de Chebyshev como función base	82
7.5.	Análisis de resultados	94
8.	CONCLUSIONES	97
	BIBLIOGRAFÍA	99
	APÉNDICES	107
A.	Diseño mediante la ubicación de los polos	108
B.	Programas de MATLAB	111
B.1.	Programa para MLP	111
B.2.	Programa para red polinomial compacta	112
B.3.	Programa para entrenamiento de la red VPBF	113
B.4.	Función para generar las funciones base de Volterra	113
B.5.	Función para evaluar la red VPBF	114
B.6.	Programa para entrenamiento de la red CBF	115
B.7.	Función para generar las funciones base de Chebyshev	115
B.8.	Función para evaluar la red CBF	116
B.9.	Función para la reducción por Gram - Schmidt	116
B.10.	Programa para importar datos a MATLAB desde un archivo de texto	117
C.	Programas de LabVIEW	118
C.1.	Programa para adquirir datos del PISC	118
C.2.	Programa para convertir de formato .bin a formato .txt	119
D.	Glosario	120

ÍNDICE DE TABLAS

6.1. Parámetros del prototipo de laboratorio.	48
7.1. Relación entre factor de olvido y número de datos significativos a la aproximación por MCR.	57
7.2. Datos del diseño de experimentos MLP (simulación).	59
7.3. Análisis de variancia para MLP (simulación).	59
7.4. Recursos de cómputo para MLP simulación.	61
7.5. Correlación MLP simulación	61
7.6. Datos del diseño de experimentos MLP (experimento).	64
7.7. Análisis de variancia para MLP (experimento).	65
7.8. Recursos de cómputo para MLP experimento.	67
7.9. Correlación MLP experimento	67
7.10. Datos del diseño de experimentos VPBF (simulación).	70
7.11. Análisis de variancia para red VPBF (simulación).	71
7.12. Recursos de cómputo para red VPBF, simulación.	73
7.13. Correlación red VPBF, simulación	73
7.14. Datos del diseño de experimentos VPBF (experimento).	76
7.15. Análisis de variancia para red VPBF (experimento).	77
7.16. Recursos de cómputo para red VPBF, experimento.	79
7.17. Correlación red VPBF, experimento	79
7.18. Datos del diseño de experimentos CBF (simulación).	82
7.19. Análisis de variancia para red CBF (simulación).	83
7.20. Recursos de cómputo para red CBF, simulación.	85
7.21. Correlación red CBF, simulación	85
7.22. Datos del diseño de experimentos CBF (experimento).	88
7.23. Análisis de variancia para red CBF (experimento).	89
7.24. Recursos de cómputo para red CBF, experimento.	91
7.25. Correlación red CBF, EXP	91
7.26. Comparación de complejidad de cálculo para MLP, VPBF y CBF.	96
7.27. Coeficiente de correlación lineal de Pearson para el mejor desempeño.	96
7.28. Coeficiente R^2 para MLP, VPBF y CBF. Datos experimentales.	96

ÍNDICE DE FIGURAS

1.1.	Un péndulo invertido en un carro móvil.	3
2.1.	Esquema de identificación.	8
3.1.	Péndulo invertido sobre un carro.	13
3.2.	Péndulo simple.	16
3.3.	Diagrama esquemático de una realimentación de estado.	22
3.4.	Diagrama a bloques del prototipo físico del PISC.	23
4.1.	Esquema de una neurona.	25
4.2.	Arquitectura de un perceptrón multicapa con una capa oculta.	27
4.3.	Gráfico de flujo de señal resaltando los detalles de la neurona oculta j conectada a la neurona de salida k	28
4.4.	Gráfica de una tangente hiperbólica	29
4.5.	Diagrama de flujo para identificación con MLP.	31
5.1.	Diagrama de flujo de una red polinomial (Volterra).	36
5.2.	Algoritmo de generación de funciones base (Volterra).	37
5.3.	Diagrama de flujo de una red polinomial (Chebyshev).	38
5.4.	Algoritmo de generación de funciones base (Chebyshev).	39
6.1.	Vehículo del RT124.	46
6.2.	Vista lateral del vehículo.	47
6.3.	Diagrama de flujo de algoritmo compacto.	48
6.4.	Diagrama general del sistema.	49
6.5.	Perturbaciones no acotadas.	50
6.6.	Subsistema para la dinámica del PISC.	51
6.7.	Datos adquiridos de la simulación.	52
6.8.	Identificación mediante red neuronal artificial.	53
6.9.	Sistema experimental RT124.	54
6.10.	Sistema experimental y computadora.	55
6.11.	Datos adquiridos del experimento.	55
7.1.	Efectos ordenados para el diseño 2^3 MLP (simulación).	60
7.2.	Comportamiento de los efectos principales, MLP (simulación).	60
7.3.	Comportamiento de las interacciones, MLP (simulación).	61
7.4.	Posición angular del péndulo, MLP (simulación).	62
7.5.	Desplazamiento del carro, MLP (simulación).	62

7.6. Velocidad angular del péndulo, MLP (simulación).	63
7.7. Velocidad lineal del carro, MLP (simulación).	63
7.8. Efectos ordenados para el diseño 2^3 MLP (experimento).	65
7.9. Comportamiento de los efectos principales, MLP (experimento).	66
7.10. Comportamiento de las interacciones, MLP (experimento).	66
7.11. Posición angular del péndulo, MLP (experimento).	68
7.12. Desplazamiento del carro, MLP (experimento).	68
7.13. Velocidad angular del péndulo, MLP (experimento).	69
7.14. Velocidad lineal del carro, MLP (experimento).	69
7.15. Efectos ordenados para el diseño 2^3 VPBF (simulación).	71
7.16. Comportamiento de los efectos principales, VPBF (simulación).	72
7.17. Comportamiento de las interacciones, VPBF (simulación).	72
7.18. Posición angular del péndulo, VPBF (simulación).	74
7.19. Desplazamiento del carro, VPBF (simulación).	74
7.20. Velocidad angular del péndulo, VPBF (simulación).	75
7.21. Velocidad lineal del carro, VPBF (simulación).	75
7.22. Efectos ordenados para el diseño 2^3 VPBF (experimento).	77
7.23. Comportamiento de los efectos principales, VPBF (experimento).	78
7.24. Comportamiento de las interacciones, VPBF (experimento).	78
7.25. Posición angular del péndulo, VPBF (experimento).	80
7.26. Desplazamiento del carro, VPBF (experimento).	80
7.27. Velocidad angular del péndulo, VPBF (experimento).	81
7.28. Velocidad lineal del carro, VPBF (experimento).	81
7.29. Efectos ordenados para el diseño 2^3 CBF (simulación).	83
7.30. Comportamiento de los efectos principales, CBF (simulación).	84
7.31. Comportamiento de las interacciones, CBF (simulación).	84
7.32. Posición angular del péndulo, CBF (simulación).	86
7.33. Desplazamiento del carro, CBF (simulación).	86
7.34. Velocidad angular del péndulo, CBF (simulación).	87
7.35. Velocidad lineal del carro, CBF (simulación).	87
7.36. Efectos ordenados para el diseño 2^3 CBF (experimento).	89
7.37. Comportamiento de los efectos principales, CBF (experimento).	90
7.38. Comportamiento de las interacciones, CBF (experimento).	90
7.39. Posición angular del péndulo, CBF (experimento).	92
7.40. Desplazamiento del carro, CBF (experimento).	92
7.41. Velocidad angular del péndulo, CBF (experimento).	93
7.42. Velocidad lineal del carro, CBF (experimento).	93

I. INTRODUCCIÓN

1.1. Motivación

La identificación de sistemas dinámicos complejos es una parte importante en la teoría de control. Este interés recae en la necesidad de dar nuevas soluciones a problemas de hace tiempo en control automático, trabajar con sistemas más y más complejos, satisfacer criterios de diseño más estrictos; y cumplir con los puntos anteriores con cada vez menos conocimiento a priori de la planta. En este contexto, se ha estado realizando un gran esfuerzo dentro del área de identificación de sistemas hacia el desarrollo de modelos no lineales de los procesos reales.

Se sabe que en las pasadas décadas los modelos lineales han sido ampliamente utilizados en identificación de sistemas. Sin embargo, la mayoría de los sistemas encontrados en la práctica son no lineales. En muchos casos, los modelos lineales no son aptos para representar esos procesos reales y los modelos no lineales tienen que ser considerados. Es por tal que el estudio, análisis y modelado de sistemas físicos debe realizarse tomando en cuenta sus relaciones no lineales para obtener modelos que representen con mayor precisión el sistema de interés. También, como se puede notar, entre mejor sea el modelo más cercanos serán el desempeño teórico y el desempeño práctico.

1.2. Objeto de estudio

Los sistemas de control moderno usualmente requieren un conocimiento muy estructurado acerca de la planta a ser controlada; ese conocimiento debe ser representado en términos de ecuaciones diferenciales o en diferencias. Esta descripción matemática del sistema dinámico es llamado modelo (Alanis *et al.*, 2010). Básicamente hay dos maneras de obtener un modelo matemático; puede ser desarrollado de una manera deductiva usando le-

yes físicas, o puede ser inferido de un conjunto de datos recolectados durante un experimento práctico.

Para generar un modelo matemático de un sistema físico se recurre a técnicas de identificación. Los modelos lineales han sido ampliamente usados en identificación de sistemas, pero se tiene el problema de que los sistemas utilizados en la práctica poseen la propiedad de linealidad solo en un cierto rango de operación, todos los sistemas físicos son no lineales en cierto grado. La identificación de sistemas no lineales es más difícil que la de sistemas lineales y por tanto se recurre a técnicas como las redes neuronales artificiales.

En este trabajo se utiliza un péndulo invertido simple sobre un carro móvil (PISC) para llevar a cabo experimentación sobre identificación de sistemas no lineales. Un PISC consiste en una varilla delgada acoplada a un carro móvil que se puede desplazar horizontalmente en el plano $x - y$ (Mohandas y Paritala, 2006; Chaturvedi y Röck, 2008; Milton *et al.*, 2009; Sutradhar *et al.*, 2010).

Un péndulo normal es estable en su posición vertical hacia abajo, un péndulo en su posición vertical invertida es inherentemente inestable y debe ser balanceado, para que pueda mantenerse hacia arriba. Del PISC, Figura 1.1, se toman en cuenta: 1) las variables que definen el estado, posición angular del péndulo (θ), posición del carro (x) y sus respectivas velocidades, 2) los parámetros del péndulo, masa del carro (M), masa del péndulo (m), que se tomará en cuenta como uniformemente distribuida, longitud desde la base del péndulo al centro de masa (l), y 3) la entrada al sistema que es la fuerza aplicada al carro (u).

El PISC es un sistema multivariable no lineal con inestabilidad inherente en su posición invertida (Sutradhar *et al.*, 2010). Como consecuencia, las técnicas lineales no pueden modelar las dinámicas no lineales del sistema. Debido a su naturaleza no lineal, el PISC es usado para ilustrar muchas de las ideas que surgen en el campo del control no lineal (Muskinja y Tovornik, 2006). Las características del péndulo invertido hacen la identificación y control del mismo un desafío (Mohandas y Paritala, 2006). Básicamente el objetivo es estabilizarlo, es decir, mantener un equilibrio o balance (Milton *et al.*, 2009), como ocurre en el propulsor de un cohete, un barco para perforación y extracción de petróleo, o simplemente caminar (Anderson, 1989; Abrahantes *et al.*, 2007).

Por otra parte, tanto el modelado como la identificación de sistemas son parte de la

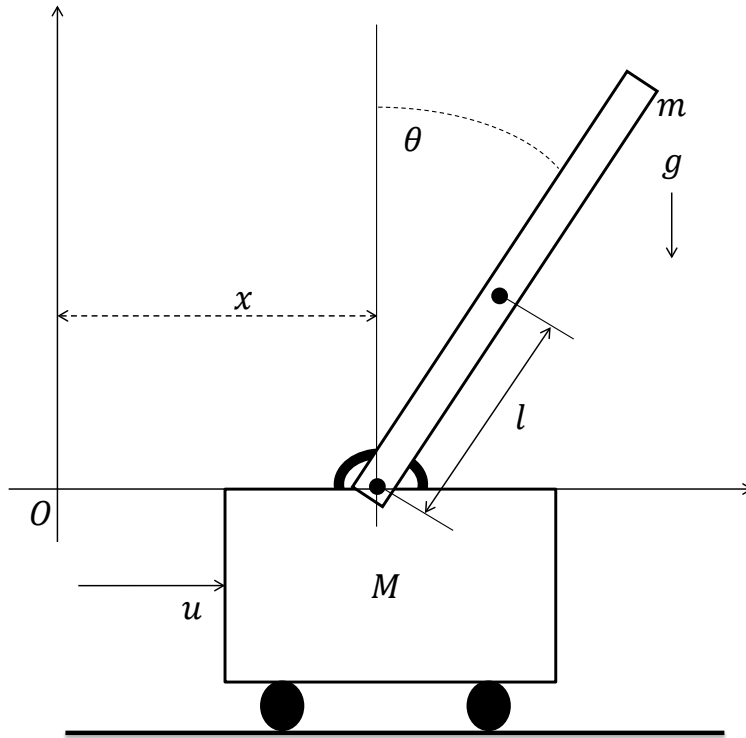


Figura 1.1: Un péndulo invertido en un carro móvil.

teoría de control (Narendra y Parthasarathy, 1989; Ranković y Nikolić, 2008). La búsqueda de un modelo puede incluir un análisis muy detallado de la física del sistema. La identificación del sistema es el procedimiento que desarrolla modelos de un sistema dinámico basándose en las señales de entrada y de salida del sistema.

El propósito de modelar el comportamiento dinámico de sistemas mecánicos es obtener agudeza en la respuesta de un sistema para una entrada dada (Riedmiller, 1993). Por otra parte, la idea básica de la teoría adaptable es estimar las incertidumbres en la planta, con señales medidas “en línea” (Hojati y Gazor, 2002). Las incertidumbres en un sistema pueden ser su dinámica o sus parámetros. Sin embargo, la teoría adaptable convencional solo puede hacer frente a sistemas con estructura dinámica conocida, pero parámetros desconocidos (Sastry y Bodson, 1989).

En años recientes, técnicas de inteligencia computacional, tales como redes neuronales artificiales (RNAs), lógica difusa (FL) y algoritmos neuro-difusos (ANFs) se han convertido en herramientas muy efectivas de identificación y control de plantas no lineales.

Recientemente las redes neuronales se han convertido en una herramienta atractiva que puede ser utilizada para construir un modelo de procesos no lineales complejos. Esto es porque las redes neuronales tienen una habilidad inherente de aprender y aproximar una función no lineal arbitrariamente bien (Zeng *et al.*, 2013). Por consecuencia, provee una posible vía de modelado de procesos no lineales complejos.

En este trabajo se aplican algoritmos de identificación de modelos no lineales, con estructura de red neuronal, a los datos de entrada - salida obtenidos de un sistema real PISC. Debido a que trabajos realizados anteriormente solo aplican redes neuronales pre-alimentadas (FNNs), también conocidas como *feedforward*, para resolver el problema (Mohandas y Paritala, 2006; Sutradhar *et al.*, 2010), y estas redes son aplicadas fuera de línea, surge la idea de utilizar una red neuronal con estructura polinomial que sea más fácil de implementar y a su vez se pueda aplicar en línea. Así, se pretende mejorar los resultados obtenidos mediante FNNs para un prototipo físico, además de generar un algoritmo de identificación compacto con menor consumo de recursos computacionales y de *hardware* a la hora de implementarse en un sistema embebido.

1.3. Descripción del problema

En un PISC una varilla delgada está unida a un carro por medio de un pivote, lo que permite que el péndulo pueda girar libremente en el plano x-y. Múltiples autores se enfocan únicamente en el control del péndulo utilizando como modelo el obtenido matemáticamente con ecuaciones de Euler (Anderson, 1989) o ecuaciones de Lagrange (Bogdanov, 2004) despreciando la fricción en el pivote (Milton *et al.*, 2009), los posibles desgastes del motor de corriente directa (C.D.), así como las posibles variaciones que pueda sufrir el sistema.

Dado que la fidelidad con la que un modelo matemático representa el comportamiento de la planta real depende de la precisión y las dinámicas tomadas en cuenta por quien realiza el modelo, en caso de una variación grande en el sistema no lineal, el modelo dejaría de ser eficiente. Debido a lo anterior se recurre comúnmente a otras formas de identificación de modelos no lineales, como la FL o las RNAs.

En algunos casos no hay suficientes reglas difusas disponibles, a partir de los cono-

cimientos expertos humanos, para construir un controlador difuso. Además, si la dinámica de la planta tiene grandes variaciones en el tiempo (aún siendo lentas) el controlador difuso (no adaptable) no tendrá un desempeño satisfactorio (Hojati y Gazor, 2002) dando lugar a la utilización de redes neuronales que gracias a su capacidad de aprendizaje pueden imitar el comportamiento de una planta para así poder identificar su modelo.

1.4. Objetivo

Identificar el modelo dinámico no lineal de un péndulo invertido simple, en su punto de operación inestable, mediante redes neuronales multicapa y polinomiales y realizar una comparación de los resultados obtenidos para demostrar la eficiencia de la técnica de identificación propuesta.

1.5. Hipótesis

Se puede mejorar el método de identificación de modelos dinámicos no lineales, expresados en series temporales, utilizando una red neuronal con estructura polinomial capaz de adaptarse a variaciones en el tiempo. Además se puede demostrar que la red propuesta es más eficiente y fácil de implementar que la comúnmente utilizada.

1.6. Contribución

En este trabajo se presenta una estructura compacta de red neuronal polinomial. Debido a la forma de la estructura propuesta se pueden utilizar como funciones base los polinomios de Volterra o los polinomios de Chebyshev sin necesidad de modificar la estructura. También una función de activación a la salida de la red puede ser modificada para cubrir requerimientos del usuario. El algoritmo de aprendizaje de la red permite que se implemente en línea. Se muestra que la capacidad de identificación es superior que la del MLP, también se muestra que la complejidad de cálculo es menor. Además, se realizó un diseño experimental factorial, de los resultados de este diseño se observan las tendencias de las redes neuronales y nos da una idea clara del patrón de comportamiento que siguen. Finalmente, se mostró por

métodos estadísticos que la red con funciones base polinomiales de Chebyshev es mejor que la red multicapa y la red con polinomios de Volterra.

1.7. Distribución de capítulos

En el Capítulo 1 se presenta una introducción donde se presenta el objetivo de este trabajo, algoritmos y sistema a utilizar, hipótesis y contribución del proyecto. En el Capítulo 2 se presentan los antecedentes referentes a este trabajo. El Capítulo 3, Capítulo 4 y Capítulo 5 presentan el marco teórico de este trabajo, en ellos se presenta la descripción matemática del sistema y las técnicas de identificación con redes neuronales utilizadas en este trabajo. En el Capítulo 6 se presenta la metodología de este trabajo, aquí se explica cómo se realizó la simulación del sistema con base a lo presentado en el Capítulo 3, también se explica cómo se llevó a cabo la adquisición de datos del sistema real, el cual es un RT-124 de la marca G.U.N.T. Los resultados de simulación y experimento se presentan en el Capítulo 7, también se presenta el diseño experimental y se proporcionan gráficas de tendencia para cada red neuronal. Finalmente, en el Capítulo 8 se presentan las conclusiones generales del trabajo.

II. REVISIÓN DE LITERATURA

El campo de las redes neuronales tiene sus raíces en la neurobiología. La estructura y funcionalidad de las redes neuronales ha sido motivada por la arquitectura del cerebro humano (Liu, 2001).

Definición 2.0.1 *Una red neuronal es una estructura paralela, procesadora de información, que consta de elementos de procesamiento interconectados con canales de señal unidireccionales, llamados conexiones, que tienen un peso.*

Desde 1986, las redes neuronales han sido aplicadas a la identificación de sistemas dinámicos no lineales. El modelado de sistemas dinámicos o “identificación de procesos” es una de las principales aplicaciones de las redes neuronales. Así, en lugar de usar un modelo matemático se implementa una identificación con redes neuronales. De acuerdo con (Liu *et al.*, 1998) se puede representar cualquier función no lineal con una aproximación por red neuronal más un error de aproximación, Figura 2.1.

Definición 2.0.2 *La identificación del sistema es el procedimiento que desarrolla modelos de un sistema dinámico basándose en las señales de entrada y de salida del sistema (Riedmiller, 1993).*

Narendra y Parthasarathy (1990) demostraron que las redes neuronales pueden ser usadas efectivamente para la identificación y control de sistemas dinámicos no lineales, los resultados presentados en simulación indican que los métodos que sugieren, para identificación y control de sistemas dinámicos no lineales de bajo orden, pueden ser muy efectivos.

FNNs de dos y tres capas fueron utilizadas por Kuschewski *et al.* (1993) para la identificación de sistemas dinámicos no lineales, estas redes utilizaban un algoritmo de adaptación de pesos generalizado. Se utilizaron simulaciones para validar el método.

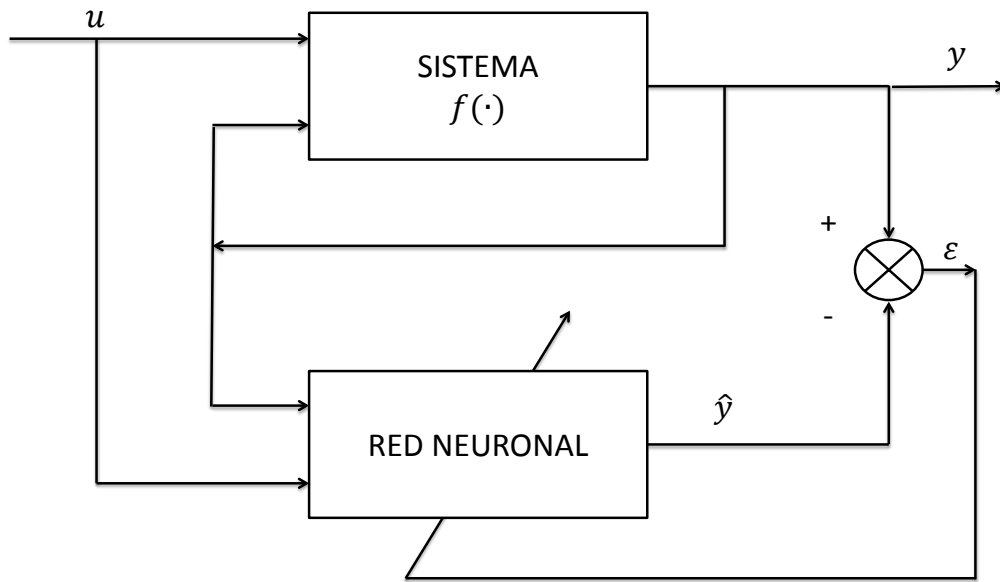


Figura 2.1: Esquema de identificación.

Después, Verschueren (1995) usa una FNN para identificar el modelo de un péndulo invertido usando el método Levenberg-Marquardt para optimizar la red, seguido a esto, utiliza dos redes neuronales para el control del péndulo, los resultados son presentados únicamente en simulación.

Entre tanto, en Luo y Billings (1995) y Luo *et al.* (1996) se presenta un método de ventaneo de datos que utiliza la estructura polinomial de un modelo no lineal autorregresivo con entrada externa (NARX). La estructura polinomial y otros conceptos introducidos en estos trabajos serían de mucha importancia en investigaciones posteriores sobre redes neuronales polinomiales.

Jagannathan y Lewis (1996) presentan una red neuronal multicapa en tiempo discreto para identificación de sistemas dinámicos no lineales.

Un esquema de identificación utilizando redes neuronales con polinomios de Volterra como funciones base (VPBF) es propuesto por Liu *et al.* (1998). Se desarrolla un algoritmo de aprendizaje de pesos recursivo, en línea, para ajustar los pesos de manera que el modelo identificado pueda adaptarse a variaciones en las características y puntos de operación

en sistemas no lineales. El procedimiento de identificación es ilustrado usando ejemplos simulados. Este trabajo tiene como inconvenientes la complejidad del algoritmo de aprendizaje y el consumo de memoria requerido. El mismo año, se reporta en Lee y Jeng (1998) el largo tiempo de entrenamiento de las FNNs y su poca utilidad en aplicaciones en línea. Proponen un modelo, al cual llaman “unificado”, de redes neuronales con polinomios de Chebyshev como función base.

Onder Efe y Kaynak (1999) investigan la identificación de sistemas no lineales con FNN, redes neuronales con funciones base radiales (RBFNN), redes neuronales Runge-Kutta y sistemas adaptativos de inferencia Neuro-Difusa.

Lee y Teng (2000) proponen una estructura de red neuronal difusa recurrente (RFNN) para identificar y controlar sistemas dinámicos no lineales. Esta red realiza inferencia difusa utilizando reglas difusas, se realimenta con conexiones a la segunda capa de la red neuronal. Los resultados para la RFNN se presentan después de aplicarla a diversas simulaciones.

La red neuronal adaptable con retraso es usada para identificación de sistemas no lineales en Yazdizadeh y Khorasani (2002). Cuatro arquitecturas son propuestas para identificar diferentes clases de sistemas no lineales.

Un método de identificación para modelos no lineales en forma de redes neuronales difusas es presentado por Oh *et al.* (2003). Las redes neuro-difusas combinan las reglas difusas “si - entonces” con las redes neuronales convencionales.

La identificación de sistemas no lineales por medio de redes neuronales recurrentes (RNN), de una sola capa y multicapa (ambas en tiempo discreto), es estudiada en Yu (2004). Por otro lado en Yu *et al.* (2004) se propone un esquema de control con base en la identificación propuesta por Liu *et al.* (1998), aunque este trabajo no propone nada nuevo en el ámbito de identificación muestra una aplicación de la red polinomial de Volterra.

Posteriormente, un nuevo enfoque de control para sistemas dinámicos no lineales discretos, el cual depende de la identificación de un modelo discreto de sistema por una FNN con una capa oculta es presentado por Canelon *et al.* (2005).

Kenne *et al.* (2006) presentan un nuevo esquema para estimación de parámetros de una gran clase de sistemas no lineales usando una RBFNN.

Por su parte, Mohandas y Paritala (2006) llevan a cabo la simulación de un péndulo invertido usando redes neuronales para su identificación. Utilizan una FNN para modelar el sistema multivariable, el tipo de entrenamiento es mediante retro-propagación (BP). Mediante SIMULINK se lleva a cabo la simulación.

Siguiendo con la red polinomial de Chebyshev, Purwar *et al.* (2007) logran compactar la estructura de red presentada por Lee y Jeng (1998) y un año después en Purwar *et al.* (2008) proponen un controlador, basado en esa estructura de red, para un robot manipulador.

Un método para identificación de sistemas no lineales variantes en el tiempo, basado en redes neuronales Bayesian-Gaussian (BGNN), es desarrollado por Liu y Peng (2009). Para validar el método proponen dos sistemas no lineales dinámicos variantes en el tiempo y presentan resultados en simulación.

Más reciente, Sutradhar *et al.* (2010) identifican el sistema no lineal de un péndulo invertido mediante una FNN con el método Levenberg-Marquardt. La red neuronal es entrenada usando el error entre las salidas del modelo y las salidas de la planta. Utilizan un prototipo físico del péndulo invertido del cual obtienen datos de la entrada y la salida y a partir de eso desarrollan la FNN en MATLAB, el error entre el modelo matemático y el modelo de la red neuronal está dentro de los límites aceptables, pero aun con la posibilidad de reducirse, además, no hacen ningún aporte nuevo al área de identificación ya que su trabajo es igual al de Mohandas y Paritala (2006) pero utilizando datos reales.

También, en México, Alanis *et al.* (2010) presentan un trabajo que se enfoca en la identificación de sistemas no lineales en tiempo discreto por medio de redes neuronales recurrentes de alto orden. Se incluye un análisis de estabilidad basado en el enfoque de Lyapunov para el algoritmo de aprendizaje de la RNA. Las aplicaciones del esquema que proponen se ilustran con simulaciones.

Subudhi y Jena (2011) presentan un trabajo donde muestran resultados prácticos de la efectividad de enfoques computacionales combinados con redes neuronales para la identificación de sistemas no lineales. Utilizan una FNN y prueban sus métodos en un sistema de un grado de libertad.

Por parte del CINEVESTAV del IPN de México, Cordova y Yu (2012) presentan una red neuronal que utiliza como función de activación una transformada Haar Wavelet, dada

la estructura de red se puede considerar que es una FNN, y modelan sistemas no lineales discretos en forma del modelo NARX.

Aún con todas las diversas RNAs propuestas por diferentes investigadores a lo largo de los años, las FNN siguen siendo las más aplicadas, trabajos recientes con esta estructura son: Awan *et al.* (2012), donde se utiliza una FNN para estimar la demanda de energía a futuro en una industria, y Choudhary *et al.* (2012), que utilizan una FNN para predecir la fuerza en un disipador de energía tipo BRB (*Buckling-Restrained Braces*). Ambos trabajos realizan una identificación fuera de línea debido a la estructura y pre-procesamiento de datos necesario por este tipo de red (pre-alimentada).

Hay que hacer énfasis en que la mayoría de los trabajos a los que se hace referencia en ésta sección solo prueban sus algoritmos mediante simulaciones y los pocos que utilizan datos reales no se prueban en línea.

III. DESCRIPCIÓN MATEMÁTICA DEL SISTEMA

3.1. Introducción

También llamado problema de carro y poste, se compone de una varilla delgada unida a un carro móvil. Un péndulo normal es estable cuando cuelga hacia abajo, un péndulo invertido vertical es inherentemente inestable, y debe ser balanceado activamente con el fin de permanecer en posición vertical hacia arriba, típicamente moviendo el carro horizontalmente, como parte de un sistema con retroalimentación. Es un sistema inestable que sirve para demostrar los aspectos teóricos y prácticos de la teoría de control (Chaturvedi y Röck, 2008).

El péndulo invertido simple ha sido ampliamente utilizado en los laboratorios de control para demostrar la efectividad de los sistemas de control (controles PID, redes neuronales, control difuso, algoritmos genéticos) en analogía con el control de muchos sistemas reales (Zhong y Röck, 2001). Las ecuaciones dinámicas para el PISC pueden obtenerse usando las “Ecuaciones de Lagrange” a partir de la energía cinética y potencial del sistema (Fantoni y Lozano, 2002; Tewari, 2002).

En este Capítulo se presenta el modelo matemático del sistema, es necesario describir el PISC en forma de ecuaciones ya que se requieren para realizar una simulación. Dado que el péndulo debe mantenerse cerca de su punto de operación inestable se linealiza el sistema en ese punto de operación y se desarrolla un controlador por realimentación de estados. Si el sistema no es controlado la posición del péndulo tenderá a dirigirse hacia abajo (punto de operación estable), por consecuencia el control es necesario para mantenerlo cerca del punto de operación inestable, más aún, una vez controlado se deben aplicar perturbaciones acotadas para cumplir la condición de excitación persistente (Åström y Eykhoff, 1971) necesaria para que los pesos de una red neuronal converjan a los pesos ideales, es decir, para que la identificación converja a la señal real.

Una observación importante es que para el experimento real al momento de la adquisición de datos la señal de entrada al sistema es voltaje, para que concuerde con la simulación hay que transformarla a términos de fuerza, por lo cual se presenta también el modelo del sistema electro-mecánico de tracción, donde se obtiene una relación voltaje - fuerza para el carro del PISC.

3.2. Modelo matemático del sistema

3.2.1 Modelado por segunda ley de Newton

Considere el péndulo invertido de la Figura 3.1. El pivote del péndulo está montado en un carro que se puede mover en dirección horizontal (Yoshida, 1999; Muskinja y Tovornik, 2006). Se considera que la masa del péndulo está distribuida uniformemente a lo largo del mismo por lo cual tiene un centro de masa, el cual se asume que es el centro geométrico, e inercia alrededor de este punto.

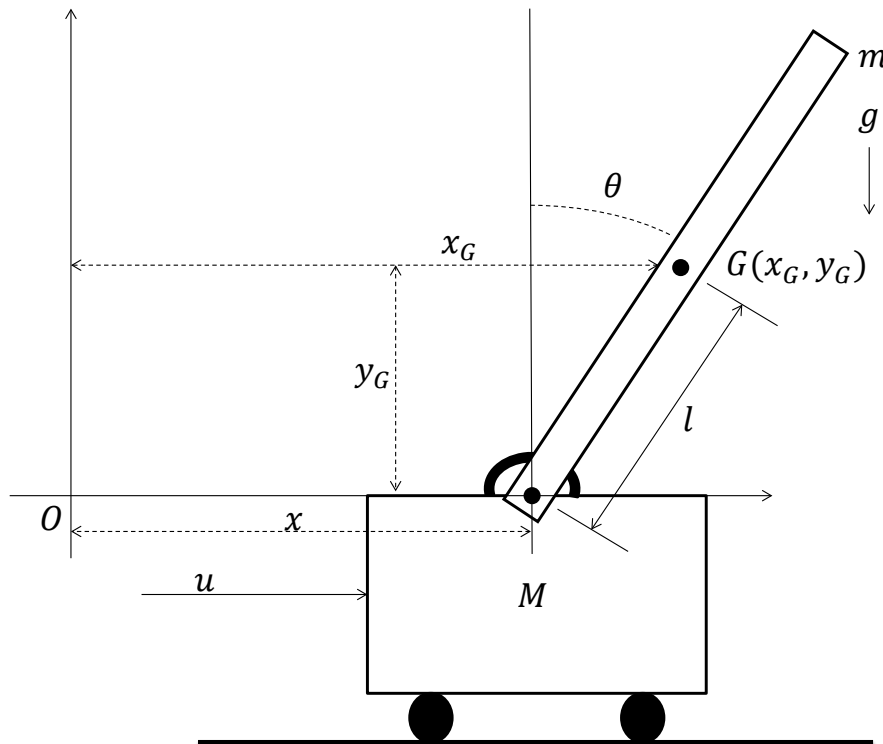


Figura 3.1: Péndulo invertido sobre un carro.

La nomenclatura para la Figura 3.1 es:

- M y m , masa del carro y masa del péndulo, respectivamente;
- G , centro de masa del péndulo;
- I , inercia del péndulo alrededor de su centro de masa;
- l , distancia del pivote al centro de masa del péndulo;
- $x(t)$, posición del carro (se mide desde el origen hasta la posición del pivote en el carro);
- x_G , coordenada del centro de masa del péndulo en el eje horizontal;
- y_G , coordenada del centro de masa del péndulo en el eje vertical;
- $\theta(t)$, posición angular del péndulo (definido desde la línea vertical);
- g , fuerza de gravedad;
- $u(t)$, fuerza aplicada al carro.

El péndulo invertido es inestable de tal manera que aun cuando su posición inicial este en una vecindad del punto de operación éste caerá a menos que una fuerza de control adecuada se aplique (Loría y Panteley, 2006). Por simplicidad en las ecuaciones, en el resto de esta sección $x = x(t)$, $\theta = \theta(t)$ y $u = u(t)$; por otra parte, las derivadas temporales se expresarán: $da/dt = \dot{a}$, $d^2a/dt^2 = \ddot{a}$, y así sucesivamente. Se definen las coordenadas del centro de masa del pendulo (x_G, y_G) (Ogata, 2002; Fantoni y Lozano, 2002):

$$x_G = x + l \sin \theta \quad (3.1)$$

$$y_G = l \cos \theta \quad (3.2)$$

$$\dot{x}_G = \dot{x} + \dot{\theta} l \cos \theta \quad (3.3)$$

$$\dot{y}_G = -\dot{\theta} l \sin \theta \quad (3.4)$$

Para obtener el modelo matemático del péndulo se utiliza el enfoque de energía. Primero se obtiene la energía cinética del mecanismo completo, la cual está dada como:

$$K = k_{pe} + k_{ca} \quad (3.5)$$

donde k_{pe} es la energía cinética del péndulo y k_{ca} es la energía cinética del carro. La k_{pe} se compone de una parte traslacional y otra rotacional, de manera que:

$$k_{pe} = \frac{1}{2}m (\dot{x}_G^2 + \dot{y}_G^2) + \frac{1}{2}I\dot{\theta}^2 \quad (3.6)$$

y la k_{ca} se compone únicamente de una parte traslacional:

$$k_{ca} = \frac{1}{2}M\dot{x}^2 \quad (3.7)$$

de manera que sustituyendo (3.3) y (3.4) en (3.6) y después sustituyendo (3.6) y (3.7) en (3.5) se obtiene:

$$K = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m \left[(\dot{x} + \dot{\theta}l \cos \theta)^2 + (-\dot{\theta}l \sin \theta)^2 \right] + \frac{1}{2}I\dot{\theta}^2 \quad (3.8)$$

expandiendo los términos cuadráticos de (3.8):

$$K = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}m \left[\dot{x}^2 + 2\dot{x}\dot{\theta}l \cos \theta + \dot{\theta}^2l^2 \cos^2 \theta + \dot{\theta}^2l^2 \sin^2 \theta \right] \quad (3.9)$$

simplificando (3.9) y aplicando identidades trigonométricas se tiene que:

$$K = \frac{1}{2}(M + m)\dot{x}^2 + m\dot{x}\dot{\theta}l \cos \theta + \frac{1}{2}(ml^2 + I)\dot{\theta}^2 \quad (3.10)$$

Ahora, como el carro se mueve en un plano horizontal se considera que su energía potencial es cero (Fantoni y Lozano, 2002), por lo que la energía potencial del mecanismo completo es igual a la energía potencial del péndulo. En la Figura 3.2 se pueden observar los vectores \bar{F} y \bar{d} , y de la definición básica de energía se sabe que la energía potencial está dada por:

$$P = \bar{F} \cdot \bar{d} \quad (3.11)$$

De la Figura 3.2 se sabe que G es el centro de masa, θ la posición angular respecto a la vertical, \bar{F} es el vector de fuerza (peso del péndulo) desde el centro de masa, \bar{d} es el vector

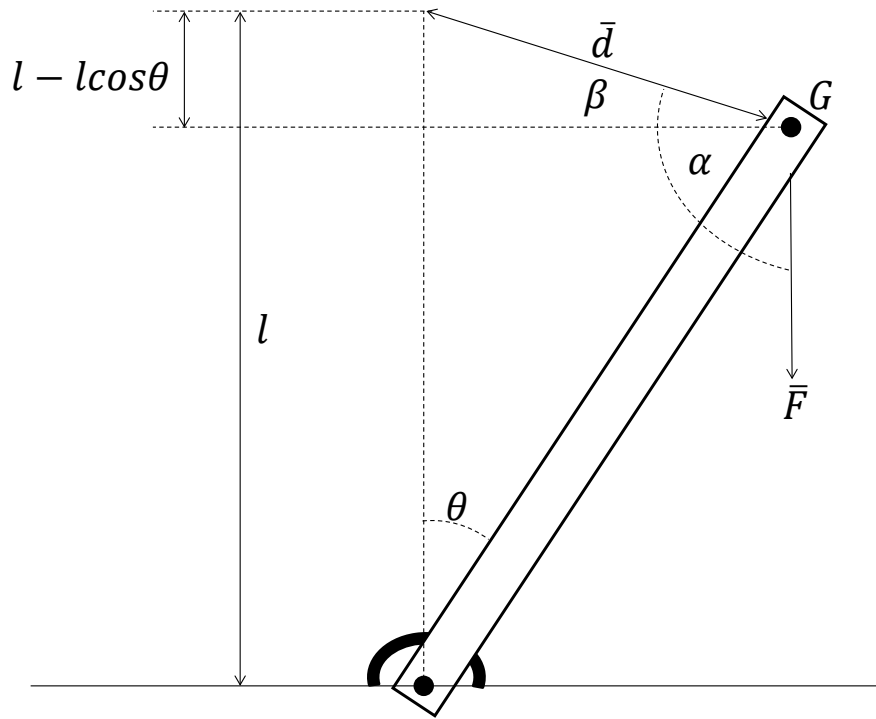


Figura 3.2: Péndulo simple.

distancia desde la posición actual (del centro de masa) del péndulo hasta su posición invertida, α es el ángulo desde el vector \vec{F} hasta el vector \vec{d} y por último β es el ángulo desde la horizontal del centro de masa hasta el vector \vec{d} .

Se define:

$$\vec{F} = mg \quad (3.12)$$

sustituyendo en (3.11) se tiene:

$$P = \vec{F} \cdot \vec{d} = mg |\vec{d}| \cos \alpha \quad (3.13)$$

haciendo una sustitución trigonométrica se tiene:

$$P = \vec{F} \cdot \vec{d} = mg (-|\vec{d}| \sin \beta) \quad (3.14)$$

finalmente se tiene:

$$P = -mg(l - l \cos \theta) = mgl(\cos \theta - 1) \quad (3.15)$$

se define el Lagrangiano del siguiente modo:

$$L = K - P = \frac{1}{2} (M + m) \dot{x}^2 + m \dot{x} \dot{\theta} l \cos \theta + \frac{1}{2} (ml^2 + I) \dot{\theta}^2 - mgl (\cos \theta - 1) \quad (3.16)$$

Para obtener el modelo matemático se debe sustituir el Lagrangiano en las ecuaciones de Euler-Lagrange (Wells, 1972; Kelly *et al.*, 2005) como se muestra a continuación:

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = u \quad (3.17)$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0 \quad (3.18)$$

entonces:

$$\frac{\partial L}{\partial \dot{x}} = (M + m) \dot{x} + ml \dot{\theta} \cos \theta \quad (3.19)$$

$$\frac{\partial L}{\partial x} = 0 \quad (3.20)$$

$$\frac{\partial L}{\partial \dot{\theta}} = ml \dot{x} \cos \theta + (ml^2 + I) \dot{\theta} \quad (3.21)$$

$$\frac{\partial L}{\partial \theta} = mgl \sin \theta - ml \dot{x} \sin \theta \quad (3.22)$$

y por lo tanto:

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} = (M + m) \ddot{x} + ml \ddot{\theta} \cos \theta - ml \dot{\theta}^2 \sin \theta \quad (3.23)$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} = ml \ddot{x} \cos \theta - ml \dot{x} \dot{\theta} \sin \theta + (ml^2 + I) \ddot{\theta} \quad (3.24)$$

Por último, sustituyendo (3.20), (3.22), (3.23) y (3.24) en (3.17) y (3.18) se obtienen las ecuaciones diferenciales que rigen el sistema:

$$(M + m) \ddot{x} + ml \ddot{\theta} \cos \theta - ml \dot{\theta}^2 \sin \theta = u \quad (3.25)$$

$$ml \ddot{x} \cos \theta + (ml^2 + I) \ddot{\theta} - mgl \sin \theta = 0 \quad (3.26)$$

Suponiendo que $ml^2 \gg I$, es decir, que I es despreciable, y simplificando se obtiene (Tewari, 2002):

$$(M + m) \ddot{x} + ml \ddot{\theta} \cos \theta - ml \dot{\theta}^2 \sin \theta = u \quad (3.27)$$

$$m \ddot{x} \cos \theta + ml \ddot{\theta} - mg \sin \theta = 0 \quad (3.28)$$

3.2.2 Ecuación de estado no lineal del PISC

De (3.28) tenemos:

$$\ddot{\theta} = \frac{1}{ml} (mg \sin \theta - m\ddot{x} \cos \theta) \quad (3.29)$$

sustituyendo (3.29) en (3.27) se tiene:

$$(M + m)\ddot{x} + (mg \sin \theta - m\ddot{x} \cos \theta) \cos \theta - ml\dot{\theta}^2 \sin \theta = u \quad (3.30)$$

se expande la ecuación y se aplican identidades trigonométricas:

$$(M + m)\ddot{x} + mg \sin \theta \cos \theta + m\ddot{x} \sin^2 \theta - m\ddot{x} - ml\dot{\theta}^2 \sin \theta = u \quad (3.31)$$

agrupando y eliminando términos se tiene:

$$(M + m \sin^2 \theta)\ddot{x} + mg \sin \theta \cos \theta - ml\dot{\theta}^2 \sin \theta = u \quad (3.32)$$

finalmente, despejando:

$$\ddot{x} = \frac{1}{M + m \sin^2 \theta} (u + ml\dot{\theta}^2 \sin \theta - mg \sin \theta \cos \theta) \quad (3.33)$$

Después, se sustituye (3.33) en (3.29) para dejar la última en términos de θ :

$$\ddot{\theta} = \frac{1}{ml} \left[mg \sin \theta - \frac{m \cos \theta}{M + m \sin^2 \theta} (u + ml\dot{\theta}^2 \sin \theta - mg \sin \theta \cos \theta) \right] \quad (3.34)$$

reacomodando y simplificando se tiene:

$$\ddot{\theta} = \frac{mg \sin \theta \cos^2 \theta - ml\dot{\theta}^2 \sin \theta \cos \theta - u \cos \theta + Mg \sin \theta + mg \sin^3 \theta}{(M + m \sin^2 \theta) l} \quad (3.35)$$

aplicando identidades trigonométricas se reduce a:

$$\ddot{\theta} = \frac{(M + m)g \sin \theta - ml\dot{\theta}^2 \sin \theta \cos \theta - u \cos \theta}{(M + m \sin^2 \theta) l} \quad (3.36)$$

Se define un vector de estados (Tewari, 2002):

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta \\ x \\ \dot{\theta} \\ \dot{x} \end{bmatrix} \in \mathbb{R}^4 \quad (3.37)$$

Utilizando (3.33) y (3.36) es posible escribir la ecuación de estado del PISC (Coron, 2010) como:

$$\dot{\mathbf{x}} = f(\mathbf{x}, u) = \frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ \frac{(M+m)g \sin x_1 - mlx_3^2 \sin x_1 \cos x_1 - u \cos x_1}{(M+m \sin^2 x_1)l} \\ \frac{-mg \sin x_1 \cos x_1 + mlx_3^2 \sin x_1 + u}{M+m \sin^2 x_1} \end{bmatrix} \quad (3.38)$$

3.2.3 Aproximación lineal

Aunque aproximar una ecuación de estado no lineal a una ecuación de estado lineal es un método muy útil, se debe mencionar que su principal desventaja es que los resultados que se obtienen sólo son válidos en una región restringida del espacio de trabajo del proceso que se desea controlar (Ogata, 2002). En el caso del péndulo en su posición invertida, esto significa que aunque si se permite que el péndulo se mueva, estos movimientos no deben ser tales que la configuración del péndulo sufra cambios demasiado grandes respecto a su configuración de equilibrio.

Considere la ecuación diferencial (Coron, 2010):

$$\dot{x} = f(x, u) \quad (3.39)$$

donde:

- $x \in \mathbb{R}^n$ es un vector de n dimensiones que representa los estados del sistema;
- $u \in \mathbb{R}^p$ es un vector de p dimensiones que representa la entrada de la ecuación diferencial.

Definición 3.2.1 *Un punto de operación es aquella pareja (x^*, u^*) tal que $f(x^*, u^*) = 0$. Esto significa que la solución de la ecuación diferencial puede permanecer en “reposo” en el valor constante x^* , porque $\dot{x}^* = f(x^*, u^*) = 0$, si se aplican las entradas adecuadas u^* (p entradas) que también resultan ser constantes.*

Como (3.38) está dada de la forma de (3.39), siguiendo la Definición 3.2.1, los puntos de operación de (3.38) se obtienen al hacer:

$$\dot{\mathbf{x}}^* = f(\mathbf{x}^*, u^*) = \begin{bmatrix} x_3^* \\ x_4^* \\ \frac{(M+m)g \sin x_1^* - mlx_3^{*2} \sin x_1^* \cos x_1^* - u^* \cos x_1^*}{(M+m \sin^2 x_1^*)l} \\ \frac{-mg \sin x_1^* \cos x_1^* + mlx_3^{*2} \sin x_1^* + u^*}{M+m \sin^2 x_1^*} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.40)$$

Del primer y segundo renglón de (3.40) se sabe que:

$$x_3^* = \dot{\theta}^* = 0 \quad (3.41)$$

$$x_4^* = \dot{x}^* = 0 \quad (3.42)$$

Del tercer y cuarto renglón de (3.40) se tiene que:

$$(M + m)g \sin x_1^* - mlx_3^{*2} \sin x_1^* \cos x_1^* - u^* \cos x_1^* = 0 \quad (3.43)$$

$$u^* = mg \sin x_1^* \cos x_1^* - mlx_3^{*2} \sin x_1^* \quad (3.44)$$

sustituyendo (3.44) en (3.43):

$$(M + m)g \sin x_1^* - mg \sin x_1^* \cos^2 x_1^* = 0 \quad (3.45)$$

$$\sin x_1^* (Mg + mg - mg \cos^2 x_1^*) = 0 \quad (3.46)$$

$$\sin x_1^* = 0 \quad (3.47)$$

por lo tanto

$$x_1^* = \theta^* = n\pi, n \in \mathbb{Z} \quad (3.48)$$

después, sustituyendo (3.48) en (3.44):

$$u^* = 0 \quad (3.49)$$

Finalmente como x_2^* no afecta directamente en (3.40) se observa que puede tomar cualquier valor, es decir:

$$x_2^* = x^* = c \in \mathbb{R} \quad (3.50)$$

Ahora, teniendo los puntos de operación donde se desea trabajar como:

$$\mathbf{x}^* = \begin{bmatrix} x_1^* \\ x_2^* \\ x_3^* \\ x_4^* \end{bmatrix} = \begin{bmatrix} \theta^* \\ x^* \\ \dot{\theta}^* \\ \dot{x}^* \end{bmatrix} = \begin{bmatrix} n\pi \\ c \\ 0 \\ 0 \end{bmatrix}, u^* = 0 \quad (3.51)$$

La aproximación lineal del sistema (3.38) en el punto de operación (3.51) está dada como (Ogata, 2002; Tewari, 2002):

$$\dot{z} = Az + Bw \quad (3.52)$$

con:

$$A = \left. \frac{\partial f(\mathbf{x}, u)}{\partial \mathbf{x}} \right|_{\substack{\mathbf{x}^* \\ u^*}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix} \bigg|_{\substack{\mathbf{x}^* \\ u^*}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{(M+m)g}{Ml} & 0 & 0 & 0 \\ \frac{-mg}{M} & 0 & 0 & 0 \end{bmatrix}, \quad (3.53)$$

$$B = \left. \frac{\partial f(\mathbf{x}, u)}{\partial u} \right|_{\substack{\mathbf{x}^* \\ u^*}} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \\ \frac{\partial f_3}{\partial u} \\ \frac{\partial f_4}{\partial u} \end{bmatrix} \bigg|_{\substack{\mathbf{x}^* \\ u^*}} = \begin{bmatrix} 0 \\ 0 \\ \frac{-1}{Ml} \\ \frac{1}{M} \end{bmatrix} \quad (3.54)$$

donde se ha definido:

$$z = \mathbf{x} - \mathbf{x}^* \quad (3.55)$$

$$w = u - u^* \quad (3.56)$$

La ecuación de salida del sistema esta dada entonces como (Ogata, 2002; Tewari, 2002):

$$y = Cz + Dw \quad (3.57)$$

donde $C \in \mathbb{R}^{1 \times 4}$ y $D = [0]$.

Se verifica que el sistema sea controlable con la matriz de controlabilidad (Tewari, 2002; Coron, 2010), $P_C = [B \ AB \ A^2B \ A^3B] \in \mathbb{R}^{n \times n}$. Su determinante es:

$$\det(P_C) = -\frac{g^2}{M^4 l^4} \neq 0 \quad (3.58)$$

por lo tanto el par (A, B) si es controlable.

3.2.4 Controlador por retroalimentación completa del estado

Si un sistema es completamente controlable, sus valores característicos pueden ser asignados de forma arbitraria a través de la realimentación de estado (Kuo, 1996). La realimentación completa del estado se refiere a un controlador que genere un vector de entradas, u , de acuerdo a una ley de control (Tewari, 2002).

Se define un controlador:

$$u = -K\mathbf{x} \quad (3.59)$$

el cual se muestra en la Figura 3.3, y una matriz de lazo cerrado:

$$A_{CL} = A - BK \quad (3.60)$$

donde:

- \mathbf{x} es el vector de estado de la planta y pertenece a $\mathbb{R}^{n \times 1}$,
- u es el control escalar,
- K es la matriz de realimentación con elementos de ganancia constantes y pertenece a $\mathbb{R}^{1 \times n}$,
- A y B son las matrices de coeficientes de la ecuación de estado.

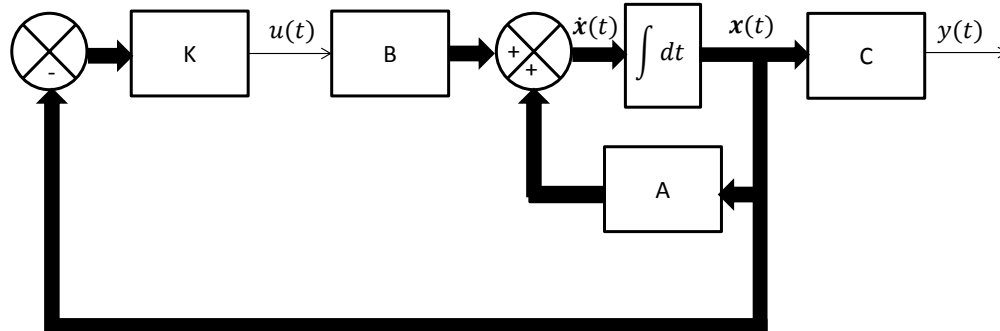


Figura 3.3: Diagrama esquemático de una realimentación de estado.

Debido a lo obtenido en (3.58) es posible asignar de manera arbitraria los valores propios de (3.60), los cuales corresponden a los polos en lazo cerrado (Tewari, 2002), a través del controlador propuesto en (3.59).

3.3. Modelado del sistema de movimiento lineal

En este trabajo, el péndulo invertido utiliza un servomotor de C.D. que está acoplado a la llanta del carro con una banda de plástico, para mover el carro con la fuerza $u = u(t)$, como se muestra en la Figura 3.4.

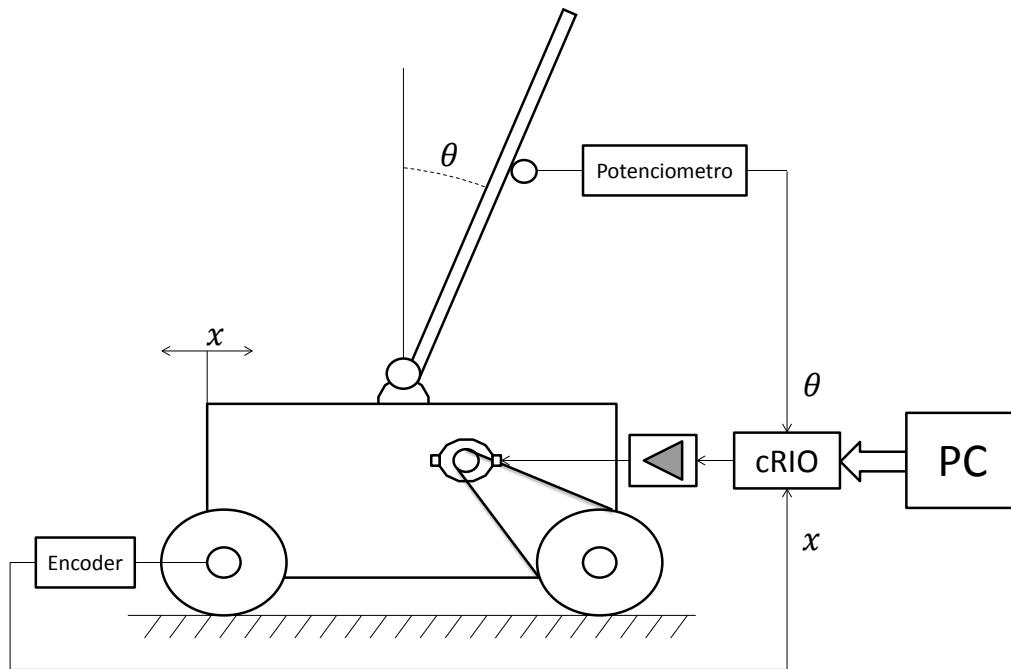


Figura 3.4: Diagrama a bloques del prototipo físico del PISC.

3.3.1 Modelado del servomotor de C.D.

El motor es controlado variando el voltaje de armadura $V_{in} = V_{in}(t)$. El torque generado por el motor es proporcional a la corriente de armadura $i_a = i_a(t)$. Entonces:

$$T_m = K_i i_a \quad (3.61)$$

donde K_i es la constante de torque. Las ecuaciones eléctricas del motor son:

$$V_{in} - e_b = L_a \dot{i}_a + R_a i_a \quad (3.62)$$

$$e_b = K_b \dot{\theta}_m = K_b \omega_m \quad (3.63)$$

La ecuación de torque para la posición de la flecha del motor θ_m es

$$T_m - T_L = J_m \ddot{\theta}_m + B_m \dot{\theta}_m \quad (3.64)$$

donde, $T_m = T_m(t)$ es el torque del motor, $T_L = T_L(t)$ es el torque utilizado para generar la fuerza $u(t)$ para mover el carro, K_b es la constante de fuerza contra electromotriz, J_m es el momento de inercia del rotor con carga, B_m es el coeficiente de amortiguación, L_a es la inductancia de armadura, R_a es la resistencia de armadura, y finalmente $\theta_m = \theta_m(t)$ y $\omega_m = \omega_m(t)$ son la posición y velocidad (respectivamente) de la flecha del motor.

3.3.2 Modelado completo del sistema de movimiento

Las ecuaciones que describen el comportamiento de el sistema combinado que consiste en el péndulo invertido y el motor de C.D. se han obtenido convirtiendo las variables rotacionales de la flecha del motor (θ_m, T_L) a variables traslacionales del carro (x, u) (Sutradhar *et al.*, 2010):

$$T_L = ru, \quad \theta_m = \frac{x}{nr} \quad (3.65)$$

Para el sistema acoplado, r es el radio de la llanta y n es la relación de la transmisión. Por lo tanto, a partir de la ecuaciones (3.64) y (3.65), la ecuacion para el torque del sistema combinado es:

$$T_m = J_m \frac{\ddot{x}}{nr} + B_m \frac{\dot{x}}{nr} + nru \quad (3.66)$$

Usando las ecuaciones (3.61), (3.62) y (3.63), y asumiendo inductancia de armadura cero,

$$T_m = K_i \left[\frac{V_{in} - K_b \dot{\theta}_m}{R_a} \right] = K_i \left[\frac{V_{in} - K_b \left(\frac{\dot{x}}{nr} \right)}{R_a} \right] \quad (3.67)$$

Ahora las ecuaciones (3.27), (3.28), (3.66) y (3.67) describen el modelo no lineal del sistema del péndulo invertido con un motor de C.D. A partir de las ecuaciones (3.66) y (3.67) se despeja la fuerza aplicada u como:

$$u = \frac{K_i V_{in} - R_a J_m \left(\frac{\ddot{x}}{nr} \right) - K_i K_b \left(\frac{\dot{x}}{nr} + \frac{x}{nr} \right)}{nr} \quad (3.68)$$

IV. IDENTIFICACIÓN CON RED NEURONAL MULTICAPA

4.1. Introducción

En este Capítulo se presenta la fundamentación teórica de redes neuronales prealimentadas con estructura perceptrón multicapa. Se da un marco general del funcionamiento de las redes neuronales multicapa y se presentan las ecuaciones para modelado. Se proponen las funciones de activación para cada capa. El algoritmo de entrenamiento de la red es el de retro-propagación.

Una red neuronal se compone de unidades llamadas neuronas. Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida, Figura 4.1.

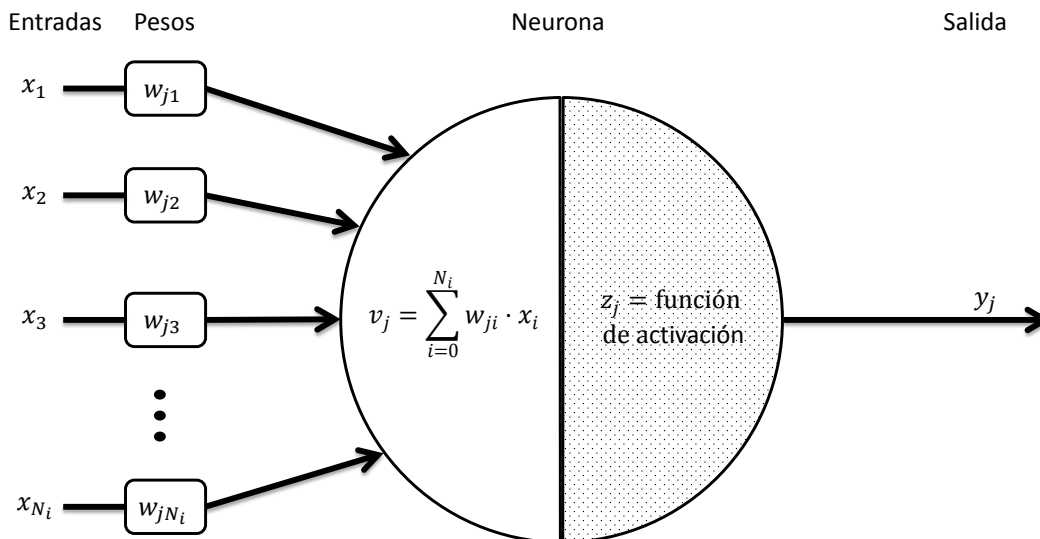


Figura 4.1: Esquema de una neurona.

La identificación basada en las RNAs parte de un conjunto de datos de entrada sufi-

cientemente significativo y el objetivo es conseguir que la red aprenda automáticamente las propiedades deseadas. En este sentido, el diseño de la red tiene menos que ver con cuestiones como los flujos de datos y la detección de condiciones, y más que ver con cuestiones tales como la selección del modelo de red, las variables a incorporar y el preprocesamiento de la información que formará el conjunto de entrenamiento. Asimismo, el proceso por el que los parámetros de la red se adecuan a la resolución de cada problema no se denomina genéricamente programación sino que se suele denominar entrenamiento neuronal.

El perceptrón multicapa (MLP) es por mucho la red neuronal más bien conocida y más popular de entre todos los paradigmas de redes neuronales existentes (Manry *et al.*, 2001).

El MLP es una variante del modelo original propuesto por Rosenblatt (1958). Un modelo de red neuronal MLP consiste en una red pre-alimentada (FNN) con capas. Cada neurona en un MLP tiene una función de activación no lineal que es, comúnmente, continuamente diferenciable. Se ha demostrado que un MLP con suficientes unidades no lineales en una sola capa oculta puede aproximar cualquier función no lineal (Hornik *et al.*, 1989). Un ejemplo de un MLP con una sola capa oculta se muestra en la Figura 4.2. Cada círculo representa una neurona individual, a cada neurona en la capa oculta se le llama unidad oculta.

4.2. Modelado con red pre-alimentada

Una red multicapa pre-alimentada (*feedforward*) consiste en unidades acomodadas en capas que solo tienen conexiones de avance hacia unidades en capas subsecuentes. La estructura neuronal de este tipo de red se muestra en la Figura 4.3, donde $\vec{x} = (x_1, \dots, x_i, \dots, x_{N_i})^T \in \mathbb{R}^{N_i \times 1}$ es el vector de entradas, $\vec{t} = (t_1, \dots, t_k, \dots, t_{N_o})^T \in \mathbb{R}^{N_o \times 1}$ es el vector de salidas deseadas, $\vec{y} = (y_1, \dots, y_k, \dots, y_{N_o})^T \in \mathbb{R}^{N_o \times 1}$ es el vector de salidas de la red neuronal, N_i es el número total de entradas a la red neuronal, N_h es el número total de neuronas en la capa oculta, N_o es el número total de salidas de la red neuronal, N_s es el número total de muestras de entrenamiento. De acuerdo con la Figura 4.3 para

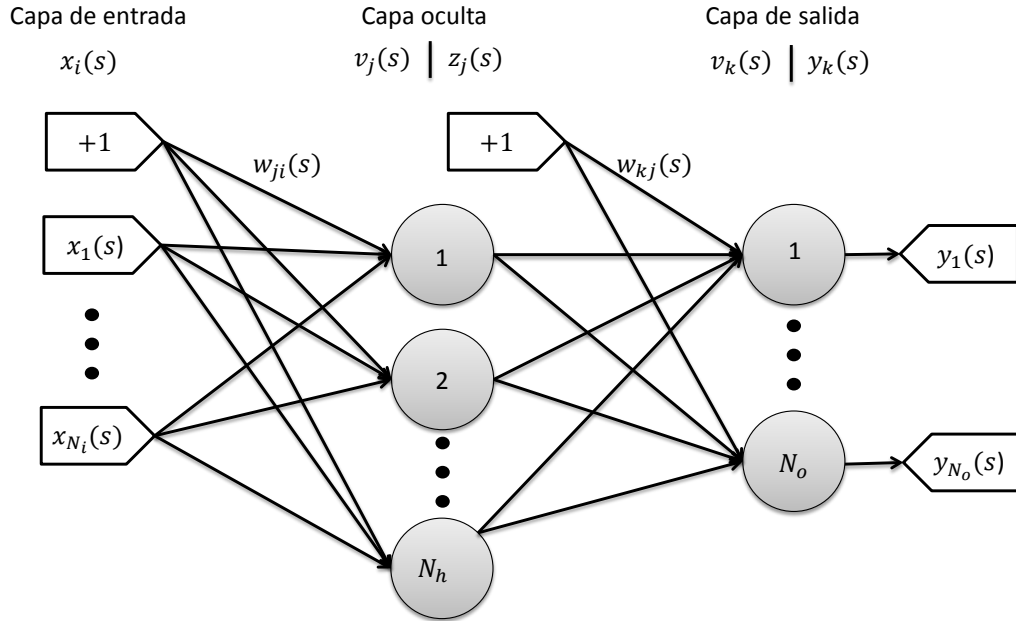


Figura 4.2: Arquitectura de un perceptrón multicapa con una capa oculta.

la j -ésima unidad oculta, el campo inducido local $v_j(s)$ y la salida de activación $z_j(s)$ son:

$$v_j(s) = \sum_{i=0}^{N_i} w_{ji}(s) x_i(s), \quad 1 \leq s \leq N_s \quad (4.1)$$

$$z_j(s) = \varphi_j(v_j(s)) \quad (4.2)$$

donde $w_{ji}(s)$ denota el peso de la conexión de la i -ésima unidad de entrada a la j -ésima unidad oculta, x_0 es $+1$ y w_{j0} es el término de umbral (bias). Para la k -ésima unidad de salida, el campo inducido local $v_k(s)$ y la salida de la red $y_k(s)$ son:

$$v_k(s) = \sum_{j=0}^{N_h} w_{kj}(s) z_j(s), \quad 1 \leq s \leq N_s \quad (4.3)$$

$$y_k(s) = \varphi_k(v_k(s)) \quad (4.4)$$

donde $w_{kj}(s)$ denota el peso de la conexión de la j -ésima unidad oculta a la k -ésima unidad de salida, z_0 es $+1$ y w_{k0} es el término de umbral (bias). Para redes MLP, las funciones de activación $\varphi_j(\cdot)$ y $\varphi_k(\cdot)$ utilizadas son las siguientes:

$$\varphi_j(v_j(s)) = \frac{2}{(1 + e^{-2v_j(s)})} - 1 \quad (4.5)$$

$$\varphi_k(v_k(s)) = v_k(s) \quad (4.6)$$

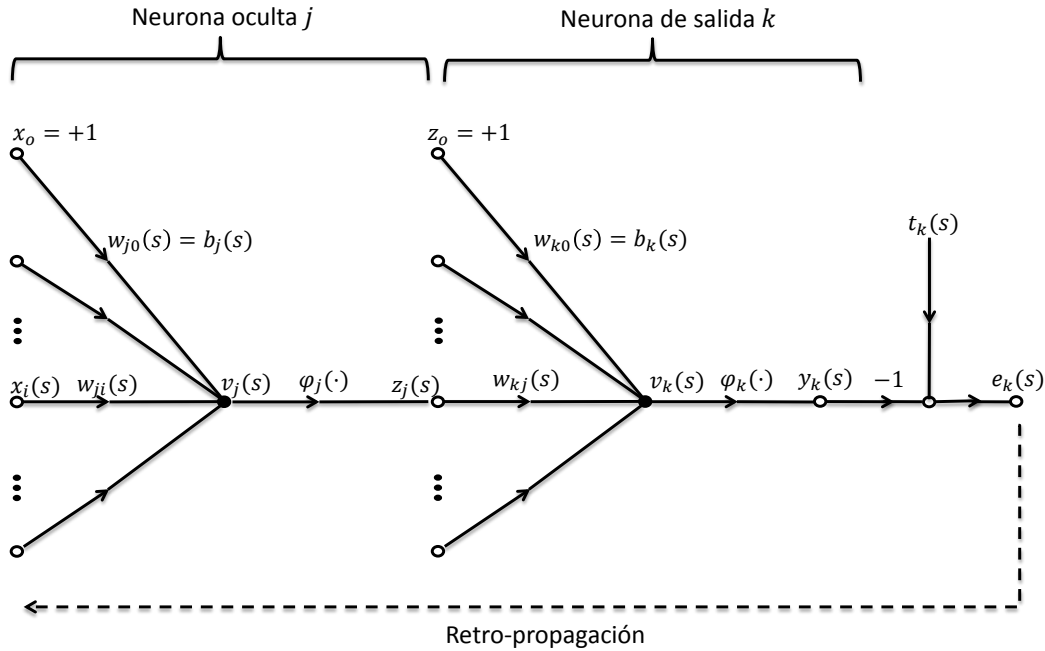


Figura 4.3: Gráfico de flujo de señal resaltando los detalles de la neurona oculta j conectada a la neurona de salida k .

Cita 4.2.1 “...Cada neurona en la capa oculta debe incluir una función de activación no lineal. Es importante enfatizar que la no linealidad debe ser suave, es decir, continuamente diferenciable...(Haykin, 1999)”

En términos básicos, la diferenciabilidad es el único requisito que una función de activación debe cumplir. Dado lo anterior, la función tangente hiperbólica en (4.5), Figura 4.4, y la función lineal en (4.6) cumplen para ser utilizadas como funciones de activación para la capa oculta y la capa de salida, respectivamente. Además una propiedad importante de (4.5) es que normaliza las entradas a la red neuronal entre -1 y +1, mientras que (4.6) deja a la salida en su rango completo para poder ser comparada correctamente con la salida deseada.

A fin de entrenar la red neuronal, la suma instantánea de errores cuadráticos para la capa de salida está definido como:

$$E(s) = \frac{1}{2} \sum_{k=1}^{N_o} e_k^2(s) \tag{4.7}$$

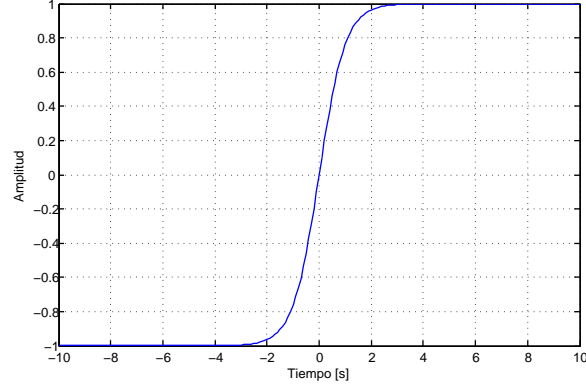


Figura 4.4: Gráfica de una tangente hiperbólica

donde $e_k(s) = t_k(s) - y_k(s)$ es el error en la k -ésima unidad de salida. El desempeño total de la red neuronal MLP, medido como un error de media cuadrática (MSE), puede ser escrito como:

$$E_{av} = \frac{1}{2N_s} \sum_{s=1}^{N_s} \sum_{k=1}^{N_o} e_k^2(s) \quad (4.8)$$

El objetivo es ajustar los pesos $w_{ji}(s)$ y $w_{kj}(s)$ para minimizar el error E_{av} . Esto lleva a un problema de optimización de mínimos cuadrados. Básicamente la actualización de pesos se realiza mediante:

$$w_{kj}(s+1) = w_{kj}(s) + \Delta w_{kj}(s) \quad (4.9)$$

$$w_{ji}(s+1) = w_{ji}(s) + \Delta w_{ji}(s) \quad (4.10)$$

El algoritmo de retro-propagación aplica una corrección $\Delta w_{kj}(s)$ y $\Delta w_{ji}(s)$ a los pesos sinápticos $w_{kj}(s)$ y $w_{ji}(s)$, respectivamente. Esta corrección es proporcional a la derivada parcial del error cuadrático $E(s)$ respecto a los pesos, utilizando una relación de aprendizaje η .

Comentario 4.2.1 *El parámetro de relación de aprendizaje η sirve para modificar la amplitud del cambio en la estimación de los pesos sinápticos mediante el algoritmo de retro-propagación. Según Haykin (1999) a menos valor de η menores los cambios en los pesos produciendo una trayectoria suave pero lenta. Por el contrario, un parámetro η muy grande además de acelerar la convergencia también puede ocasionar que la red se vuelva inestable, por ejemplo, oscilatoria.*

De manera que:

$$\Delta w_{kj}(s) = -\eta \frac{\partial E(s)}{\partial w_{kj}(s)} = \eta \delta_k(s) z_j(s) \quad (4.11)$$

$$\Delta w_{ji}(s) = -\eta \frac{\partial E(s)}{\partial w_{ji}(s)} = \eta \delta_j(s) x_i(s) \quad (4.12)$$

donde:

$$\delta_k(s) = e_k(s) \varphi'_k(v_k(s)) = e_k(s) \quad (4.13)$$

$$\delta_j(s) = \varphi'_j(v_j(s)) \sum_{k=1}^{N_o} \delta_k(s) w_{kj}(s) = \varphi_j(v_j(s)) [1 - \varphi_j(v_j(s))] \sum_{k=1}^{N_o} \delta_k(s) w_{kj}(s) \quad (4.14)$$

la diferenciación con respecto al argumento de $\varphi(\cdot)$ se denota como $\varphi'(\cdot)$. Dado que (4.6) denota una relación lineal, para preservar la salida estimada sin normalizar, su derivada (4.13) es igual a 1.

De forma que se pueden reescribir las ecuaciones (4.9) y (4.10) como:

$$w_{kj}(s+1) = w_{kj}(s) + \eta \delta_k(s) z_j(s) \quad (4.15)$$

$$w_{ji}(s+1) = w_{ji}(s) + \eta \delta_j(s) x_i(s) \quad (4.16)$$

El proceso se repite presentando a la red nuevas épocas de muestras de entrenamiento hasta que se cumpla un criterio para detener el entrenamiento. En este trabajo el criterio para detener el entrenamiento de la red es el número de épocas. El orden de presentación para las muestras de entrenamiento debe ser aleatorio de época en época.

De acuerdo con Dreyfus *et al.* (2005):

Propiedad 4.2.1 *Cualquier función acotada, suficientemente regular, puede ser aproximada uniformemente con precisión arbitraria en una región finita de espacio variable, por una red neuronal con una sola capa de neuronas ocultas con la misma función de activación, y una neurona de salida lineal.*

Entonces, como lo menciona la propiedad 4.2.1, cualquier función no lineal puede ser estimada mediante una red neuronal más un error de aproximación, el error dependerá de la precisión que se desee. Sin embargo, como lo mencionan Haykin (1999) y Dreyfus *et al.* (2005), no hay manera de saber cuántas neuronas en la capa oculta son las necesarias para

alcanzar la precisión deseada. Por otra parte, según lo mencionado en el capítulo 4 de Haykin (1999) el aprendizaje con retro-propagación es una aplicación del método estadístico llamado aproximación estocástica, consecuentemente tiende a converger pero lentamente. Utilizando este método se corre el riesgo de quedar atrapado en un mínimo local sin llegar a estar cerca siquiera del mínimo global. En el capítulo 8 del libro de Rumelhart y McClelland (1987), los autores afirman que es raro quedar atrapado en un mínimo local en un problema práctico. También, debido a lo establecido en el comentario 4.2.1, no hay garantía de que la aproximación de red neuronal utilizando retro-propagación converja a los valores reales en tiempo finito. Para más información sobre la derivación de las reglas de aprendizaje ver el capítulo 6 de Fausett (1994) y el capítulo 4 de Haykin (1999). El proceso de identificación mediante MLPs se ilustra en la Figura 4.5.

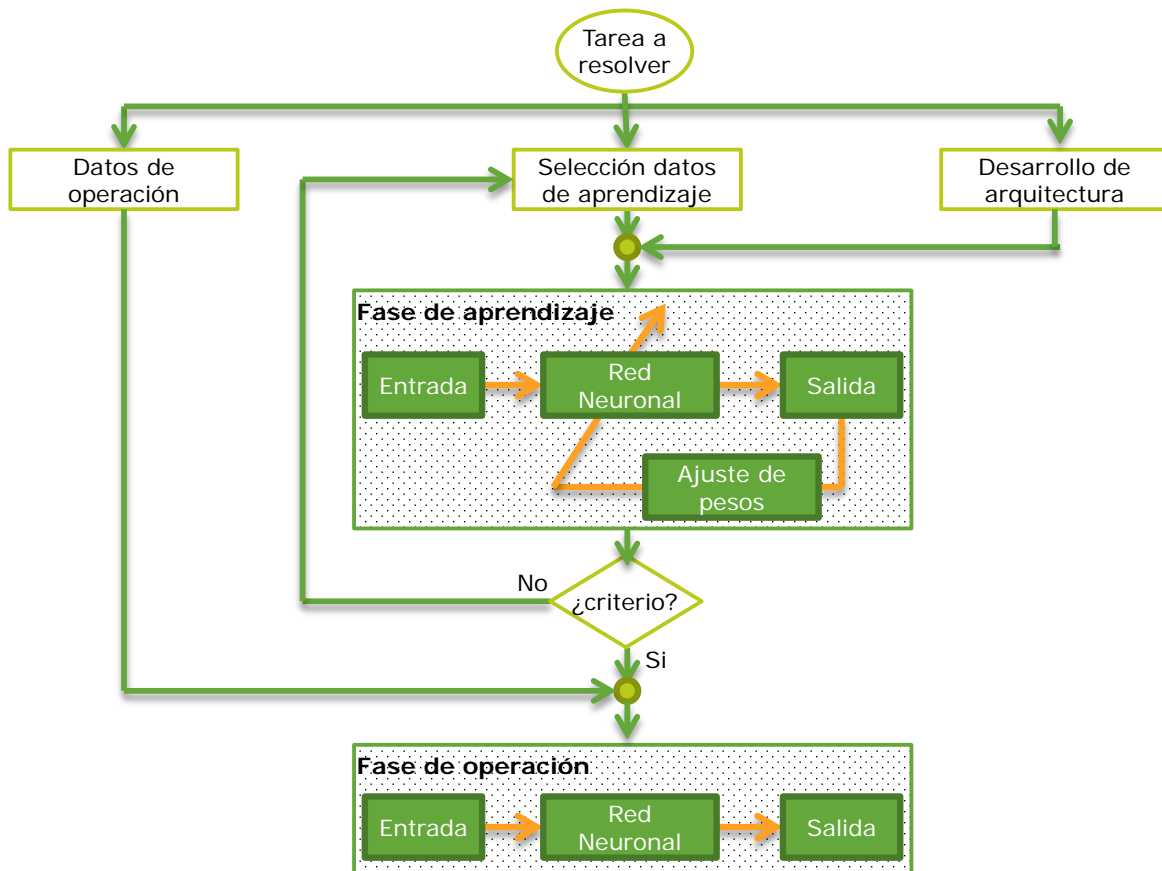


Figura 4.5: Diagrama de flujo para identificación con MLP.

V. IDENTIFICACIÓN CON RED NEURONAL POLINOMIAL

5.1. Introducción

Un área de rápido crecimiento en años recientes ha sido las redes neuronales artificiales. Este enfoque tiene pocas restricciones sobre el tipo de mapeo de entrada - salida que se puede aprender. La mayoría de las técnicas de identificación no lineal utilizando redes neuronales son fuera de línea lo cual significa que la estructura y parámetros del modelo (de red) son ajustados después de la identificación basada en un conjunto de datos de entrada - salida.

Para tener un buen desempeño de identificación tanto la estructura como los parámetros del modelo (de red) necesitan modificarse en respuesta a variaciones de las características de la planta y punto de operación. Cuando una identificación se hace fuera de línea, se cuenta con un vector finito de datos, por lo consiguiente, cualquier variación que se haya presentado en el sistema queda incluida en el vector de datos y la red neuronal se adapta. Sin embargo, cuando la identificación es en línea, siempre puede existir un dato para el cual la red aún no se entrene. Lo anterior se menciona debido a que una red neuronal se compone de interconexiones con pesos, se asume que existe un vector de pesos ideales que aproximará de la mejor manera a la función real, pero, si el sistema varía en el tiempo los pesos ideales podrían modificarse, por consecuencia es necesario que el método de actualización de pesos sea no solo en línea sino también adaptable. Recientemente, se han desarrollado nuevos algoritmos que operan en una ventana de datos y pueden utilizarse en línea para seguir y adaptarse a variaciones en la estructura del modelo o topología y actualizar los parámetros estimados o pesos en línea.

En este capítulo se presenta el marco teórico de las redes neuronales polinomiales. Primeramente se expone el modelo NARX el cual permite representar un sistema no lineal discreto en función de sus entradas y salidas y respectivos retrasos. Después, se presenta la

teoría de los polinomios de Volterra y los polinomios de Chebyshev y los algoritmos para generarlos. Se presentan dos métodos de actualización de pesos, mínimos cuadrados ortogonales para identificación fuera de línea y mínimos cuadrados recursivos para identificación en línea. Finalmente se presenta el algoritmo de reducción de Gram - Schmidt, si la identificación es fuera de línea se puede aplicar éste método, se propone una tolerancia de error deseada y el algoritmo agrupa las funciones base polinomiales de acuerdo a qué tan significativas son en la aproximación de manera que sólo sea necesario implementar (posteriormente, en línea) las funciones que más aportan, reduciendo el tamaño de la red.

Las expansiones de polinomios multivariados son ampliamente utilizadas en aproximación de funciones, particularmente cuando la entrada es unidimensional (Liu, 2001). La representación funcional de una expansión polinomial en una red neuronal esta descrita por:

$$f(a) = \hat{f}(a; p) + e(a^{k+1}) \quad (5.1)$$

$$\hat{f}(a; p) = \sum_{j=1}^N w_j \varphi_j(a) \quad (5.2)$$

donde $p = [w_1, w_2, \dots, w_N]$ y φ_j es el conjunto de funciones base formadas de los términos de entrada polinomiales, N es el número de funciones base polinomiales, k es el orden de la expansión polinomial, $e(a^{k+1})$ denota el error de aproximación causado por el orden alto ($\geq k + 1$) del vector de entrada. Las funciones base son esencialmente polinomios de ordenes cero, uno y más altos del vector de entrada $a \in \mathbb{R}^N$. Una diferencia importante entre redes polinomiales y otras redes es que las funciones base polinomiales por sí mismas son no parametrizadas y por lo tanto no es requerida la adaptación de las funciones base durante el aprendizaje.

5.2. Modelo NARX

Por otra parte, considere el sistema discreto no lineal descrito por:

$$X_{t+1} = G(X_t, u_t) \quad (5.3)$$

$$y_t = h(X_t, u_t) \quad (5.4)$$

donde $G(\cdot)$ es un vector de funciones no lineales, $h(\cdot)$ una función no lineal, X_t el vector de estados, y_t la salida y u_t la entrada. Basado en la relación de entrada y salida de un sistema, el sistema discreto no lineal (5.3), (5.4) puede ser expresado también por un modelo no lineal autorregresivo con entrada externa (NARX) (Ranković y Nikolić, 2008) el cual es:

$$y_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-n}, u_{t-1}, u_{t-2}, \dots, u_{t-m}) \quad (5.5)$$

donde $f(\cdot)$ es alguna función no lineal, n y m son los correspondientes retardos máximos.

5.3. Modelado no lineal por redes con polinomio de Volterra (VPBF)

Es bien sabido que las redes neuronales proveen buenas técnicas de aproximación de funciones no lineales. La estructura de identificación no lineal mediante redes neuronales se muestra en la Figura 2.1. Aquí, se asume que la función no lineal $f(\cdot)$ en el modelo (5.5) es aproximada por una red neuronal de una sola capa, la cual consiste de una combinación lineal de funciones base:

$$\hat{f}(x_t) = \sum_{k=1}^N \omega_k \varphi_k(x_t) \quad (5.6)$$

donde $x_t = [y_{t-1}, y_{t-2}, \dots, y_{t-n}, u_{t-1}, u_{t-2}, \dots, u_{t-m}]$, $\varphi_k(x_t)$ es la función base y ω_k el peso.

De acuerdo con el teorema de aproximación universal (Haykin, 1999), existen un número finito de funciones base de manera que la red neuronal puede aproximar la función no lineal de manera precisa. Se ha demostrado que cualquier precisión requerida en la aproximación puede ser alcanzada usando un número adecuado de funciones base no lineales independientes (Yu *et al.*, 2004), en este caso particular, las funciones polinomiales de Volterra. Hay que mencionar que Karl Weierstrass (Weierstrass, 1885) afirma que:

Teorema 5.3.1 *Cualquier función continua definida en un intervalo cerrado puede ser uniformemente aproximada de manera tan precisa como se desee por una función polinomial.*

También, para el teorema anterior, existen una generalización y una prueba simplificada realizadas por Marshall H. Stone (Stone, 1948).

La representación de la función no lineal $\hat{f}(x_t)$ está dada por:

$$\begin{aligned} \hat{f}(x_t) = & \omega_1 + \omega_2 y_{t-1} + \cdots + \omega_{n+1} y_{t-n} + \omega_{n+2} u_{t-1} + \cdots + \omega_{n+m+1} u_{t-m} \\ & + \omega_{n+m+2} y_{t-1}^2 + \omega_{n+m+3} y_{t-1} y_{t-2} + \cdots + \omega_N u_{t-m}^l \end{aligned} \quad (5.7)$$

donde l es el orden del sistema y N el número de funciones base. Hay que notar que (5.7) es equivalente a (5.6). El conjunto de funciones base polinomiales de Volterra es:

$$\begin{aligned} & [\varphi_1, \varphi_2, \varphi_3, \cdots, \varphi_{n+1}, \varphi_{n+2}, \cdots, \varphi_{n+m+1}, \varphi_{n+m+2}, \varphi_{n+m+3}, \cdots, \varphi_N](x_t) \\ & = [1, y_{t-1}, y_{t-2}, \cdots, y_{t-n}, u_{t-1}, \cdots, u_{t-m}, y_{t-1}^2, y_{t-1} y_{t-2}, \cdots, u_{t-m}^l] \end{aligned} \quad (5.8)$$

y el número de funciones base polinomiales está dado por:

$$N = \frac{(n+m+l)!}{l!(n+m)!} \quad (5.9)$$

Utilizando la red neuronal VPBF, la función no lineal $f(\cdot)$ puede ser obtenida por:

$$f(x_t) = \hat{f}(x_t) + o(x_t^l) \quad (5.10)$$

donde $o(x_t^l)$ es el error de aproximación. Incrementando el orden l , el número N de funciones base se vuelve más grande. Así, el problema es cómo estimar la función $\hat{f}(x_t)$ utilizando una red neuronal de tamaño adecuado para que la precisión de la aproximación esté dentro del límite requerido. La Figura 5.1 muestra una representación gráfica de (5.10).

La generación de las funciones base de Volterra depende del orden del polinomio que se desee de una manera que no puede escribirse como un algoritmo recursivo. A continuación se definirán funciones que servirán para generar las VPBFs mientras que en la Figura 5.2 se muestra el algoritmo para generar el polinomio de Volterra y por consecuencia la función base φ_k correspondiente.

Se define:

$$\Delta_1 = y(t-1), \quad \Delta_2 = y(t-2), \quad \cdots, \quad (5.11)$$

$$\Delta_n = y(t-n) \quad (5.12)$$

$$\Delta_{n+1} = u(t-1), \quad \Delta_{n+2} = u(t-2), \quad \cdots, \quad (5.13)$$

$$\Delta_{n+m} = u(t-m) \quad (5.14)$$

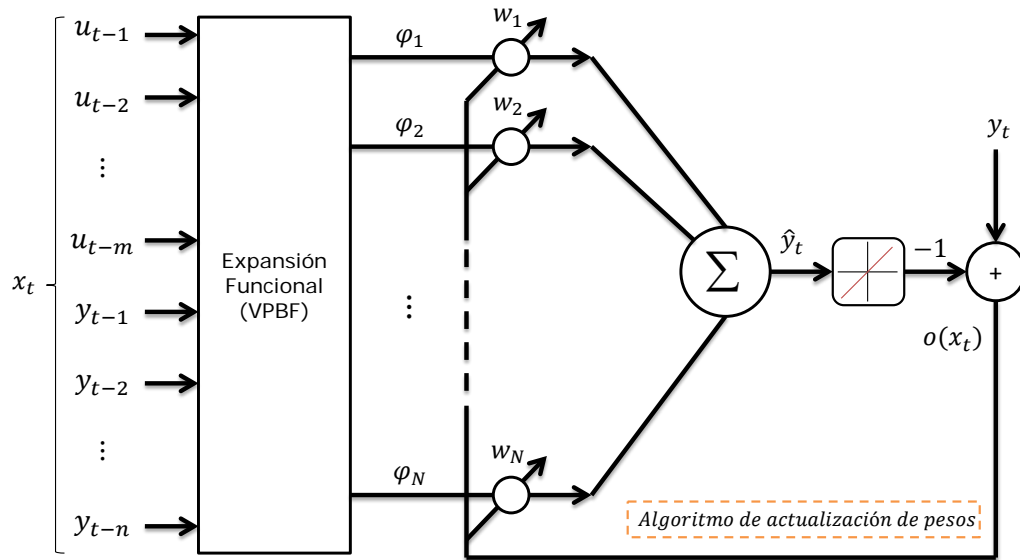


Figura 5.1: Diagrama de flujo de una red polinomial (Volterra).

Expresando las funciones base en forma vectorial, como en (5.8), se tiene:

$$\phi = [\varphi_1 \ \varphi_2 \ \dots \ \varphi_N]^T \quad (5.15)$$

5.4. Modelado no lineal por redes con polinomio de Chebyshev (CBF)

Aun cuando se han reportado algunas pruebas rigurosas de la capacidad de aproximación de funciones no lineales por medio de redes neuronales pre-alimentadas convencionales, estos enfoques usualmente requieren un largo tiempo de entrenamiento bajo las estructuras en las que se proponen, por consecuencia no son útiles para un esquema de aprendizaje en línea.

Para hacer frente al problema antes mencionado, se utiliza una red neuronal con funciones base polinomiales. De esta manera la identificación de sistemas dinámicos no lineales se lleva a cabo eliminando las capas ocultas (utilizadas en una red pre-alimentada convencional como la FNN) y utilizando en su lugar una expansión de la entrada al sistema mediante

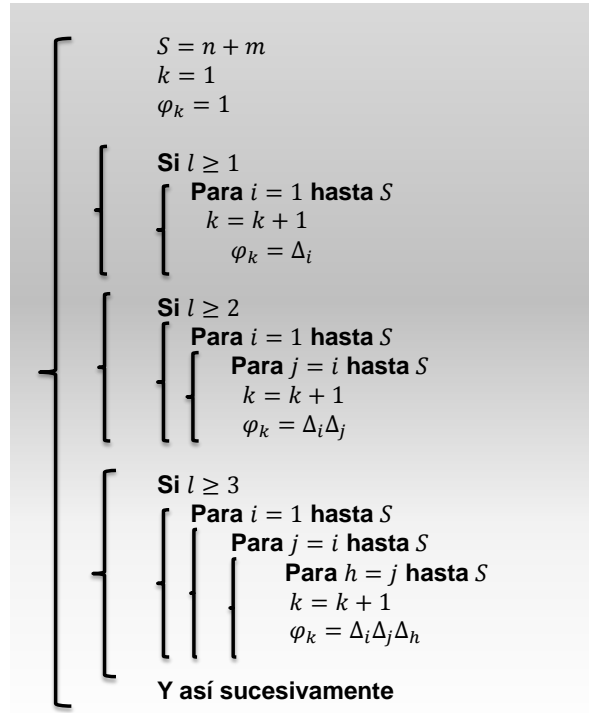


Figura 5.2: Algoritmo de generación de funciones base (Volterra).

polinomios de Chebyshev. El modelo de red neuronal con polinomios de Chebyshev (o de Volterra) es lineal en los pesos y no lineal en sus entradas (funciones base).

De nuevo considere el sistema discreto no lineal presentado en las ecuaciones (5.3) y (5.4), a partir de la expansión del sistema en forma NARX, dada en (5.5), se puede asumir que la función no lineal $f(\cdot)$ en el modelo (5.5) es aproximada por una red neuronal de una sola capa, la cual consiste de una combinación lineal de funciones base:

$$\hat{y}_t = \hat{f}(x_t) = \sum_{k=1}^N \omega_k \varphi_k(x_t) \quad (5.16)$$

donde $x_t \in \mathbb{R}^{1 \times S}$ es un vector formado por las entradas y salidas pasadas, $\varphi_k(x_t)$ es la función base y ω_k el peso. La representación de la función no lineal $\hat{f}(x_t)$ se puede ver en un esquema gráfico en la Figura 5.3. El número de funciones base polinomiales de Chebyshev está dado por:

$$N = 1 + (n + m)l \quad (5.17)$$

donde n y m son los retardos máximos en la salida y en la entrada, respectivamente, y l es el orden del polinomio. Utilizando la red neuronal CBF, la función no lineal $f(\cdot)$, presentada en

(5.5), puede ser obtenida como

$$f(x_t) = \hat{f}(x_t) + o(x_t^l) \quad (5.18)$$

donde $o(x_t^l)$ es el error de aproximación.

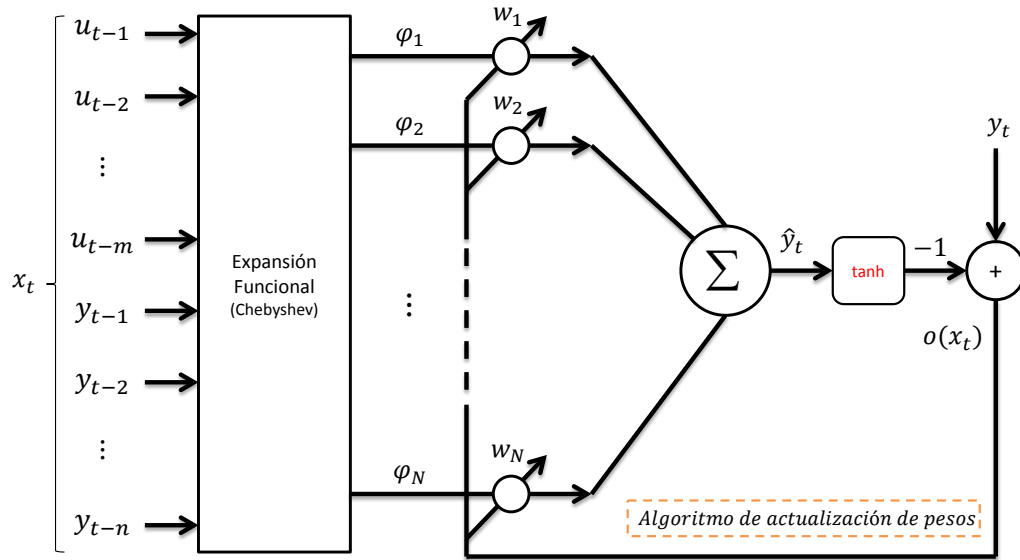


Figura 5.3: Diagrama de flujo de una red polinomial (Chebyshev).

De acuerdo con Purwar *et al.* (2007), Xiuchun *et al.* (2009), Li y He (2010) y Jiang *et al.* (2012), los polinomios de Chebyshev, a diferencia de los de Volterra, se pueden generar con una fórmula recursiva:

$$T_{i+1}(x_t) = 2x_j T_i(x_t) - T_{i-1}(x_t), \quad T_0(x_t) = 1, \quad T_1(x_t) = x_j \quad (5.19)$$

donde se define x_t como:

$$\begin{aligned} x_t &= [x_1, x_2, x_3, \dots, x_{S-1}, x_S] \\ &= [y_{t-1}, y_{t-2}, \dots, y_{t-n}, u_{t-1}, u_{t-2}, \dots, u_{t-m}] \end{aligned} \quad (5.20)$$

y $S = n + m$. El vector de funciones base, $\phi \in \mathbb{R}^{N \times 1}$ se define entonces como:

$$\begin{aligned} \phi &= [\varphi_1, \varphi_2, \dots, \varphi_N]^T \\ &= [1, T_1(x_1), T_2(x_1), \dots, T_1(x_2), T_2(x_2), \dots]^T \end{aligned} \quad (5.21)$$

donde T_i es un polinomio de Chebyshev. El algoritmo para generar los polinomios de Chebyshev se muestra en la Figura 5.4. Al igual que en las redes VPBF, se puede llegar a un mínimo en el error de aproximación, dentro de un límite requerido, variando el número de polinomios, N .

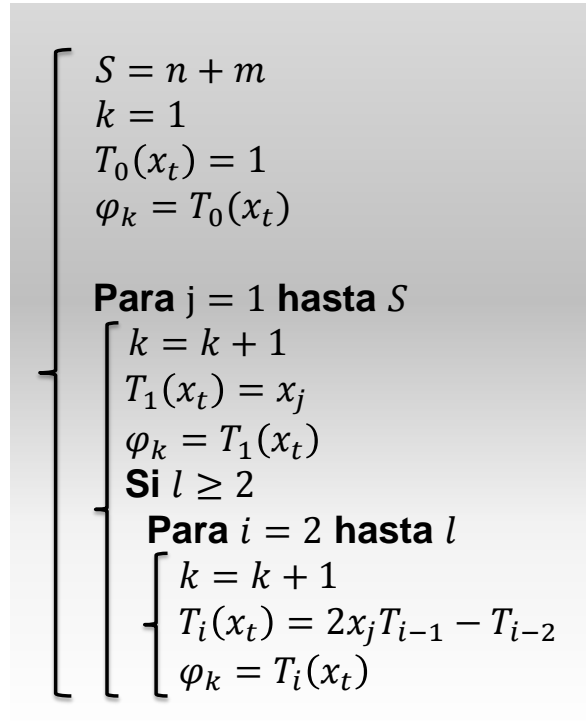


Figura 5.4: Algoritmo de generación de funciones base (Chebyshev).

5.5. Algoritmo de actualización de pesos

5.5.1 Mínimos cuadrados ortogonales (MCO)

Se presenta una técnica de entrenamiento fuera de línea para redes polinomiales, utilizando el algoritmo de mínimos cuadrados ortogonales (Billings *et al.*, 1989). De acuerdo con Patrikar y Provence (1996), Liu *et al.* (1998) y Liu (2001) se asume que un conjunto de datos de entrada-salida $(u_t, y_t, t = 1, 2, \dots, M)$ del sistema es dado. Con base en (5.10) y (5.18), la relación entrada salida puede ser escrita de forma compacta como la siguiente forma vectorial:

$$Y = \Phi(x)W + O(x^l) \quad (5.22)$$

donde el vector de salida $Y \in \mathbb{R}^{M \times 1}$, el vector de pesos $W \in \mathbb{R}^{N \times 1}$, el vector de error de aproximación $O(x^l) \in \mathbb{R}^{M \times 1}$ y la matriz de funciones base $\Phi(x) \in \mathbb{R}^{M \times N}$ son, respectivamente:

$$Y = [y_1 \ y_2 \ \cdots \ y_M]^T \quad (5.23)$$

$$W = [w_1 \ w_2 \ \cdots \ w_N]^T \quad (5.24)$$

$$O(x^l) = [o(x_1^l) \ o(x_2^l) \ \cdots \ o(x_M^l)]^T \quad (5.25)$$

$$\Phi(x) = \begin{bmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_N(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_N(x_2) \\ \vdots & \vdots & \cdots & \vdots \\ \varphi_1(x_M) & \varphi_2(x_M) & \cdots & \varphi_N(x_M) \end{bmatrix} \quad (5.26)$$

El vector de pesos W se puede estimar minimizando la norma Euclidiana:

$$\hat{W} = \arg \min_W \|Y - \Phi(x)W\|_2 \quad (5.27)$$

la ecuación (5.27) es una solución de mínimos cuadrados.

Los vectores $\Phi_i = [\varphi_i(x_1), \varphi_i(x_2), \dots, \varphi_i(x_M)]^T$, para $i = 1, 2, \dots, N$, forman un conjunto de vectores base, y la solución de mínimos cuadrados ortogonales, \hat{W} , satisface la condición que $\Phi(x)\hat{W}$ será la proyección de Y en el espacio abarcado por los vectores de funciones base Φ_i . El método de mínimos cuadrados ortogonales involucra la transformación de un conjunto de vectores base Φ_i en un conjunto de vectores base ortogonales, y así se hace posible calcular la contribución individual, de cada vector base, a la salida deseada. Una descomposición ortogonal de la matriz $\Phi(x)$ resulta en:

$$\Phi(x) = PQ \quad (5.28)$$

donde $P = [P_1, P_2, \dots, P_N]$ es una matriz de $M \times N$ con columnas ortogonales y Q es una matriz triangular superior de $N \times N$ con 1 en la diagonal y 0 debajo de la diagonal, esto es:

$$Q = \begin{bmatrix} 1 & q_{12} & q_{13} & \cdots & q_{1N} \\ 0 & 1 & q_{23} & \cdots & q_{2N} \\ 0 & 0 & 1 & \cdots & q_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (5.29)$$

La matriz P satisface:

$$P^T P = D \quad (5.30)$$

donde D es una matriz diagonal con elementos d_i :

$$d_i = P_i^T P_i, \quad 1 \leq i \leq N \quad (5.31)$$

Utilizando la ecuación (5.31), se puede reescribir la ecuación (5.22) como:

$$Y = PV + O(x^l) \quad (5.32)$$

$$W = Q^{-1}V \quad (5.33)$$

donde $V = [v_1, v_2, \dots, v_N]^T \in \mathbb{R}^{N \times 1}$. Se puede observar que la estimación óptima $\hat{V} = [\hat{v}_1, \hat{v}_2, \dots, \hat{v}_N]^T$ del vector V es:

$$\hat{v}_i = \frac{Y^T P_i}{P_i^T P_i}, \quad i = 1, 2, \dots, N \quad (5.34)$$

de manera que $\|Y - P\hat{V}\|_2$ es mínima. El correspondiente vector de pesos óptimo es:

$$\hat{W} = Q^{-1}\hat{V} \quad (5.35)$$

La relación de reducción de error debido a P_i (Billings *et al.*, 1989) se puede expresar por:

$$r_i = \frac{\hat{v}_i P_i^T Y}{Y^T Y} \quad (5.36)$$

Esta relación ofrece una manera simple y efectiva de buscar un subconjunto de funciones base significativas.

5.5.2 Mínimos cuadrados recursivos (MCR)

El problema de identificación consiste en establecer un modelo (de identificación) adecuado y ajustar los parámetros (pesos) del modelo para optimizar una función de desempeño con base en el error entre las salidas de la planta y la salida estimada (identificada). Una red neuronal de una sola capa, como lo es una red con expansión polinomial, es lineal en los pesos y no lineal en las funciones base (Purwar *et al.*, 2007).

El método de mínimos cuadrados recursivos con factor de olvido puede utilizarse como algoritmo de aprendizaje para la actualización de pesos en línea.

Comentario 5.5.1 De acuerdo con Purwar et al. (2008) para la identificación se asume que existe un vector de pesos ideales, que no es necesario conocer, al cual el algoritmo de aprendizaje aproximará.

Dado que el sistema a identificar puede sufrir variaciones en el tiempo, el vector de pesos ideal podría sufrir variaciones consecuencia de lo anterior. Según Åström y Wittenmark (1994) la estimación mediante mínimos cuadrados recursivos cuando los parámetros son variantes en el tiempo, lentamente, da lugar a la aplicación del factor de olvido exponencial. La función de desempeño para ser minimizada está dada por:

$$E = \sum_{i=1}^M \lambda^{M-i} |\epsilon(i)|^2 \quad (5.37)$$

donde E es la función de error de mínimos cuadrados ponderados, M es el número total de muestras, λ es el factor de olvido.

El algoritmo para el modelo en tiempo discreto está dado por Purwar et al. (2007):

$$\hat{W}_t = \hat{W}_{t-1} + k_t \epsilon_t \quad (5.38)$$

$$k_t = \frac{\lambda^{-1} P_{t-1} \phi_t}{1 + \lambda^{-1} \phi_t^T P_{t-1} \phi_t} \quad (5.39)$$

$$\epsilon_t = y_t - \hat{y}_t \quad (5.40)$$

$$P_t = \lambda^{-1} P_{t-1} - \lambda^{-1} k_t \phi_t^T P_{t-1} \quad (5.41)$$

siendo y_t la salida medida de la planta y \hat{y}_t la salida estimada por la red neuronal para todo $t = 1, 2, \dots, M$, el vector de pesos estimados es $\hat{W}_t \in \mathbb{R}^{N \times 1}$ definido $\hat{W}_t = [w_1 \ w_2 \ \dots \ w_N]^T$, el vector de funciones base se denota como $\phi_t \in \mathbb{R}^{N \times 1}$ definido $\phi_t = [\varphi_1 \ \varphi_2 \ \dots \ \varphi_N]^T$ y $P_t \in \mathbb{R}^{N \times N}$, $k_t \in \mathbb{R}^{N \times 1}$ es un vector de ganancias; N representa el número de funciones base (polinomios).

Para mas información sobre el método de MCR ver Yoon et al. (2011), para el análisis de estabilidad del método de actualización de pesos con MCR ver Purwar et al. (2007).

5.6. Reducción por método de Gram-Schmidt

El método de MCO sirve no únicamente para actualizar los pesos de la red neuronal. Es posible utilizar el método de MCR para actualizar los pesos de la red y después, fuera de

línea, se puede utilizar el método de MCO para estimar la relación de reducción de error de la red neuronal y con esto poder obtener los términos significativos a la aproximación con el fin de reducir el tamaño de la expansión polinomial de la red. Para lo antes mencionado, se utiliza el método de Gram - Schmidt clásico (Liu, 2001). El algoritmo de Gram - Schmidt clásico se describe a continuación:

1. En el primer paso, para $i = 1, 2, \dots, N$, calcular:

$$P_1^{(i)} = \Phi_i \quad (5.42)$$

$$V_1^{(i)} = \frac{Y^T P_1^{(i)}}{\left(P_1^{(i)}\right)^T P_1^{(i)}} \quad (5.43)$$

$$r_1^{(i)} = \frac{V_1^{(i)} \left(P_1^{(i)}\right)^T Y}{Y^T Y} \quad (5.44)$$

Encontrar:

$$s_1 = \arg \max\{r_1^{(i)}, i = 1, 2, \dots, N\} \quad (5.45)$$

y seleccionar:

$$P_1 = P_1^{(s_1)} = \Phi_{s_1} \quad (5.46)$$

2. En el k-ésimo paso, donde $k \geq 2$ para $i = 1, 2, \dots, N, i \neq s_1, \dots, i \neq s_{k-1}$, calcular:

$$\alpha_{jk}^{(i)} = \frac{\Phi_i^T P_j}{(P_j)^T P_j}, \quad j = 1, 2, \dots, k \quad (5.47)$$

$$P_k^{(i)} = \Phi_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} P_j \quad (5.48)$$

$$V_k^{(i)} = \frac{Y^T P_k^{(i)}}{\left(P_k^{(i)}\right)^T P_k^{(i)}} \quad (5.49)$$

$$r_k^{(i)} = \frac{V_k^{(i)} \left(P_k^{(i)}\right)^T Y}{Y^T Y} \quad (5.50)$$

Encontrar:

$$s_k = \arg \max\{r_k^{(i)}, i = 1, 2, \dots, N, i \neq s_1, \dots, i \neq s_{k-1}\} \quad (5.51)$$

y seleccionar:

$$P_k = P_1^{(s_k)} = \Phi_{s_k} - \sum_{j=1}^{k-1} \alpha_{jk}^{(s_k)} P_j \quad (5.52)$$

3. El procedimiento concluye en el L -ésimo paso cuando:

$$1 - \sum_{j=1}^L r_j < e_0 \quad (5.53)$$

donde $0 < e_0 < 1$ es la tolerancia elegida. Esto genera un modelo subconjunto que contiene L términos significativos.

5.7. Notas del capítulo

En este punto del trabajo es conveniente aclarar al lector algunas cuestiones:

- En todos los trabajos relacionados con redes neuronales polinomiales, presentados en el Capítulo 2, proponen su propia estructura de red. En este trabajo la estructura de red y el algoritmo de aprendizaje son únicos independientemente de si se utilizan polinomios de Volterra o de Chebyshev.
- La inclusión de un factor de olvido en el método de mínimos cuadrados recursivos hace que los datos recientes tengan un peso unitario, pero los datos retrasados n veces tendrán un peso de λ^n .
- Una desventaja de incluir un factor de olvido es que aun cuando el error de la red no contenga ninguna información nueva para actualizar los pesos de la red, los datos seguirán siendo olvidados.
- Para poder hacer una reducción por medio de Gram-Schmidt es necesario entrenar primero la red fuera de línea por lo cual es conveniente que se utilicen datos de entrenamiento que cubran todo el rango en el que opera el sistema.
- La red tiene que tener una entrada que cumpla la condición de excitación persistente, además se recomienda que esta entrada cubra todas las zonas del rango en el que opera el sistema.

- De acuerdo con (Purwar *et al.*, 2007) con la siguiente función de Lyapunov:

$$V_t = \lambda^{M-t} \tilde{W}_t^T P_t^{-1} \tilde{W}_t$$

con $\tilde{W}_t = W_t^* - \hat{W}_t$ se asegura (mediante el segundo método de Lyapunov) la convergencia de los pesos estimados hacia los pesos ideales ($\hat{W}_t \rightarrow W_t^*$, $t \rightarrow \infty$) mediante el método de mínimos cuadrados recursivos con factor de olvido.

VI. METODOLOGÍA

6.1. Prototipo de laboratorio

El sistema PISC utilizado en este trabajo es el RT-124 de la marca G.U.N.T., este modelo forma parte de una serie de sistemas de enseñanza desarrollados por la marca (G.U.N.T.) en colaboración con el Departamento de Automatización e Informática de la *Harz University of Applied Studies and Research*. El dispositivo cuenta con un potenciómetro giratorio y un encoder, como sensores, y un motor de C.D., como actuador.

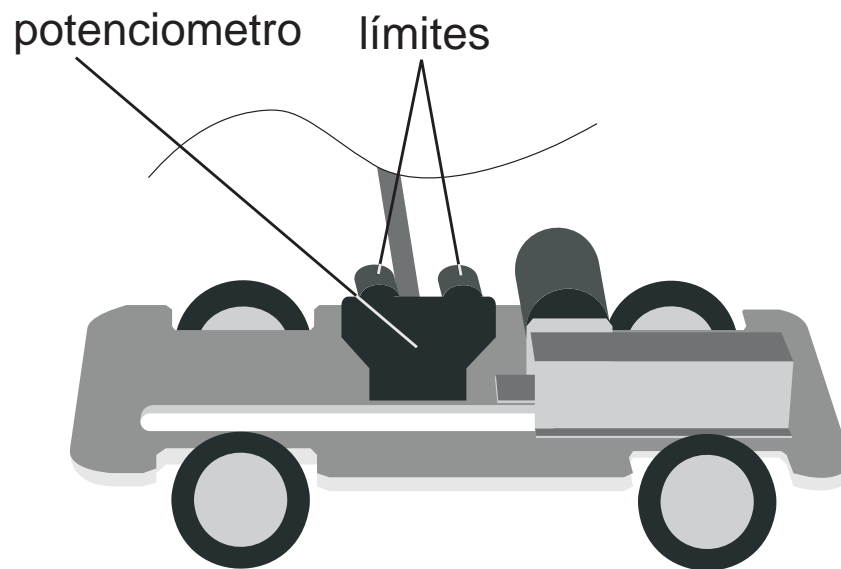


Figura 6.1: Vehículo del RT124.

El pivote del péndulo es un balero, dos toques limitan la desviación máxima, Figura 6.1. El potenciómetro giratorio está incorporado directamente con el péndulo, esto sirve para sensar el ángulo en función del voltaje. El motor mueve las dos llantas traseras y, por consecuencia, el vehículo. El encoder giratorio está localizado bajo el vehículo y es utilizado para determinar su posición en el plano horizontal, Figura 6.2.

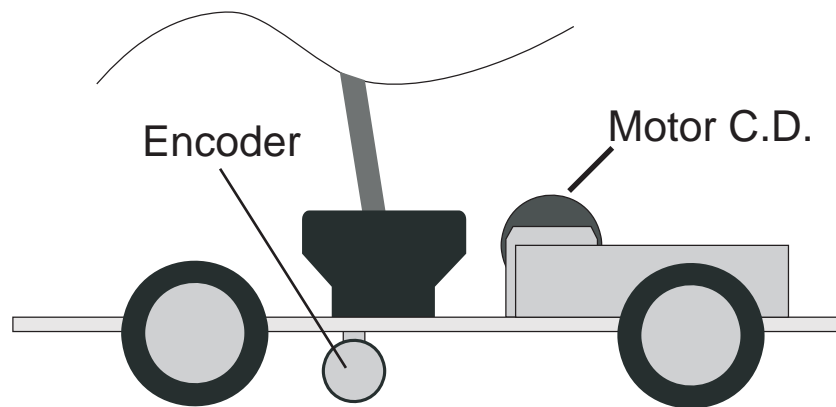


Figura 6.2: Vista lateral del vehículo.

En la Tabla 6.1 se muestran todos los parámetros físicos del sistema, estos parámetros se obtuvieron de las especificaciones presentadas en G.U.N.T. (2008).

Como ya se mencionó anteriormente, para identificar el modelo cerca del punto inestable se necesita aplicar algún tipo de control al PISC, no se puede excitar la entrada sin hacer control porque el péndulo caerá. Una vez que se tiene controlado el sistema cerca de su punto de operación es necesario que tenga perturbaciones que hagan que se mueva dentro del rango de operación inestable.

6.2. Algoritmo compacto para red polinomial

Con base en lo presentado en el Capítulo 5 se diseñó un algoritmo compacto de red neuronal polinomial. El objetivo de este algoritmo es generar una red neuronal polinomial donde sólo se tenga que cambiar el tipo de expansión funcional y la función de activación en la salida, cual si fueran módulos, de esta forma no se tiene que generar un nuevo algoritmo para cada tipo de expansión. La función de activación sirve para normalizar la salida dentro de un rango o dejar la salida tal cual, depende de las necesidades del usuario final. En la Figura 6.3 se muestra una representación gráfica del algoritmo mencionado. En el apéndice B.2, se muestra el código del algoritmo. En los apéndices B.4 y B.7, se muestra el código para generar las funciones base de Volterra y Chebyshev, respectivamente.

Tabla 6.1: Parámetros del prototipo de laboratorio.

Símbolo	Parámetro	Valor
M	Masa del carro	2.00 Kg
m	Masa del péndulo	0.10 Kg
l	Largo del péndulo	1.00 m
r	Radio de la llanta	0.0335 m
g	Aceleración debida a la gravedad	9.81 m/s ²
B_m	Coefficiente de amortiguación de la flecha	0.0001568 Nm/rad/s
R_a	Resistencia de armadura	1.8 Ω
K_i	Constante de torque	0.0168 Nm/A
K_b	Constante contra electromotriz	0.0168 V/rad/s
J_m	Momento de inercia del rotor	0.000011 Kg m ²
n	Radio de transmision	0.2
P	Potenciómetro para la posición angular	5 k Ω
E	Encoder rotacional	50 ppr

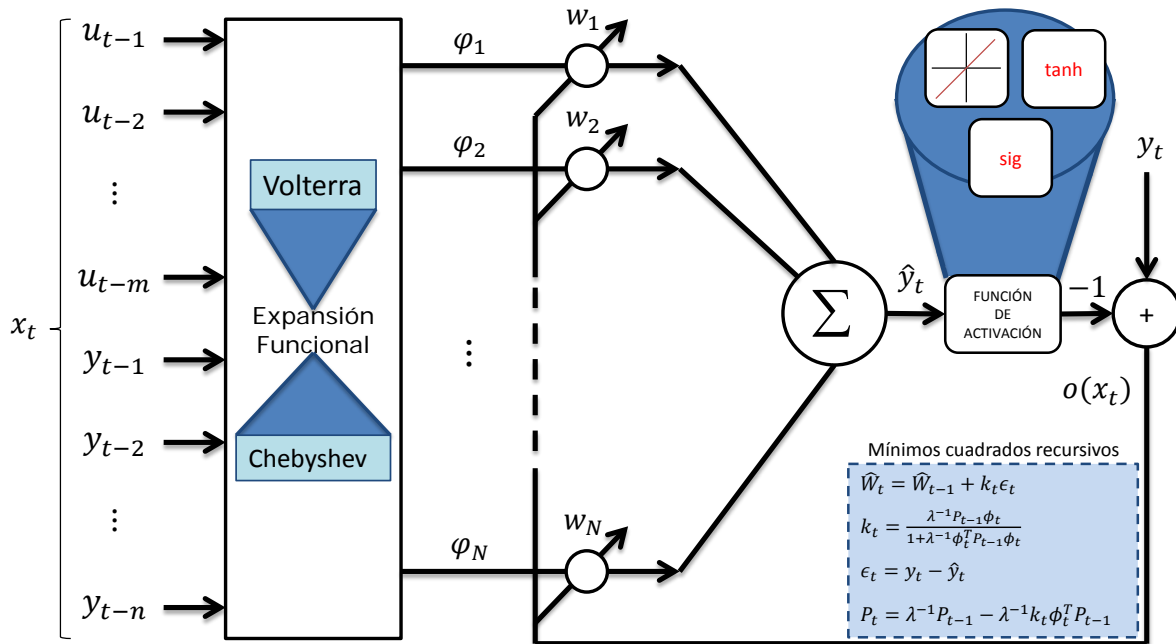


Figura 6.3: Diagrama de flujo de algoritmo compacto.

6.3. Simulación

Para probar los algoritmos de identificación generados se realizó una simulación, del PISC, en SIMULINK. Con el fin de hacer la simulación lo más realista posible al modelo del PISC se le incluyó: los modelos de fricción en la llanta del carro y en la unión rotativa del péndulo, la saturación del actuador y perturbaciones no controladas en el actuador.

En la Figura 6.4 se muestra el diagrama a bloques general, este diagrama incluye la dinámica no lineal del péndulo, un control por retroalimentación completa del estado, una saturación del actuador que representa la fuerza de tracción máxima del carro, acotada entre $-12N$ y $+12N$, y finalmente las perturbaciones. Dado que el sistema debe cumplir con la condición de excitación persistente para poder ser identificado correctamente se aplicó una perturbación externa, aleatoria, cada 2.5 segundos, directamente sobre el péndulo.

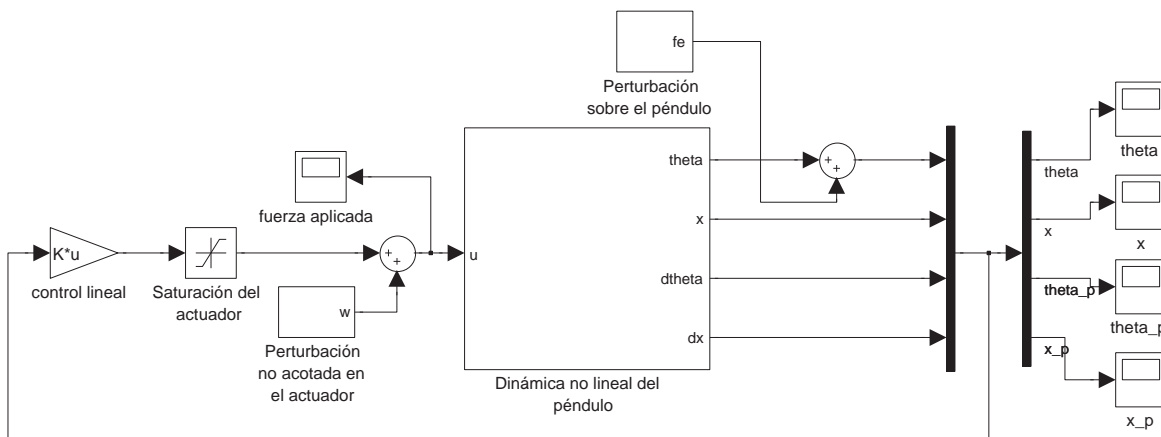


Figura 6.4: Diagrama general del sistema.

Las ganancias del controlador lineal se eligieron mediante un diseño por ubicación de los polos (ver apéndice A), siendo las ganancias utilizadas para la simulación las que se muestran en (A.16). Los parámetros de configuración de la simulación se ajustaron de la siguiente manera:

- Tiempo de muestreo = $10ms$
- Tiempo total de simulación = $75s$
- Método de integración = ode1(Euler)

Dado que el tiempo de muestreo es fijo a $10ms$ no es posible insertar las perturbaciones a intervalos de tiempo diferente de manera directa. El contenido de los bloques de perturbaciones se muestra en la Figura 6.5. Las perturbaciones están compuestas de ruido blanco aleatorio y un bloque de transición el cual inserta la perturbación en el sistema cada cierto intervalo.

Comentario 6.3.1 *El ruido blanco es una señal aleatoria (proceso estocástico) que se caracteriza por el hecho de que sus valores de señal en dos tiempos diferentes no guardan correlación estadística. Como consecuencia su gráfica es plana. Esto significa que la señal contiene todas las frecuencias y todas ellas muestran la misma potencia.*

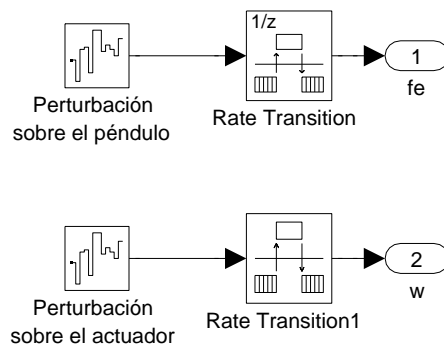


Figura 6.5: Perturbaciones no acotadas.

Para la perturbación no acotada sobre el actuador:

- Amplitud del ruido = 0.1
- Tiempo de inserción al sistema = 0.1 s

Para la perturbación sobre el péndulo:

- Amplitud del ruido = 0.0005
- Tiempo de inserción al sistema = 2.5 s

El subsistema que compone el bloque “Dinámica no lineal del péndulo” se muestra en la Figura 6.6. Este subsistema está compuesto de una *MATLAB function*, en la cual se han programado las ecuaciones (3.33) y (3.36) dando como salida \ddot{x} y $\ddot{\theta}$.

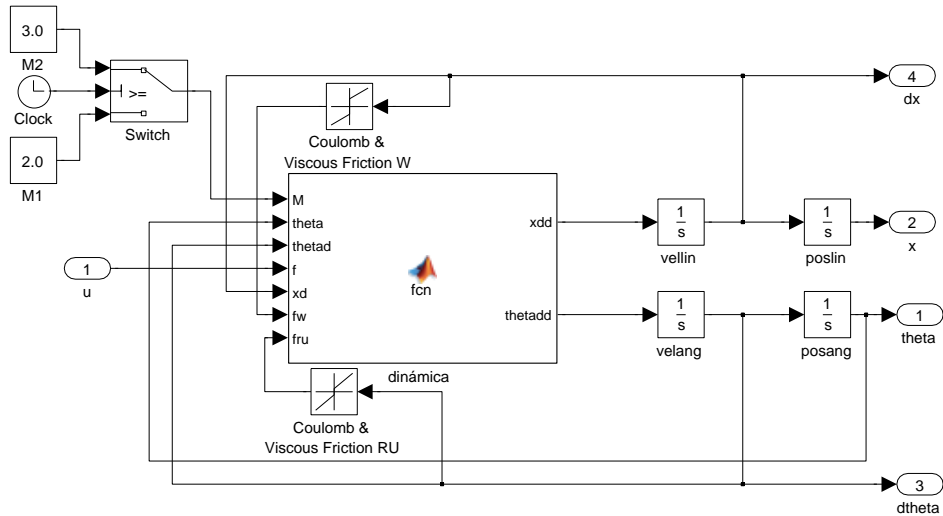


Figura 6.6: Subsistema para la dinámica del PISC.

A las ecuaciones (3.33) y (3.36), se agrega la fricción entre la llanta del carro y el suelo y también la fricción en la unión rotativa (potenciometro) del péndulo. Las fricciones se simulan con un bloque integrado de SIMULINK, llamado *Coulomb & Viscous Friction*, que responde a la ecuación:

$$F = \text{sign}(a) (B |a| + C) \quad (6.1)$$

donde F representa la fuerza de fricción, a es la señal de entrada, B el coeficiente de fricción viscosa y C es el valor de la fricción de Coulomb. Los valores utilizados en las fricciones son los siguientes:

Para la fricción entre la llanta y el suelo:

- $C = 0.05$
- $B = 0.06$

Para la fricción en la unión rotativa:

- $C = 0.001$
- $B = 0.006$

Las fricciones se suman (o restan, según sea el caso) directamente con las funciones que componen \ddot{x} y $\ddot{\theta}$, como se observa en la Figura 6.6, seguido a esto, las salidas, se integran

una vez para tener las variables de estado \dot{x} y $\dot{\theta}$, correspondientes a la velocidad del carro y del péndulo, respectivamente. Se integra una vez más y se obtienen las otras dos variables de estado, x y θ , que representan posición del carro y de péndulo, respectivamente.

Debido a que este trabajo se enfoca a identificar un sistema variante en el tiempo, se propone realizar un cambio en algún parámetro físico del sistema. Por sencillez se eligió variar la masa del carro, M . En la simulación esto se logra mediante un bloque *Switch* que es activado por un *Clock*, de manera que al segundo 70 de simulación la masa variará de $M1 = 2kg$ a $M2 = 3kg$.

Gracias a los *Scope* de SIMULINK los datos de simulación pueden ser guardados en arreglos finitos, de manera que podemos tener un vector para la entrada, u , y cada estado, θ , x , $\dot{\theta}$, \dot{x} . Cada vector será de dimensiones 7501×1 debido a que $t = 1, 2, \dots, 7501$. Los datos adquiridos de la simulación se muestran en la Figura 6.7.

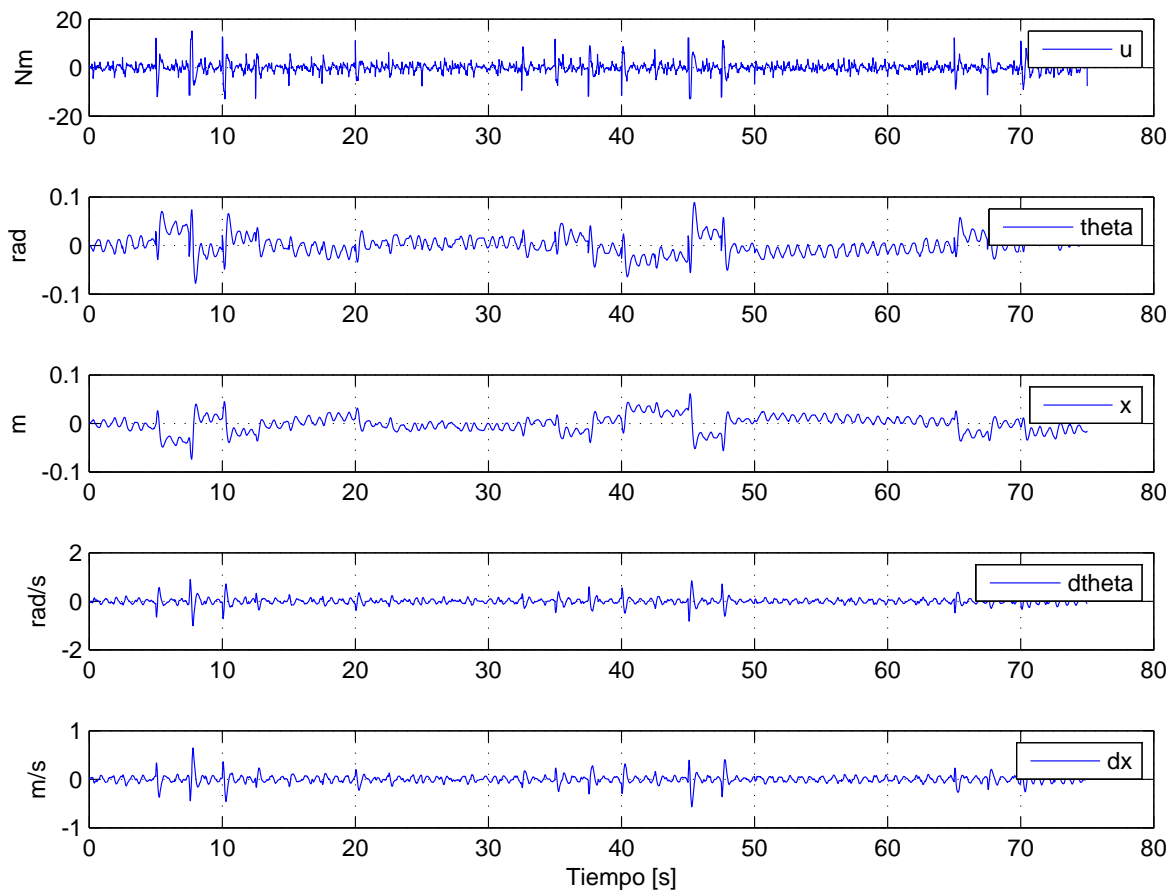


Figura 6.7: Datos adquiridos de la simulación.

Estos datos se aplican directamente a los algoritmos de red MLP, red VPBF y red CBF propuestos en las secciones anteriores (ver los programas en los apéndices B.1 y B.2). Los resultados se presentan y discuten en el siguiente capítulo de este trabajo.

6.4. Experimento

Como ya se mencionó anteriormente, se utilizó un sistema didáctico PISC (RT-124) de G.U.N.T. Este prototipo tiene un módulo de lógica difusa integrado en su circuito de control, para más información ver G.U.N.T. (2008), de manera que se puede utilizar un controlador difuso automático para estabilizar el péndulo alrededor de su punto de operación (posición vertical invertida). Hay que recordar que al proceso de identificación no le afecta el tipo de controlador que se utilice para mantener al péndulo equilibrado; la identificación solo toma muestras de la entrada al sistema y de la salida del sistema, independientemente de como se genere la entrada, Figura 6.8.

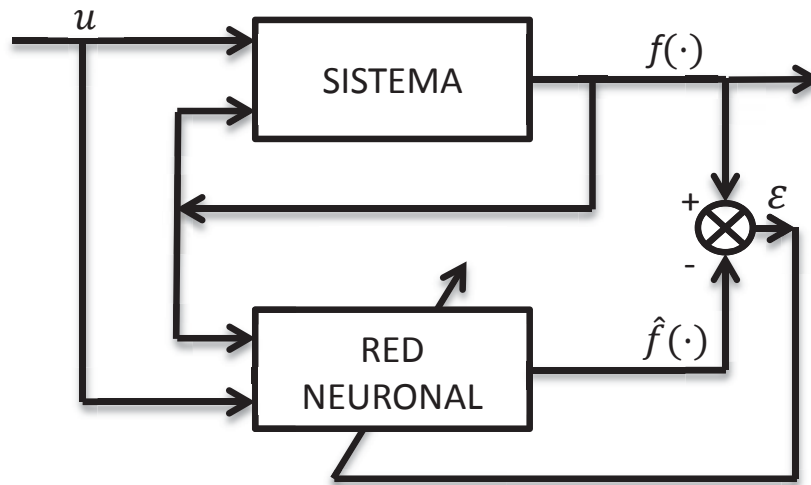


Figura 6.8: Identificación mediante red neuronal artificial.

Los parámetros del PISC se muestran en la Tabla 6.1. El sistema controlado fue perturbado continuamente para cumplir la condición de excitación persistente. Un sistema cRIO-9074, procesador para aplicaciones en RT y FPGAs en ambiente *LabVIEW*, de *National Instruments* con tarjetas para entradas digitales NI 9411 (utilizada para la lectura de *encoder*) y entradas analógicas NI 9205 (utilizada para la lectura de señal de voltaje generada por

potenciometro) se utilizó para adquirir los datos experimentales del péndulo invertido con un tiempo de muestreo $T_s=10\text{ms}$ durante 75 segundos. Los datos experimentales se obtuvieron utilizando el software de tiempo real (RT) de *LabVIEW 2012* (apéndice C.1). En la Figura 6.9 y en la Figura 6.10 se muestran fotografías del RT-124, cRIO y el *software* de adquisición.

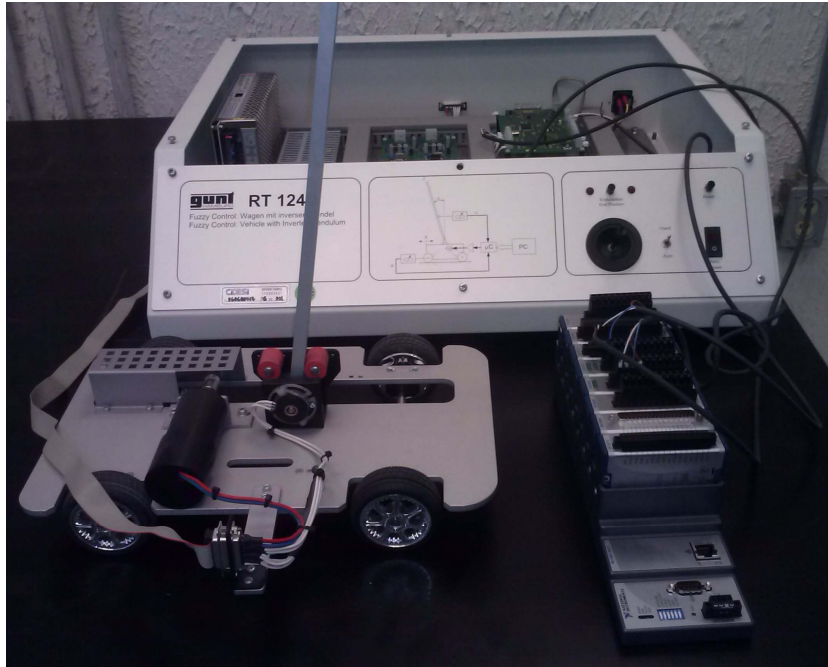


Figura 6.9: Sistema experimental RT124.

Los datos adquiridos se guardaron en arreglos de dimensión finita en un archivo de texto (apéndice C.2), al igual que los datos de simulación. Se pueden apreciar los datos reales en la Figura 6.11. Se aplicó un filtro antialiasing (filtro pasa-bajas analógico) de 50Hz para limpiar la señal antes de hacer la conversión analógica-digital.

Los datos adquiridos se aplican directamente a los algoritmos de red MLP, red VPBF y red CBF propuestos en las secciones anteriores (ver los programas en los apéndices B.1 y B.2). Los resultados se presentan y discuten en el siguiente capítulo.

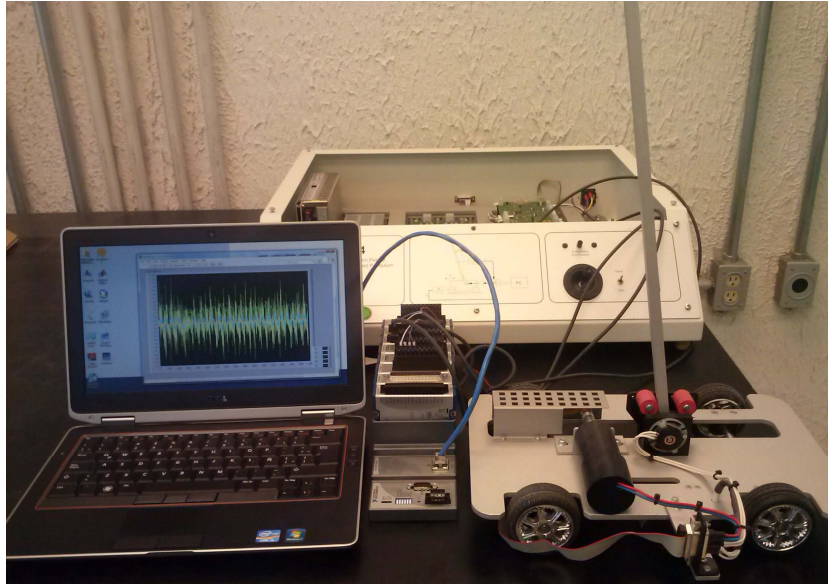


Figura 6.10: Sistema experimental y computadora.

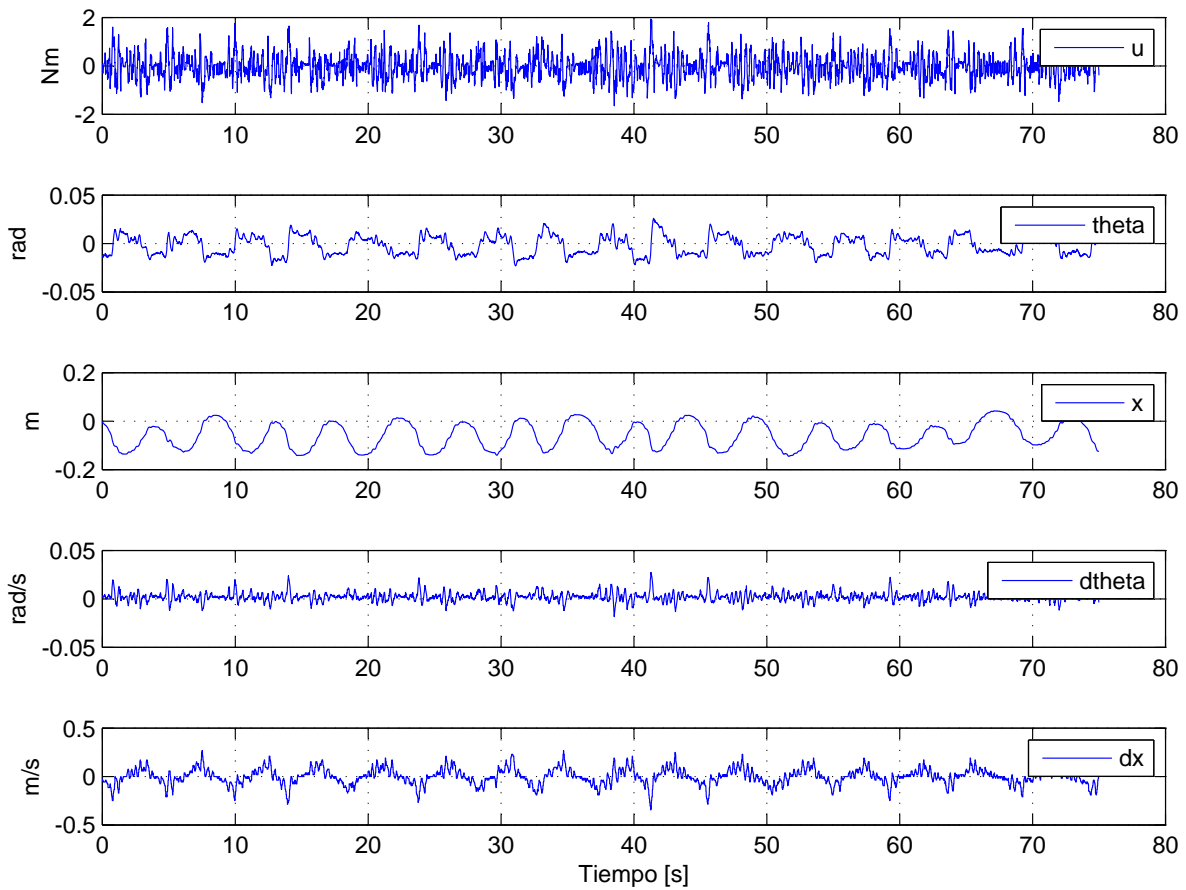


Figura 6.11: Datos adquiridos del experimento.

VII. RESULTADOS Y DISCUSIÓN

7.1. Introducción

En este capítulo se presentan los resultados que se obtuvieron de aplicar datos de simulación y datos reales (presentados en el capítulo 6) como entrada a la red MLP y la red polinomial compacta desarrollada (apéndice B.2). También se desarrolló un diseño experimental 2^k para cada red, tanto en simulación como en experimento real, donde se muestran las tendencias de cada red neuronal. Se presentan tablas estadísticas que muestran el error de media cuadrática (MSE) para cada red neuronal, también se presentan gráficas de tendencia resultantes del diseño experimental y la gráfica correspondiente a la mejor aproximación de cada red. El motivo por el que se utiliza el MSE es porque la red multicapa lo calcula forzosamente en su algoritmo de actualización (retro-propagación), es fácil de obtener en la red polinomial, y además es un promedio de errores instantáneos. Para concluir el capítulo, se analizan los resultados, se hacen comparaciones estadísticas en cuanto al error mínimo alcanzado en relación a la complejidad de la red y se concluye cuál red es mejor.

■ Simulación

Se utilizan los datos adquiridos de la simulación para estudiar el desempeño de los algoritmos propuestos en los capítulos 4 y 5. Las simulaciones se realizaron en MATLAB 7.12.0 (R2011a), instalado en una computadora Gateway NV57H15m, con sistema operativo Windows 7 64 bits, los datos obtenidos se muestran en la Figura 6.7. Se aplicó cada par (entrada y estado) de series temporales al algoritmo MLP, presentado en el apéndice B.1, y se realizó un diseño de experimentos factorial utilizando Minitab 16. Los mismos datos entrada-estado, se aplicaron a las redes VPBF y CBF, mediante el algoritmo compacto propuesto en el apéndice B.2, también se realizó un diseño de experimentos para éstas redes.

- Experimento

Se presentan los resultados en términos del MSE para los datos presentados en la sección 6.4. Como ya se mencionó antes, los datos fueron adquiridos mediante un sistema cRIO y se guardaron en un archivo “.txt”. Así como con los datos de simulación, los datos del experimento se aplicaron a los algoritmos de redes neuronales propuestos en los apéndices B.1 y B.2. De igual manera se realizó el diseño experimental factorial 2^3 a cada RNA, donde cada variable del sistema se considera una réplica del experimento con la red.

Dado que el algoritmo compacto que se propone en el apéndice B.2 utiliza como actualización de pesos el algoritmo MCR el factor de olvido (λ) debe ser seleccionado de manera correcta para el buen funcionamiento de la red. En la Tabla 7.1 se muestran los valores típicos y las muestras sobre las que tiene efecto el factor de olvido, fue propuesta originalmente por Aguado Behar y Martínez Iranzo (2003).

Tabla 7.1: Relación entre factor de olvido y número de datos significativos a la aproximación por MCR.

F. Olvido λ	Número de datos significativos	F. Olvido λ	Número de datos significativos
0.999	1155	0.994	195
0.998	580	0.993	165
0.997	385	0.992	145
0.996	290	0.991	134
0.995	230	0.990	125

En este capítulo se utilizan tablas de diseño de experimentos y de análisis de variancia. La primera presenta todos los MSE’s resultantes de la red neuronal, en ella se identifica cada variable del sistema haciendo variar tres factores de la red y se resalta en negro el mejor resultado en relación al error de aproximación y la complejidad de cálculo requerida. El segundo tipo de tabla presenta el análisis de variancia del diseño experimental, esta tabla da resultados estadísticos que nos permiten saber con claridad qué factor o factores tienen mayor influencia sobre la variación del error de aproximación de la red neuronal. La columna

Fuente se refiere a los factores del diseño de experimentos, *SS* es la suma de cuadrados, *GL* son los grados de libertad, *MSS* es el cuadrado medio, *F* es el cociente entre el *MSS* de cada factor y el del error, *P* es el nivel de significancia.

También se presentan diversas gráficas. El *Diagrama de Pareto* presenta los efectos estandarizados que produce cada factor sobre el error de aproximación de la red, básicamente informa de manera visual cuáles factores son realmente significativos en la variación del MSE. La *Gráfica de efectos principales* muestra las tendencias, o comportamiento, del MSE respecto a la variación de factores en la red neuronal; puede servir para dar una idea de cuáles serían los factores para una aproximación óptima por parte de la red neuronal. La *Gráfica de interacción*, al igual que la de efectos principales, muestra las tendencias del MSE pero esta vez interactuando la modificación de los tres factores en la red neuronal.

Se presentan las gráficas de la prueba de validación de las redes neuronales, éstas gráficas para cada variable corresponden a los datos resaltados en negro en las tabas de diseño de experimentos.

7.2. Red NARX prealimentada

■ Simulación

En la Tabla 7.2 se muestran los resultados obtenidos de aplicar los datos de simulación (Figura 6.7) a la red NARX MLP. La respuesta es el error de aproximación de la red (MSE) para cada variable. Se realiza un diseño experimental factorial, 2^3 , los datos para realizar este diseño son los totales de las cuatro réplicas, se consideró cada variable como una réplica. Los factores para el diseño de experimentos son: retrasos en la entrada y salida del sistema ($n = m$), número de neuronas ocultas (N_h) y factor de aprendizaje (η), a lo largo de dos niveles. En la Tabla 7.3 se puede observar el análisis de variancia para la red NARX prealimentada con datos de simulación.

En la Figura 7.1 se muestran los efectos estandarizados, de cada factor, al error de aproximación. Las tendencias de los efectos principales del diseño experimental se muestran en la Figura 7.2. En la Figura 7.3 se muestra la gráfica de interacciones de los factores del diseño experimental y sus respectivas tendencias.

Tabla 7.2: Datos del diseño de experimentos MLP (simulación).

Respuesta:		Factor B: N_h			
MSE		10		50	
Factor C:		Factor A: η		Factor A: η	
$m = n$		0.1	0.5	0.1	0.5
2	θ	8.56E-06	9.86E-06	1.27E-05	1.52E-05
	x	3.68E-06	3.01E-06	1.08E-05	7.11E-06
	$\dot{\theta}$	3.75E-04	1.23E-04	8.70E-03	1.20E-03
	\dot{x}	4.61E-05	9.73E-05	2.20E-03	5.91E-04
4	θ	2.60E-05	2.60E-05	4.55E-05	3.42E-05
	x	8.57E-06	8.65E-06	1.47E-05	1.17E-05
	$\dot{\theta}$	7.97E-04	7.32E-04	3.30E-03	2.20E-03
	\dot{x}	2.54E-04	2.54E-04	9.75E-04	7.11E-04

Tabla 7.3: Análisis de variancia para MLP (simulación).

Fuente	SS	GL	MSS	F	P
A	3.61E-06	1	3.61E-06	2.01	0.171
B	9.30E-06	1	9.30E-06	5.17	0.034
C	5.00E-07	1	5.00E-07	0.28	0.603
AB	3.27E-06	1	3.27E-06	1.81	0.192
AC	1.93E-06	1	1.93E-06	1.07	0.312
BC	1.48E-06	1	1.48E-06	0.82	0.375
ABC	1.80E-06	1	1.80E-06	1	0.328
Error residual	3.78E-05	21	1.80E-06		
Total	8.47E-05	31			

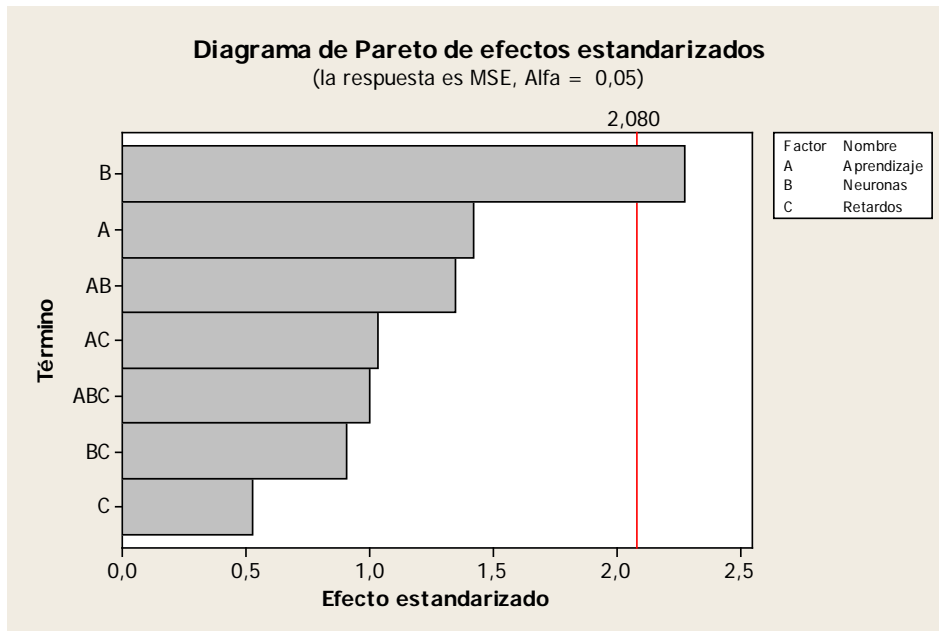


Figura 7.1: Efectos ordenados para el diseño 2^3 MLP (simulación).

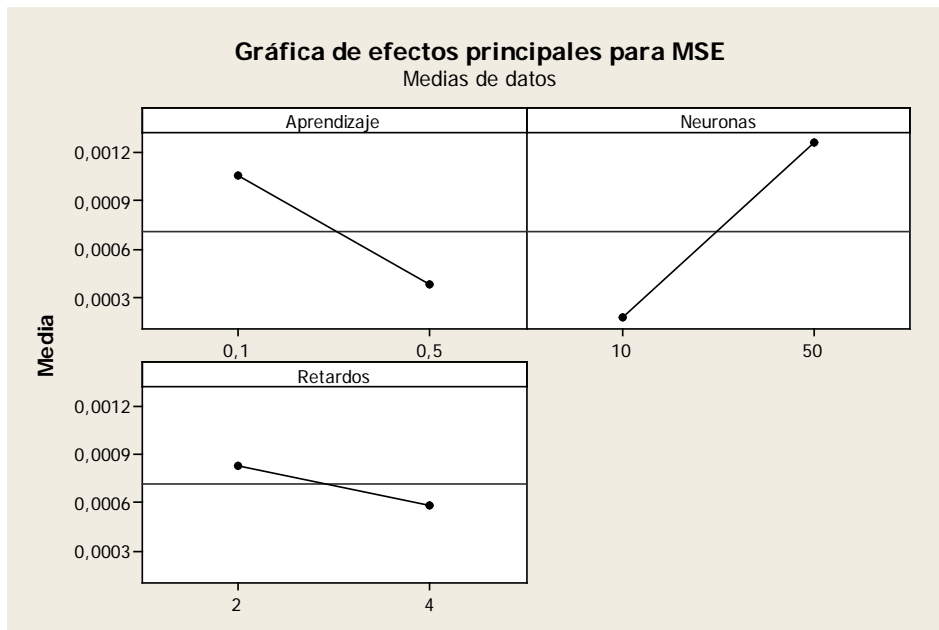


Figura 7.2: Comportamiento de los efectos principales, MLP (simulación).

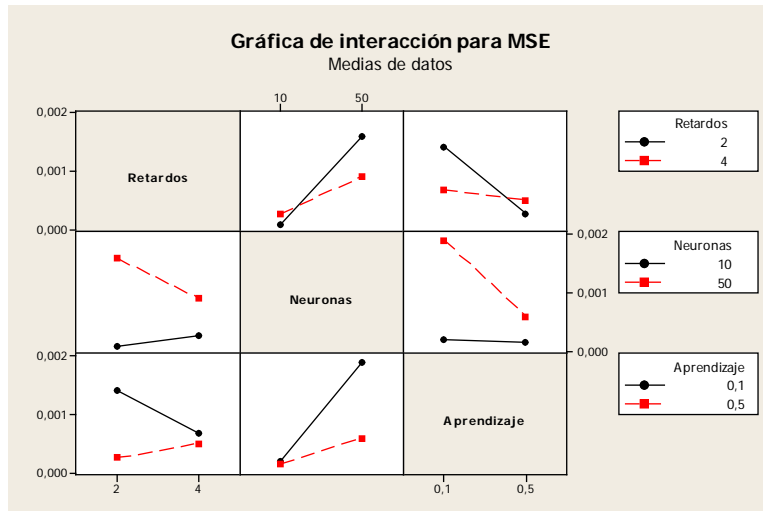


Figura 7.3: Comportamiento de las interacciones, MLP (simulación).

En la Tabla 7.4 se muestran los recursos computacionales requeridos por la red (para los datos resaltados en la Tabla 7.2). En la Tabla 7.5 se muestra el coeficiente de correlación Pearson entre cada variable real y cada variable estimada.

Tabla 7.4: Recursos de cómputo para MLP simulación.

Número de	MLP (simulación)
Neuronas ocultas	10
Pesos	50
Retrasos totales	4
Tanh	10

Tabla 7.5: Correlación MLP simulación

Variable	MLP
θ	0.9826
x	0.9849
$\dot{\theta}$	0.9957
\dot{x}	0.9968

En las Figuras 7.4, 7.5, 7.6 y 7.7 se muestran las gráficas para los datos resaltados en la Tabla 7.2. Se muestran los últimos 15 segundos del experimento, equivalentes a la prueba de validación de la red.

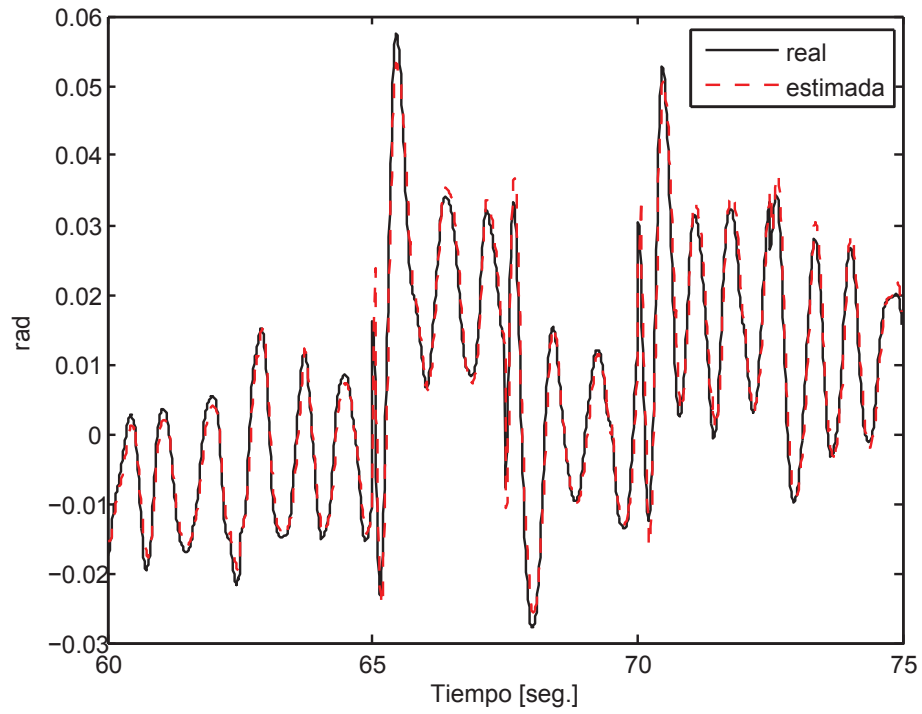


Figura 7.4: Posición angular del péndulo, MLP (simulación).

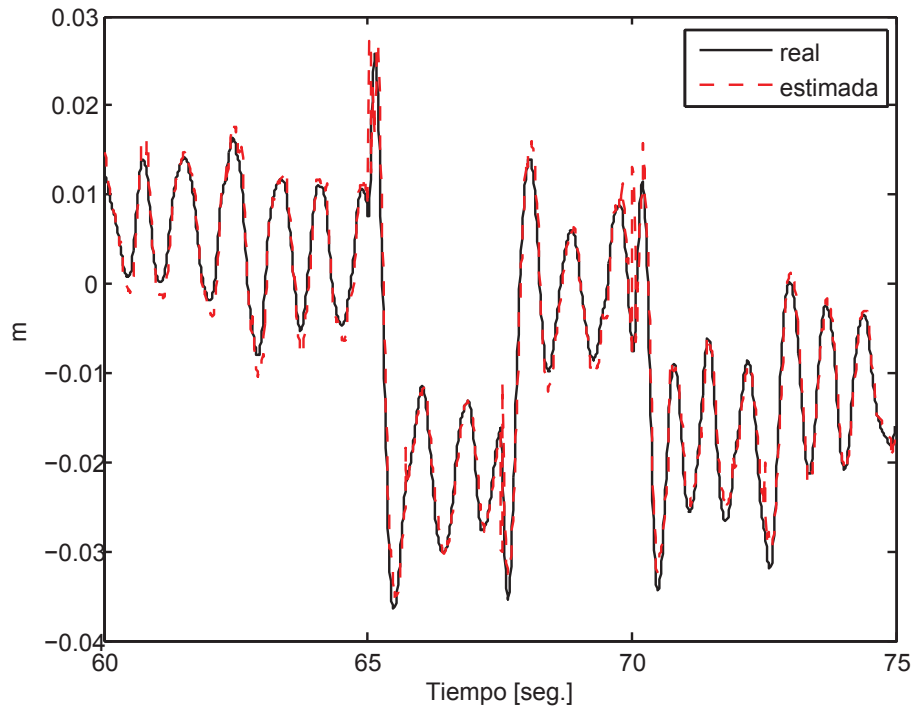


Figura 7.5: Desplazamiento del carro, MLP (simulación).

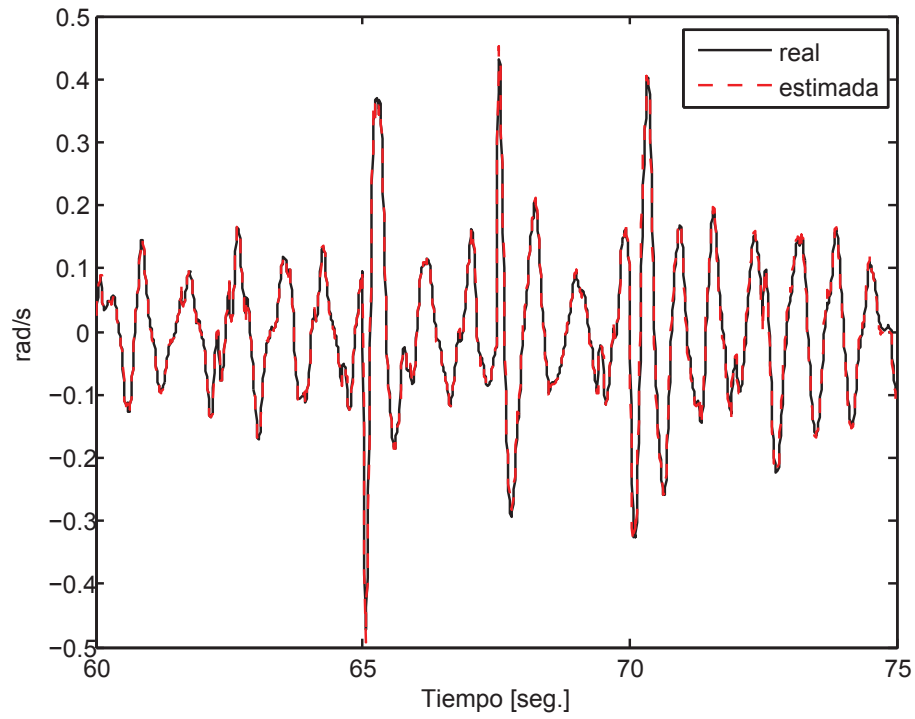


Figura 7.6: Velocidad angular del péndulo, MLP (simulación).

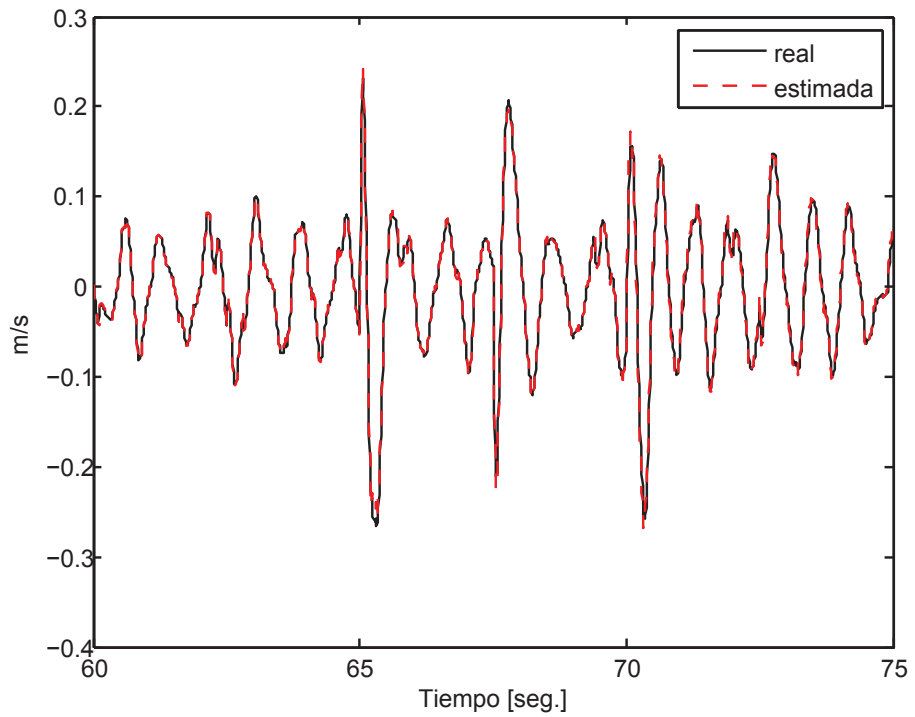


Figura 7.7: Velocidad lineal del carro, MLP (simulación).

- Experimento

En la Tabla 7.6 se muestran los datos de error de aproximación para la prueba de validación de la red, estos corresponden a los últimos 15 segundos del experimento. Los valores resaltados son los que se consideran como la mejor aproximación en relación a los recursos computacionales requeridos por la red.

Tabla 7.6: Datos del diseño de experimentos MLP (experimento).

Respuesta:		Factor B: N_h			
MSE		10		50	
Factor C:		Factor A: η		Factor A: η	
$m = n$		0.1	0.5	0.1	0.5
2	θ	7.86E-07	4.57E-07	2.02E-06	9.27E-07
	x	3.00E-06	4.69E-05	3.83E-04	4.62E-05
	$\dot{\theta}$	2.91E-06	2.88E-06	3.53E-06	3.12E-06
	\dot{x}	3.37E-04	3.47E-04	5.62E-04	4.90E-04
4	θ	1.29E-06	1.14E-06	3.40E-06	2.79E-06
	x	1.49E-05	2.24E-05	2.66E-05	1.33E-04
	$\dot{\theta}$	7.34E-06	7.56E-06	8.09E-06	7.53E-06
	\dot{x}	4.61E-04	5.14E-04	5.68E-04	7.28E-04

Se realizó el diseño de experimentos del cual se obtuvieron los datos presentados en el análisis de variancia de la Tabla 7.7.

Se puede observar que el error residual no es grande y que el factor que más efecto tiene sobre la variación del MSE de la aproximación es el número de neuronas en la capa oculta (N_h), esto se puede visualizar en la Figura 7.8. En la Figura 7.9 se presenta una gráfica de tendencias de los efectos principales del diseño experimental. En la Figura 7.10 se muestran las tendencias de las interacciones de los factores. En términos generales el MLP sigue tendencias muy similares tanto para datos de simulación como para datos adquiridos de un experimento real.

Tabla 7.7: Análisis de variancia para MLP (experimento).

Fuente	SS	GL	MSS	F	P
A	0.00E+00	1	0.00E+00	0.00	0.947
B	4.00E-08	1	4.00E-08	6.51	0.019
C	0.00E+00	1	0.00E+00	0.34	0.565
AB	0.00E+00	1	0.00E+00	0.30	0.588
AC	1.00E-08	1	1.00E-08	2.11	0.161
BC	0.00E+00	1	0.00E+00	0.42	0.526
ABC	1.00E-08	1	1.00E-08	2.02	0.170
Error residual	1.40E-07	21	1.00E-08		
Total	1.59E-06	31			

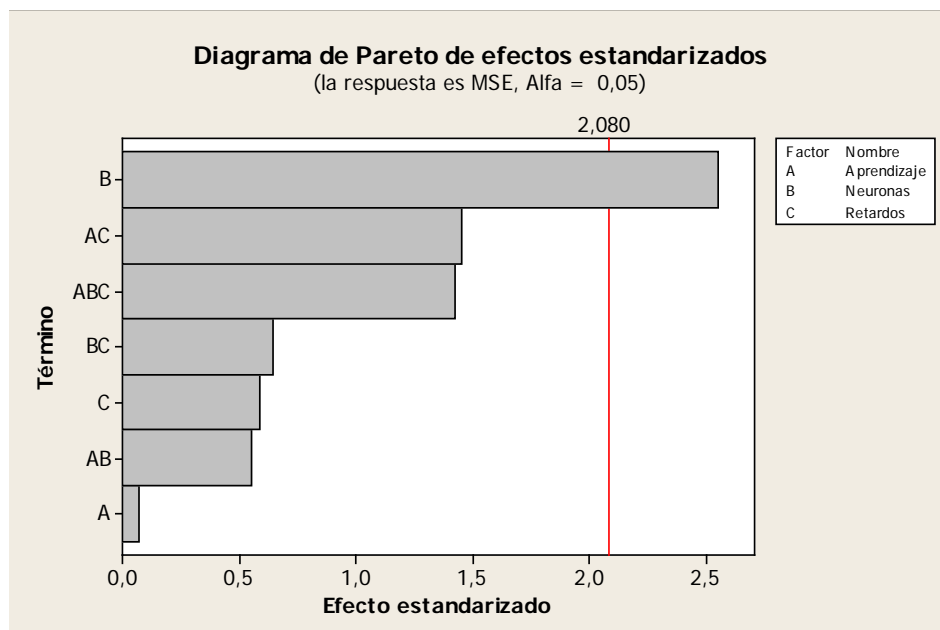


Figura 7.8: Efectos ordenados para el diseño 2^3 MLP (experimento).

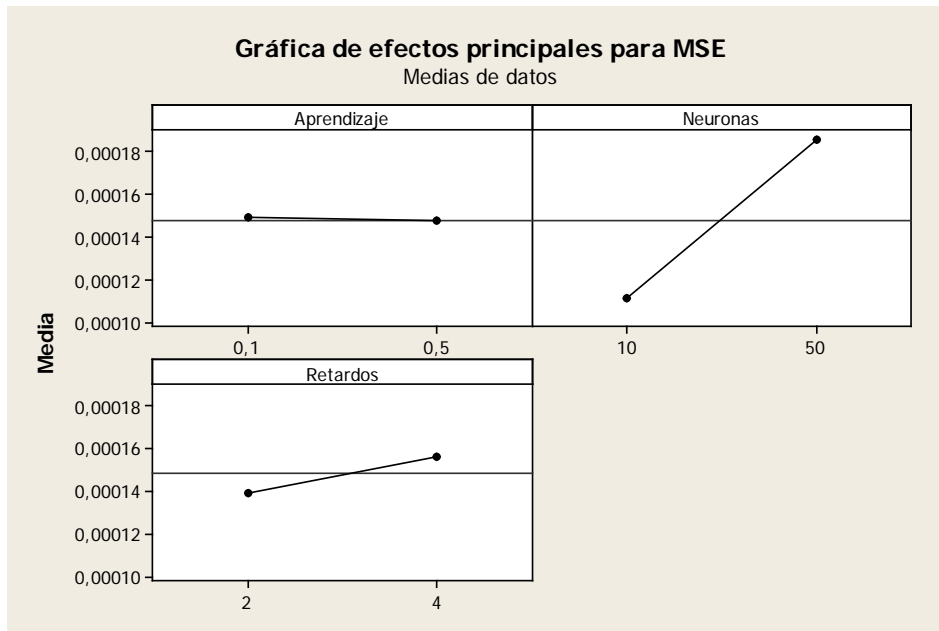


Figura 7.9: Comportamiento de los efectos principales, MLP (experimento).

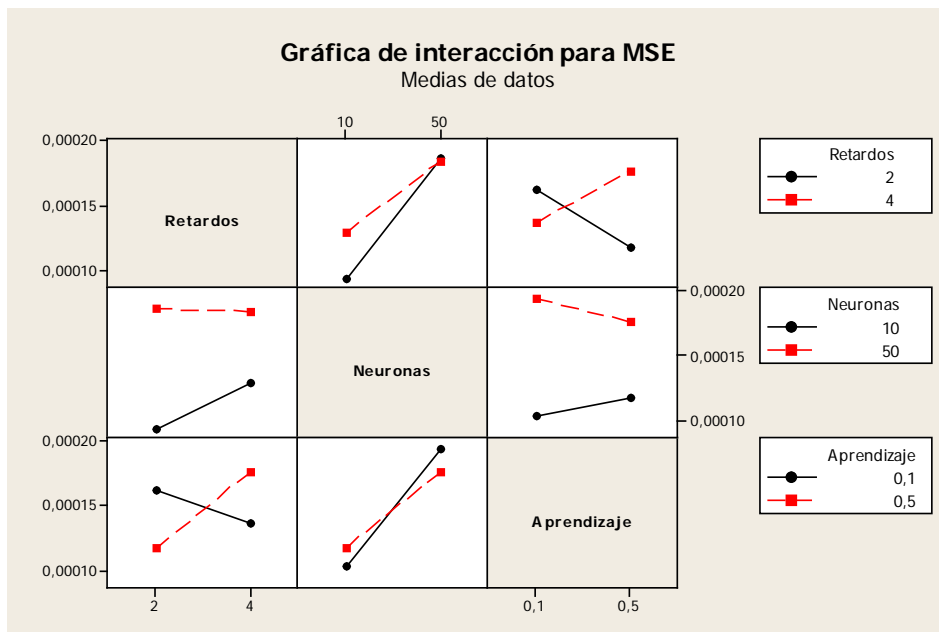


Figura 7.10: Comportamiento de las interacciones, MLP (experimento).

En la Tabla 7.8 se muestran los recursos computacionales requeridos por la red (para los datos resaltados en la Tabla 7.6). En la Tabla 7.9 se muestra el coeficiente de correlación Pearson entre cada variable real y cada variable estimada.

Tabla 7.8: Recursos de cómputo para MLP experimento.

Número de	MLP (simulación)
Neuronas ocultas	10
Pesos	50
Retrasos totales	4
Tanh	10

Tabla 7.9: Correlación MLP experimento

Variable	MLP
θ	0.9963
x	0.9989
$\dot{\theta}$	0.9167
\dot{x}	0.9602

En las Figuras 7.11, 7.12, 7.13 y 7.14 se muestran las gráficas para la mejor aproximación con el MLP, en términos del MSE, para cada variable del péndulo (resaltado en la Tabla 7.6). En la sección 7.5 se presenta un análisis de resultados mas detallado.

■ Observaciones:

1. De las Figuras 7.1 y 7.8 se observa que el número de neuronas en la capa oculta es lo que más genera variación en el error de aproximación.
2. De las Tablas 7.2 y 7.6 se observa que entre más neuronas ocultas no mejora la aproximación sino que empeora, esto se atribuye a errores de cálculo numérico.
3. De acuerdo con (Haykin, 1999, p. 230) el número mínimo necesario de muestras de entrenamiento para esta red es de 5200 (cuando se utilizan 10 neuronas ocultas), se eligió 6000 mil para cumplir cuando la red utiliza 50 neuronas ocultas.

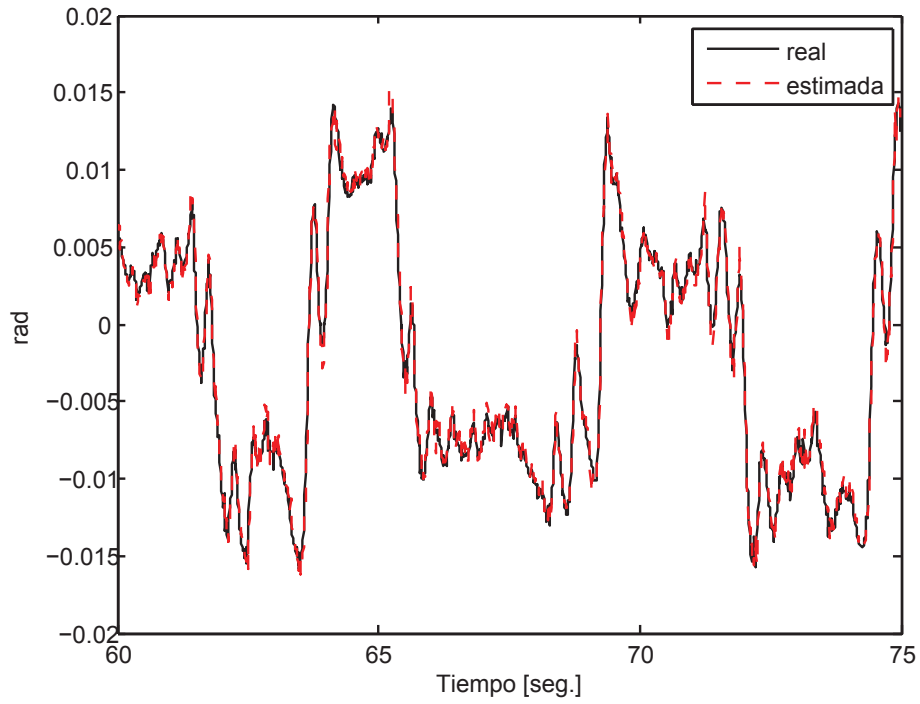


Figura 7.11: Posición angular del péndulo, MLP (experimento).

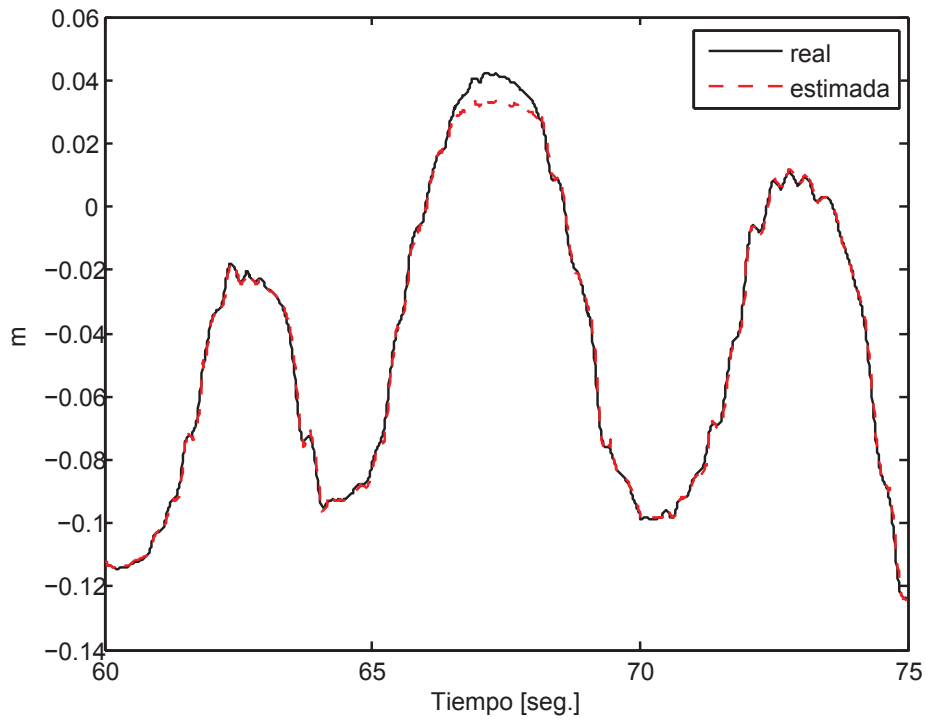


Figura 7.12: Desplazamiento del carro, MLP (experimento).

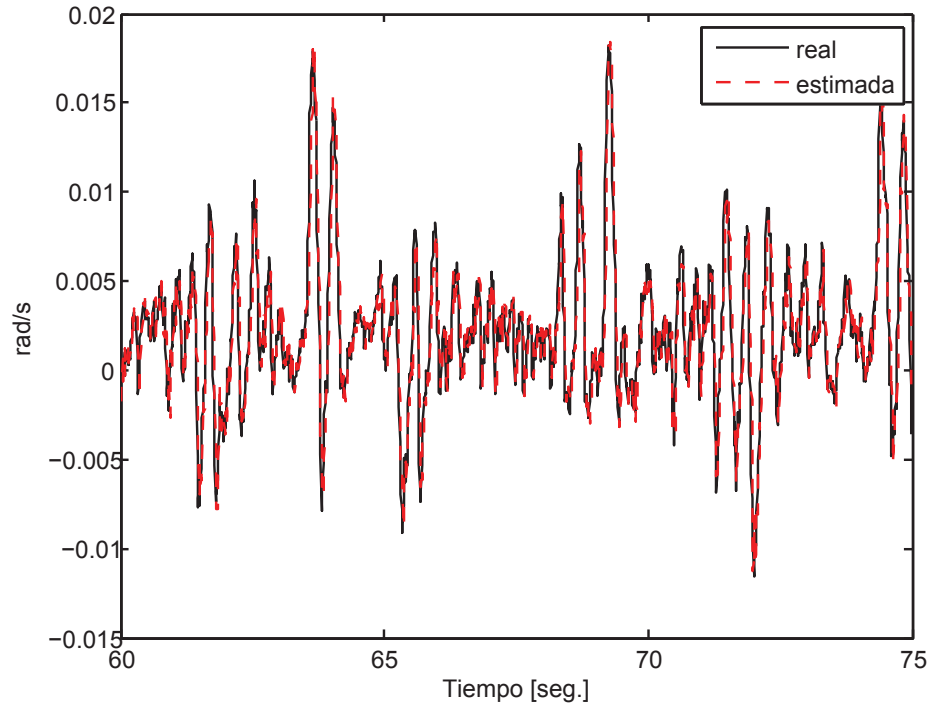


Figura 7.13: Velocidad angular del péndulo, MLP (experimento).

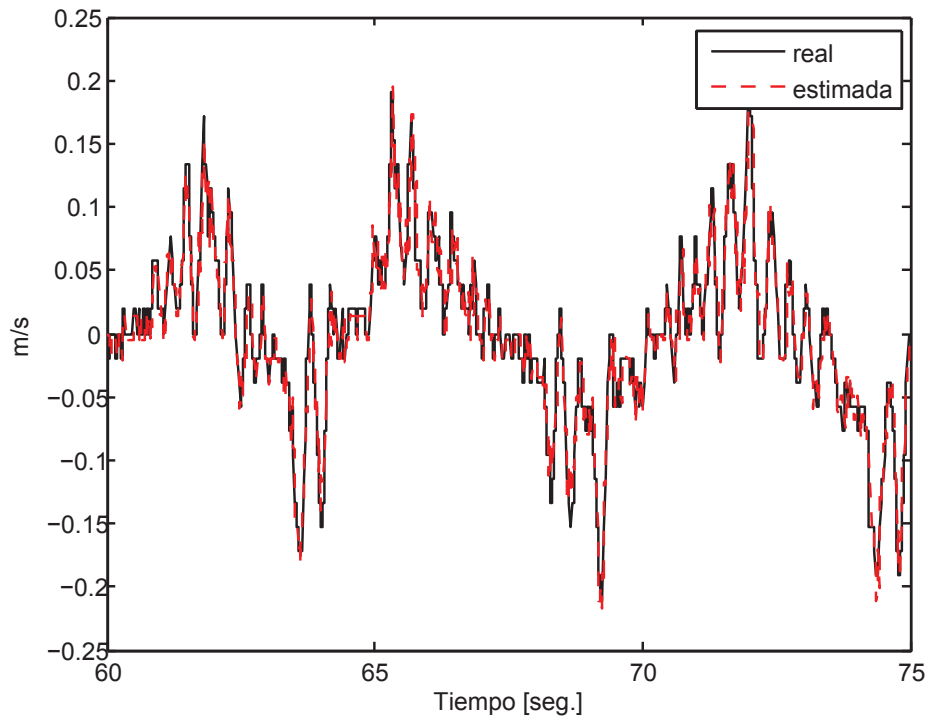


Figura 7.14: Velocidad lineal del carro, MLP (experimento).

7.3. Red con polinomio de Volterra como función base

- Simulación

Para este tipo de red también se realizó un diseño de experimentos 2^3 , los factores que influyen en la red VPBF son: los retrasos de E/S ($n = m$), el orden del polinomio (l) y el factor de olvido (λ). Se tomaron los datos presentados en la Figura 6.7 para realizar la prueba de red. En la Tabla 7.10, se muestran los resultados en función del MSE para la red VPBF al aplicarle los datos de simulación del PISC, los valores resaltados son los que presentan una mejor aproximación con menor uso de recursos de cómputo. Cada variable se considera una réplica, pues lo que se evalúa es la red neuronal no el PISC.

Tabla 7.10: Datos del diseño de experimentos VPBF (simulación).

Respuesta:		Factor B: l			
MSE		2		3	
Factor C:		Factor A: λ		Factor A: λ	
$m = n$		0.990	0.999	0.990	0.999
2	θ	6.68E-06	3.00E-06	1.13E-04	2.97E-06
	x	1.87E-07	1.04E-06	1.58E-07	9.60E-07
	$\dot{\theta}$	1.35E-05	6.85E-05	8.95E-01	3.69E-05
	\dot{x}	1.22E-04	2.01E-05	9.11E-01	9.45E-06
4	θ	9.83E-01	2.32E-06	1.02E+00	1.11E-04
	x	2.00E-03	9.25E-07	1.02E+00	7.23E-06
	$\dot{\theta}$	1.00E+00	7.06E-05	1.04E+00	7.08E-04
	\dot{x}	1.01E+00	6.34E-05	1.03E+00	2.10E-03

De la Tabla 7.10 se observa un compromiso entre los factores, cuando se tienen dos retardos de entrada-salida, para un orden polinomial dos los resultados son mejores si se utiliza un factor de olvido bajo, pero, si se utiliza un orden polinomial tres los resultados mejoran al utilizar un factor de olvido alto. Por otra parte, cuando se tienen cuatro retardos de entrada-salida, para un orden polinomial indistinto los resultados mejoran al utilizar un factor de olvido alto.

De la Tabla 7.11 y la Figura 7.15 se pueden observar los efectos de cada factor de diseño experimental.

Tabla 7.11: Análisis de variancia para red VPBF (simulación).

Fuente	SS	GL	MSS	F	P
A	2.48E+00	1	2.48E+00	41.93	0.000
B	2.67E-01	1	2.67E-01	4.52	0.046
C	8.78E-01	1	8.78E-01	14.85	0.001
AB	2.66E-01	1	2.66E-01	4.50	0.046
AC	8.76E-01	1	8.76E-01	14.81	0.001
BC	1.48E-02	1	1.48E-02	0.25	0.622
ABC	1.51E-02	1	1.51E-02	0.25	0.619
Error residual	1.24E+00	21	5.91E-02		
Total	6.35E+00	31			

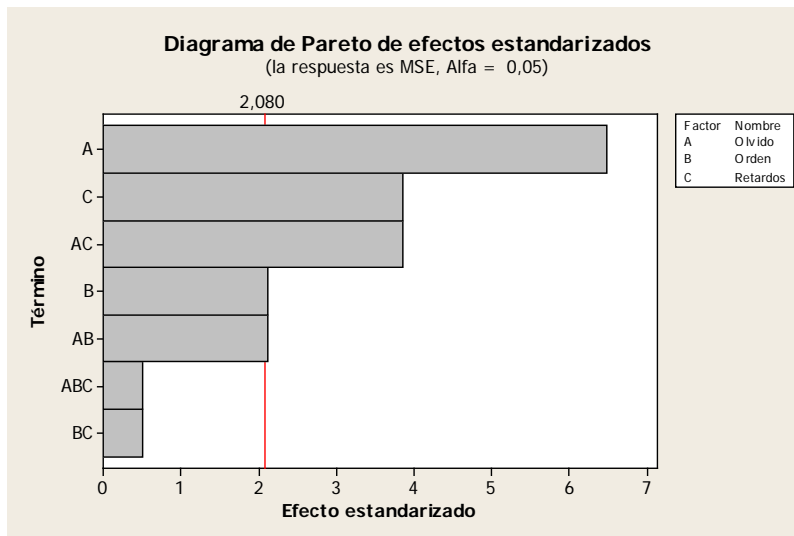


Figura 7.15: Efectos ordenados para el diseño 2^3 VPBF (simulación).

Se determinó que los factores que mas influyen en el MSE de la aproximación son: el número de retardos de entrada - salida, el factor de olvido y la interacción de ambos. Es de notarse que también el orden del polinomio y la interacción olvido - orden son significativos, aunque tienen un efecto menor.

En las Figuras 7.16 y 7.17 se muestran las gráficas de tendencia para los factores principales y las interacciones del diseño experimental de esta red.

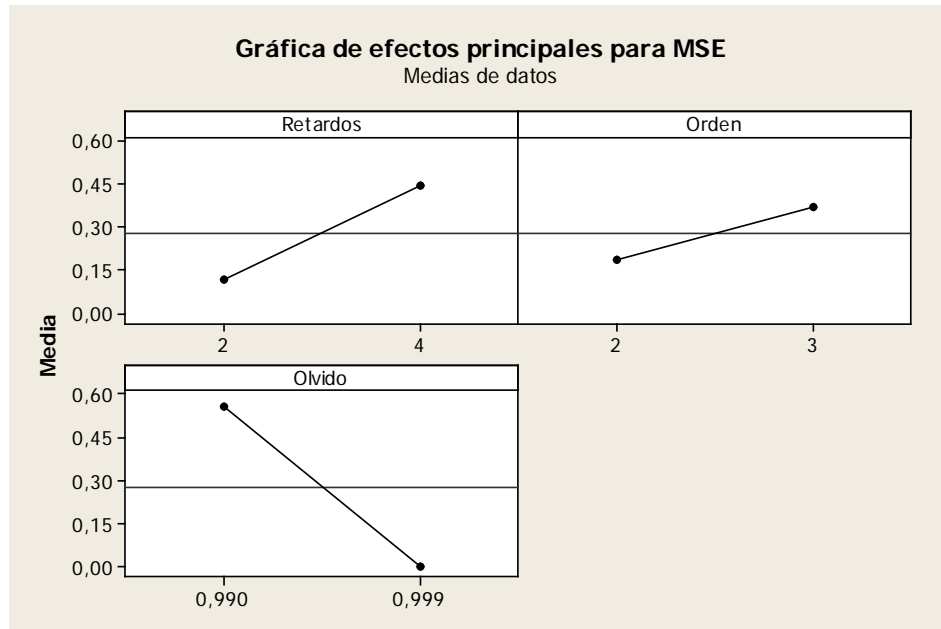


Figura 7.16: Comportamiento de los efectos principales, VPBF (simulación).

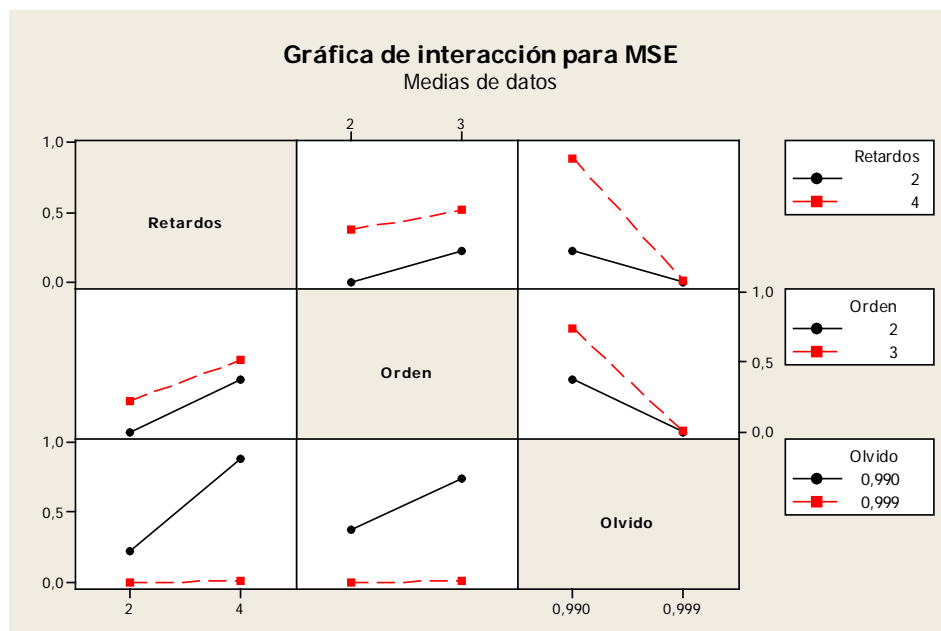


Figura 7.17: Comportamiento de las interacciones, VPBF (simulación).

En la Tabla 7.12 se muestran los recursos computacionales requeridos por la red (para los datos resaltados en la Tabla 7.10).

Tabla 7.12: Recursos de cómputo para red VPBF, simulación.

Número de	VPBF (simulación)
Funciones base	15
Pesos	15
Retrasos totales	4
Tanh	1

En la Tabla 7.13 se muestra el coeficiente de correlación Pearson entre cada variable real y cada variable estimada.

Tabla 7.13: Correlación red VPBF, simulación

Variable	VPBF
θ	0.9883
x	0.9995
$\dot{\theta}$	0.9997
\dot{x}	0.9894

Como se puede observar al compararlas Tablas 7.4 y 7.12 la red multicapa requiere de una complejidad de cómputo mayor que la polinomial. Además, al comparar las Tablas 7.5 y 7.13, la red polinomial propuesta con funciones base de Volterra tiene una mejor aproximación, en términos de error instantáneo, aunque hay que resaltar que la velocidad lineal, para los datos de simulación, es aproximada mejor por la red MLP.

En las Figuras 7.18, 7.19, 7.20 y 7.21 se muestran las gráficas para la mejor aproximación con la red VPBF, en términos del error cuadrático, para cada variable del péndulo, resaltado en la Tabla 7.10. Únicamente se grafican los resultados de la prueba de validación de la red, equivalente a los últimos 15 segundos de los datos adquiridos. Se utiliza exactamente la misma relación de entrenamiento - validación que en la red MLP.

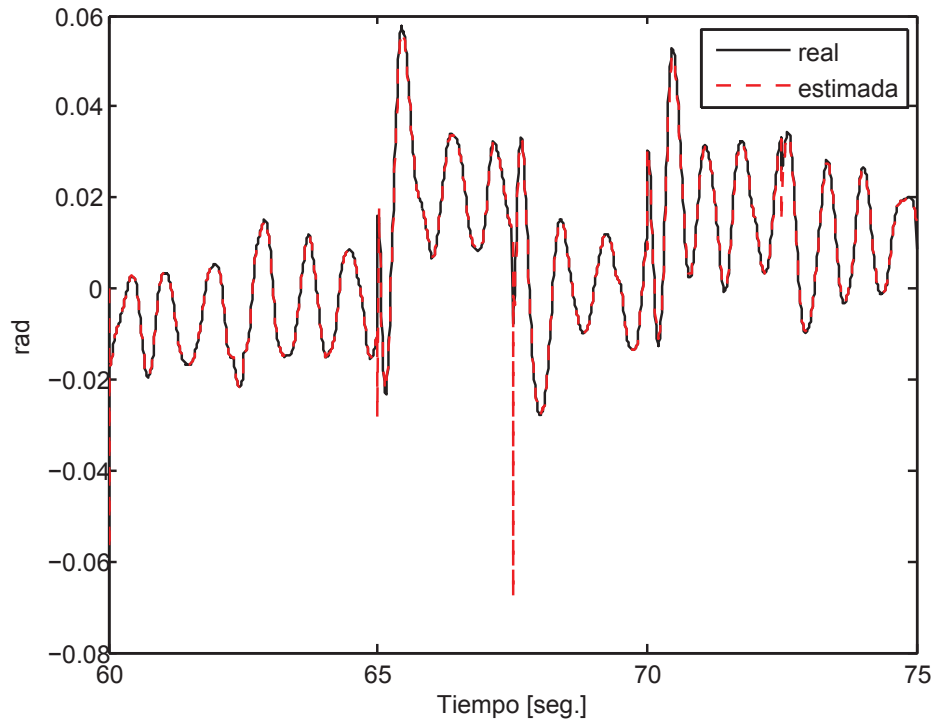


Figura 7.18: Posición angular del péndulo, VPBF (simulación).

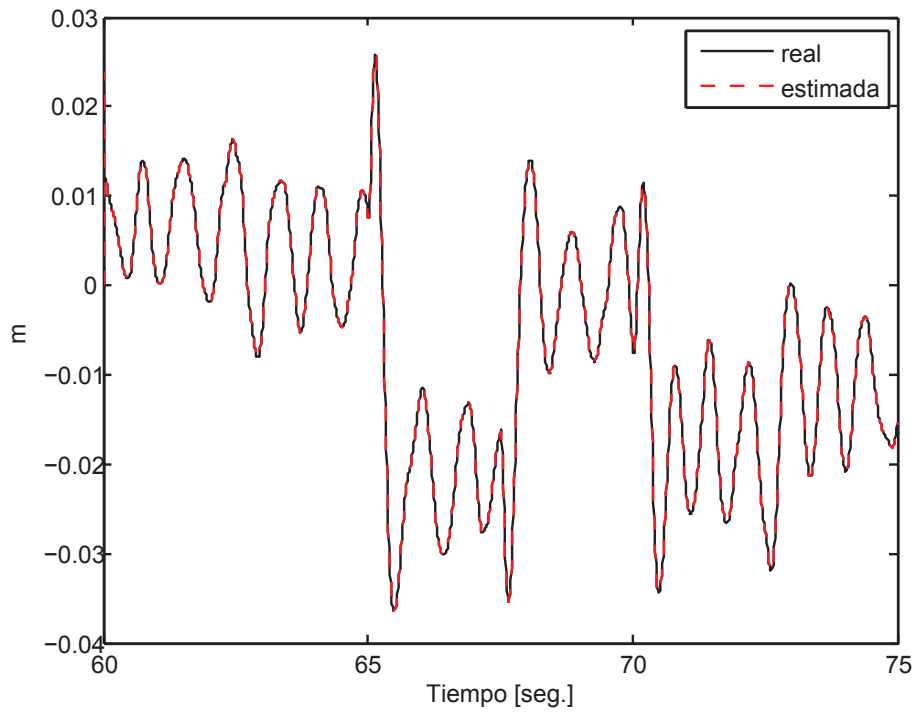


Figura 7.19: Desplazamiento del carro, VPBF (simulación).

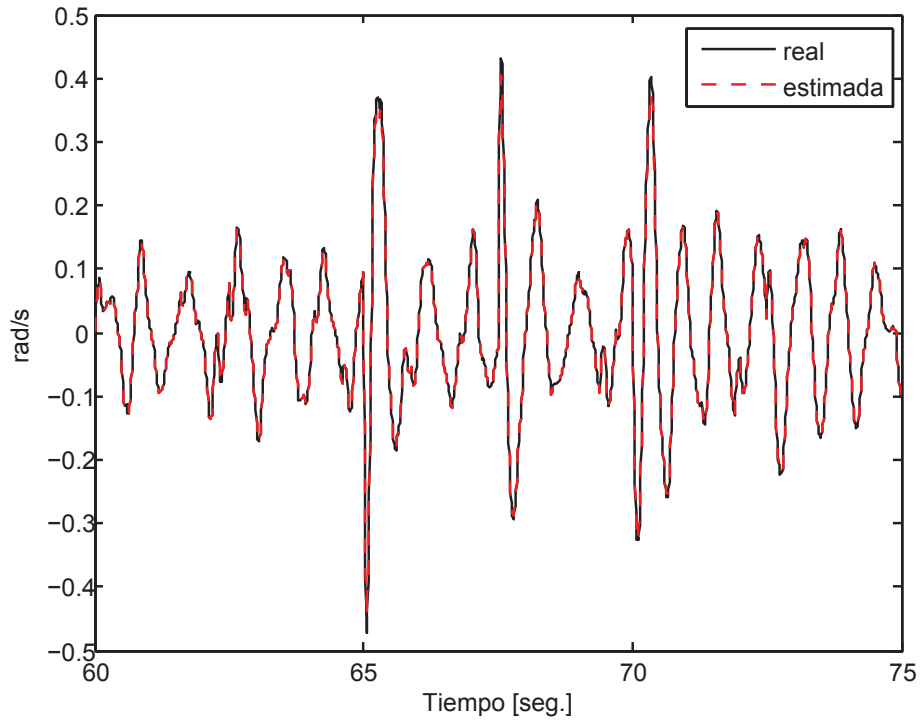


Figura 7.20: Velocidad angular del péndulo, VPBF (simulación).

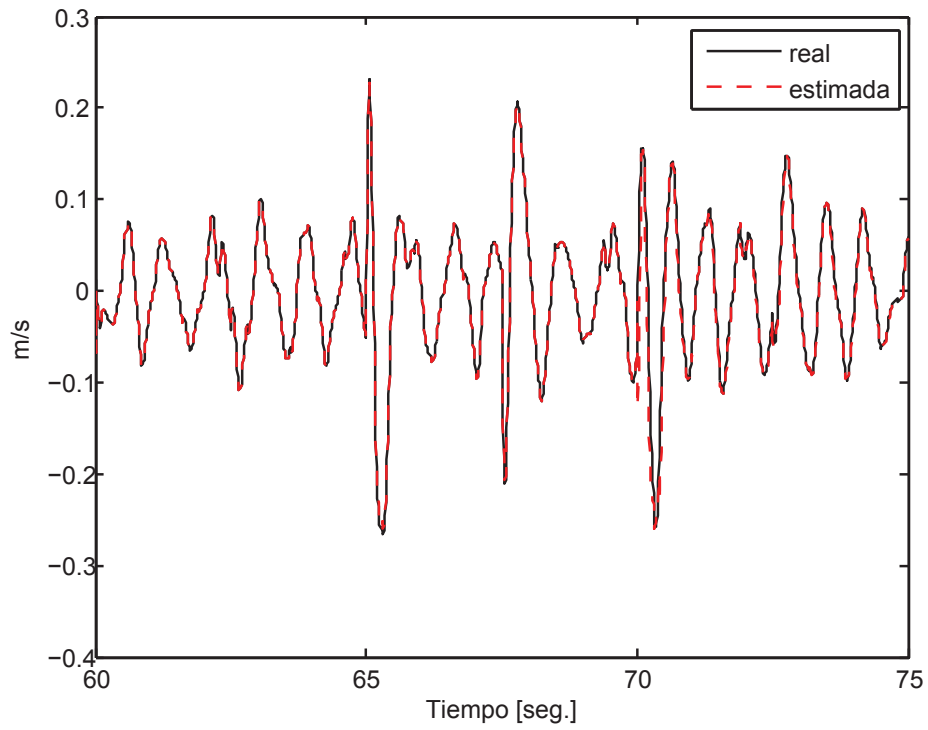


Figura 7.21: Velocidad lineal del carro, VPBF (simulación).

■ Experimento

En la Tabla 7.14 se muestran los datos de error de aproximación para la prueba de validación de la red VPBF. Los valores resaltados son los que se consideran como la mejor aproximación en relación a los recursos computacionales requeridos por la red. Los factores utilizados para el diseño de experimentos son los mismos que se utilizaron en la simulación.

Tabla 7.14: Datos del diseño de experimentos VPBF (experimento).

Respuesta:		Factor B: l			
MSE		2		3	
Factor C:		Factor A: λ		Factor A: λ	
$m = n$		0.990	0.999	0.990	0.999
2	θ	3.11E-07	2.37E-07	1.21E-06	2.44E-07
	x	1.54E-05	1.03E-05	1.06E+00	1.03E-05
	$\dot{\theta}$	9.24E-07	1.47E-06	1.09E-06	1.46E-06
	\dot{x}	1.96E-04	1.83E-04	1.99E-04	1.81E-04
4	θ	4.26E-07	3.05E-07	3.62E-05	2.87E-07
	x	8.66E-04	1.04E-05	1.04E+00	1.08E-05
	$\dot{\theta}$	9.86E-07	1.28E-06	2.94E-06	1.24E-06
	\dot{x}	1.00E+00	1.64E-04	1.03E+00	1.54E-04

Se realizó el diseño de experimentos del cual se obtuvieron los datos presentados en el análisis de variancia de la Tabla 7.15.

En la Figura 7.22 se puede observar que el factor que más efecto tiene sobre la variación del MSE de la aproximación es el factor de olvido. Así como en la Figura 7.15, correspondiente a la simulación, se puede observar que el factor de olvido es lo que más influencia tiene sobre el error de aproximación de la red. Lo antes mencionado no implica una influencia positiva o negativa sino más bien una variación en el error de media cuadrática.

En la Figura 7.23 se presenta una gráfica de tendencia de los efectos principales del diseño experimental. En la Figura 7.24 se muestran las tendencias de las interacciones de los factores.

Tabla 7.15: Análisis de variancia para red VPBF (experimento).

Fuente	SS	GL	MSS	F	P
A	5.35E-01	1	5.35E-01	5.26	0.032
B	1.43E-01	1	1.43E-01	1.40	0.250
C	1.27E-01	1	1.27E-01	1.24	0.277
AB	1.43E-01	1	1.43E-01	1.40	0.250
AC	1.27E-01	1	1.27E-01	1.24	0.277
BC	0.00E+00	1	3.00E-06	0.00	0.996
ABC	0.00E+00	1	3.00E-06	0.00	0.996
Error residual	2.14E+00	21	1.02E-01		
Total	3.75E+00	31			

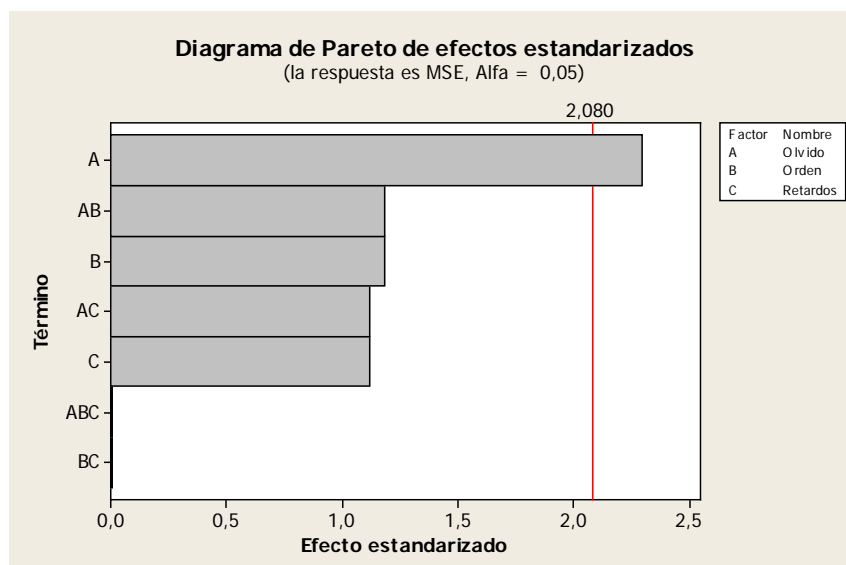


Figura 7.22: Efectos ordenados para el diseño 2^3 VPBF (experimento).

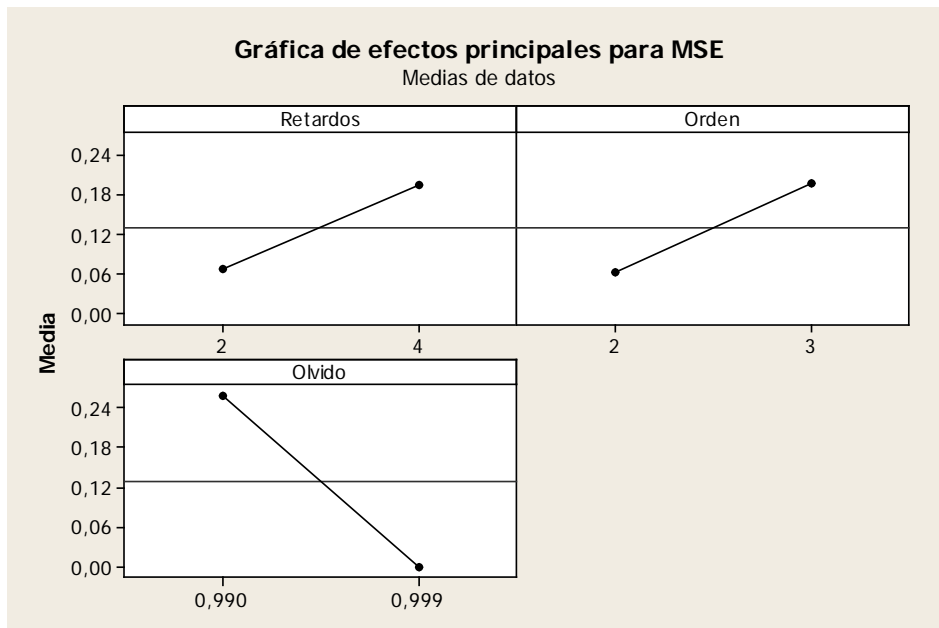


Figura 7.23: Comportamiento de los efectos principales, VPBF (experimento).

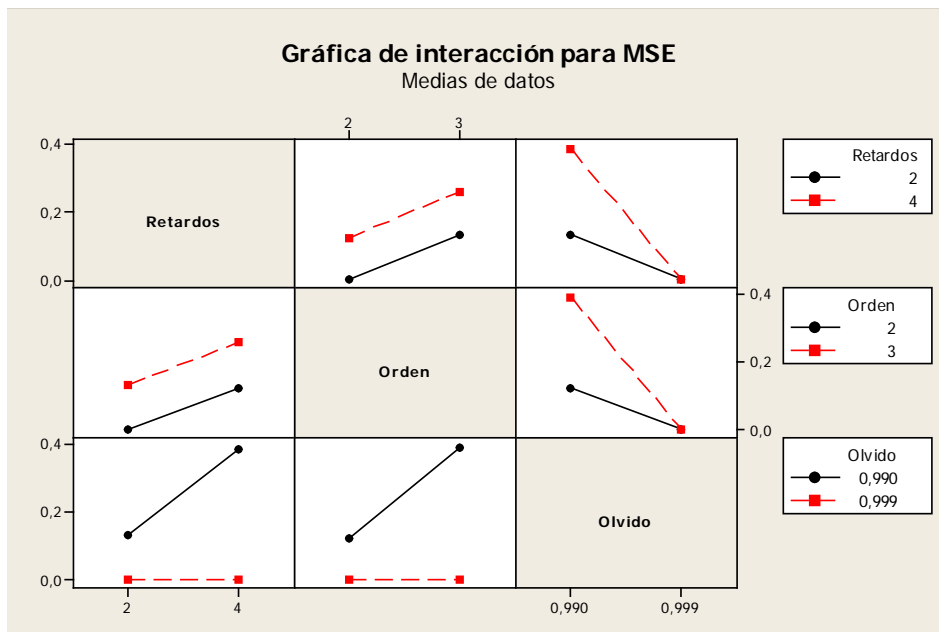


Figura 7.24: Comportamiento de las interacciones, VPBF (experimento).

Las Figuras 7.23 y 7.24 muestran las tendencias de variación del MSE al cambiar los valores de los parámetros importantes de la red compacto. Lo anterior es importante debido a que da una idea más clara de como elegir los parámetros para una implementación mejor.

En la Tabla 7.16 se muestran los recursos computacionales requeridos por la red (para los datos resaltados en la Tabla 7.14).

Tabla 7.16: Recursos de cómputo para red VPBF, experimento.

Número de	VPBF (experimento)
Funciones base	15
Pesos	15
Retrasos totales	4
Tanh	1

En la Tabla 7.17 se muestra el coeficiente de correlación Pearson entre cada variable real y cada variable estimada.

Tabla 7.17: Correlación red VPBF, experimento

Variable	VPBF
θ	0.9975
x	0.9968
$\dot{\theta}$	0.9735
\dot{x}	0.9792

Como se puede observar al compararlas Tablas 7.8 y 7.16 la red multicapa requiere de una complejidad de cómputo mayor que la polinomial. Además, al comparar las Tablas 7.9 y 7.17, la red polinomial propuesta con funciones base de Volterra sigue teniendo una mejor aproximación, aunque la posición lineal es aproximada un poco mejor por la red MLP.

En las Figuras 7.25, 7.26, 7.27 y 7.28 se muestran las gráficas para la mejor aproximación con la red VPBF, en términos del MSE, para cada variable del péndulo (resaltado en la Tabla 7.14).

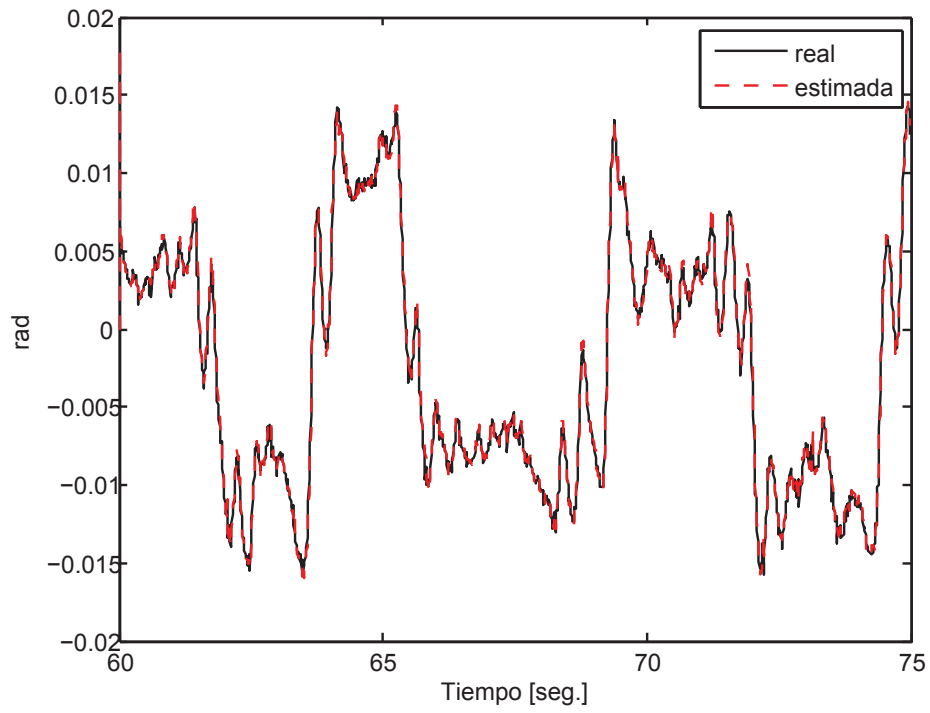


Figura 7.25: Posición angular del péndulo, VPBF (experimento).

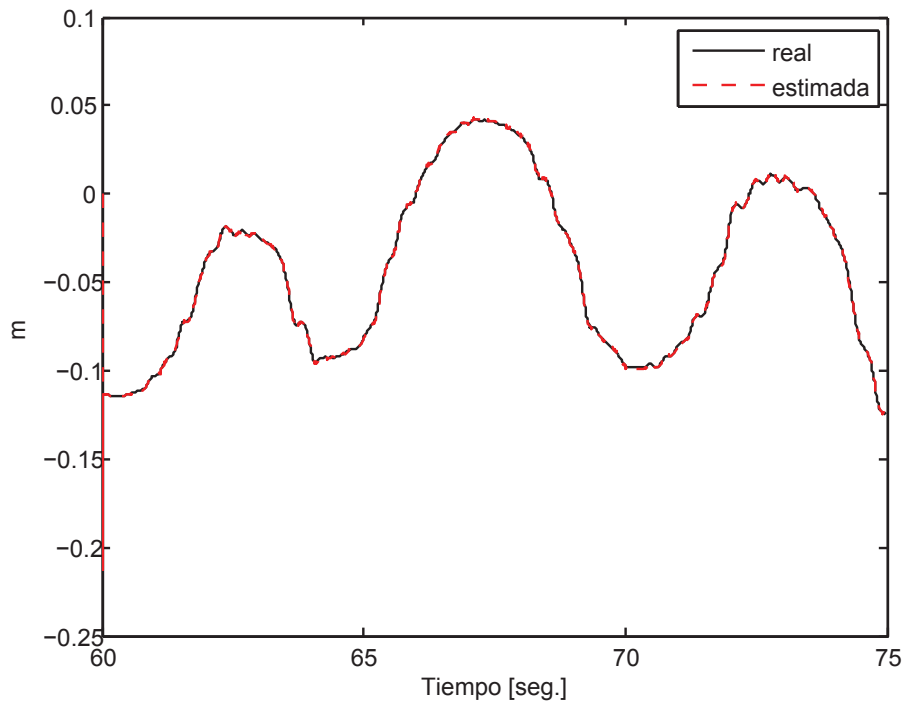


Figura 7.26: Desplazamiento del carro, VPBF (experimento).

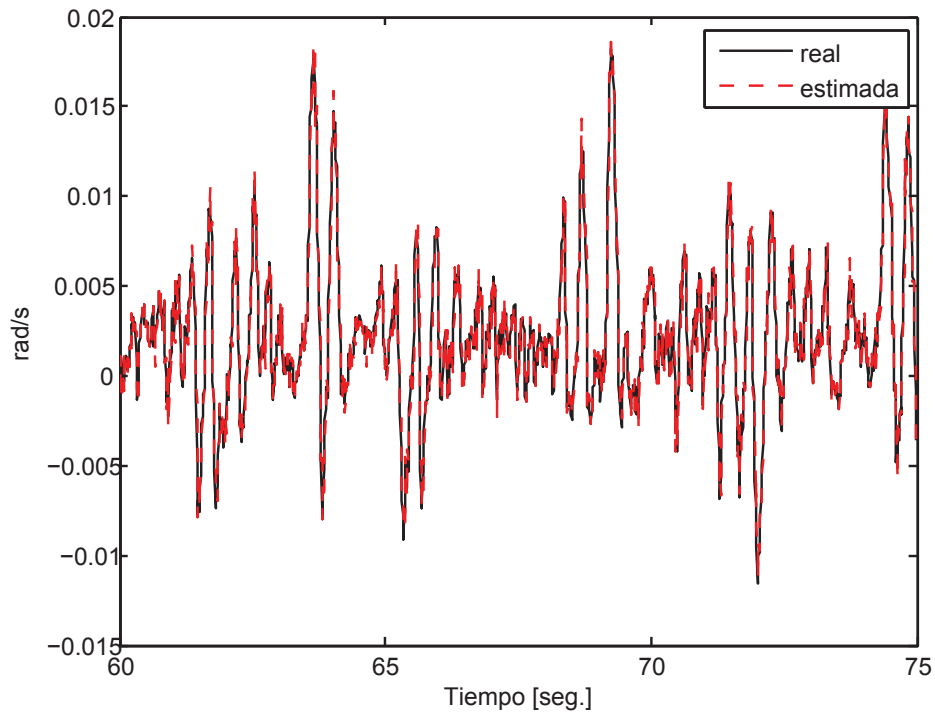


Figura 7.27: Velocidad angular del péndulo, VPBF (experimento).

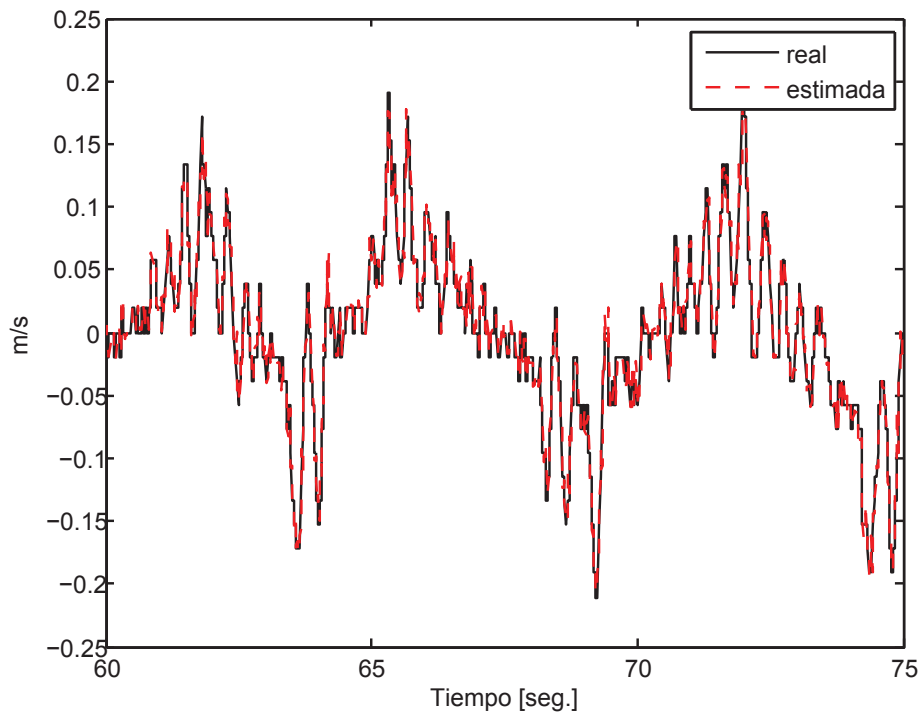


Figura 7.28: Velocidad lineal del carro, VPBF (experimento).

7.4. Red con polinomio de Chebyshev como función base

■ Simulación

Se realizó un diseño de experimentos 2^3 , los factores que influyen en la red CBF son: los retrasos de E/S ($n = m$), el orden del polinomio (l) y el factor de olvido (λ), los mismos que para la red VPBF. En la Tabla 7.18 se muestran los datos del error de aproximación de éste tipo de red, los valores resaltados son los que tienen una mejor aproximación tomando en cuenta los recursos de cómputo necesarios.

Tabla 7.18: Datos del diseño de experimentos CBF (simulación).

Respuesta:		Factor B: l			
MSE		2		3	
Factor C:		Factor A: λ		Factor A: λ	
$m = n$		0.990	0.999	0.990	0.999
2	θ	2.09E-06	2.97E-06	1.99E-06	1.75E-06
	x	1.89E-07	1.03E-06	1.90E-07	3.97E-07
	$\dot{\theta}$	1.31E-05	6.00E-05	2.25E-06	3.62E-04
	\dot{x}	1.72E-06	2.03E-05	1.70E-06	2.16E-05
4	θ	1.48E-06	2.22E-06	1.01E+00	1.34E-06
	x	3.14E-06	9.14E-07	1.01E+00	1.60E-07
	$\dot{\theta}$	3.48E-04	3.35E-05	2.37E-06	2.70E-03
	\dot{x}	1.80E-06	5.47E-05	2.21E-06	5.36E-05

En la Tabla 7.19 se muestra el análisis de variancia para el diseño experimental realizado.

A diferencia que en la red VPBF, se concluye que todos los factores y sus interacciones producen efectos prácticamente igual de significativos en la aproximación. Lo anterior se puede observar en la Figura 7.29.

En las Figuras 7.30 y 7.31 se muestran las gráficas de tendencias de los factores e interacciones del diseño experimental. Para los datos de simulación es difícil llegar a una conclusión clara de cómo afectan los factores a la variación del error.

Tabla 7.19: Análisis de variancia para red CBF (simulación).

Fuente	SS	GL	MSS	F	P
A	1.27E-01	1	1.27E-01	2.99	0.098
B	1.28E-01	1	1.28E-01	3.01	0.098
C	1.28E-01	1	1.28E-01	3.01	0.098
AB	1.27E-01	1	1.27E-01	2.99	0.099
AC	1.27E-01	1	1.27E-01	2.99	0.098
BC	1.28E-01	1	1.28E-01	3.00	0.098
ABC	1.27E-01	1	1.27E-01	2.99	0.098
Error residual	8.94E-01	21	4.26E-02		
Total	1.91E+00	31			

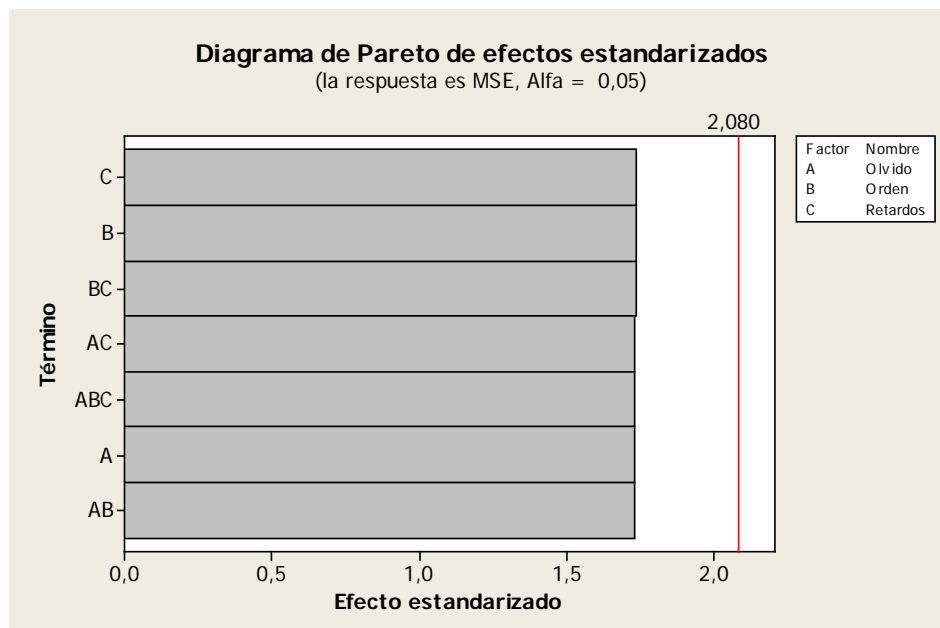


Figura 7.29: Efectos ordenados para el diseño 2^3 CBF (simulación).

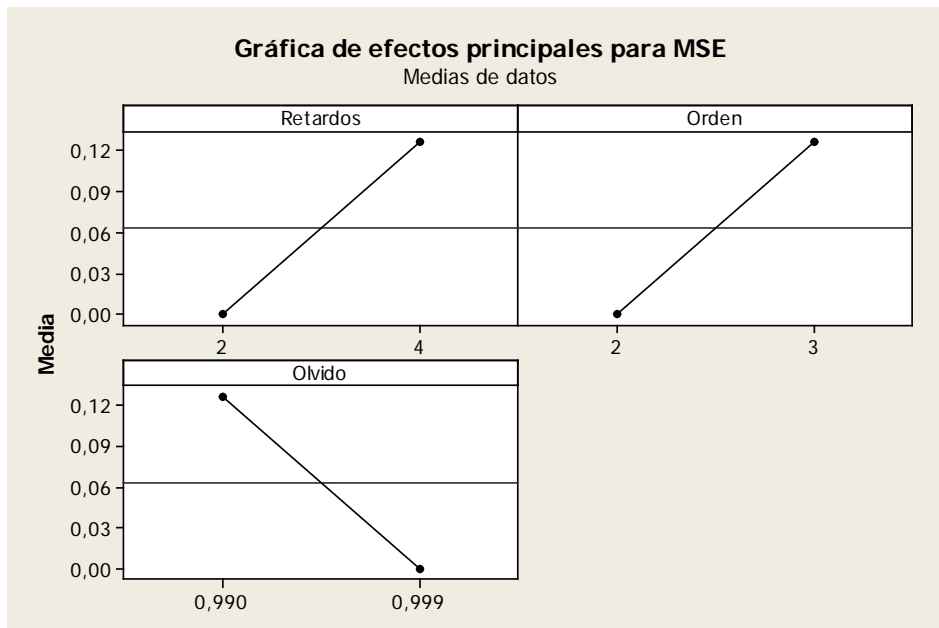


Figura 7.30: Comportamiento de los efectos principales, CBF (simulación).

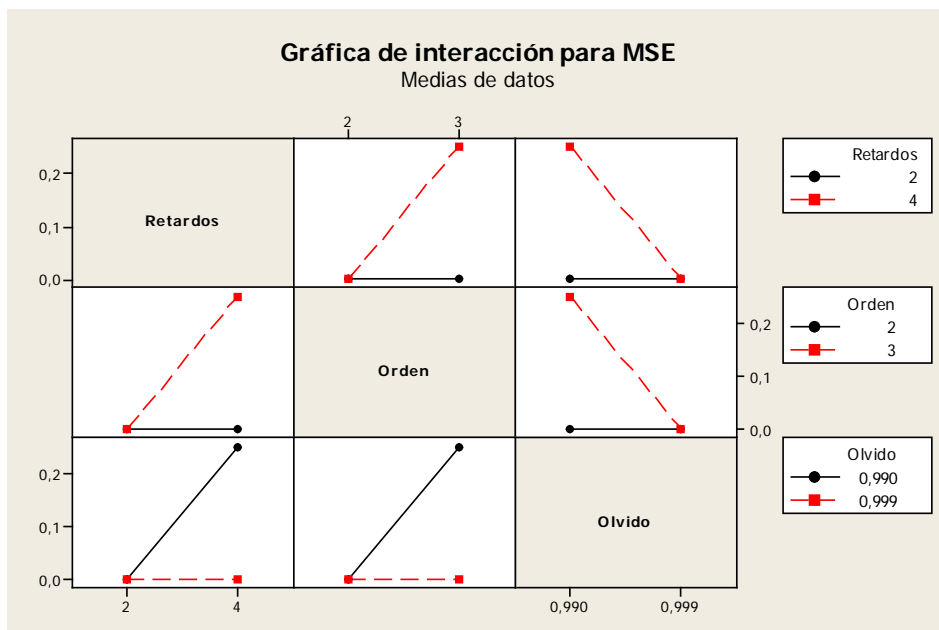


Figura 7.31: Comportamiento de las interacciones, CBF (simulación).

En la Tabla 7.20 se muestran los recursos computacionales requeridos por la red (para los datos resaltados en la Tabla 7.18).

Tabla 7.20: Recursos de cómputo para red CBF, simulación.

Número de	CBF (simulación)
Funciones base	9
Pesos	9
Retrasos totales	4
Tanh	1

En la Tabla 7.21 se muestra el coeficiente de correlación Pearson entre cada variable real y cada variable estimada.

Tabla 7.21: Correlación red CBF, simulación

Variable	CBF
θ	0.9963
x	0.9995
$\dot{\theta}$	0.9997
\dot{x}	0.9999

Como se puede observar al compararlas Tablas 7.4 y 7.20 la red multicapa requiere de una complejidad de cómputo mayor que la polinomial. Además, al comparar las Tablas 7.5 y 7.21, la red polinomial propuesta con funciones base de Chebyshev tiene una mejor aproximación, en términos de error instantáneo. También hay que mencionar que los resultados, para datos de simulación, reflejan una mejor aproximación cuando se utilizan polinomios de Chebyshev que cuando se utilizan polinomios de Volterra.

En las Figuras 7.32, 7.33, 7.34 y 7.35 se muestran las gráficas para la mejor aproximación con la red CBF, en términos del MSE, para cada variable del péndulo, resaltado en la Tabla 7.18. Únicamente se grafican los resultados de la prueba de validación de la red.

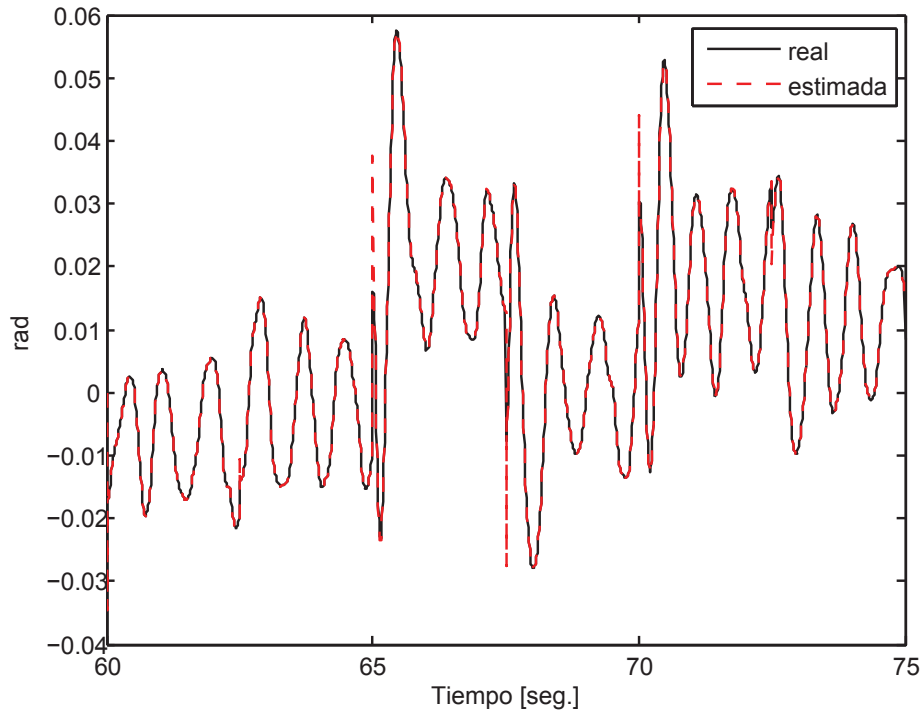


Figura 7.32: Posición angular del péndulo, CBF (simulación).

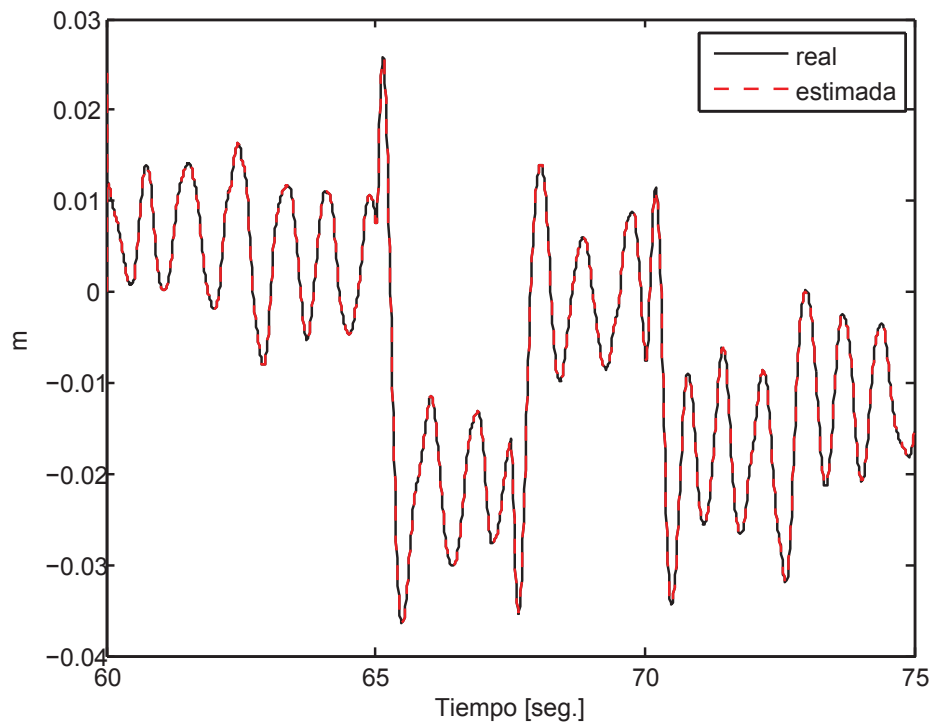


Figura 7.33: Desplazamiento del carro, CBF (simulación).

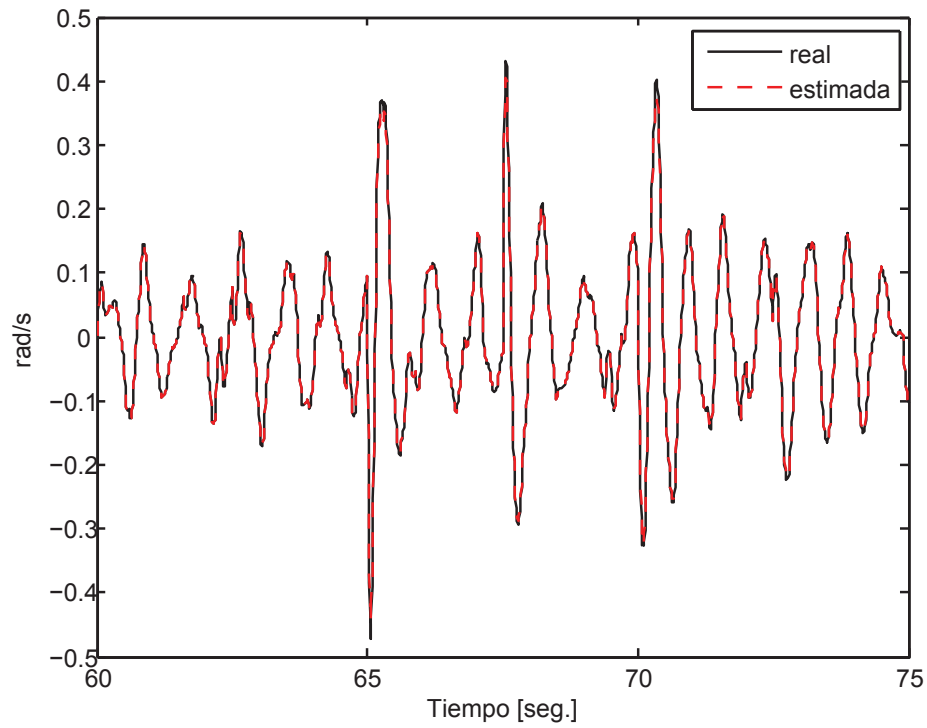


Figura 7.34: Velocidad angular del péndulo, CBF (simulación).

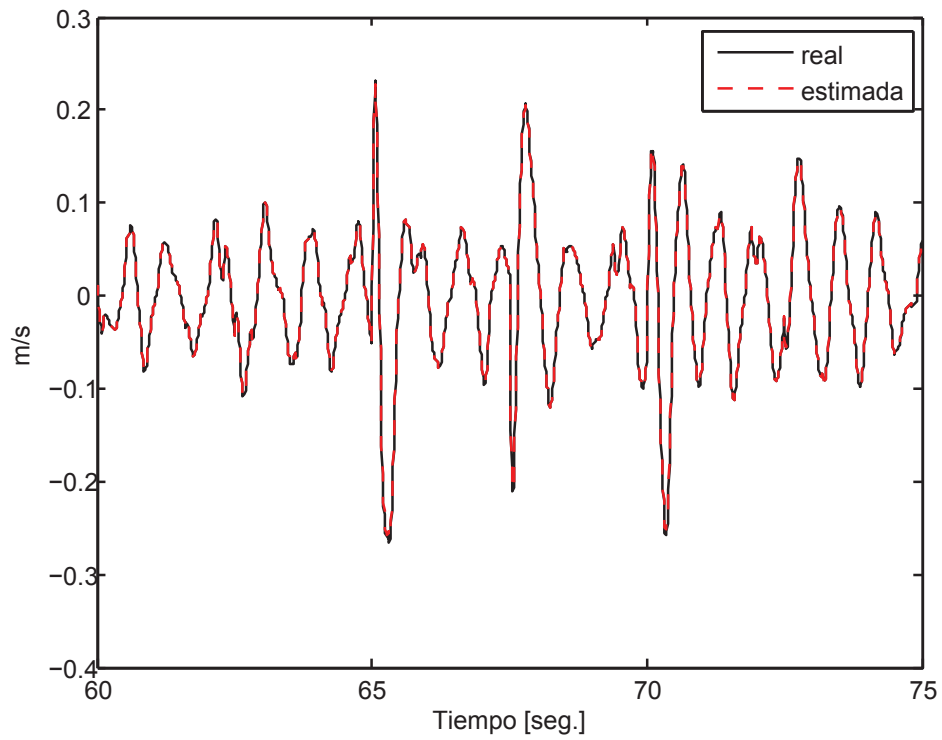


Figura 7.35: Velocidad lineal del carro, CBF (simulación).

■ Experimento

También se realizó un diseño de experimentos 2^3 , los factores que influyen son los mismos que en la simulación. En la Tabla 7.22, se muestran los datos del error de aproximación, los valores resaltados son los que tienen una mejor aproximación tomando en cuenta los recursos de cómputo necesarios.

Tabla 7.22: Datos del diseño de experimentos CBF (experimento).

Respuesta:		Factor B: l			
MSE		2		3	
Factor C:		Factor A: λ		Factor A: λ	
$m = n$		0.990	0.999	0.990	0.999
2	θ	2.22E-07	2.38E-07	2.20E-07	2.20E-07
	x	1.37E-05	1.03E-05	1.29E-05	9.36E-06
	$\dot{\theta}$	9.16E-07	1.47E-06	9.38E-07	1.01E-06
	\dot{x}	1.87E-04	1.82E-04	1.91E-04	1.82E-04
4	θ	2.18E-07	3.07E-07	2.17E-07	2.25E-07
	x	1.22E-05	1.03E-05	1.20E-05	1.16E-05
	$\dot{\theta}$	8.53E-07	1.32E-06	9.01E-07	8.50E-07
	\dot{x}	1.75E-04	1.63E-04	1.80E-04	1.62E-04

En la Tabla 7.23 se muestra el análisis de variancia. Los resultados del análisis de variancia son más ilustrativos para los datos experimentales que para los datos de simulación.

En la Figura 7.36 se pueden observar los efectos de los factores del diseño experimental. El factor que más influye en el MSE de la aproximación es el número de retardos de entrada - salida.

En la Figura 7.37 se presenta una gráfica con las tendencias de cada factor principal del diseño experimental. En la Figura 7.38 se muestran las tendencias de las interacciones de los factores.

En la Tabla 7.24 se muestran los recursos computacionales requeridos por la red (para los datos resaltados en la Tabla 7.22).

Tabla 7.23: Análisis de variancia para red CBF (experimento).

Factor	SS	GL	MSS	F	P
A	8.48E-11	1	8.48E-11	3.02	0.097
B	1.28E-12	1	1.28E-12	0.04	0.843
C	1.22E-10	1	1.22E-10	4.56	0.045
AB	2.96E-12	1	2.96E-12	4.00	0.757
AC	4.12E-12	1	4.12E-12	0.10	0.721
BC	2.44E-13	1	2.44E-13	0.13	0.000
ABC	6.64E-15	1	6.64E-15	0.00	0.000
Error residual	1.82E-07	21	7.57E-09		
Total	1.82E-07	31			

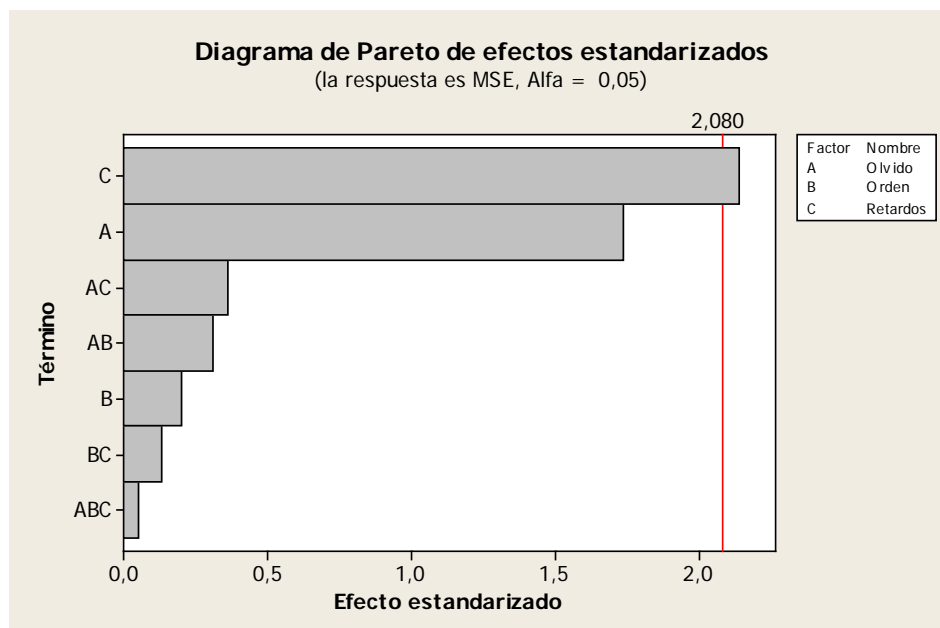


Figura 7.36: Efectos ordenados para el diseño 2^3 CBF (experimento).

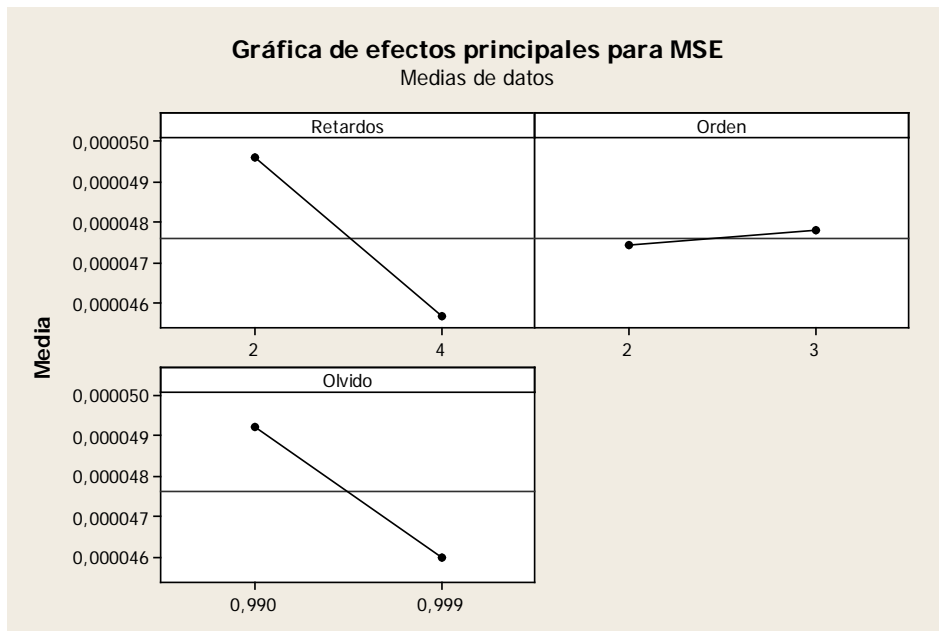


Figura 7.37: Comportamiento de los efectos principales, CBF (experimento).

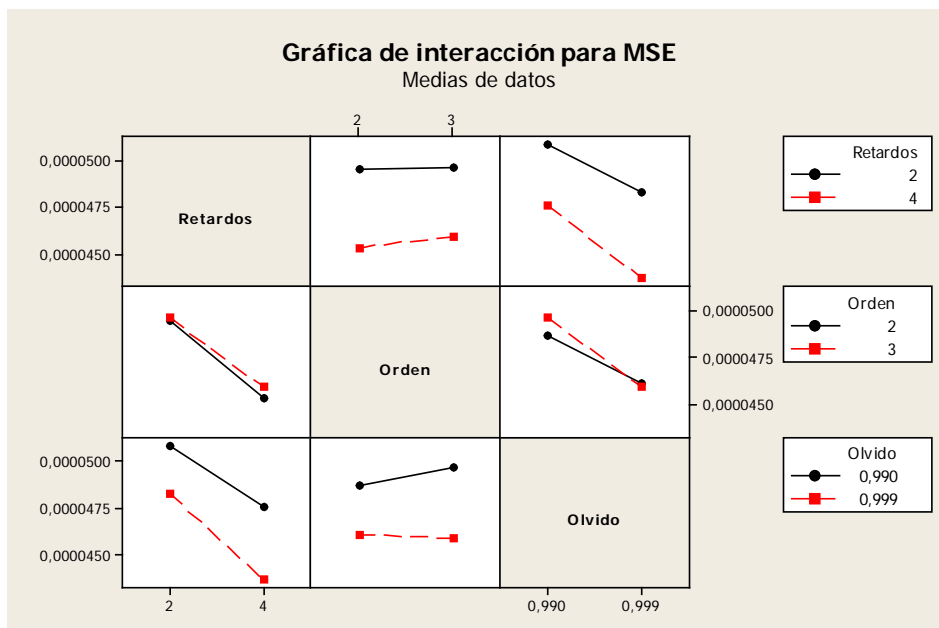


Figura 7.38: Comportamiento de las interacciones, CBF (experimento).

Tabla 7.24: Recursos de cómputo para red CBF, experimento.

Número de	CBF (experimento)
Funciones base	9
Pesos	9
Retrasos totales	4
Tanh	1

En la Tabla 7.25 se muestra el coeficiente de correlación Pearson entre cada variable real y cada variable estimada.

Tabla 7.25: Correlación red CBF, EXP

Variable	CBF
θ	0.9982
x	0.9971
$\dot{\theta}$	0.9737
\dot{x}	0.9801

Como se puede observar al compararlas Tablas 7.8 y 7.24 la red multicapa requiere de una complejidad de cómputo mayor que la polinomial. Además, al comparar las Tablas 7.9 y 7.25, la red polinomial propuesta con funciones base de Chebyshev tiene una mejor aproximación, en términos de error instantáneo, que la red MLP y la red con polinomios de Volterra. También hay que la red MLP aproxima mejor el desplazamiento lineal que cuando se utilizan polinomios de Volterra o de Chebyshev.

Finalmente en las Figuras 7.39, 7.40, 7.41 y 7.42 se muestran las gráficas para la mejor aproximación con la red CBF, en términos del MSE, para cada variable del péndulo.

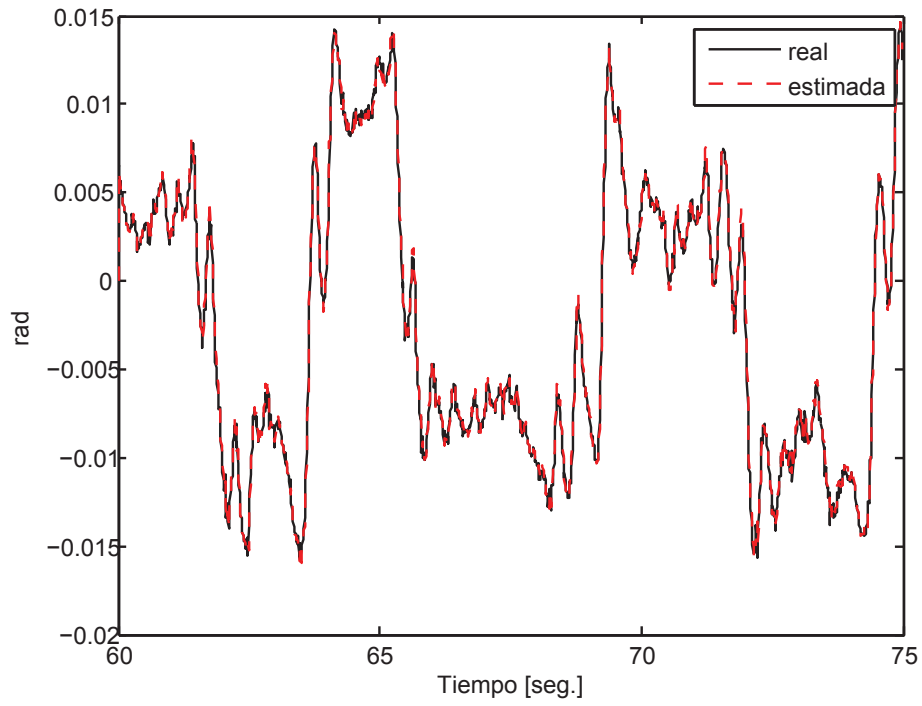


Figura 7.39: Posición angular del péndulo, CBF (experimento).

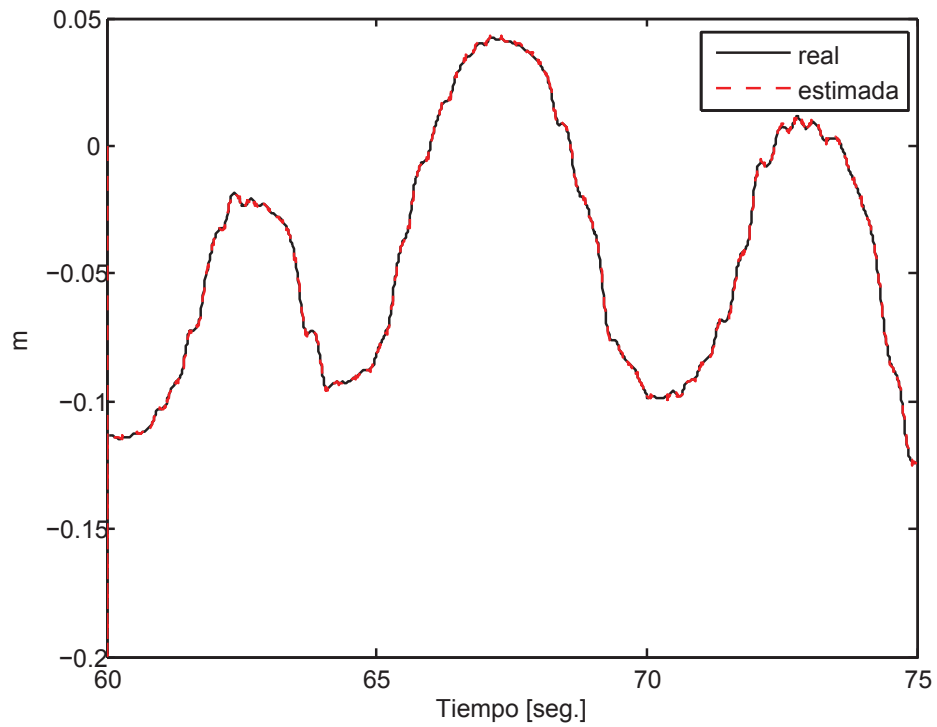


Figura 7.40: Desplazamiento del carro, CBF (experimento).

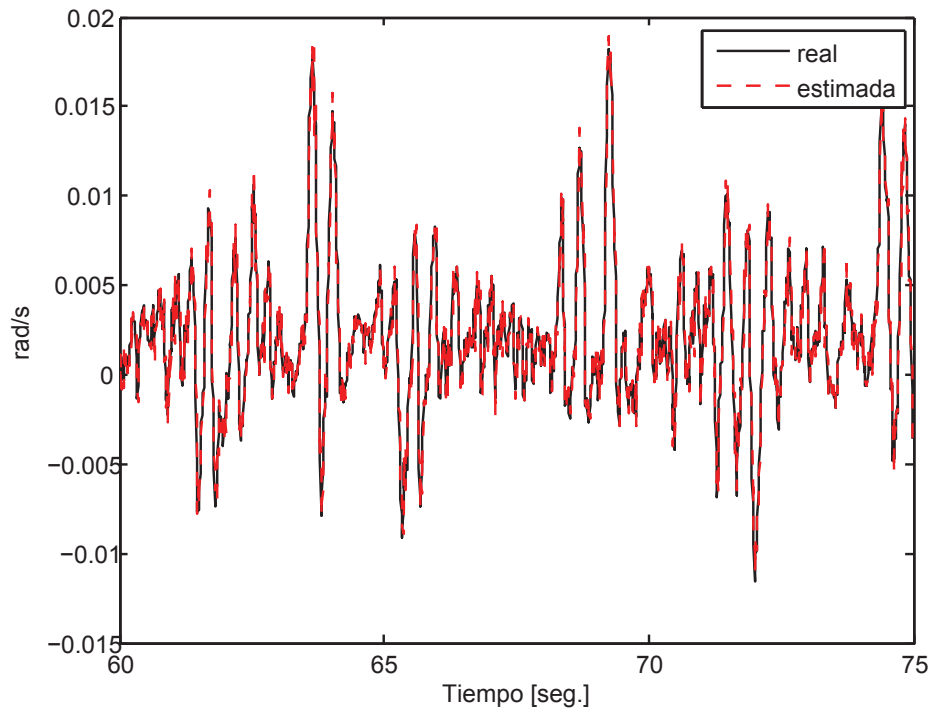


Figura 7.41: Velocidad angular del péndulo, CBF (experimento).

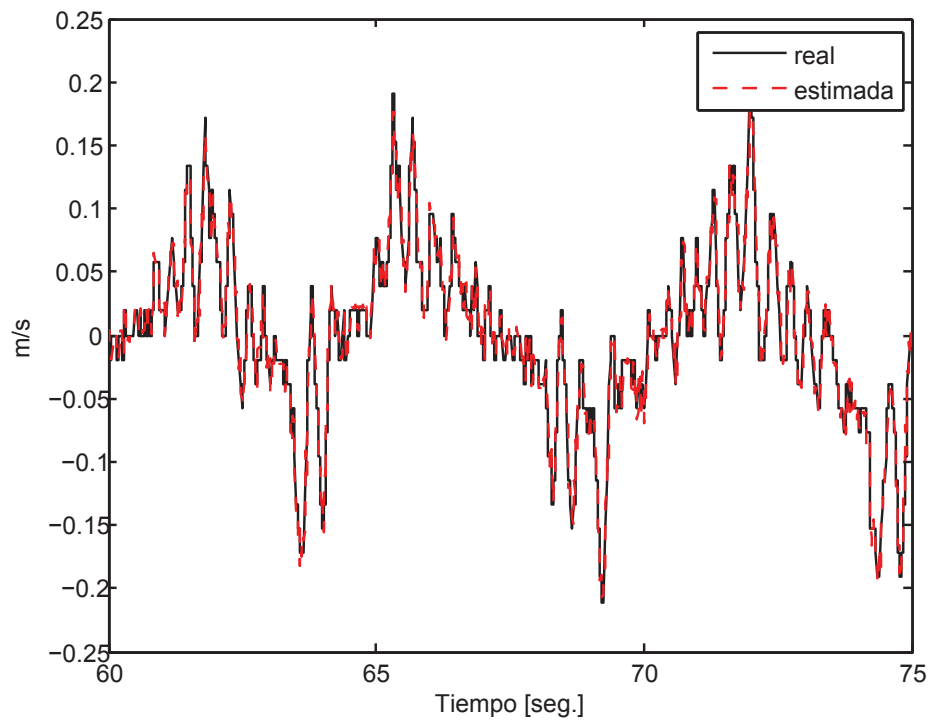


Figura 7.42: Velocidad lineal del carro, CBF (experimento).

7.5. Análisis de resultados

Tanto en la simulación como en el experimento se observa que para una red NARX prealimentada una mayor cantidad de neuronas en la capa oculta no implica, necesariamente, una mejor aproximación, tal y como lo discuten Mohandas y Paritala (2006), Sutradhar *et al.* (2010) y Choudhary *et al.* (2012), este fenómeno se atribuye principalmente a errores de cálculo matemático (truncamientos y redondeos). Si se comparan las gráficas de efectos principales para MSE presentadas en las Figuras 7.2 y 7.9 se puede observar lo mencionado anteriormente. De las Figuras 7.1 y 7.8 se puede concluir que tanto para la simulación como para el experimento real, el factor que más efecto tiene sobre el error de aproximación es el número de neuronas en la capa oculta que, dado el diseño experimental realizado, al ser mayor, lejos de disminuir el MSE, lo aumenta. También hay que observar que el número de retardos de entrada-salida prácticamente no tiene influencia sobre el error de aproximación del MLP.

Por otro lado, es importante el hecho de que los resultados obtenidos para este tipo de red son pobremente reproducibles debido a que los datos se presentan de manera aleatoria en cada época de entrenamiento, por lo que los resultados antes mencionados aunque son similares no son idénticos al reproducir el experimento con exactamente los mismos datos de entrada a la red y los mismos parámetros de red.

También se observa que esta red es de pobre desempeño si se pretende aplicar en un experimento en tiempo real debido a la forma en la que se entrena, puesto que necesitaría todos los datos del experimento completo para entrenarse más de una época.

La red NARX prealimentada, sin embargo, tiene una tendencia uniforme en sus resultados y esto se puede comprobar al comparar las Figuras 7.3 y 7.10, donde ante datos de entrada muy diferentes se tiene una tendencia muy similar en cuanto a aproximación se refiere.

Analizando el algoritmo compacto para red neuronal propuesto en el apéndice B.2 se observa su eficiencia ante diferentes expansiones polinomiales y su capacidad para tener o no alguna función de activación en su salida, dependiendo del usuario final. En este trabajo se utilizó una función tangente hiperbólica.

En particular para el polinomio de Volterra como función base (VPBF) se observa que la mejor aproximación para cada variable del péndulo se obtiene con los mismos parámetros de red, tanto para la simulación como para el experimento, esto se puede observar en las Tablas 7.10 y 7.14. Es decir, independientemente de los datos de entrada que se proporcionen la red tiene la misma tendencia de aproximación, lo anterior se puede comprobar comparando las Figuras 7.17 y 7.24.

Pero, aunque las tendencias de respuesta son las mismas, el diseño de experimento muestra (en las Tablas 7.11 y 7.15) que, para la simulación y para el experimento real, aunque el factor más significativo en la aproximación sea el factor de olvido, los demás factores que tienen efecto sobre el error de aproximación son diferentes. Esto último no es necesariamente malo, pero sí indica que el error de aproximación no sigue una distribución normal y hay que tomar en cuenta esta propiedad dependiendo del tipo de sistema al que se pretenda aplicar la red.

Para el polinomio de Chebyshev como función base (CBF) se observa que existe una tendencia en los resultados, si se comparan las Tablas 7.18 y 7.22 se nota que, independientemente de los datos de entrada a la red, la tendencia del MSE es la misma. Pero al igual que con la VPBF, los análisis de variancia, presentados en las Tablas 7.19 y 7.23, muestran que no hay similitud entre los resultados obtenidos de cada diseño experimental (el de la simulación y el del experimento real). Más aun, las Figuras 7.29, 7.30 y 7.31 al ser comparadas con las Figuras 7.36, 7.37 y 7.38 reflejan una diferencia notable en cuanto a los efectos y tendencias de los factores del diseño experimental, por lo que se puede deducir que la distribución de los errores de aproximación no sigue una tendencia normal.

Por otra parte, si se hace una comparación directa de complejidad de cálculo, la red Chebyshev siempre tiene mejores resultados utilizando menos recursos que las demás, tanto para datos de simulación como datos de experimento, esto se puede observar en la Tabla 7.26 (sólo se toman en cuenta los resultados de las Tablas de diseño experimental que están resaltados).

Además, la red Chebyshev también mostró superioridad en aproximación, tanto para datos de simulación como para datos de experimento, esto se puede observar en la Tabla 7.27 donde se presentan los coeficientes de correlación lineal para los datos resaltados en las Tablas

Tabla 7.26: Comparación de complejidad de cálculo para MLP, VPBF y CBF.

Número de	MLP	VPBF	CBF
Neuronas ocultas	10	-	-
Funciones base	-	15	9
Pesos	50	15	9
Retardos totales	4	4	4
Tanh	10	1	1

de diseño experimental, los cuales se consideran el mejor resultado de cada red tomando en cuenta los recursos de cómputo requeridos. El coeficiente de Pearson sirve como un índice que puede utilizarse para medir el grado de relación de dos variables, en este caso la salida deada y la salida estimada por la red neuronal.

Tabla 7.27: Coeficiente de correlación lineal de Pearson para el mejor desempeño.

Variable	Datos de Simulación			Datos Experimentales		
	MLP	VPBF	CBF	MLP	VPBF	CBF
θ	0.9826	0.9883	0.9963	0.9963	0.9975	0.9982
x	0.9849	0.9995	0.9995	0.9989	0.9968	0.9971
$\dot{\theta}$	0.9957	0.9997	0.9997	0.9167	0.9735	0.9737
\dot{x}	0.9968	0.9894	0.9999	0.9602	0.9792	0.9801

Por último, se presenta el coeficiente de determinación (R^2) en la Tabla 7.28, sólo los resultados para los datos experimentales. Se calcula de acuerdo a la siguiente fórmula (Kutner *et al.*, 2004):

$$R^2 = 1 - \frac{SSE}{SST}$$

donde $SSE = \sum^n (Y - \hat{Y})^2$ y $SST = (n - 1) \text{var}(\hat{Y})$.

Tabla 7.28: Coeficiente R^2 para MLP, VPBF y CBF. Datos experimentales.

Variable	MLP	VPBF	CBF
θ	0.9926	0.9950	0.9964
x	0.9978	0.9936	0.9942
$\dot{\theta}$	0.8403	0.9477	0.9481
\dot{x}	0.9220	0.9588	0.9606

VIII. CONCLUSIONES

En este trabajo de investigación se presentó un algoritmo compacto de red neuronal polinomial y su respectivo diseño de experimentos y gráficas de tendencia, así también se comparó como una red neuronal artificial de uso común y se mostró que el algoritmo propuesto tiene un mejor desempeño en términos de error de aproximación y posibilidad de implementación en línea. Algunas ventajas del algoritmo de red polinomial propuesto en este trabajo son: la posibilidad de cambiar la expansión polinomial por cualquiera que se desee sin tener que cambiar o adaptar el resto del algoritmo, la posibilidad de tener una función de activación en la salida, un método rápido y confiable de actualización de pesos, y una reproducibilidad total en el sentido de que administrando exactamente los mismos datos de entrada a la red, con los mismos parámetros, en experimentos diferentes, se obtienen exactamente los mismos resultados. Esta última propiedad mencionada no la tienen las redes neuronales artificiales comunes como la red NARX prealimentada con estructura de perceptrón multicapa.

Resultados para una simulación aplicados a cada red neuronal son presentados, los cuales coinciden con los resultados para datos experimentales reales aplicados a cada red. La creencia común de que una red prealimentada aumenta su capacidad de aproximación al agregar más neuronas en la capa oculta es negada, al menos para las que tienen estructura en modelo NARX, con base en el diseño experimental realizado tanto para simulación como experimento real. Los resultados del diseño experimental concuerdan con resultados obtenidos por Mohandas y Paritala (2006), Sutradhar *et al.* (2010) y Choudhary *et al.* (2012), cada uno para plantas diferentes.

También se concluye que, efectivamente, como se propuso inicialmente en la hipótesis, una red neuronal polinomial tiene mejor desempeño que una red neuronal prealimentada en cuanto a aproximación de modelos se refiere. Y también el algoritmo compacto de red polinomial que se propone en este trabajo es más sencillo que cualquier algoritmo de red

prealimentada.

No se requiere conocimiento a priori del sistema para aplicar la red neuronal polinomial, pero si se llegara a contar con un modelo discreto del sistema a identificar se podría incluso identificar los parámetros del sistema.

A manera de posibles trabajos futuros se propone realizar control robusto utilizando una red neuronal polinomial. Otro posible trabajo futuro que se puede proponer es utilizar redes polinomiales, que dentro de este trabajo solo fueron utilizadas para aproximar datos en series temporales, para clasificación de modelos altamente variables en el tiempo.

Finalmente, se puede decir que se cumplió exitosamente con el objetivo general, propuesto en un principio, y que la red neuronal con polinomios de Chebyshev como funciones base es la que tuvo un mejor desempeño de entre las estructuras presentadas en este trabajo.

BIBLIOGRAFÍA

- Abrahantes, M., J. Mulder and K. Butter. 2007. Modeling, Identification and Control of an Under Actuated Inertial Wheel Pendulum. System Theory, 2007. SSST '07. Thirty-Ninth Southeastern Symposium on. :1 –5.
- Aguado Behar, Alberto and Miguel Martínez Iranzo. 2003. Identificación y control adaptativo. Pearson Educación, S.A.
- Alanis, Alma Y., Edgar N. Sanchez, Alexander G. Loukianov and Esteban A. Hernandez. 2010. Discrete-time recurrent high order neural networks for nonlinear identification. Journal of the Franklin Institute 347(7):1253 – 1265.
- Anderson, C.W. 1989. Learning to control an inverted pendulum using neural networks. Control Systems Magazine, IEEE 9(3):31 –37.
- Åström, K.J. and Bjorn Wittenmark. 1994. Adaptive Control. 2nd ed. Addison-Wesley Longman Publishing Co. Boston, U.S.A.
- Åström, KJ and Peter Eykhoff. 1971. System identification - a survey. Automatica 7(2):123–162.
- Awan, S.M., Z.A. Khan, M. Aslam, W. Mahmood and A. Ahsan. 2012. Application of NARX based FFNN, SVR and ANN Fitting models for long term industrial load forecasting and their comparison. Industrial Electronics (ISIE), 2012 IEEE International Symposium on. :803–807.
- Billings, S. A., S. Chen and M. J. Korenberg. 1989. Identification of MIMO non-linear systems using a forward-regression orthogonal estimator. International Journal of Control 49(6):2157–2189.

- Bogdanov, A. 2004. Optimal control of a double inverted pendulum on a cart. OHSU Technical Report, Department of Computer Science and Electrical Engineering, OGI School of Science and Engineering.
- Canelon, J.I., L.S. Shieh and N.B. Karayiannis. 2005. A new approach for neural control of nonlinear discrete dynamic systems. *Information Sciences* 174(3):177–196.
- Chaturvedi, DK and H. Röck. 2008. Generalized Neuron Based Control of the Inverted Pendulum on a Cart. XXXII NATIONAL SYSTEMS CONFERENCE, NSC. :730–733.
- Choudhary, I., K. Assaleh and M. AlHamaydeh. 2012. Nonlinear AutoRegressive eXogenous Artificial Neural Networks for predicting Buckling restrained braces force. *Mechatronics and its Applications (ISMA), 2012 8th International Symposium on.* :1–5.
- Cordova, Juan and Wen Yu. 2012. Two Types of Haar Wavelet Neural Networks for Nonlinear System Identification. *Neural Processing Letters* 35:283–300.
- Coron, J.M. 2010. On the Controllability of Nonlinear Partial Differential Equations. PROCEEDINGS OF THE INTERNATIONAL CONGRESS OF MATHEMATICIANS. 901 :238–264.
- Dreyfus, Gérard *et al.* 2005. *Neural networks: methodology and applications.* Springer Berlin.
- Fantoni, I. and R. Lozano. 2002. *Non-linear control for underactuated mechanical systems.* Springer, Great Britain.
- Fausett, L.V. 1994. *Fundamentals of neural networks: architectures, algorithms, and applications.* Prentice-Hall Englewood Cliffs, NJ.
- G.U.N.T. 2008. Experiment Instructions RT124 Fuzzy Control: Carrier Vehicle with Inverted Pendulum.
- Haykin, S. 1999. *Neural Networks: A Comprehensive Foundation.* Pearson Education New York, U.S.A.

- Hojati, M. and S. Gazor. 2002. Hybrid adaptive fuzzy identification and control of nonlinear systems. *Fuzzy Systems, IEEE Transactions on* 10(2):198–210.
- Hornik, K., M. Stinchcombe and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5):359–366.
- Jagannathan, S. and F.L. Lewis. 1996. Identification of nonlinear dynamical systems using multilayered neural networks. *Automatica* 32(12):1707–1712.
- Jiang, Lian Lian, D.L. Maskell and J.C. Patra. 2012. Chebyshev Functional Link Neural Network-based modeling and experimental verification for photovoltaic arrays. *Neural Networks (IJCNN), The 2012 International Joint Conference on*. :1–8.
- Kelly, R., V. Santibáñez and A. Loria. 2005. *Control of robot manipulators in joint space*. Springer, London.
- Kenne, G., T. Ahmed-Ali, F. Lamnabhi-Lagarrigue and H. Nkwawo. 2006. Nonlinear systems parameters estimation using radial basis function network. *Control engineering practice* 14(7):819–832.
- Kuo, Benjamin C. 1996. *Sistemas de control automático (7ma ed.)*. Prentice-Hall Hispanoamericana.
- Kuschewski, J.G., S. Hui and S.H. Zak. 1993. Application of feedforward neural networks to dynamical system identification and control. *Control Systems Technology, IEEE Transactions on* 1(1):37–49.
- Kutner, Michael, Christopher Nachtsheim, John Neter and William Li. 2004. *Applied Linear Statistical Models*. 5 ed. McGraw-Hill.
- Lee, Ching-Hung and Ching-Cheng Teng. 2000. Identification and control of dynamic systems using recurrent fuzzy neural networks. *Fuzzy Systems, IEEE Transactions on* 8(4):349–366.

- Lee, Tsu-Tian and Jin-Tsong Jeng. 1998. The Chebyshev-polynomials-based unified model neural networks for function approximation. *Systems, Man, and Cybernetics, Part B: Cybernetics*, IEEE Transactions on 28(6):925–935.
- Li, Mu and Yigang He. 2010. Nonlinear system identification using adaptive Chebyshev neural networks. *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*. 1 :243–247.
- Liu, G. P., V. Kadiramanathan and S. A. Billings. 1998. On-line identification of nonlinear systems using Volterra polynomial basis function neural networks. *Neural networks : the official journal of the International Neural Network Society* 11(9):1645 – 1657.
- Liu, G.P. 2001. *Nonlinear Identification and Control: A Neural Network Approach*. Springer, Verlag London.
- Liu, Yijian and Chen Peng. 2009. Time-Variation Nonlinear System Identification Based on Bayesian-Gaussian Neural Network. *Natural Computation, 2009. ICNC '09. Fifth International Conference on*. 1 :353 –357.
- Loría, Antonio and Elena Panteley. 2006. *6 Stability, Told by Its Developers*. *Advanced Topics in Control Systems Theory*. Springer :199–258.
- Luo, W and S.A. Billings. 1995. Adaptive model selection and estimation for nonlinear systems using a sliding data window. *Signal Processing* 46(2):179–202.
- Luo, W, S.A. Billings and K.M. Tsang. 1996. On-line structure detection and parameter estimation with exponential windowing for nonlinear systems. *European Journal of Control* (2):291–304.
- Manry, M., H. Chandrasekaran and C.H. Hsieh. 2001. *Handbook of Neural Network Signal Processing*. :2–1 to 2–28.
- Milton, J., J.L. Cabrera, T. Ohira, S. Tajima, Y. Tonosaki, C.W. Eurich and S.A. Campbell. 2009. The time-delayed inverted pendulum: implications for human balance control. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 19(2):1–12.

- Mohandas, K.P. and Suresh Paritala. 2006. Simulation of Inverted Pendulum Using Neural Networks for Identification. Proceedings of the 4th Faculty of Architecture and Engineering Symposim. :621–627.
- Muskinja, N. and B. Tovornik. 2006. Swinging up and stabilization of a real inverted pendulum. Industrial Electronics, IEEE Transactions on 53(2):631 – 639.
- Narendra, K.S. and K. Parthasarathy. 1989. Adaptive identification and control of dynamical systems using neural networks. Decision and Control, 1989., Proceedings of the 28th IEEE Conference on. :1737 –1738 vol.2.
- Narendra, K.S. and K. Parthasarathy. 1990. Identification and control of dynamical systems using neural networks. Neural Networks, IEEE Transactions on 1(1):4 –27.
- Ogata, Katsuhiko. 2002. Modern Control Engineering (4th ed.). Prentice-Hall, Inc.
- Oh, S.K., W. Pedrycz and H.S. Park. 2003. Hybrid identification in fuzzy-neural networks. Fuzzy Sets and Systems 138(2):399–426.
- Onder Efe, M. and O. Kaynak. 1999. A comparative study of neural network structures in identification of nonlinear systems. Mechatronics 9(3):287–300.
- Patrikar, A. and J. Provenca. 1996. Nonlinear system identification and adaptive control using polynomial networks. Mathematical and Computer Modelling 23(1):159–173.
- Purwar, S., I. N. Kar and A. N. Jha. 2007. On-line system identification of complex systems using Chebyshev neural networks. Applied Soft Computing 7(1):364–372.
- Purwar, S., I. N. Kar and A. N. Jha. 2008. Adaptive output feedback tracking control of robot manipulators using position measurements only. Expert systems with applications 34(4):2789–2798.
- Ranković, V.M. and I.Ž. Nikolić. 2008. Identification of nonlinear models with feed forward neural network and digital recurrent network. FME Transactions 36(2):87–92.

- Riedmiller, M. 1993. Controlling an inverted pendulum by neural plant identification. *Systems, Man and Cybernetics*, 1993. 'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on. :473 –478 vol.4.
- Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* 65(6):386.
- Rumelhart, David E and James L McClelland. 1987. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Foundations. 1 MIT press.
- Sastry, S. and M. Bodson. 1989. *Adaptive control: stability, convergence, and robustness*. Prentice-Hall, Inc.
- Stone, Marshall H. 1948. The generalized Weierstrass approximation theorem. *Mathematics Magazine* 21(5):237–254.
- Subudhi, Bidyadhar and Debashisha Jena. 2011. A differential evolution based neural network approach to nonlinear system identification. *Applied Soft Computing* 11(1):861 – 871.
- Sutradhar, A., A. Sengupta and V.R. Challa. 2010. Identification of servo-driven inverted pendulum system using neural network. *India Conference (INDICON), 2010 Annual IEEE*. :1 –4.
- Tewari, Ashish. 2002. *Modern Control Design With MATLAB and SIMULINK*. JOHN WILEY & SONS, LTD, England.
- Verschueren, E.R.M. 1995. *Identification and Control of an Inverted Pendulum Using Neural Networks*. Master thesis Technische Universiteit Eindhoven.
- Weierstrass, Karl. 1885. Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen Veränderlichen. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin* 2:633–639.
- Wells, D. A. 1972. *Teoría y problemas de dinámica de Lagrange*. McGraw-Hill, Colombia.

- Xiuchun, Xiao, Jiang Xiaohua and Zhang Yunong. 2009. An algorithm for designing Chebyshev neural network. *Computing, Communication, Control, and Management*, 2009. CCCM 2009. ISECS International Colloquium on. 2 :206–209.
- Yazdizadeh, A. and K. Khorasani. 2002. Adaptive time delay neural network structures for nonlinear system identification. *Neurocomputing* 47(1):207–240.
- Yoon, H., Brett. E. Bateman and B. N. Agrawal. 2011. Laser beam jitter control using recursive-least-squares adaptive filters. *Journal of Dynamic Systems, Measurement, and Control* 133:041001–1.
- Yoshida, K. 1999. Swing-up control of an inverted pendulum by energy-based methods. *Proceedings of the American Control Conference*, 1999. 6 :4045 –4047 vol.6.
- Yu, Ping, Pu Han, Ze Dong and Zeng-Zhou Jia. 2004. Volterra basic function network based adaptive controller. *Machine Learning and Cybernetics*, 2004. *Proceedings of 2004 International Conference on*. 2 :729–732.
- Yu, W. 2004. Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms. *Information sciences* 158:131–147.
- Zeng, Xiangping, Haiquan Zhao, Weidong Jin, Zhengyou He and Tianrui Li. 2013. Identification of nonlinear dynamic systems using convex combinations of multiple adaptive radius basis function networks. *Measurement* 46(1):628 – 638.
- Zhong, Wei and H. Röck. 2001. Energy and passivity based control of the double inverted pendulum on a cart. *Control Applications*, 2001. (CCA '01). *Proceedings of the 2001 IEEE International Conference on*. :896 –901.

APÉNDICES

A. DISEÑO MEDIANTE LA UBICACIÓN DE LOS POLOS

En lugar de especificar sólo los polos dominantes en lazo cerrado (enfoque del diseño convencional), el enfoque actual de ubicación de polos especifica todos los polos en lazo cerrado. Sin embargo, hay un costo asociado con ubicar todos los polos en lazo cerrado, porque hacerlo requiere de mediciones exitosas de todas las variables de estado, o bien requiere de la inclusión de un observador de estado en el sistema (Ogata, 2002).

Una representación lineal de la planta está dada por las ecuaciones (3.52) y (3.57), de donde se obtuvieron las matrices de coeficientes A y B , ecuaciones (3.53) y (3.54) respectivamente. También se definen:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (\text{A.1})$$

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{A.2})$$

La única entrada, $u(t)$, es una fuerza horizontal aplicada al carro, y las dos salidas son la posición angular del péndulo, $\theta(t)$, y la posición horizontal del carro, $x(t)$. El vector de estados para esta planta de cuarto orden es $\mathbf{x}(t)$, (3.37). Se asumen los valores numéricos de la planta de la siguiente manera: $M = 2Kg$, $m = 0,1Kg$, $l = 0,5m$, y $g = 9,8m/s^2$. Sustituyendo en (3.53) y (3.54) se obtiene:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 20,58 & 0 & 0 & 0 \\ -0,49 & 0 & 0 & 0 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0,5 \end{bmatrix} \quad (\text{A.3})$$

Para determinar la matriz de ganancias de realimentación K que obliga a los valores característicos de A_{CL} a ser los valores deseados primeramente se debe verificar que el

sistema sea de estado completamente controlable, lo cual se demostró en (3.58). Ahora, hay que determinar los valores característicos de la matriz A , para eso se determina su polinomio característico:

$$|sI - A| = s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0 \quad (\text{A.4})$$

Se obtiene el polinomio característico:

$$s^4 - 20,58s^2 = 0 \quad (\text{A.5})$$

y se define el vector a :

$$a = \begin{bmatrix} 0 & 0 & -20,58 & 0 \end{bmatrix} \quad (\text{A.6})$$

Sacando las raíces del polinomio característico del sistema se obtiene la ubicación de los polos, la planta es inestable ya que tiene un polo con parte real positiva (y también un par de polos en $s = 0$). La tarea del controlador por realimentación de estados es estabilizar la planta. Para hacer el sistema en lazo cerrado estable se seleccionan los polos en lazo cerrado como:

$$V = \begin{bmatrix} -7,853 + 3,2528i \\ -7,853 - 3,2528i \\ -7,853 + 7,853i \\ -7,853 - 7,853i \end{bmatrix} \quad (\text{A.7})$$

Se desea ubicar los polos en lazo cerrado de tal manera que el polinomio característico es el siguiente:

$$|sI - A_{CL}| = |sI - A + BK| = s^n + \alpha_{n-1}s^{n-1} + \dots + \alpha_1s + \alpha_0 \quad (\text{A.8})$$

Dado que (A.7) son los polos deseados de A_{CL} se obtiene el polinomio característico como:

$$\alpha^4 + 31,4\alpha^3 + 442,3\alpha^2 + 3071,9\alpha + 8911,3 = 0 \quad (\text{A.9})$$

y se define el vector α :

$$\alpha = \begin{bmatrix} 31,4 & 442,3 & 3071,9 & 8911,3 \end{bmatrix} \quad (\text{A.10})$$

Se pueden conocer los parametros de:

$$K = [k_1 \ k_2 \ \cdots \ k_n] \quad (\text{A.11})$$

de la siguiente manera:

$$k_1 = \alpha_{n-1} - a_{n-1}; \ k_2 = \alpha_{n-2} - a_{n-2}; \ \cdots \ k_{n-1} = \alpha_1 - a_1; \ k_n = \alpha_0 - a_0 \quad (\text{A.12})$$

Expresado en forma vectorial (Tewari, 2002):

$$K = \alpha - a \quad (\text{A.13})$$

Pero dado que la representación de la planta en espacio de estados no esta dada en la forma canónica controlable, se debe utilizar la fórmula de ubicación de polos de Ackermann (Tewari, 2002):

$$K = (\alpha - a)P_{CT}P_C^{-1} \quad (\text{A.14})$$

Donde P_{CT} esta definida como:

$$P_{CT} = \begin{bmatrix} 1 & -a_{n-1} & -a_{n-2} & \cdots & -a_2 & -a_1 \\ 0 & 1 & -a_{n-1} & \cdots & -a_3 & -a_2 \\ 0 & 0 & 1 & \cdots & -a_4 & -a_3 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -a_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \quad (\text{A.15})$$

Sustituyendo todos los valores en (A.14) se obtiene entonces:

$$K = \begin{bmatrix} -917,5060 & -909,3161 & -188,1431 & -313,4622 \end{bmatrix} \quad (\text{A.16})$$

B. PROGRAMAS DE MATLAB

B.1. Programa para MLP

```
clc; clear all; close all;
% DATOS REALES
%-----%
importfile('DATOSsim.txt');%carga en matlab la base de datos
time(:,1)=x(:,1);%vector de tiempo de la prueba
THETA=x(:,3); %vector de salida
dTHETA=x(:,5); %vector de salida
X=x(:,4); %vector de salida
dX=x(:,6); %vector de salida
u=x(:,2); %vector de entrada
M = (size(time)*[1;0])-((size(time)*[1;0])-7500);
%75seg Ts=10ms
time=time(1:M);
u=u(1:M);
X=X(1:M);
dX=dX(1:M);
THETA=THETA(1:M);
dTHETA=dTHETA(1:M);

% DATOS DE ENTRENAMIENTO
tt=time(1:6000);
ut=u(1:6000);
Xt=X(1:6000);
dXt=dX(1:6000);
THETAt=THETA(1:6000);
dTHETAt=dTHETA(1:6000);

% DATOS DE VALIDACION
tv=time(6001:7500);
uv=u(6001:7500);
Xv=X(6001:7500);
dXv=dX(6001:7500);
THETAv=THETA(6001:7500);
dTHETAv=dTHETA(6001:7500);
%-----%
% PARÁMETROS DE LA RED
nu = 3; % retrasos en la entrada
ny = nu; % retrasos en la salida
Nh = 10; % neuronas ocultas
trainFcn = 'traingd'; % algoritmo de entrenamiento
Xnf = tonndata(ut,false,false);
%entrada de entrenamiento (0-60)seg
Tf = tonndata(dTHETAt,false,false); % salida de entrenamiento
ERR = 9e-07; % error deseado entrenamiento
EP = 1000; % épocas de entrenamiento (máximo)
LR = 0.5; % factor de aprendizaje
Xnfv = tonndata(uv,false,false);
```

```

% entrada de validación (60-75) seg
Tfv = tonndata(dTHETA_v,false,false); % salida de validación
rho1 = 'tansig';
rho2 = 'purelin';
%-----%
%RED
net = narxnet(nu, ny, Nh);
net.layers{1}.transferFcn = rho1;
net.layers{2}.transferFcn = rho2;
[Xs,Xi,Ai,Ts] = preparets(net,Xnf,{},Tf,{});
net.trainFcn = trainFcn;
net.trainParam.goal = ERR;
net.trainParam.epochs = EP;
net.trainParam.lr = LR;
net.trainParam.min_grad = 1e-6;
net.performFcn = 'mse';
[net,tr]= train(net,Xs,Ts,Xi,Ai);
tr;
[Xvs,Xvi,Avi,Tvs] = preparets(net,Xnfv,{},Tfv,{});
Y = net(Xvs,Xvi,Avi);
%perf = perform(net,Tvs,Y);
y = cell2mat(Tvs);
yhat = cell2mat(Y);
e2 = y-yhat;
MSE = mse(e2)
plot(tv(nu+1:1500), y, 'r', tv(nu+1:1500), yhat, 'k')
RESULTADOS(:,1) = tv(nu+1:1500);
RESULTADOS(:,2) = yhat;
RESULTADOS(:,3) = y;

```

B.2. Programa para red polinomial compacta

```

%Estructura general de RED + Gram Schmidt
clc; clear all; close all;

n = 4; %retardos en la salida
l = 3; %Orden del sistema
lambda = 0.5; %factor de olvido
m = n; %Retardos en la entrada
S = n+m; %numero de entradas

importfile('DATOS_R1.txt'); %carga en matlab la base de datos
time(:,1)=x(:,1); %vector de tiempo de la prueba
output(:,1)=x(:,6); %vector de salida
%vellin(:,1)=x(:,5); %vector de velocidad lineal
input(:,1)=x(:,2); %vector de entrada

M = 6001; %Muestras de simulación
Mt = 7501 - M; %15 seg.
t = time(1:M);
u = input(1:M);
y = output(1:M);
tt = time(M+1:7501);
ut = input(M+1:7501);
yt = output(M+1:7501);

ERR=0.00;

%[Y, ye, W, phi, N] = TrainCheb(u,y,M, l, S, lambda); %entrenamiento
%[Yt, yet, phi_t] = EvalCheb(ut,yt,Mt, l, S, W); %prueba

```

```

[Y, ye, W, fhi, N] = TrainVPBF(u,y,M, l, S, lambda); %entrenamiento
[Yt, yet, fhit] = EvalVPBF(ut,yt,Mt, l, S, W); %prueba

Y=Y';
ye = ye';
Yt=Yt';
yet = yet';

[NRSS, NRSSt, L] = NRSSError(fhi,fhit,Y,Yt,N,ERR);
MSE = mse(ye-Y);
MSEt = mse(yet-Yt)

plot(t,ye,'r',t,y,'k');
plot(tt,yet,'r',tt,yt,'k');

```

B.3. Programa para entrenamiento de la red VPBF

```

function [y, ye, W, FHI, N] = TrainVPBF(input,output,M, l, S, lambda)
N=factorial(S+1)/(factorial(l)*factorial(S));
for t=1:M
    u(t) = input(t,1);
    y(t) = output(t,1);
    if t==1
        P = eye(N);
        W = zeros(N,1); %vector de pesos inicial
    end
    [fhi] = ObtenerDatosVPBF(u,y,t,S,l);
    ye(t) = tansig(W'*fhi');
    %ye(t) = W'*fhi';
    k(:,1) = (inv(lambda)*P*fhi')/(1+(inv(lambda)*fhi*P*fhi'));
    P = (inv(lambda)*P)-(inv(lambda)*k(:,1)*fhi*P);
    e(t) = y(t)-ye(t);
    W = W + k(:,1)*e(t);
    FHI(t,:)=fhi(:);
end

```

B.4. Función para generar las funciones base de Volterra

```

function [fhi] = ObtenerDatosVPBF(u,y,t,S,l)
if S==4
    if t==1
        D = [0,0,0,0];
    elseif t==2
        D = [y(t-1),0,u(t-1),0];
    elseif t>2
        D = [y(t-1),y(t-2),u(t-1),u(t-2)];
    end
elseif S==6
    if t==1
        D = [0,0,0,0,0,0];
    elseif t==2
        D = [y(t-1),0,0,u(t-1),0,0];
    elseif t==3
        D = [y(t-1),y(t-2),0,u(t-1),u(t-2),0];
    elseif t>3
        D = [y(t-1),y(t-2),y(t-3),u(t-1),u(t-2),u(t-3)];
    end
end

```



```

end
elseif S==8
if t==1
D = [0,0,0,0,0,0,0,0];

elseif t==2
D = [y(t-1),0,0,0,u(t-1),0,0,0];

elseif t==3
D = [y(t-1),y(t-2),0,0,u(t-1),u(t-2),0,0];

elseif t==4
D = [y(t-1),y(t-2),y(t-3),0,u(t-1),u(t-2),u(t-3),0];
elseif t>4
D = [y(t-1),y(t-2),y(t-3),y(t-4),u(t-1),u(t-2),u(t-3),u(t-4)];
end
end

con = 1;
fhi(con) = 1;
if l >=1
for i=1:S
con = con + 1;
fhi(con) = D(i);
end
end
if l >=2
for i=1:S
for j=i:S
con = con + 1;
fhi(con) = D(i)*D(j);
end
end
end
if l >= 3
for i=1:S
for j=i:S
for k=j:S
con = con + 1;
fhi(con) = D(i)*D(j)*D(k);
end
end
end
end
if l==4
for i=1:S
for j=i:S
for k=j:S
for h=k:S
con = con + 1;
fhi(con) = D(i)*D(j)*D(k)*D(h);
end
end
end
end
end
end
end

```

B.5. Función para evaluar la red VPBF

```
function [y, ye, FHI] = EvalVPBF(input,output,M, l, S, W)
```

```

for t=1:M
    u(t) = input(t,1);
    y(t) = output(t,1);
    [fhi] = ObtenerDatosVPBF(u,y,t,S,1);
    ye(t) = tansig(W'*fhi');
    %ye(t) = W'*fhi';
    FHI(t,:)=fhi(:);
end

```

B.6. Programa para entrenamiento de la red CBF

```

function [y, ye, W, FHI, N] = TrainCheb(input,output,M, l, S, lambda)
N=((S*l)+1);
for t=1:M
    u(t) = input(t,1);
    y(t) = output(t,1);
    if t==1
        P = eye(N);
        W = zeros(N,1); %vector de pesos inicial
    end
    [fhi] = ObtenerDatosCheb(u,y,t,S,1);
    ye(t) = tansig(W'*fhi');
    %ye(t) = W'*fhi';
    k(:,1) = (inv(lambda)*P*fhi')/(1+(inv(lambda)*fhi*P*fhi'));
    P = (inv(lambda)*P)-(inv(lambda)*k(:,1)*fhi*P);
    e(t) = y(t)-ye(t);
    W = W + k(:,1)*e(t);
    FHI(t,:)=fhi(:);
end

```

B.7. Función para generar las funciones base de Chebyshev

```

function [fhi] = ObtenerDatosCheb(u,y,t,S,1)
if S==4
    if t==1
        D = [0,0,0,0];
    elseif t==2
        D = [y(t-1),0,u(t-1),0];
    elseif t>2
        D = [y(t-1),y(t-2),u(t-1),u(t-2)];
    end
elseif S==6
    if t==1
        D = [0,0,0,0,0,0];
    elseif t==2
        D = [y(t-1),0,0,u(t-1),0,0];
    elseif t==3
        D = [y(t-1),y(t-2),0,u(t-1),u(t-2),0];
    elseif t>3
        D = [y(t-1),y(t-2),y(t-3),u(t-1),u(t-2),u(t-3)];
    end
elseif S==8
    if t==1
        D = [0,0,0,0,0,0,0,0];
    elseif t==2

```

```

        D = [y(t-1),0,0,0,u(t-1),0,0,0];

elseif t==3
    D = [y(t-1),y(t-2),0,0,u(t-1),u(t-2),0,0];

elseif t==4
    D = [y(t-1),y(t-2),y(t-3),0,u(t-1),u(t-2),u(t-3),0];
elseif t>4
    D = [y(t-1),y(t-2),y(t-3),y(t-4),u(t-1),u(t-2),u(t-3),u(t-4)];
end
end

con = 1;
T(1)=1;
fhi(con) = T(1);
for j=1:S
    con = con + 1;
    T(2)=D(j);
    fhi(con) = T(2);
    for i=2:l
        con = con + 1;
        T(i+1)=2*D(j)*T(i)-T(i-1);
        fhi(con)=T(i+1);
    end
end
end

```

B.8. Función para evaluar la red CBF

```

function [y, ye, FHI] = EvalCheb(input,output,M, l, S, W)
for t=1:M
    u(t) = input(t,l);
    y(t) = output(t,l);
    [fhi] = ObtenerDatosCheb(u,y,t,S,l);
    ye(t) = tansig(W'*fhi');
    %ye(t) = W'*fhi';
    FHI(t,:)=fhi(:);
end
end

```

B.9. Función para la reducción por Gram - Schmidt

```

function [NRSS, NRSSt, L]=NRSSError(fhi,fhit,Y,Yt,N,ERR)
L = 1;
SUM = 1;

for k=1:N
    maxi = -100;
    for i=1:N
        if k == 1
            P(:,i) = fhi(:,i);
            V(i) = Y'*P(:,i)/(P(:,i)'*P(:,i));
            r(i) = V(i)*V(k,i)*P(:,i)'*P(:,i)/(Y'*Y);
            %r(i) = V(i)*P(:,i)'*Y/(Y'*Y);
            if maxi < r(i)
                maxi = r(i);
                argmax = i;
            end
        end
        else
            if isequal(i == s(:),zeros(k-1,1))
                sum = 0;
            end
        end
    end
end

```

```

        for j=1:k-1
            alfa(j,i) = fhi(:,i)'*Pe(:,j)/(Pe(:,j)'*Pe(:,j));
            sum = sum + alfa(j,i)*Pe(:,j);
        end
        P(:,i) = fhi(:,i) - sum;
        V(i) = Y'*P(:,i)/(P(:,i)'*P(:,i));
        r(i) = V(i)*V(i)*P(:,i)'*P(:,i)/(Y'*Y);
        %r(i) = V(i)*P(:,i)'*Y/(Y'*Y);
        if maxi < r(i)
            maxi = r(i);
            argmax = i;
        end
    end
end
end

R(k) = maxi;
s(k) = argmax;
Pe(:,k) = P(:,s(k));

if SUM > ERR
    if k == 1
        Q(k,k) = 1;
    else
        Q(k,k) = 1;
        aux = [alfa(:,argmax);1];
        Q(:,k) = aux;
    end
    Ve(k) = V(argmax);
    FHI(:,k) = fhi(:,argmax);
    %FHIt(:,k) = fhit(:,argmax); %prueba de red
    L = k;
end
SUM = SUM - R(k);
error(k) = SUM;
end

s;
L;
SUM;
NRSS=error(L);

[FHIt,erro]=EvalRed1(fhit,Yt,s,N,L);

NRSS=erro(L);

```

B.10. Programa para importar datos a MATLAB desde un archivo de texto

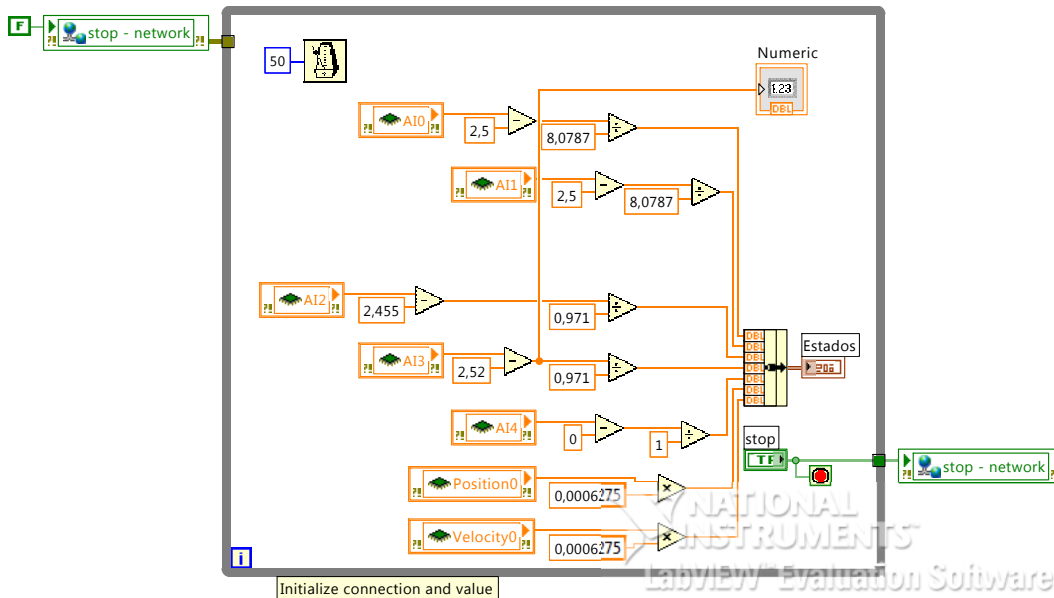
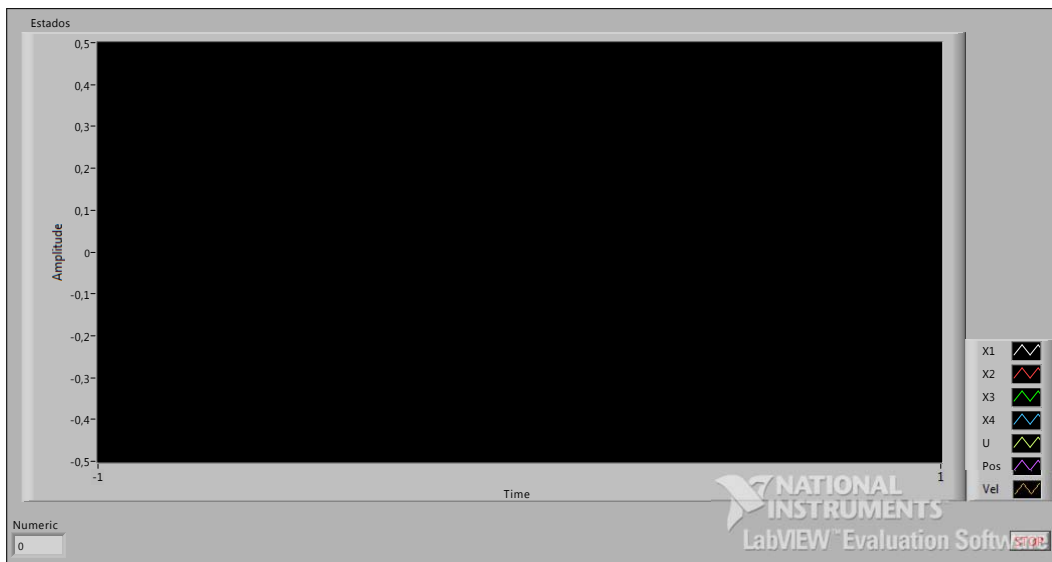
```

function importfile(fileToRead1)
rawData1 = importdata(fileToRead1);
[name] = fileparts(fileToRead1);
newData1.(genvarname(name)) = rawData1;
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('base', vars{i}, newData1.(vars{i}));
end
end

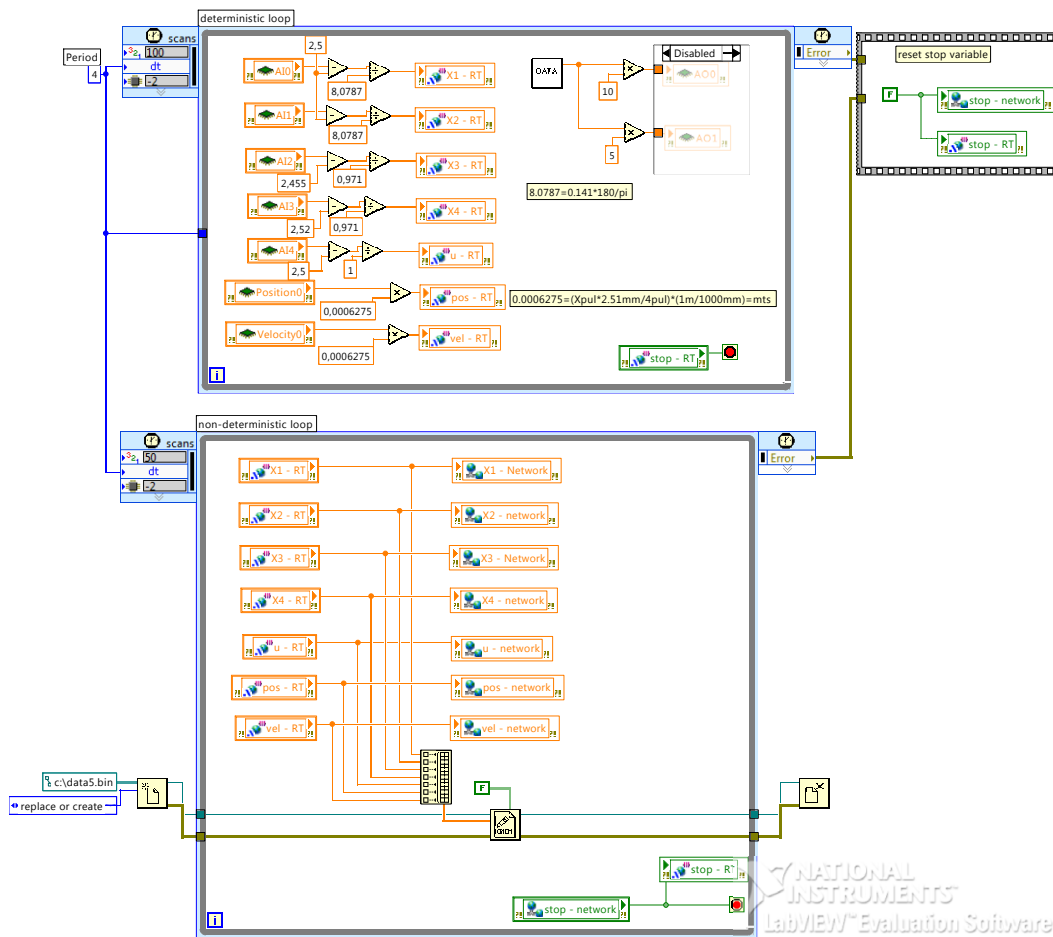
```

C. PROGRAMAS DE LABVIEW

C.1. Programa para adquirir datos del PISC



C.2. Programa para convertir de formato .bin a formato .txt



D. GLOSARIO

ANF	Algoritmo neuro-difuso
BGNN	Red neuronal Bayesian-Gaussian
BP	Retro-propagación
C.D.	Corriente directa
CBF	Polinomio de Chebyshev como función base
FL	Lógica difusa (Fuzzy Logic)
FNN	Red neuronal pre-alimentada (feedforward)
GL	Grados de libertad
MCO	Mínimos cuadrados ortogonales
MCR	Mínimos cuadrados recursivos
MLP	Perceptrón multicapa
MSE	Error de media cuadrática
MSS	Media de suma de cuadrados
NARX	No lineal autorregresivo con entrada externa
NRSS	Suma de cuadrados residual normalizada
PISC	Péndulo invertido simple sobre un carro móvil
RBFFNN	Red neuronal con funciones base radiales
RFNN	Red neuronal difusa recurrente
RNA	Red neuronal artificial
RNN	Red neuronal recurrente
RT	Tiempo real
SS	Suma de cuadrados
VPBF	Polinomio de Volterra como funciones base