Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Control Automático

Tesis

Redes Neuronales Recurrentes Complejas para Identificación y Control de Sistemas No Lineales

Que presenta

Jose Francisco Vega Lopez

Para obtener el grado de

Maestro en Ciencias

En la especialidad de

Control Automático

Director de tesis

Dr. Ieroham Solomon Barouh

Ciudad de México Octubre 2017

Agradecimientos

En principio, quiero agradecer al CONACyT y al CINVESTAV IPN, por darme la oportunidad de continuar con mis estudios de posgrado.

También reconozco al personal en general del departamento de control automático, ya que siempre hubo un buen trato en la prestación de servicios por parte de ellos.

Asimismo, quiero agradecer a los doctores del departamento de control automático, en especial a los doctores con los que tuve la oportunidad de tomar algún curso; obtuve muchos conocimientos de sus clases y los considero un gran ejemplo a seguir.

Agradezco a mi director de tesis, quien siempre estuvo dispuesto a ayudarme y atender mis dudas durante el tiempo en que estuvimos trabajando juntos.

Finalmente me gustaría reconocer a mi familia, quienes siempre me han brindado su apoyo y me han motivado a dar lo mejor de mí en cualquier cosa que haga, han sido un gran soporte y una parte fundamental de mi vida.

Abstract

In this thesis, the concept of artificial neural networks with parameters in the domain of complex numbers is taken up again. Specifically, it deals with a neural network topology, with which some works have already been presented in terms of the identification and control of systems; but among them there are some differences, mainly in the construction of the topology of the networks, besides not mentioning the sufficient details for its application. For these reasons, it is now intended to clearly describe the topology in the complex domain, together with sufficient details for its use.

Also, the extension to the complex domain of another similar neural network topology is done, which is more efficient for systems identification and control applications. Since this neural network was developed to be able to work taking into account the effect of the disturbances that arise in these processes, without the need to apply an independent signal filtering process to the network.

To achieve this, this work is divided into four chapters. The first chapter deals with the antecedents of the development of artificial neural networks, in order to understand their importance and emphasize the benefits they have. In the second chapter, the neural network topology that served as the basis for this work is taken up and its extension to the complex domain is shown. Subsequently in the third chapter, details are given to apply the topologies that were developed in the identification of systems, something similar is done in the fourth chapter but for the systems control. Finally, based on the results obtained, conclusions and some ideas are given for possible future work.

Resumen

En esta tesis, se retoma el concepto de redes neuronales artificiales con parámetros en el dominio de los números complejos. En específico, se trata con una topología de red neuronal, con la cual ya se habían presentado algunos trabajos en cuanto a la identificación y control de sistemas; pero entre ellos se tienen algunas diferencias, principalmente en la construcción de la topología de las redes, además de que no se mencionan los suficientes detalles para su aplicación. Por estos motivos, ahora se pretende describir de forma clara a la topología en el dominio complejo, junto con los suficientes detalles para su uso.

Asimismo, se hace la extensión al dominio complejo de otra topología similar de red neuronal, la cual es más eficiente para aplicaciones de identificación y control de sistemas. Ya que, esta red neuronal se desarrolló para poder trabajar tomando en cuenta el efecto de las perturbaciones que surgen en estos procesos, sin la necesidad de tener que aplicar un proceso de filtrado de señales independiente a la red.

Para lograr lo anterior, este trabajo se divide en cuatro capítulos. El primer capítulo trata de los antecedentes del desarrollo de redes neuronales artificiales, con lo que se busca comprender su importancia y recalcar las bondades que éstas tienen. En el segundo capítulo, se retoma la topología de red neuronal que sirvió de base para realizar este trabajo y se muestra su extensión al dominio complejo. Posteriormente en el tercer capítulo, se mencionan los detalles para aplicar las topologías que se desarrollaron en la identificación de sistemas, algo similar se hace en el cuarto capítulo pero para el de control de sistemas. Por último, en base a los resultados obtenidos, se dan conclusiones y algunas ideas para posibles trabajos futuros.

Índice General

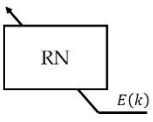
Índice de Figuras

Índice de Tablas

Siglas y Acrónimos

RNA	Red Neuronal Artificial.
RN	Red Neuronal.
RNC	Red Neuronal Compleja.
RNRE	Red Neuronal Recurrente Entrenable.
RNREE	Red Neuronal Recurrente Entrenable Extendida.
KFRNN	Siglas en ingles de “Kalman Filter Recurrent Neural Network”, red neuronal recurrente de filtro de Kalman.
CVNN	Siglas en ingles de “Complex Valued Neural Network”, red neuronal valuada en el dominio complejo.
CVRNN	Siglas en ingles de “Complex Valued Recurrent Neural Network”, red neuronal recurrente en el dominio complejo.
CVKFRNN	Siglas en ingles de “Complex Valued Kalman Filter Recurrent Neural Network”, red neuronal recurrente de filtro de Kalman en el dominio complejo.
BP	Siglas en ingles de “BackPropagation”, algoritmo del error recursivo o algoritmo de propagación hacia atrás.
LM	Siglas en ingles de “Levenberg-Marquardt”, algoritmo Levenberg-Marquardt.
CVBP	Siglas en ingles de “Complex Valued BackPropagation”, algoritmo de propagación hacia atrás en el dominio complejo.
CVLM	Siglas en ingles de “Complex Valued Levenberg-Marquardt”, algoritmo Levenberg-Marquardt en el dominio complejo.
SISO	Siglas en ingles de “Single Input Single Ouput”, sistema una entrada-una salida.
MIMO	Siglas en ingles de “Multi-Input, Multi-Ouput”, sistema de múltiples entradas-múltiples salidas.

Simbología

$Y \in \mathbb{R}$	Y pertenece al conjunto de los números reales.
$U \in \mathbb{C}$	U pertenece al conjunto de los números complejos.
X_{RE}	Parte real de X.
X_{IM}	Parte imaginaria de X.
i	Unidad imaginaria, se tiene que $i = \sqrt{-1}$.
A^T	Transpuesta de A.
C^*	Transpuesta conjugada de C.
$\lvert B \rvert$	Módulo de B.
D^{-1}	Matriz inversa de D (cuando D es invertible).
$\Gamma'(\cdot)$	Derivada de la función $\Gamma(\cdot)$.
	Matriz F de pesos bajo entrenamiento.
	Red neuronal bajo entrenamiento, minimizando a $E(k)$.

Capítulo 1. Antecedentes de Redes Neuronales Artificiales

En este capítulo se habla de las redes neuronales artificiales en general, con la finalidad de dar a conocer los antecedentes necesarios, para la comprensión del tema en el cual está basada esta tesis. Se aborda desde el desarrollo inicial de las redes neuronales, sus características principales y se describen sus procesos más importantes.

1.1. Redes Neuronales Artificiales

Las redes neuronales artificiales, conocidas como RNA o RN, son poderosas herramientas usadas dentro de diversos campos, entre los que destacan la informática, la robótica, el control de sistemas entre otros. El desarrollo inicial de las RN se inspiró en los sistemas nerviosos de los seres vivos (redes neuronales biológicas), ya que estos sistemas presentan características interesantes entre las que destacan: la memorización, la especialización, el paralelismo, la robustez, la densidad, el aprendizaje y la generalización.

A partir de diversos estudios sobre el sistema nervioso, tales como [1] y [2], se han desarrollado diversos modelos matemáticos que describen el comportamiento de las redes neuronales biológicas. Por ejemplo, en [3] se puede ver que las redes neuronales biológicas pueden interpretarse como un sistema dinámico, el cual depende de su ambiente y que la respuesta de cada neurona en la red depende del efecto que producen las otras neuronas conectadas dentro de la propia red.

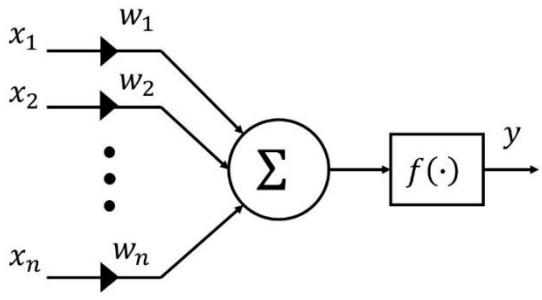


Figura 1.1 Representación gráfica de la neurona, unidad fundamental de una RNA.

Desde el punto de vista computacional, una RNA son todos aquellos modelos matemáticos y de cálculo que intentan imitar las capacidades y características de sus semejantes biológicos. En [4], [5] y [6] se dan más detalles sobre el origen y desarrollo histórico de las RNA, en esta tesis solo se habla de los conceptos más destacados.

Una RNA está formada por elementos relativamente simples de cálculo, conocidos como neuronas, las cuales se conectan entre ellas con una cierta organización, a esta organización se le

llama topología. Destacan dos tipos de neuronas que son los perceptrones y los ADALINE (revisar [2] y [7]), siendo los primeros las estructuras más usadas. En la figura 1.1 se puede ver de forma gráfica el modelo básico de una neurona, que matemáticamente se puede expresar como:

$$y = f\left(\sum_{i}^{n} x_i w_i\right) \tag{1.1}$$

Comparando la figura 1.1 y la ecuación (1.1) destacan algunos elementos, los cuales se describen a continuación:

- Los elementos de entrada de la neurona denotados por x_i, también conocidos como conjunto de sinapsis, cada elemento de este conjunto es ponderado por un peso sináptico w_i respectivamente. En algunas ocasiones se incluye una entrada constante con valor igual a -1 o 1 y ésta es ponderada por un peso w_0, al cual se le puede nombrar de dos formas; cuando la entrada vale 1 a w_0 se le conoce como bias, o bien, a w_0 se le llama umbral si es que la entrada tiene el valor de -1.
- Se distingue un sumador, el cual es el cuerpo de la neurona y se encarga de sumar todas las entradas ponderadas.
- Una función $f(\cdot)$, que es conocida como función de activación, esta función se encarga de generar la señal de salida y de la neurona. Se puede interpretar al argumento de la función como una combinación lineal de las entradas de la neurona.

Existen diversos tipos de funciones de activación. Para que una función matemática pueda ser considerada una función de activación, ésta debe de cumplir como requisitos ser derivable y acotada, usualmente se usan funciones no lineales. Entre las funciones más utilizadas están: la función escalón, la función sigmoide, la función tangente hiperbólica y la función saturación. Las funciones mencionadas anteriormente tienen la propiedad de que su derivada está en función de ellas mismas, lo cual es aprovechado para realizar el ajuste de los parámetros de la red, proceso del que se hablara más adelante.

Una característica importante de las neuronas, es que en cada uno de los pesos sinápticos que las conforman se almacena información sobre la forma en que responden ante los estímulos producidos por sus entradas. Esta información es utilizada y modificada por medio un proceso conocido como aprendizaje, el cual modifica a los parámetros internos de la RNA.

Ya que se ha hablado de los elementos que conforman a una RNA, se puede dar una definición formal de ésta, la cual se puede encontrar en [4].

Definición 1.1: *una RNA es un procesador masivo y en paralelo construido de unidades de procesamiento sencillas, la cual tiene una gran disposición natural de almacenar conocimiento experimental y hacerlo disponible para su uso.*

Las RNA presentan las siguientes propiedades:

- No linealidad: una red compuesta de neuronas no lineales interconectadas entre ellas es no lineal.
- Mapeo Entrada-Salida: la RNA construye un mapeo de los elementos utilizados en su entrada y sus correspondientes respuestas.
- Adaptabilidad: las RNA pueden adaptarse a su ambiente, una RNA que opera en un ambiente específico puede ser reajustada para tratar con cambios menores en el ambiente en el que opera.
- Información contextual: la información adquirida por la red queda representada por el estado de activación de las neuronas, así como la estructura global de la red.
- Tolerancia a fallas: si una neurona o sus conexiones son dañadas, no necesariamente se afecta el desempeño global de la red. La RNA exhibe una degradación en su desempeño gradual en vez del fallo total en su funcionamiento.
- Implementación a gran escala.

1.1.1. Clasificación de las RNA

Existen diversos tipos de clasificación para las redes neuronales, pero la más empleada se basa en la manera en que la información se transmite en la red, de esta forma se tienen a las RNA con conexiones hacia adelante y a las RNA recurrentes.

Las RNA con conexiones hacia adelante, también conocidas como RNA estáticas, tienen como característica principal que la respuesta de la red depende únicamente de sus entradas, es decir, solo se realiza un mapeo desde el espacio de entrada al espacio de salida de la red, por lo que la respuesta de la RN no depende del tiempo en el cual se aplica.

En general, las RNA estáticas se emplean en tareas de clasificación de patrones, aproximación de funciones y memoria asociativa. Dentro de las RNA estáticas existen las RNA multicapa y las RNA de funciones radiales básicas. Las RNA multicapa se construyen por neuronas organizadas en capas (ver figura 1.2 inciso A, se muestra un ejemplo con una capa de entrada, dos capas ocultas y una capa de salida, se puede notar que los nodos tienen una estructura similar a la mostrada en la figura 1.1), en donde cada capa de la red tiene como entrada a todas o a un conjunto de las salidas de la capa anterior. Este tipo de topología tiene algunas desventajas, tales como no tener una estructura fija y que el ajuste de sus parámetros es lento.

Por otro lado, una RNA de funciones radiales básicas es una alternativa a las RNA multicapa (ver figura 1.2 inciso B). El ajuste de los pesos de este tipo de red es más rápido, porque tiene una estructura fija conformada exactamente por tres capas en el siguiente orden: la capa de entrada, que consiste en las entradas de la red x_i; la capa oculta formada por neuronas que tienen como función de activación una función radial básica $G(\cdot)$ (por ejemplo la campana de Gauss), estas

neuronas son distribuidas equitativamente; y por ultimo esta la capa de salida, que consiste en sumar las salidas de las neuronas de la capa oculta. Cada capa está completamente vinculada con la siguiente, las conexiones entre la capa de entrada y la capa oculta no son ponderadas, mientras que las conexiones entre la capa oculta y la de salida si son ponderadas. Una gran desventaja de este tipo de red es que en ocasiones su topología es complicada de aplicar.

Las RNA recurrentes, también denominadas RNA dinámicas, nacen a partir de que hay problemas que necesitan que un sistema tenga una respuesta dependiente de su estado anterior, por lo que se recurre a estructuras que tengan una dinámica interna propia. Entre las tareas que destacan para las RN recurrentes se encuentran la predicción de series, la identificación y control de sistemas dinámicos. Dentro de las RNA recurrentes existen diversos modelos, los cuales se inspiran de las topologías de las RN estáticas, razón por la cual tienen muchas veces la misma forma que las RNA estáticas, con la diferencia de que las RNA dinámicas también toman como entradas a retados de las salidas producidas por la misma red. Entre las topologías que destacan se encuentra la red de Hopfield y la máquina de Boltzman, en [5] y [8] se muestran más detalles de éstas.

Además de las topologías descritas anteriormente existen tambien topologías hibridas, es decir, se combinan las estructuras de las RNA recurrentes y las estructuras de las RNA estáticas, tales como las redes de Jordán y Elman, ver en [4], así como algunas redes que se han desarrollado recientemente como en [9] y [10].

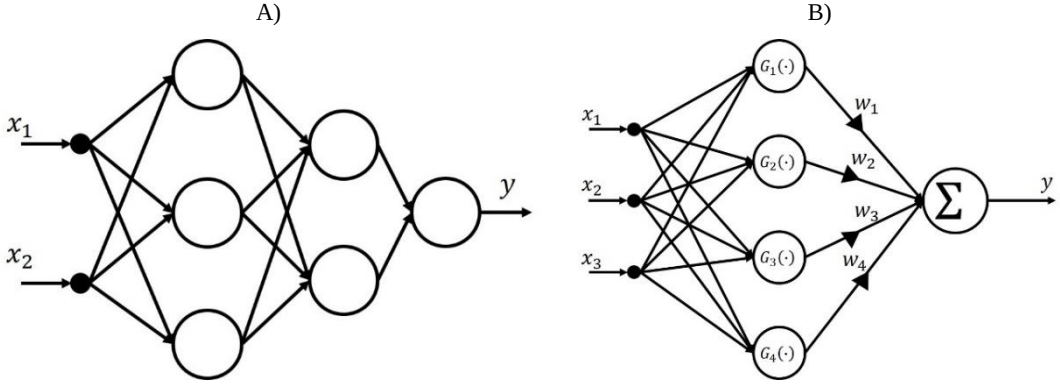


Figura 1.2 Topologías de RNA estáticas.
A) RNA multicapa. B) RNA de funciones radiales básicas.

Independiente de la topología con la que se llegue a trabajar, las RNA en general se definen en tiempo discreto, debido a que se implementan computacionalmente; los datos con los que trabajan las RNA son tomados a partir de señales o patrones muestreados. Por lo que, cuando se trabaja con RNA se hace uso en muchas ocasiones del teorema de muestreo de Shannon [11], el cual establece lo siguiente: si la frecuencia de muestreo es lo suficientemente alta, en comparación con la componente de más alta frecuencia que se incluye en la señal en tiempo continúo, las características de amplitud de la señal en tiempo continuo se pueden preservar en la envolvente

de la señal muestreada. Entonces, para conocer el periodo de muestreo T_o necesario para poder recrear una señal en tiempo continuo, se usa la siguiente formula:

$$T_o \leq \frac{1}{2f_m} \tag{1.2}$$

Donde f_m representa la componente de más alta frecuencia del sistema. Mientras se cumpla la desigualdad (1.2), se garantiza que, las características de las señales en tiempo continuo se conservan para su uso en las RNA.

Las RNA han tenido un gran número de aplicaciones, debido principalmente a que su estructura fundamental, la neurona, se acopla perfectamente a la descripción del teorema universal de aproximación [4], que estipula lo siguiente:

Teorema 1.1: *Sea $\varphi(\cdot)$ una función continua, no constante, acotada y monótona creciente. Sea I_{mo} un hipercubo unitario de dimensión m_o. El espacio de funciones continuas en I_{mo} es denotado por $C(I_{mo})$. Entonces, dada cualquier función $f \in C(I_{mo})$ y $\varepsilon > 0$, existe un entero M y conjuntos de constantes reales α_i, β_i y w_{ij} donde $i = 1, \dots, m_1$ y $j = 1, \dots, m_0$ tal que se define:*

$$F(x_1, \dots, x_{m_o}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{i=1}^{m_0} w_{ij} x_j + \beta_i\right)$$

Con una realización aproximada de la función $f(\cdot)$, esto es:

$$\left|F(x_1, \dots, x_{m_0}) - f(x_1, \dots, x_{m_0})\right| < \varepsilon$$

Para todos $x_1, x_2, \dots, x_{m_0}$ que yasen en el espacio de entradas de la red.

Por lo anterior, las RNA son atractivas en aplicaciones de identificación y control de sistemas, siendo de los primeros trabajos en estos temas el mostrado en [12]. Posteriormente, en [13] se exponen diversas topologías de RNA aplicadas en diversos tipos de sistemas mecánicos, en [14] se le dan a las RNA un enfoque más hacia el control adaptable de sistemas. En otros trabajos como en [15], [16] y [17] se muestra que las RNA se pueden acoplar con la lógica difusa, obteniendo resultados favorables. Otros trabajos más recientes hacen uso de topologías hibridas de RNA, como en [18], [19], [20], [21], [22], [23] y [24] en donde se emplean a las redes como estimadores de los parámetros de diversos sistemas; a partir de la información obtenida de las redes se definen leyes de control para tales sistemas.

1.1.2. Aprendizaje y Generalización las RNA

Se tienen dos procesos importantes para las RNA, el aprendizaje y la generalización. El aprendizaje de una RN se refiere al proceso a través del cual los parámetros de la red se modifican; es un proceso continuo de estimulación y reducción de un criterio de error dependiente de la tarea que se quiere llevar a cabo. El proceso de generalización, trata sobre la RN produciendo una respuesta razonable a estímulos que no se presentan durante el proceso de aprendizaje; si la RN aprendió el comportamiento de un sistema usando determinados datos, se espera que, al cambiar dichos datos la RN sea capaz de producir una respuesta similar a la del sistema con esa misma información.

Entrando más en detalle en el proceso de aprendizaje, también conocido como entrenamiento de la RNA, se puede resumir de la siguiente forma: la RN es estimulada por su ambiente, en consecuencia, la RN realiza cambios internos y debido a esos cambios la RN responde de una forma diferente. Hay dos formas para llevar a cabo el aprendizaje:

- En línea: a partir de un proceso en tiempo real, se toman datos de su dinámica y conforme el proceso va evolucionando la RN va cambiando sus parámetros para adaptarse a éste. Tiene como desventaja que la RN puede tardar en reproducir la dinámica deseada, y a veces es difícil ajustar el proceso de aprendizaje.
- Fuera de línea: en este caso, se toman datos guardados de la dinámica deseada y con esta información se entrena a la RN. Debido a esto, el entrenamiento de la RN se puede realizar por lotes o épocas de entrenamiento (una época de entrenamiento es la colección de datos de la dinámica que se quiere aprender). La ventaja de hacer el proceso de aprendizaje fuera de línea es que se puede hacer cíclico, hasta obtener los resultados deseados. Pero como desventaja, se tiene que en los datos guardados cabe la posibilidad de que estos no contengan la suficiente información de la dinámica deseada, dando como consecuencia que el entrenamiento de la RN no se haga correctamente.

El proceso de aprendizaje de una RNA se basa en paradigmas o modelos, así como en los algoritmos para realizar este proceso. Un paradigma de aprendizaje se refiere a la forma en que interactúa la RNA con su ambiente, se basa en los algoritmos encargados de ejecutar la tarea de ajuste de pesos sinápticos de la RNA. Existen tres tipos de paradigmas:

- Aprendizaje supervisado: se supone la existencia de un maestro que tiene como función orientar el aprendizaje de la RN, proporciona ejemplos de lo que se desea que sea aprendido y la red asume que estos ejemplos están libres de error. Una vez que se asimilaron los ejemplos (entrenamiento), la RN debe de ser capaz de reproducir la causa-efecto aprendida sin error o con algún grado aceptable de error (generalización), en especial cuando se estimule a la RN con muestras diferentes a las proporcionadas en los ejemplos.

- Aprendizaje reforzado: la RN tiene la oportunidad de saber, después de un conjunto de acciones de causa-efecto, si se lograron cumplir con los objetivos deseados. En términos simples, existe un elemento de crítica al final del proceso en vez de tener ejemplos durante el entrenamiento.
- Aprendizaje no supervisado: se carece de un elemento externo que indique que debe ser aprendido, se considera que la RN es capaz de extraer las características que necesita de la misma información que se proporciona para el entrenamiento y añade ésta a su comportamiento.

A continuación, se nombran los algoritmos más significativos que se encargan del ajuste de pesos sinápticos de una RNA, en [4] estos algoritmos se muestran con más detalle y se habla de algunos otros.

- Aprendizaje por corrección del error: ha servido de base para diversos algoritmos más avanzados, hace uso de una señal de error generada por la diferencia entre una salida deseada y la salida que ofrece la RN. Tomando una neurona r, construida por un nodo en la capa de salida de una RN, con forma similar a la mostrada en la figura 1.1, la señal de error se describe a continuación:

$$e_r(k) = y_r^d(k) - y_r(k) \tag{1.3}$$

Con:

- $e_r(k)$, señal de error producida por la neurona r.
- y_r^d, salida deseada para la neurona r.
- y_r, salida de la neurona r.

El objetivo es minimizar una función de costo que toma en cuenta a todas las neuronas de la capa de salida, la función a minimizar es el error cuadrático y se define como:

$$\varepsilon(k) = \frac{1}{2}\sum_r e_r^2(k) \tag{1.4}$$

La forma en que se minimiza a la ecuación (1.4) debe de estar en términos de los pesos sinápticos, para lo cual se utiliza la regla delta [4]:

$$\Delta w_{ij}(k) = w_{ij}(k+1) - w_{ij}(k) = \eta e_i(k) x_j(k) \tag{1.5}$$

Donde a η se le llama índice de aprendizaje o velocidad del algoritmo de aprendizaje, este parámetro asegura la estabilidad del algoritmo.

Entonces, el ajuste de pesos sinápticos (1.5) tiende a ir en la dirección del mínimo global de la superficie del error formada por la función (1.4); si la RN consta de neuronas lineales, la superficie del error es una función cuadrática con un mínimo global. Pero si

las neuronas son no lineales, la superficie del error tendrá un mínimo global y posiblemente varios mínimos locales, que podrían representar un problema: ya que el algoritmo inicia en un punto arbitrario de la superficie del error y va paso a paso en busca del mínimo global, pero puede caer en un mínimo local y quedarse ahí.

- Aprendizaje basado en memoria: la experiencia pasada de la salida de RN, se guarda explícitamente en una memoria de ejemplos entrada-salida $\{x_i y_i^d\}_{i=1}^N$. Cuando la respuesta de una entrada no conocida x_{test} es requerida, el algoritmo responde recuperando y analizando los datos de entrenamiento que se encuentran dentro de una vecindad de x_{test}.
- Aprendizaje Hebbiano: el ajuste de los pesos sinápticos entre dos neuronas conectadas entre sí dentro de una RN se incrementa selectivamente, si es que ambas son activadas simultáneamente. El ajuste se decrementa si las neuronas se activan en tiempos diferentes.
- Aprendizaje competitivo: las neuronas en cada capa de una RN compiten entre ellas para elegir cuál de todas alcanza su valor máximo de salida. Es llevado a cabo por conexiones recurrentes de activación e inhibición por parte de las neuronas vecinas.
- Aprendizaje de Boltzman: este aprendizaje es de tipo estocástico. Si las neuronas de una RN con una topología recurrente operan de forma binaria, con los estados de encendido y apagado denotados por 1 y -1 respectivamente. Para el ajuste de pesos sinápticos, se puede considerar una función de energía de la siguiente forma:

$$E = -\frac{1}{2}\sum_{j,n,j\neq n} w_{nj}x_n x_j \qquad (1.6)$$

Con x_j como el estado de la neurona j, w_{nj} como el peso sináptico entre las neuronas n y j. La red elije una neurona al azar en cada iteración del proceso de entrenamiento, cambiando su estado x_j a $-x_j$ en una temperatura virtual T, siguiendo la regla de probabilidad:

$$P(x_n \to -x_n) = \frac{1}{1 + e_n^{-\Delta E T^{-1}}} \qquad (1.7)$$

Con la aplicación repetida de esta regla, la red llegará a un "equilibrio térmico" por lo que el entrenamiento habrá terminado.

También existen algoritmos de aprendizaje que emplean métodos de optimización, los cuales se basan en el algoritmo de corrección por error como se mencionó antes y son usados con frecuencia actualmente. Estos algoritmos consisten, básicamente, en tomar la función del error medio

cuadrático (1.4) en términos de los pesos sinápticos w y a partir de ésta encontrar a los pesos sinápticos óptimos $\bar{\bar{w}}$, de tal forma que se satisface la siguiente expresión:

$$\varepsilon(\bar{\bar{w}}) \leq \varepsilon(w) \tag{1.8}$$

Lo anterior se interpreta como un problema de optimización sin restricciones, donde se busca la solución óptima, minimizando a la ecuación de costo $\varepsilon(w)$ respecto a w. La condición necesaria de optimización se muestra en (1.9) y corresponde a un algoritmo de optimización de primer orden.

$$\nabla\varepsilon(\bar{\bar{w}}) = 0 \tag{1.9}$$

Los algoritmos de optimización sin restricciones, basados en el descenso local iterativo, inician desde una condición inicial $w(0)$ generando así una secuencia de pesos, de tal forma que la función de costo es reducida en cada iteración (ecuación (1.10)).

$$\varepsilon\big(w(k+1)\big) \leq \varepsilon\big(w(k)\big) \tag{1.10}$$

Eventualmente se espera llegar a la solución óptima $\bar{\bar{w}}$, pero como se mencionó anteriormente esto no es seguro, ya que el algoritmo puede quedarse en un mínimo local que se considere suficientemente bueno.

Se tiene entonces, un conjunto de pesos sinápticos iniciales $w(0)$, se calcula la dirección de máxima variación del error que se define como el gradiente $\nabla\varepsilon\big(w(k)\big)$ en $w(0)$; el ajuste de pesos se efectúa en sentido contrario del indicado por el gradiente (dirección de máximo decrecimiento). De esta manera, se espera producir un descenso por la superficie formada por error hasta llegar a un mínimo. Lo anterior queda expresado de la siguiente forma:

$$\begin{aligned} w(k+1) &= w(k) - \eta\nabla\varepsilon(w) \\ w(k+1) &= w(k) - \eta g(k) \end{aligned} \tag{1.11}$$

La condición de optimización mostrada en (1.9) se debe cumplir. Para mostrar esto, se inicia un desarrollo a partir de (1.10), en donde se realiza una expansión en series de Taylor de primer orden alrededor de $w(k)$ para aproximar a $\varepsilon\big(w(k+1)\big)$.

$$\varepsilon\big(w(k+1)\big) \approx \varepsilon\big(w(k)\big) + g^T(k)\Delta w(k) \tag{1.12}$$

Definiendo:

$$\begin{aligned} g(k) &= \nabla\varepsilon(w) \\ \Delta w(k) &= w(k+1) - w(k) \end{aligned}$$

Y continuando con el desarrollo, se tiene:

$$\varepsilon\big(w(k+1)\big) = \varepsilon\big(w(k)\big) + \eta g^T(k) g(k) \qquad (1.13)$$
$$\varepsilon\big(w(k+1)\big) = \varepsilon\big(w(k)\big) + \eta \|g(k)\|^2$$

El parámetro η tiene un gran peso en el comportamiento del algoritmo de aprendizaje. Cuando η es pequeño se tiene una respuesta sobreamortiguada por parte del algoritmo, por lo que la trayectoria que sigue $w(k)$ es suave. Si η es grande, la respuesta del algoritmo es subamortiguada, por lo que $w(k)$ sigue una trayectoria oscilatoria. En [4], se ha establecido que η debe de encontrarse en un conjunto de valores entre cero y uno, si se cumple esta condición se considera que el algoritmo tiene menos probabilidades de fallar.

El método de propagación del error hacia atrás, conocido en inglés como backpropagation (BP), es el método más popular y el más probado dentro de los algoritmos de optimización de primer orden. El funcionamiento de este algoritmo es el siguiente: se introduce a la RN un patrón de entrada, este patrón se propaga a través de toda la red para producir un patrón de salida, entonces se forma un error entre el patrón de salida deseado y el patrón que la RN produce. El error resultante se propaga hacia atrás; desde la salida de la RN, pasando por las capas intermedias de la red hasta llegar a la entrada de la misma. El error que pasa por cada neurona es proporcional a la contribución de éstas en el error total de la red, de esta forma se realiza el ajuste de parámetros de toda la RN.

A continuación, se ofrece una descripción matemática del algoritmo BP, en donde se emplea a la ecuación (1.3) y a la ecuación (1.4) como funciones de error y de costo respectivamente. Primero se extiende la descripción de la neurona hecha en (1.1), de tal forma que se tiene:

$$u_r(k) = \sum_{j=1}^{l} w_{rj} x_j \qquad (1.14)$$
$$v_r(k) = u_r(k) + b_r$$
$$y_r(k) = f_r\big(v_r(k)\big)$$

Donde r representa a la neurona de salida sobre la que se esta trabajando, j es el número de elementos de entrada (pueden ser salidas de neuronas de capas intermedias) que tiene dicha neurona, pueden ser hasta l elementos. La variable $v_r(k)$ es el cuerpo de la neurona, que toma en cuenta a las entradas ponderadas por los pesos sinápticos $u_r(k)$ y al bias o compensación b_r, en caso de que éste exista. Finalmente, f_r representa a una función de activación, necesaria para la obtención de la salida $y_r(k)$ de la RN.

El algoritmo de propagación hacia atrás aplica una corrección $\Delta w_{rj}(k)$ al peso sináptico $w_{rj}(k)$, el incremento que se realiza es proporcional al gradiente descendiente.

$$\Delta w_{rj}(k) = \frac{\partial \varepsilon(k)}{\partial w_{rj}(k)} \quad (1.15)$$

De acuerdo con la regla de la cadena del cálculo, se puede expresar a (1.15) como:

$$\frac{\partial \varepsilon(k)}{\partial w_{rj}(k)} = \frac{\partial \varepsilon(k)}{\partial e_r(k)} \frac{\partial e_r(k)}{\partial y_r(k)} \frac{\partial y_r(k)}{\partial v_r(k)} \frac{\partial v_r(k)}{\partial w_{rj}} \quad (1.16)$$

En (1.16) se tiene un factor de sensibilidad, representado por $\partial \varepsilon(k)/\partial w_{rj}(k)$, que determina la dirección en la búsqueda de los pesos sinápticos w_{rj}.

Diferenciando ambos lados de la ecuación (1.4) con respecto a $e_r(k)$ se tiene que:

$$\frac{\partial \varepsilon(k)}{\partial e_r(k)} = e_r(k) \quad (1.17)$$

Diferenciando a la ecuación (1.3) con respecto a $y_r(k)$ se tiene:

$$\frac{\partial e_r(k)}{\partial y_r(k)} = -1 \quad (1.18)$$

Diferenciando a $y_r(k)$ del conjunto de ecuaciones (1.14) con respecto a $v_r(k)$ se tiene:

$$\frac{\partial y_r(k)}{\partial v_r(k)} = f_r'\big(v_r(k)\big) \quad (1.19)$$

Diferenciando a $v_r(k)$ del conjunto de ecuaciones (1.14) con respecto a $w_{rj}(k)$ se tiene:

$$\frac{\partial v_r(k)}{\partial w_{rj}(k)} = x_j(k) \quad (1.20)$$

Entonces, en base al desarrollo anterior la expresión (1.16) se puede reescribir como:

$$\frac{\partial \varepsilon(k)}{\partial w_{rj}(k)} = -e_r(k) f_r'\big(v_r(k)\big) x_j(k) \quad (1.21)$$

La corrección $\Delta w_{rj}(k)$ que se aplica a w_{rj} queda en función de la regla delta:

$$\Delta w_{rj}(k) = -\eta \frac{\partial \varepsilon(k)}{\partial w_{rj}(k)} \quad (1.22)$$

Donde η se interpreta como la velocidad del algoritmo de aprendizaje, este parámetro tiene la misma función de la que se habló antes. De acuerdo con (1.21) y (1.22) se obtiene la siguiente ecuación:

$$\Delta w_{rj}(k) = -\eta \delta_r(k) x_j(k) \quad (1.23)$$

Con $\delta_r(k)$ como el gradiente local, que se define como:

$$\delta_r(k) = -\frac{\partial \varepsilon(k)}{\partial v_r(k)} = -\frac{\partial \varepsilon(k)}{\partial e_r(k)} \frac{\partial e_r(k)}{\partial y_r(k)} \frac{\partial y_r(k)}{\partial v_r(k)} = e_r(k) f_r'\big(v_r(k)\big) \quad (1.24)$$

Cuando la neurona r está localizada en la capa de salida de la red, la actualización de $w_{rj}(k)$ se hace usando a (1.23) y (1.24), cuando la neurona r se encuentra en una capa oculta, se redefine el gradiente local $\delta_j(k)$ para la neurona j de la capa oculta como:

$$\delta_j(k) = -\frac{\partial \varepsilon(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} = -\frac{\partial \varepsilon(k)}{\partial y_j(k)} f_j'\left(v_j(k)\right) \quad (1.25)$$

Retomando a (1.4) y diferenciándola con respecto de $y_j(k)$ se tiene:

$$\frac{\partial \varepsilon(k)}{\partial y_j(k)} = \sum_r e_r(k) \frac{\partial e_r(k)}{\partial y_j(k)} \quad (1.26)$$

Aplicando la regla de la cadena para $\partial e_r(k)/\partial y_j(k)$, (1.26) se reescribe como:

$$\frac{\partial \varepsilon(k)}{\partial y_j(k)} = \sum_r e_r(k) \frac{\partial e_r(k)}{\partial v_r(k)} \frac{\partial v_r(k)}{\partial y_j(k)} \quad (1.27)$$

Retomando a (1.3) y diferenciándola con respecto a $v_r(k)$ se tiene:

$$\frac{\partial e_r(k)}{\partial v_r(k)} = -f_r'\big(v_r(k)\big) \quad (1.28)$$

Del conjunto de ecuaciones (1.14), tomando solo a $v_r(k)$ y diferenciándola con respecto a $y_j(k)$ se tiene:

$$\frac{\partial v_r(k)}{\partial y_j(k)} = w_{rj}(k) \quad (1.29)$$

Utilizando a (1.28) y (1.29) en (1.27) se obtiene la derivada parcial que se busca:

$$\frac{\partial \varepsilon(k)}{\partial y_j(k)} = -\sum_r e_r(k) f_r'\big(v_r(k)\big) w_{rj}(k) = -\sum_r \delta_r(k) w_{rj}(k) \quad (1.30)$$

Donde $\delta_r(k)$ es el gradiente local usado en la definición en (1.24). Por último, utilizando a (1.30) se llega a una formula conocida como "propagación dacia atrás para el gradiente local $\delta_j(k)$" descrita como:

$$\delta_j(k) = f_j'\left(v_j(k)\right)\sum_r \delta_r(k)w_{rj}(k) \quad (1.31)$$

Cuando la neurona j esta oculta. La corrección $\Delta w_{ji}(k)$ del peso sináptico $w_{ji}(k)$ que conecta a la neurona i con la j se define por la regla delta:

$$\Delta w_{ij}(k) = \eta\delta_j(k)x_i(k) \quad (1.32)$$

Donde el cálculo del gradiente local depende de si se trata de una neurona de la capa de salida r o de una neurona de alguna capa oculta j. Si es el gradiente local de una neurona de la capa de salida se utiliza a (1.24) y si es de una capa oculta se utiliza a (1.31).
En [4], se muestra una modificación para (1.32), en la que se agrega el término momento α, para corregir la oscilación propia del algoritmo BP:

$$\Delta w_{ji}(k) = \eta\delta_j(k)x_i(k) + \alpha\Delta w_{ji}(k-1) \quad (1.33)$$

Existen también los algoritmos de optimización de segundo orden, los cuales se basan en algún método como el de Newton (ecuación 1.34), en donde el objetivo del algoritmo es minimizar una aproximación cuadrática de $\varepsilon(w)$ empleando la expansión en series de Taylor de segundo orden alrededor de $w(k)$.

$$\Delta\varepsilon\big(w(k)\big) \coloneqq \varepsilon\big(w(k+1)\big) - \varepsilon\big(w(k)\big) \approx g^T(k)\Delta w(k) + \frac{1}{2}\Delta w^T(k)H(k)\Delta w(k) \quad (1.34)$$

Donde $H(k)$ es la matriz Hessiana de $\varepsilon(k)$. Diferenciando a (1.34) con respecto a Δw, el cambio $\Delta\varepsilon$ se minimiza cuando:

$$g(k) + H(k)\Delta w(k) = 0 \quad (1.35)$$

Que implica:

$$\Delta w(k) = -H^{-1}(k)g(k) \quad (1.36)$$

Pero también:

$$w(k+1) = w(k) - H^{-1}(k)g(k) \quad (1.37)$$

La ventaja de este método de actualización de pesos sinápticos, es que no presenta oscilaciones y usa menos iteraciones que los anteriores [5]. También se tienen algunas desventajas, tales como que la ecuación (1.4) debe de ser dos veces diferenciable con respecto a los pesos sinápticos y que la matriz Hessiana debe ser definida positiva e invertible.
Por lo anterior, se han desarrollado métodos conocidos como "cuasi Newton", que son similares a los métodos de primer orden. Estos métodos se describen a partir de las siguientes ecuaciones:

$$w(k+1) = w(k) + \eta s(k) \quad (1.38)$$

$$s(k) = -S(k)g(k) \quad (1.39)$$

Donde, $s(k)$ es un vector de dirección definido en términos del gradiente $g(k)$ y $S(k)$ es una matriz definida positiva que se ajusta en cada iteración, de forma que $s(k)$ sea aproximado a la dirección del método de Newton de segundo orden. Un método cuasi Newton usa información del método de segundo orden de Newton, sin requerir calcular la matriz Hessiana y por tanto su inversa, solamente se emplea a las iteraciones sucesivas $k+1$ y k de $g(k)$. Continuando con el desarrollo, se definen:

$$q(k) = g(k+1) - g(k) \quad (1.40)$$

$$\Delta w(k) = w(k+1) - w(k) \quad (1.41)$$

La curvatura de la función de error se puede aproximar por medio de la siguiente ecuación:

$$q(k) \approx \left(\frac{\partial g(k)}{\partial w(k)}\right) \Delta w(k) \quad (1.42)$$

En particular, dado un incremento linealmente independiente en el vector de pesos w, esto es $\Delta w(0), \Delta w(1), \dots, \Delta w(k-1)$, y su respectivo incremento de gradientes, la matriz Hessiana se aproxima como:

$$H(k) \approx [q(0) \quad q(1) \quad \dots \quad q(k-1)][q(0) \quad q(1) \quad \dots \quad q(k-1)]^{-1} \quad (1.43)$$

Y su inversa por medio de:

$$H^{-1}(k) \approx [\Delta w(0) \quad \Delta w(1) \quad \dots \quad \Delta w(k-1)][q(0) \quad q(1) \quad \dots \quad q(k-1)]^{-1} \quad (1.44)$$

Con (1.44) se puede obtener para (1.39) al término $S(k)$. Ante la dificultad para obtener a la matriz Hessiana y de lo complicado de los cálculos de los algoritmos de segundo orden, se han desarrollado además algoritmos híbridos, los cuales toman características tanto de los algoritmos de primer orden, así como de los de segundo orden. Entre ellos destaca el algoritmo de Levenberg-Marquardt (LM), ver [25] y [26].

Para hacer una descripción del algoritmo de LM se retoma a la ecuación del error cuadrático (1.4), el vector gradiente y la matriz Hessiana se definen de la siguiente forma:

$$\varepsilon\big(w(k)\big) = \sum_{j=1}^{L} e_j^2\big(w(k)\big) \quad (1.45)$$

$$g(k) = J^T(k)E\big(w(k)\big) \quad (1.46)$$

$$H(k) = J^T(k)J(k) + S\big(w(k)\big) \quad (1.47)$$

Con $J(k)$ como la matriz jacobina de la red, $E\big(w(k)\big)$ es un vector de errores de las salidas de la red. La matriz $S\big(w(k)\big)$ se calcula de la siguiente forma:

$$S\big(w(k)\big) = \sum_{j=1}^{L} e_j^2\big(w(k)\big)\nabla e_j\big(w(k)\big) \quad (1.48)$$

Para el método de Newton se supone $S\big(w(k)\big) \approx 0$, por lo que la actualización de pesos (1.36), usando a (1.46) y (1.47) se reescribe de la siguiente forma:

$$\Delta w(k) = -[J^T(k)J(k)]^{-1}J^T(k)E\big(w(k)\big) \quad (1.49)$$

La ecuación (1.49) es una variante conocida como el método de Gauss-Newton, a la cual se le debe de agregar un término de regularización ρ para obtener la actualización de pesos usando por el método de Levenberg-Marquardt:

$$\Delta w(k) = -[J^T(k)J(k) + \rho I]^{-1}J^T(k)E\big(w(k)\big) \quad (1.50)$$

Donde I es una matriz identidad de dimensiones adecuadas, ρ puede variar con cada iteración y depende de (1.45); si ésta aumenta ρ es multiplicado por un coeficiente β, si disminuye entonces ρ es dividido por β. Para valores grandes de ρ, el método que se ha descrito se asemeja a los métodos de gradiente descendiente de primer orden, mientras que para valores pequeños se asemeja los algoritmos de segundo orden. El objetivo del algoritmo LM se cumple si la función (1.45) ha sido reducida dentro de un intervalo admisible.

Existe otra variación más eficiente del algoritmo de LM, la cual se desarrolla a partir de un entrenamiento recursivo en el cual se utiliza un vector de errores instantáneos $E(k)$, definido como:

$$E\big(w(k)\big) = \Big(Y^d(k) - Y(k)\Big) \quad (1.51)$$

Con $Y^d(k)$ como el vector de salida deseado y $Y(k)$ como el vector de salida generado por la RN. Siguiendo lo establecido en [27], se puede obtener un estimado de la matriz Hessiana de la siguiente forma:

$$R(k) = \alpha(k)R(k-1) + \big(1-\alpha(k)\big)g\big(w(k)\big)g^T\big(w(k)\big) \quad (1.52)$$

Bajo las restricciones $0.97 \leq \alpha(k) \leq 1$, y renombrando a la matriz Hessiana como $R(k)$. Entonces, la actualización de pesos se toma de la siguiente forma:

$$w(k+1) = w(k) + R^{-1}(k)g\big(w(k)\big)E\big(w(k)\big) \quad (1.53)$$

Para calcular la inversa de $R(k)$, se emplea una formula resultante del lema de inversión de matrices (ver [28] y [29]), que se muestra en (1.54), debido a que es mejor hacerlo de esta forma computacionalmente hablando.

$$(A+BC)^{-1} = A^{-1} - A^{-1}B(I+CA^{-1}B)^{-1}CA^{-1} \quad (1.54)$$

Aplicando (1.54) para obtener la inversa de (1.52) se tiene:

$$P(k) = \frac{1}{\alpha(k)}\left[P(k-1) - \frac{P(k-1)g\big(w(k)\big)g^T\big(w(k)\big)P(k-1)}{\alpha(k) + g^T\big(w(k)\big)P(k-1)g\big(w(k)\big)}\right] \quad (1.55)$$

Con $P(k) = R^{-1}(k)$, a la matriz $P(k)$ se le puede interpretar como la matriz de covarianza de los pesos sinápticos estimados [27], entonces la ecuación (1.53) queda redefinida como:

$$w(k+1) = w(k) + P(k)g\big(w(k)\big)E\big(w(k)\big) \quad (1.56)$$

De (1.52) se puede deducir el algoritmo LM de forma recursiva, mediante la incorporación de un término de regularización:

$$R(k) = \alpha(k)R(k-1) + \big(1-\alpha(k)\big)\big(g\big(w(k)\big)g^T\big(w(k)\big) + \rho I\big) \quad (1.57)$$

Pero de esta forma no se puede aplicar el lema de inversión de matrices, por lo que (1.57) se modifica de la siguiente forma:

$$R(k) = \alpha(k)R(k-1) + \big(1-\alpha(k)\big)\big(g\big(w(k)\big)g^T\big(w(k)\big) + \rho Z\big) \quad (1.58)$$

Donde Z es una matriz diagonal, con todos los elementos de la diagonal igual a cero con excepción de uno, el cual cambia de posición dentro de la diagonal de la siguiente manera:

$$z_{ii} = \begin{cases} 1, & \text{cuando } i = k \bmod(N_w) + 1 \text{ y con } k > N_w \\ 0, & \text{de otra forma} \end{cases} \quad (1.59)$$

Con esta modificación en conjunto con la incorporación de nuevas variables, la ecuación (1.52) se puede reescribir de la siguiente forma:

$$R(k) = \alpha(k)R(k-1) + \big(1-\alpha(k)\big)\Omega\big(w(k)\big)\Lambda^{-1}(k)\Omega^T\big(w(k)\big) \quad (1.60)$$

$$\Omega^T\big(w(k)\big) = \begin{bmatrix} g\big(w(k)\big) & \\ 0 \quad \dots \quad 1 \quad \dots \quad 0 \end{bmatrix} \quad (1.61)$$

$$\Lambda^{-1}(k) = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix} \quad (1.62)$$

Con $\Omega(k)$ como una matriz de dimensión $N_w \times 2$, con la primera columna correspondiente al gradiente $g\big(w(k)\big)$ y con la segunda columna definida por un vector de dimensión $N_w \times 1$, este último se forma por elementos cero a excepción de uno, el cual cambia de posición dentro del vector de acuerdo con (1.59), similar que en el caso de la diagonal de Z.
Continuando con el cálculo de $S(k)$ y $P(k)$ se tiene:

$$S\big(w(k)\big) = \alpha(k)\Lambda(k) + \Omega^T\big(w(k)\big)P(k-1)\Omega\big(w(k)\big) \quad (1.63)$$

$$P(k) = \frac{1}{\alpha(k)}\big[P(k-1) - P(k-1)\Omega\big(w(k)\big)S^{-1}\big(w(k)\big)\Omega^{\mathrm{T}}\big(w(k)\big)P(k-1)\big] \quad (1.64)$$

Finalmente, se usa la misma la fórmula que se muestra en (1.56) para aplicar el algoritmo LM en forma recursiva. Se tiene que tomar en cuenta que este algoritmo, tal y como se describió es para topologías de RNA de sistemas MISO o SISO, para sistemas MIMO se tiene que modificar.
En general, para los algoritmos de optimización se deben de tener condiciones iniciales diferentes de cero para los pesos sinápticos bajo entrenamiento, así como también los parámetros que se empleen dentro de los algoritmos, ya que es recomendable que estos últimos varíen en cada iteración del algoritmo, aunque también se pueden tomar como constantes.

1.2. RNA en el Dominio de los Números Complejos

En las secciones anteriores solo se habló de las RN en el dominio de los números reales, dado que los primeros aportes sobre el tema se hicieron en este dominio y es donde hay más trabajos realizados, además de que en el dominio real las RN son relativamente fáciles de implementar. Entonces, una red neuronal compleja (RNC) es similar a una RN, con la diferencia de que sus parámetros, funciones de activación, sus patrones de entrada y sus patrones de salida están definidos en el dominio de los números complejos. Al definir los parámetros de una RN en el dominio complejo se tiene una mayor riqueza en cuanto a la información que maneja la red,

aumenta su capacidad de aprendizaje y su capacidad procesamiento. En general, la incorporación de parámetros en el dominio complejo genera un mejor rendimiento por parte de la RN.

El desarrollo de las RNC es relativamente reciente, esto debido a que son difíciles de implementar computacionalmente hablando, por lo que, conforme hay nuevos avances en el campo de la computación se avanza paralelamente en el desarrollo de las RNC.

En [30], [31], [32] y [33] se da una visión general de las RNC, desde desarrollos históricos, características y aplicaciones. Entre los primeros trabajos que se tienen de RNC destaca [34], en donde por primera vez se hace una extensión de una topología de RN en el dominio real al dominio complejo. En [35] se presenta una aplicación utilizando un algoritmo de mínimos cuadrados en su versión compleja, de forma similar en [36] se utiliza una topología de RN en el dominio complejo y se muestra una variación del algoritmo BP en ese dominio. Asimismo, en [37] se trabaja con una topología compleja, pero dividiendo los valores de los paramentos de la RN en parte real y parte imaginaria, mostrando que la derivación del algoritmo de entrenamiento puede ser más fácil de esta forma. En otros trabajos como en [38], se proponen RNC con funciones de activación que contemplan el módulo y el argumento de los parámetros de la red. En cuanto a aplicaciones de las RNC se destacan las mostradas en [22], [39], [40], [41] y [42] en donde se realizan tareas de modelado e identificación de sistemas, para el control de sistemas han destacado los resultados exhibidos en [24] y [43].

Retomando todos los trabajos mencionados en esta sección, se puede notar que los sistemas en los que se han aplicado las RNC están definidos en el dominio complejo, es decir, que se puede representar con facilidad en este dominio; tales como el procesamiento de imágenes, circuitos eléctricos, posicionamiento de robots en el eje (x, y), entre otros. Para sistemas definidos totalmente en el dominio real no hay muchos trabajos, dado que es complicado acoplar las RNC con sistemas que no están definidos en el mismo dominio.

La topología de una RNC se ve afectada en gran medida por la función de activación de ésta, por consecuencia también el algoritmo de entrenamiento. En el caso de las RN en el dominio real, solo basta que las funciones de activación sean derivables y acotadas, como se mencionó al inicio del capítulo, pero para el caso complejo se piden condiciones extras, [30] y [44]. Las funciones de activación para las RNC deben de ser holomorfas, acotadas y sin puntos de singularidad, a continuación, se da la definición de función holomorfa.

Definición 1.2: *una función compleja f, se dice holomorfa en un punto z_0 si su derivada $f'(z)$ existe en z_0 y en cada punto de una vecindad alrededor de z_0. O bien, f es holomorfa en un conjunto abierto $D \subseteq \mathbb{C}$, si su derivada existe para cada punto de este conjunto.*

Esto representa una gran desventaja para las RNC, ya que no es fácil encontrar funciones complejas que se adecuen a estas condiciones, en [45], [46] y [47] se hace un análisis de las

funciones de activación para las RNC, resultando que las funciones construidas son la mejor opción a esta problemática. Una función construida, se refiere a modificar una función de tal forma que se eliminen las propiedades que no permiten que ésta sea holomorfa y acotada al mismo tiempo, generalmente este tipo de funciones se definen en ciertos intervalos o tomando por separado la parte real e imaginaria de los parámetros de la RNC.

Los análisis, así como los algoritmos de entrenamiento para RNC se obtienen de manera similar a como se hizo para las RN en el dominio real, salvo por el uso de las funciones construidas que se acaban de mencionar. Para manejar a las funciones definidas en el dominio de los números complejos, se deben que tener más consideraciones a diferencia de las funciones en el dominio real. El cálculo de Wirtinger es una herramienta matemática por medio de la cual se pueden manejar a las funciones en el dominio complejo, en [48] se habla de esta herramienta y se muestra la manera en que se hace la manipulación de números y funciones complejas.

1.3. Motivación del Trabajo de Tesis

Una vez que se han revisado todos los antecedentes, ventajas y desventajas de las RNA en el dominio de los números reales, así como en el de los números complejos, se puede notar que las topologías de RN que utilizan parámetros complejos son superiores. A partir de eso, surge el interés usar RNC en aplicaciones de identificación y control de sistemas, en especial expandir la topología de RN presentada en [9] al dominio de los números complejos. Esta RN, conocida como RNRE, ha demostrado tener buenas propiedades en aplicaciones referentes a identificación y control de sistemas lineales y no lineales, pero solo se ha trabajado en el dominio de los números reales. Entonces, se pueden esperar mejores resultados en estas tareas al utilizar las RNRE con parámetros complejos.

A pesar de que se tienen trabajos precedentes, en donde se ha se extendido a la RNRE al dominio de los números complejos, tales trabajos presentan algunas incongruencias y falta completar algunas partes en ellos para poder emplear de forma correcta a la RNRE con parámetros complejos. Por lo que, se tiene como objetivo en este trabajo, el describir de forma clara a las topologías y dar los detalles necesarios para aplicar las RNRE con parámetros complejos.

Además, se busca dar la extensión de otra topología de red neuronal, conocida como RNREE, la cual toma como base a una RNRE, pero es más eficiente para aplicaciones de identificación de sistemas (de esto se habla más a fondo en el capítulo dos). Esta es una razón suficiente para plantear como otro objetivo, el aprovechar las características de una RNREE, en conjunto con las ventajas que da el uso de parámetros complejos en las RN para la identificación de sistemas.

El siguiente capítulo trata sobre las topologías de RN base de este trabajo, sus algoritmos de aprendizaje y la extensión al dominio complejo para éstas. En los siguientes capítulos, se mencionan las consideraciones necesarias para aplicar las topologías desarrolladas en las tareas

que se han mencionado. Se muestran simulaciones con sistemas no lineales retomados de los trabajos anteriores, con la finalidad de que se note la diferencia entre trabajar con parámetros en el dominio real y parámetros en el dominio complejo.

1.4. Conclusión

En este capítulo se habló de los conceptos y características sobresalientes de las RNA tanto para el dominio real, así como en el dominio complejo. Se describió el proceso de aprendizaje de una RNA, el cual es el más importante, debido a que en este proceso se representan todas las propiedades de una RNA y es a partir de éste que se empieza el desarrollo de los diversos trabajos que hay en este tema de investigación. Con base en todo esto, se definieron los objetivos de este trabajo, los cuales buscan establecer los requerimientos necesarios para trabajar con RNRE en el dominio complejo, para la identificación y el control de sistemas.

Capítulo 2. Topologías Basadas en Redes Neuronales Recurrentes Entrenables

Se presentan en este capítulo a las topologías de RN usadas como base para el desarrollo de esta tesis; se retoman sus características principales, sus algoritmos de entrenamiento y se muestran estas topologías de RN extendidas al dominio de los números complejos.

2.1. Topologías de RNRE

En [9] se expone una topología hibrida de RN, conocida como RNRE por las siglas de "Red Neuronal Recurrente Entrenable", ésta se compone de una etapa recurrente y una tapa estática. La figura 2.1 muestra en forma de diagrama de bloques a la topología característica de la RNRE; la parte recurrente está definida hasta la señal $Z(k)$ y la parte estática se define después de ésta. Variaciones de esta topología se muestran en [49], pero en este trabajo se limitará a tal cual se muestra en la figura 2.1, sin ningún término de compensación adicional.

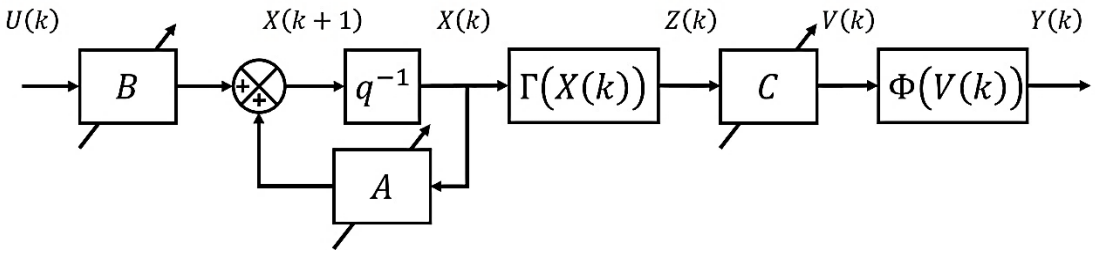


Figura 2.1 Topología de una RNRE.

Matemáticamente la topología de la RNRE se describe como:

$$X(k+1) = AX(k) + BU(k) \quad (2.1)$$

$$Z(k) = \Gamma\big(X(k)\big) \quad (2.2)$$

$$V(k) = CZ(k) \quad (2.3)$$

$$Y(k) = \Phi\big(V(k)\big) \quad (2.4)$$

Donde:

- $A \in \mathbb{R}^{n\times n}$, es una matriz diagonal de pesos sinápticos, se puede interpretar como una matriz de retroalimentación de estados de la RNRE.
- $B \in \mathbb{R}^{n\times m}$, es una matriz de pesos sinápticos que pondera a las entradas de la RNRE.
- $C \in \mathbb{R}^{l\times n}$, es una matriz de pesos sinápticos que pondera a los estados de la RNRE.

- $X(k) \in \mathbb{R}^n$, es el vector de estados internos de la RNRE.
- $Z(k) \in \mathbb{R}^n$, es un vector auxiliar de la RNRE.
- $V(k) \in \mathbb{R}^l$, es un vector auxiliar de la RNRE.
- $U(k) \in \mathbb{R}^m$, es el vector de los parámetros de entrada de la RNRE.
- $Y(k) \in \mathbb{R}^l$, es el vector de los parámetros de salida generados por la RNRE.
- $\Gamma(\cdot)$ y $\Phi(\cdot)$, son funciones de activación, ambas tangentes hiperbólicas.

Los subíndices n, m y l representan al número de neuronas en la capa recurrente, número de parámetros de entrada y número de parámetros de salida de la RNRE respectivamente.

Esta topología puede ser interpretada como un sistema cuasi lineal, en donde las no linealidades son causadas por las funciones de activación. La estabilidad de la red, durante su funcionamiento, se asegura si todos los valores propios de la matriz A se encuentran dentro del círculo unitario definido en el plano complejo. Lo anterior, es una condición de estabilidad para los sistemas dinámicos en tiempo discreto y se puede consultar en [11].

La RNRE es similar a la topología de RN desarrollada en [10], a la cual se le denomino DRNN por las siglas en ingles de "Diagonal Recurrent Neural Network" (red neuronal diagonal recurrente). Ambas topologías tienen diferencias importantes, entre las que destacan las siguientes: la ecuación de estado de la RNRE es lineal, mientras que la de la DRNN es no lineal, por lo tanto, la RNRE presenta propiedades tanto de observabilidad, así como de controlabilidad. Además, se puede usar la información de la RNRE directamente en algoritmos de control, ya que se toma como un estimador de parámetros de cualquier sistema, mientras que las propiedades de controlabilidad y observabilidad de la DRNN dependen de la función de activación con la que se trabaje. Esto hace a la RNRE una buena topología para poder trabajar en problemas de identificación de sistemas, en comparación a otras topologías que se han desarrollado a lo largo del tiempo desde el desarrollo de las RNA.

Por otra parte, en [50] se mostró que la RNRE no funciona adecuadamente cuando se tienen perturbaciones externas en los datos con los que trabaja, tales perturbaciones pueden ser ruidos en la medición de variables, entre otros factores. Una solución a esta problemática fue el incluir filtros pasa bajas en las mediciones, de esta manera se logra un mejor desempeño de la RNRE, aunque se tiene la desventaja que el ajuste de los filtros se hace independiente al ajuste de los parámetros de la red.

Posteriormente, en [51] y [52] se planteó una nueva topología de RN que toma en cuenta el efecto de perturbaciones externas sin la necesidad de agregar elementos ajenos a la red. Esta nueva topología se ideo para la identificación de sistemas y se le conoce como RNREE, por las siglas de "Red Neuronal Recurrente Entrenable Extendida", o también se le conoce como KFRNN por las siglas en ingles de "Kalman Filter Recurrent Neural Network" (red neuronal recurrente de Filtro de Kalman), debido a que se tomó como base para crear a esta topología a la estructura de

una RNRE en conjunto con las ecuaciones del filtro de Kalman. El desarrollo de la topología de la RNREE se muestra a continuación, empezando por considerar sistemas no lineales de la forma:

$$X_d(k+1) = G[X_d(k), U(k)] \quad (2.5)$$

$$Y_d(k) = F[X_d(k)] \quad (2.6)$$

Donde X_d representa a los estados del sistema, U_d representa a las señales de entrada del sistema y Y_d representa a las señales de salida del sistema. Las funciones $G[\cdot]$ y $F[\cdot]$, describen el comportamiento de los estados del sistema y el comportamiento de la salida de éste respectivamente. La representación mostrada en (2.5) y (2.6) puede ser modificada a una forma cuasi lineal, quedando redefinida como:

$$X_d(k+1) = A_d(k)X_d(k) + B_d(k)U(k) + \Theta_1(k) \quad (2.7)$$

$$Y_d(k) = C_d(k)X_d(k) + \Theta_2(k) \quad (2.8)$$

Con:

- $A_d(k) \in \mathbb{R}^{n_d \times n_d}$, matriz de retroalimentación de estados del sistema.
- $B_d(k) \in \mathbb{R}^{n_d \times m_d}$, matriz de entrada del sistema.
- $C_d(k) \in \mathbb{R}^{l_d \times n_d}$, matriz de salida del sistema.
- $X_d(k) \in \mathbb{R}^{n_d}$, vector de estados del sistema
- $U(k) \in \mathbb{R}^{m_d}$, vector de entradas del sistema.
- $Y_d \in \mathbb{R}^{l_d}$, vector de salidas del sistema.
- $\Theta_1 \in \mathbb{R}^{n_d}$ y $\Theta_2 \in \mathbb{R}^{l_d}$, representan las no linealidades de las ecuaciones (2.7) y (2.8) respectivamente.

Donde n_d representa el número de estados del sistema, m_d representa al número de entradas del sistema y l_d representa al número de salidas del sistema.

Entonces, según lo establecido en [53], si se quiere aplicar al filtro de Kalman como un estimador para los parámetros del sistema descrito por (2.7) y (2.8), se deben de cumplir los siguientes requisitos:

- Conocer la información completa tanto de la entrada y de la salida del sistema.
- Tener todos los parámetros de las ecuaciones (2.7) y (2.8).
- Conocer las condiciones iniciales, así como estadísticas (media y varianza) de Θ_1 y Θ_2.

Las ecuaciones que describen al filtro de Kalman como un estimador de parámetros y salidas son las siguientes:

$$X_e(k+1) = A_e(k)X_e(k) + K_e(k)Y_d(k) + B_d(k)U(k) \quad (2.9)$$

$$A_e(k) = A_d(k) - K_e(k)C_d(k) \quad (2.10)$$

$$Y_e(k) = C_d(k)X_e(k) \quad (2.11)$$

Con:

- $A_e(k) \in \mathbb{R}^{n_e \times n_e}$, matriz de retroalimentación de estados del sistema estimado.
- $B_e(k) \in \mathbb{R}^{n_e \times m_d}$, matriz de entrada para el sistema estimado.
- $C_e(k) \in \mathbb{R}^{l_d \times n_e}$, matriz de salida para el sistema estimado.
- $K_e(k) \in \mathbb{R}^{n_e \times l_d}$, matriz de ganancias optimas del filtro de Kalman.
- $X_e(k) \in \mathbb{R}^{n_e}$, vector de estados del sistema estimado.
- $Y_e \in \mathbb{R}^{l_d}$, vector de salidas del sistema estimado.

Donde n_e representa al número de estados estimados, K_e se calcula por medio de métodos de optimización como se muestra en [53]. Otro inconveniente, además de los requisitos para el diseño del filtro, es que el filtro de Kalman no puede estimar parámetros al mismo tiempo que filtra las perturbaciones del sistema. El proceso de filtrado de las perturbaciones se hace mediante K_e, pero al estar esta ganancia en retroalimentación se corre el riesgo de que se amplifiquen los componentes de las perturbaciones en el error, especialmente cuando la ganancia es grande.

Como solución a los problemas que presenta el filtro de Kalman, se propuso combinar a la RNRE y a las ecuaciones que describen al filtro, ya que la RNRE puede cubrir las carencias de información del sistema porque es un estimador de parámetros del mismo. Para esto, primero se reescribe al sistema de ecuaciones (2.9)-(2.11) de la siguiente forma:

$$X_e(k+1) = A_d(k)X_e(k) - K_e(k)Y_e(k) + B_1(k)U_e(k) \quad (2.12)$$

$$B_1(k) = \begin{bmatrix} B_d(k) \\ K_e(k) \end{bmatrix} \quad (2.13)$$

$$U_e(k) = \begin{bmatrix} U(k) \\ Y_d(k) \end{bmatrix} \quad (2.14)$$

$$Y_e(k+1) = A_1 Y_e(k) + Z(k) \quad (2.15)$$

$$Z(k) = C_d(k)X_e(k) \quad (2.16)$$

Donde A_1 es una matriz de retroalimentación de dimensiones adecuadas. Comparando y combinando a los elementos de las ecuaciones (2.12)-(2.16) con los elementos de las ecuaciones (2.1)-(2.4), se puede llegar a la topología de una RNREE, la cual se puede ver en la figura 2.2 y matemáticamente se expresa como:

$$X(k+1) = AX(k) + BU(k) - FY(k) \quad (2.17)$$

$$Z(k) = \Gamma\big(X(k)\big) \quad (2.18)$$

$$V(k+1) = DV(k) + CZ(k) \quad (2.19)$$

$$Y(k) = \Phi\big(V(k)\big) \quad (2.20)$$

Todos los elementos que son similares entre las ecuaciones (2.17)-(2.20) y las ecuaciones (2.1)-(2.4) se definen de la misma forma que para la topología de la RNRE, los nuevos elementos que aparecen se describen a continuación:

- $F \in \mathbb{R}^{n\times l}$, matriz de pesos sinápticos de retroalimentación global.
- $D \in \mathbb{R}^{l\times l}$, matriz diagonal de pesos sinápticos.

Al ser la matriz F de la RNREE análoga a la matriz K_e del filtro de Kalman, se pueden considerar a los algoritmos de entrenamiento de la RNREE como análogos de los métodos de optimización para calcular a K_e, de esta forma se cubren los requisitos del filtro de Kalman dando lugar a otra topología de RN.

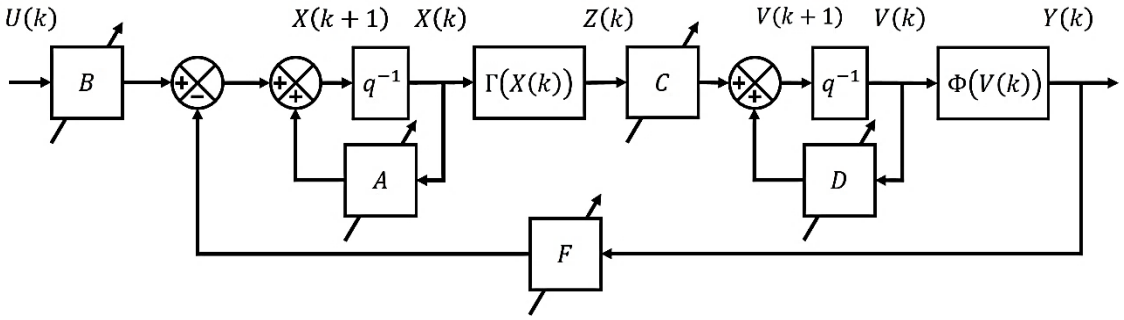


Figura 2.2 Topología de la RNREE.

Para la RNREE, se conserva la condición de estabilidad durante el funcionamiento de la RNRE para la matriz A y se añade que los valores propios de la matriz D deben de cumplir con la misma condición. Al comparar las figuras 2.1 y 2.2, resalta el hecho de que para generar a la RNREE, se cambió la etapa estática de la RNRE por otra etapa recurrente y se añadió una retroalimentación global a ésta. La primera retroalimentación de la RNREE se encarga de estimar los parámetros del sistema, la segunda retroalimentación, así como la retroalimentación global de la red se encargan de filtrar a las perturbaciones del sistema. Las propiedades de observabilidad y controlabilidad de la RNRE expuestas en [49] y [51], se conservan para la RNREE según lo establecido en [51] y [52].

2.2. Algoritmos de Entrenamiento para RNRE

Para entrenar a las topologías mostradas anteriormente, en un principio se usó el algoritmo BP. En [49] se muestra la forma de derivar este algoritmo, similarmente a como se hizo en el primer capítulo, aplicando la regla de la cadena para obtener los gradientes necesarios para el ajuste de pesos sinápticos. En el caso de topologías pequeñas de RNRE esto no representa mucha dificultad, pero cuando se trata de topologías grandes el grado de dificultad aumenta, debido a que se tienen más términos que derivar por lo que ya no es conveniente usar esta metodología. En trabajos subsecuentes, como en [51], [19] y [21] se emplea un método alternativo para la derivación del

algoritmo BP, el cual se basa en los diagramas de bloques de las topologías de RN y fue desarrollado a partir de lo mostrado en [54]. A este método se le conoce como método diagramático, este método usa la teoría de flujos gráficos para manipular diagramas de bloques formados por las topologías de RN; se considera un algoritmo de entrenamiento basado en optimización semejante al BP, ya que busca minimizar funciones de costo como la ecuación (1.24), por eso se tomó en cuenta.

El objetivo principal del método diagramático es crear una red adjunta, esta red representa en forma gráfica a los gradientes necesarios para el ajuste de parámetros de la RN. La construcción de la red adjunta implica el uso de la propagación del error entre parámetros deseados y obtenidos por la red a través de ésta. Dada una representación en diagrama de bloques de una topología de RN, para obtener la red adjunta que correspondiente a este diagrama, se debe de aplicar una inversión en el flujo de señales del diagrama en combinación con un conjunto de reglas, las cuales establecen lo siguiente:

- Las salidas del diagrama de bloques se convierten en entradas.
- Los puntos suma del diagrama a bloques se cambian a puntos de bifurcación.
- Los puntos de bifurcación del diagrama de bloques se cambian a puntos suma.
- Las funciones de una variable dentro del diagrama de bloques son cambiadas por sus derivadas, las funciones multivariables son cambiadas por sus Jacobianos.
- Los operadores de retardo dentro del diagrama de bloques son cambiados por operadores de adelanto.
- Los matrices que representan a los pesos sinápticos se sustituyen por sus transpuestas.

Aplicando lo anterior al diagrama de flujo de la figura 2.1 se consigue la red adjunta de la RNRE, que corresponde a la figura 2.3, se puede ver en esta figura que el error formado a partir de los parámetros deseados y los parámetros obtenidos de la red se propaga a través de toda la red adjunta. De esta forma se obtienen los gradientes necesarios para aplicar el algoritmo BP, el uso de esta metodología facilita en gran medida el proceso de aprendizaje y es aplicable a RNRE de cualquier tamaño, sin que aumente la complejidad del proceso.

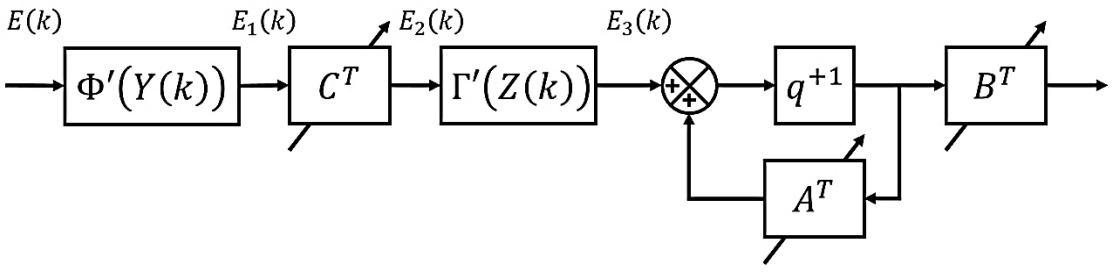


Figura 2.3 Red adjunta de una RNRE, para el algoritmo de entrenamiento BP.

A partir de la figura 2.3, se pueden definir los elementos necesarios para aplicar el algoritmo de aprendizaje BP que es definido por la siguiente ecuación:

$$W(k+1) = W(k) + \eta \Delta W(k) + \alpha \Delta W(k-1) \quad (2.21)$$

Donde:

- $W(k)$, representa una matriz de pesos sinápticos, para el caso de la RNRE puede ser A, B o C.
- η, velocidad del algoritmo de aprendizaje, se toma constante en un intervalo de valores entre cero y uno para asegurar la estabilidad del algoritmo.
- α, termino momento usado para atenuar el efecto de las oscilaciones que produce el algoritmo BP, generalmente se toma en el mismo intervalo de valores que η.
- $\Delta W(k)$, matriz de ajuste de los valores de los pesos sinápticos, utiliza los gradientes obtenidos por la red adjunta y se define igual que en el capítulo uno.

Para este algoritmo se tiene la condición de estabilidad $|W_{ij}| < W_0$, la cual indica que mientras todos los términos de las matrices de pesos sinápticos se encuentren acotados dentro de determinada frontera, el algoritmo BP convergerá de manera estable. Esta condición de estabilidad se desarrolló en [55] y se muestra a detalle en [51].

Los elementos de la red adjunta mostrada en la figura 2.3, así como las matrices de incrementos para aplicar el algoritmo BP se obtienen de la siguiente forma:

$$E(k) = Y_p(k) - Y(k) \quad (2.22)$$

$$E_1(k) = \Phi'\big(Y(k)\big)E(k) \quad (2.23)$$

$$E_2(k) = C^T E_1(k) \quad (2.24)$$

$$E_3(k) = \Gamma'\big(Z(k)\big) \quad (2.25)$$

$$\Delta C = E_1(k) Z^T(k) \quad (2.26)$$

$$\Delta B = E_3(k) U^T(k) \quad (2.27)$$

$$\Delta A = E_3(k) X^T(k) \quad (2.28)$$

En donde $Y_p(k)$ es la salida deseada, $Y(k)$ es la salida obtenida de la red, los vectores $E_1(k)$, $E_2(k)$ y $E_3(k)$ son errores a través de las distintas capas de la RNRE, además $\Phi'\big(Y(k)\big)$ y $\Gamma'\big(Z(k)\big)$ se manejan como matrices diagonales, es decir, como matrices Jacobianas.

Posteriormente, para el ajuste de los parámetros de la RNRE se utilizó el algoritmo LM. Este algoritmo, para poder ser usado en una RNRE, se desarrolló a partir de la descripción hecha en el primer capítulo (ecuación (1.56) y ecuaciones (1.60)-(1.62)) y lo expuesto en [27] y [56], tal y como se puede apreciar en [50], [51] y [19]. La ecuación que describe al algoritmo LM está dada por:

$$W(k+1) = W(k) + P_w(k) \nabla Y\big(W(k)\big) E(k) \quad (2.29)$$

Donde:

- $W(k)$, representa a un vector de pesos sinápticos, para el caso de la RNRE puede se forma por los elementos de las matrices A, B o C.
- $P_w(k)$, es la matriz cuadrada de covarianza correspondiente a A, B o C, esta matriz debe ser definida positiva y toma valores iniciales $P_w(0)$ en un rango de valores entre 1×10^{-3} y 1×10^{-6}, sus dimensiones dependen del número de elementos de $W(k)$.
- $\nabla Y\big(W(k)\big)$, es el vector gradiente de la salida de la RNRE con respecto a $W(k)$.
- $E(k)$, se toma de la misma forma que en (2.22), es el vector de error.

Los términos de los gradientes empleados en la actualización de pesos sinápticos de la red se obtienen por medio del método diagramático, al igual que para el algoritmo BP, pero tomando en cuenta algunas consideraciones. En lugar de transmitir al error $E(k)$ a través de la red adjunta, se transmite un vector $H(k)$, tal y como se aprecia en la figura 2.4; los elementos de este vector son igual a uno y la dimensión de éste es igual a la dimensión del vector de salida de red. Con estas consideraciones, se obtienen los términos del gradiente de la salida con respecto a los pesos sinápticos.

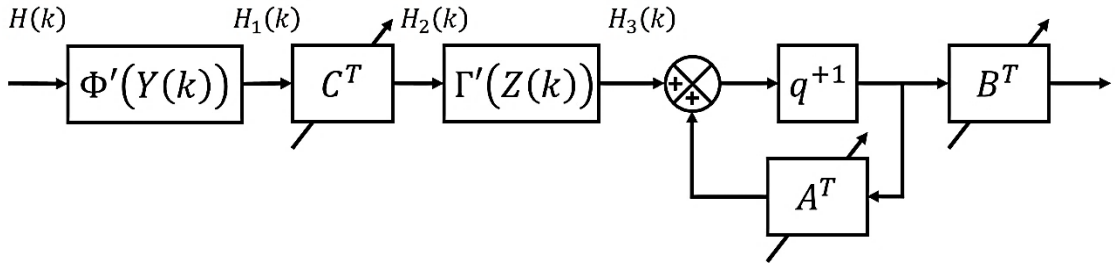


Figura 2.4 Red adjunta de una RNRE, para el algoritmo de entrenamiento LM.

Los términos de la red adjunta en la figura 2.4 para el algoritmo LM se definen como:

$$H(k) = I \quad (2.30)$$

$$H_1(k) = \Phi'\big(Y(k)\big)H(k) \quad (2.31)$$

$$H_2(k) = C^T H_1(k) \quad (2.32)$$

$$H_3(k) = \Gamma'\big(Z(k)\big)H_2(k) \quad (2.33)$$

$$\nabla Y(C) = H_1 Z^T(k) \quad (2.34)$$

$$\nabla Y(B) = H_3 U^T(k) \quad (2.35)$$

$$\nabla Y(A) = H_3 X^T(k) \quad (2.36)$$

Con H_1, H_2 y H_3 como vectores auxiliares para obtener a los gradientes de salida, $\Phi'\big(Y(k)\big)$ y $\Gamma'\big(Z(k)\big)$ se manejan igual a como se especificó para el algoritmo BP.

Se tienen algunas diferencias con respecto al algoritmo BP, siendo la principal que el algoritmo LM opera en forma vectorial; se descomponen las matrices de pesos sinápticos en vectores y

después de que se aplica la actualización de pesos dada por (2.29) se reconstruyen estos vectores en las matrices que corresponden, para que la red opere con los parámetros corregidos. De esta manera se sigue hasta que se alcanza un desempeño aceptable del error. Para obtener a la matriz de covarianza de los pesos sinápticos, $P_W(k)$, se aplican las siguientes ecuaciones:

$$\Omega^T\big(W(k)\big) = \begin{bmatrix} & \nabla Y^T\big(W(k)\big) & \\ 0 & \dots \quad 1 \quad \dots & 0 \end{bmatrix} \tag{2.37}$$

$$\Lambda^{-1}(k) = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix} \tag{2.38}$$

$$S\big(W(k)\big) = \alpha\Lambda(k) + \Omega^T\big(W(k)\big)P_W(k-1)\Omega\big(W(k)\big) \tag{2.39}$$

$$P_w(k) = \frac{1}{\alpha}\big[P_w(k-1) - P_w(k-1)\Omega\big(W(k)\big)S^{-1}\big(W(k)\big)\Omega^T\big(W(k)\big)P_w(k-1)\big] \tag{2.40}$$

Con las siguientes restricciones: $0.97 \leq \alpha \leq 1$ y $1 \times 10^{-4} \leq \rho \leq 1 \times 10^{-6}$.

Como se dijo, $P_W(k)$ es una matriz cuadrada definida positiva cuyas dimensiones son $N_W \times N_w$, siendo N_w el número de elementos del vector $W(k)$ de pesos sinápticos. Esto implica que para la ecuación (2.37) el elemento $\Omega\big(W(k)\big)$ es una matriz de dimensión $N_W \times 2$, donde la segunda columna de esta matriz se compone por elementos que cambian de posición entre ellos: estos términos son todos iguales a cero a excepción de un término igual a uno, como se puede notar, esto es similar a lo planteado por la ecuación (1.61), por lo que también sigue lo establecido en (1.59) para la posición de los elementos de esta columna. La matriz $\Lambda(k)$ mostrada en (2.38) es diagonal de dimensión 2×2, al igual que la matriz $S\big(W(k)\big)$, aunque esta última no es diagonal. Cabe aclarar que la ecuación (2.9) y el conjunto de ecuaciones (2.37)-(2.40) solamente son válidas para el caso de sistemas SISO o MISO.

Para el caso de sistemas MIMO, el vector gradiente $\nabla Y\big(W(k)\big)$ cambia a ser una matriz, la cual se compone por los vectores fila de los gradientes de cada salida estimada del sistema con respecto a la matriz de pesos sinápticos con la que se está trabajando en el ajuste. Por ende, la matriz $\Omega\big(W(k)\big)$ también cambia sus dimensiones de acuerdo con el número de salidas, pero mantiene a su última columna igual. Asimismo, se tiene un aumento en el tamaño de la matriz $\Lambda(k)$, en la cual se incrementan el número de elementos igual a uno en su diagonal de acuerdo con el número de salidas.

Por ejemplo, en el caso de un sistema MIMO de dos entradas y dos salidas, la diagonal de la matriz $\Lambda(k)$ se compone por tres elementos, dos elementos iguales a uno y al final de la diagonal el termino ρ, por lo que la dimensión de $\Lambda(k)$ ahora será de 3×3 al igual que la de $S\big(W(k)\big)$. Las dimensiones de $\Omega\big(W(k)\big)$ cambian a ser de $N_W \times 3$, donde las primeras dos columnas corresponden a los gradientes de cada salida con respecto a $W(k)$ y la ultima columna se conserva igual que en (2.37).

Según lo establecido en [19] y [57], el algoritmo LM no necesita una prueba de estabilidad, debido a que se derivó a partir del método de Newton, como se mencionó en el primer capítulo y además se le considera una extensión del algoritmo BP, por lo que la condición de estabilidad $|W_{ij}| < W_0$ se conserva. Otro detalle que resalta para este algoritmo es que la ecuación (2.29) se aplica de forma vectorial, mientras que el conjunto de ecuaciones (2.30)-(2.36) se aplican de forma matricial.

Para la RNREE también se usaron los algoritmos BP y LM en el entrenamiento de sus parámetros, tomando en cuenta a las mismas consideraciones que se hicieron para la RNRE. En la figura 2.5 se muestra la red adjunta para el algoritmo BP, mientras que en la figura 2.6 se muestra la red adjunta correspondiente al algoritmo LM.

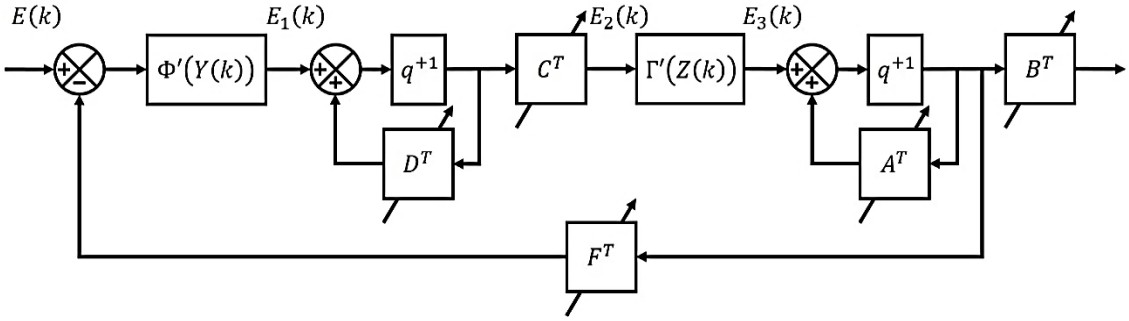


Figura 2.5 Red adjunta de una RNREE, para el algoritmo de entrenamiento BP.

Los elementos de la red adjunta mostrada en la figura 2.5 se describen de forma equivalente al conjunto de ecuaciones (2.22)-(2.28), con la incorporación de los siguientes términos para la actualización de parámetros de la red:

$$\Delta D = E_1(k)V^T(k) \quad (2.41)$$

$$\Delta F = E_3(k)Y^T(k) \quad (2.42)$$

La estabilidad del algoritmo BP para la RNREE se muestra en [51]. Para el caso del algoritmo LM se usó lo establecido en [19] y [57], tampoco se ha hecho una prueba de estabilidad para este algoritmo por las mismas razones mencionadas anteriormente para el caso de la RNRE.

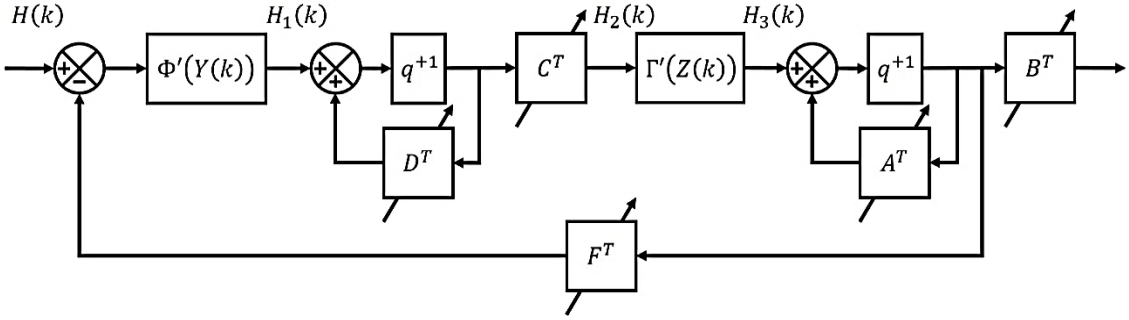


Figura 2.6 Red adjunta de una RNREE, para el algoritmo de entrenamiento LM.

Los elementos de la red adjunta para el algoritmo LM, figura 2.6, se describen similar al conjunto de ecuaciones (2.30)-(2.36), agregando los siguientes términos:

$$\nabla Y(D) = H_1 V^T(k) \quad (2.43)$$

$$\nabla Y(F) = H_3 Y^T(k) \quad (2.44)$$

La matriz de covarianza de los pesos sinápticos $P_w(k)$, al igual que para la RNRE, se obtiene usando las ecuaciones (2.37)-(2.40), considerando las mismas limitaciones para los parámetros del algoritmo LM y a las aclaraciones mencionadas para el caso de sistemas SISO, MISO y MIMO.

2.3. Topologías de RNRE en el Dominio de los Números Complejos (CVRNN)

En la motivación del trabajo de tesis se mencionó que existen trabajos anteriores, [46] y [47], en donde se hizo una extensión de la topología de una RNRE al dominio complejo. Pero en ambos trabajos se tienen detalles que no permiten aplicar las topologías de RNRE con parámetros complejos, a continuación se explica que se hizo en cada trabajo.

En [46] se elaboró un análisis entre dos funciones de activación para definir la topología de RNRE en el dominio complejo, estas funciones son las siguientes:

$$f_1(z) = \tanh(z) \quad (2.45)$$

$$f_2(z) = \tanh(z_{re}) + i\tanh(z_{im}) \quad (2.46)$$

Con $z \in \mathbb{C}$. Se escogieron estas funciones porque se ajustan a lo que ya se había establecido anteriormente en el caso real para las RNRE.

Como resultado del análisis que se mencionó antes, se tiene que la ecuación (2.44) es mejor que la ecuación (2.43) para crear topologías de RNC, debido a que la ecuación (2.43) presenta singularidades y para evitarlas ésta se tiene que definir en cierto intervalo de valores; con esto se logra hacer que esta función sea una función holomorfa, condición importante para las funciones de activación en el conjunto de los números complejos, pero se tienen perdidas de información. La ecuación (2.44) además de no tener singularidades, toma por separado la parte real y la parte imaginaria del argumento de la función compleja, lo que facilita mucho la derivación de los algoritmos de entrenamiento, al no tener que seguir estrictamente lo establecido en [48]. Más detalles de lo que se ha mencionado para (2.43) y (2.44) se pueden encontrar en [30].

Entonces, en [46] se desarrollaron dos topologías de RNRE en el dominio complejo, llamándolas a ambas CVRNN por las siglas en ingles de “Complex Valued Recurrent Neural Network” (red neuronal recurrente en el dominio complejo). Los resultados de este trabajo establecen que la

topología desarrollada a partir de (2.44), la cual se muestra en la figura 2.7, es la mejor extensión de una RNRE en el dominio complejo. Para esta topología se derivó el algoritmo de entrenamiento BP en su versión compleja, conocido como CVBP por las siglas en ingles de "Complex Valued Backpropagation" (backpropagation valuado en el dominio complejo). En este algoritmo se separa la parte real y la parte imaginaria del error, de esta forma el error se hace pasar por una red adjunta desarrollada a partir de la figura 2.7, similar a lo realizado para la RNRE.

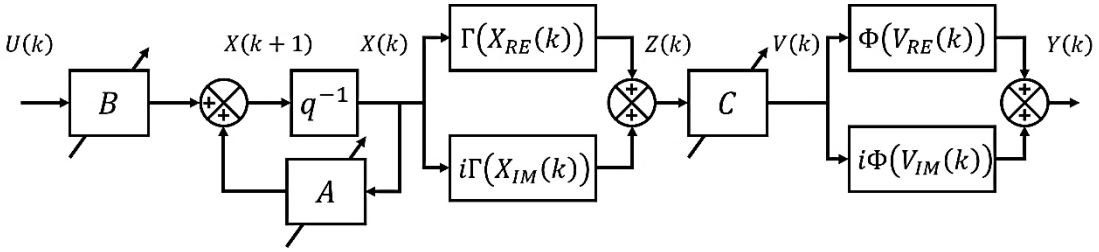


Figura 2.7 Topología de una CVRNN.

Derivado de esto, en [47] se trabajó con esta topología, pero se modificó de tal forma que se llegó a afirmar que la señal $V(k)$ pertenecía al conjunto de los números reales, por lo que el término $i\Phi(V_{IM}(k))$ no existe. Esto implica que el error con el que se efectúa el ajuste de parámetros de la red está en el dominio de los números reales, contrario a lo que se establece en [46]. Otro resultado de ese trabajo fue la derivación del algoritmo LM en su versión compleja, denominado CVLM por las siglas en ingles de "Complex Valued Levenberg-Marquardt" (Levenberg-Marquardt valuado en el dominio complejo). En este algoritmo, para obtener los términos de las matrices de covarianza se siguen las ecuaciones (2.37)-(2.40), pero separando los términos reales e imaginarios de los parámetros.

La topología que se presentó en [47] posteriormente fue aceptada por el autor de [46] y en conjunto con el autor de ésta presentaron un artículo, el cual se puede consultar en [58], ahí se muestran resultados de identificación y control de sistemas con la topología modificada, con los algoritmos CVBP y CVLM. Pero el problema es que la topología modificada en [46] es errónea, dado que no hay forma de asegurar que la parte imaginaria de la salida generada por la RN es enteramente real; siempre hay una parte imaginaria, por muy pequeña que sea y no se puede despreciar. Este error se pudo derivar a partir de que en [46] no se asegura trabajar con parámetros complejos, debido a la estructura de la CVRNN y a la forma en que se derivó el algoritmo de entrenamiento para ésta; para el entrenamiento de la red se separa la parte real e imaginaria de los componentes del error, como consecuencia de esto, sino se dan parámetros iniciales de la red que contengan parte imaginaria diferente de cero y si los parámetros de entrada de la red no contienen parte imaginaria diferente de cero, la CVRNN no generara ningún dato en el dominio de los números complejos, es decir, se continua trabajando con una RNRE. Este detalle no es fácil de percibir, a menos que se verifiquen los parámetros de la CVRNN durante su operación. Entonces

a partir de ese error, en [47] y [58] se asegura que la parte imaginaria de la salida es cero o no existe, por lo que los algoritmos de entrenamiento se definen con un error en el dominio real, lo cual es una gran equivocación; ya que si no se tiene un error en el dominio complejo entonces no se tiene forma de ajustar la parte imaginaria de los parámetros de la red y esta parte puede ir modificándose de tal forma que afecte todo el desempeño de la CVRNN, hasta el grado de hacerla inestable. Aunque esto último no ocurrió en los trabajos mencionados, porque tampoco hay la certeza de que se trabajó con parámetros complejos; los parámetros que se usan para la entrada, así como los parámetros iniciales de la CVRNN no contienen parte imaginaria diferente de cero y la CVRNN no puede generar datos complejos por sí misma, debido una vez más a la forma en cómo se definió su topología y algoritmos de entrenamiento.

Si se quiere usar la topología de una RNRE en el dominio complejo, se recomienda usar la ecuación (2.44) como función de activación, de esta forma se llega a la topología que se muestra en la figura 2.7, tal y como en un inicio se estableció en [46]. Para asegurar que la RNRE o CVRNN genere parámetros complejos durante su entrenamiento y funcionamiento, se deben de aplicar parámetros de entrada que contengan parte imaginaria distinta de cero, o bien, inicializar los parámetros de la red con parte real y parte imaginaria diferente cero. Esta es la forma correcta de hacer la extensión al dominio complejo de una RNRE, a partir de ahora en este trabajo a esta extensión también se le nombrara CVRNN y su topología se describe por las siguientes ecuaciones:

$$X(k+1) = AX(k) + BU(k) \tag{2.47}$$

$$Z(k) = \Gamma\big(X_{RE}(k)\big) + i\Gamma\big(X_{IM}(k)\big) \tag{2.48}$$

$$V(k) = CZ(k) \tag{2.49}$$

$$Y(k) = \Phi\big(V_{RE}(k)\big) + i\Phi\big(V_{IM}(k)\big) \tag{2.50}$$

Donde:

- $A \in \mathbb{C}^{n\times n}$, es una matriz diagonal de pesos sinápticos, se puede interpretar como una matriz de retroalimentación de estados de la CVRNN.
- $B \in \mathbb{C}^{n\times m}$, es una matriz de pesos sinápticos que pondera a las entradas de la CVRNN.
- $C \in \mathbb{C}^{l\times n}$, es una matriz de pesos sinápticos que pondera a los estados de la CVRNN.
- $X(k) \in \mathbb{C}^{n}$, es un vector de estados internos de la CVRNN.
- $Z(k) \in \mathbb{C}^{n}$, es un vector auxiliar de la CVRNN.
- $V(k) \in \mathbb{C}^{l}$, es un vector auxiliar de la CVRNN.
- $U(k) \in \mathbb{C}^{m}$, es el vector de los parámetros de entrada de la CVRNN.
- $Y(k) \in \mathbb{C}^{l}$, es el vector de los parámetros de salida generados por la CVRNN.
- $\Gamma(\cdot)$ y $\Phi(\cdot)$, son funciones de activación, ambas tangentes hiperbólicas.

Los subíndices n, m y l representan al número de neuronas en la capa recurrente, número de parámetros de entrada y número de parámetros de salida de la CVRNN respectivamente, al igual que para la RNRE.

En esta tesis, también se busca establecer los requerimientos necesarios para expandir a la topología de RNREE al dominio complejo. Para ello, de forma similar a lo que hizo para la CVRNN, se usara a la ecuación (2.44) como función de activación, lo que da como resultado a la topología mostrada en la figura 2.8; si se quiere asegurar que esta topología trabaje con parámetros complejos, entonces se siguen las mismas consideraciones que se dieron para la CVRNN. A esta nueva topología de RNC se le llamará a partir de ahora CVKFRNN, por las siglas en ingles de "Complex Valued Kalman Filter Recurrent Neural Network" (red neuronal recurrente de filtro de Kalman valuada en el dominio complejo). En la siguiente sección se hablará de los algoritmos de entrenamiento para esta topología, en los capítulos subsecuentes se dan algunos resultados de simulaciones empleando a esta red para identificación y control de sistemas no lineales. Las siguientes ecuaciones describen a la CVKFRNN:

$$X(k+1) = AX(k) + BU(k) - Y(k) \quad (2.51)$$

$$Z(k) = \Gamma\big(X_{RE}(k)\big) + i\Gamma\big(X_{IM}(k)\big) \quad (2.52)$$

$$V(k+1) = DV(k) + CZ(k) \quad (2.53)$$

$$Y(k) = \Phi\big(V_{RE}(k)\big) + i\Phi\big(V_{IM}(k)\big) \quad (2.54)$$

Todas matrices y vectores de las ecuaciones (2.51)-(2.54) que son iguales a los términos de las ecuaciones (2.47)-(2.50) se definen de la misma forma que se hizo para la CVRNN, los términos diferentes se definen como:

- $F \in \mathbb{C}^{n\times l}$, matriz de pesos sinápticos de retroalimentación global.
- $D \in \mathbb{C}^{l\times l}$, matriz diagonal de pesos sinápticos.

Las condiciones de estabilidad durante el funcionamiento de la red, tanto de la RNRE y de la RNREE se conservan para la CVRNN y la CVKFRNN respectivamente, así como también se conservan las propiedades expuestas al principio de este capítulo por ser solo extensiones de las primeras topologías.

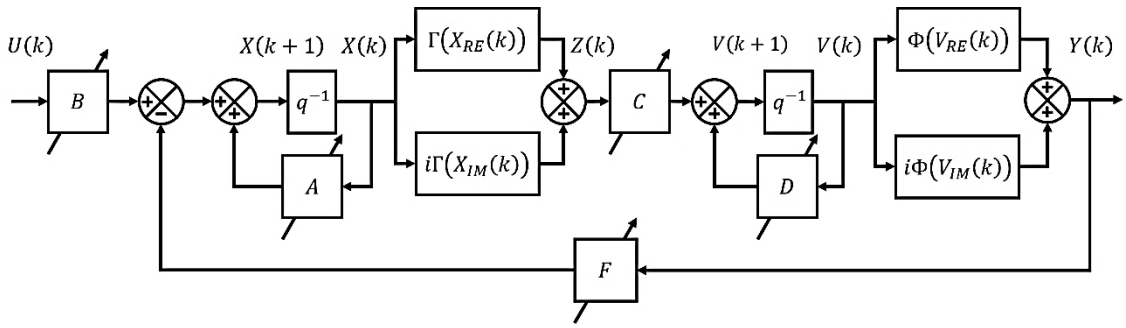


Figura 2.8 Topología de una CVKFRNN.

2.4. Algoritmos de Entrenamiento para CVRNN

Para el entrenamiento de la CVRNN, se propone usar tanto al algoritmo CVBP desarrollado en [46], así como al algoritmo CVLM que se obtuvo en [47], pero con pequeñas modificaciones para este último porque se tiene una topología distinta a la que se usó para su derivación.

Primero se describe al algoritmo CVBP, para la actualización de los pesos sinápticos este algoritmo emplea a la ecuación (2.21). De igual forma que para la RNRE, se genera una red adjunta empleando el método diagramático para obtener los términos de este algoritmo de aprendizaje. En cuanto a la obtención de la red adjunta, se tiene que para el caso complejo se usa la operación transpuesta conjugada en vez de la operación transpuesta para las matrices de pesos sinápticos.

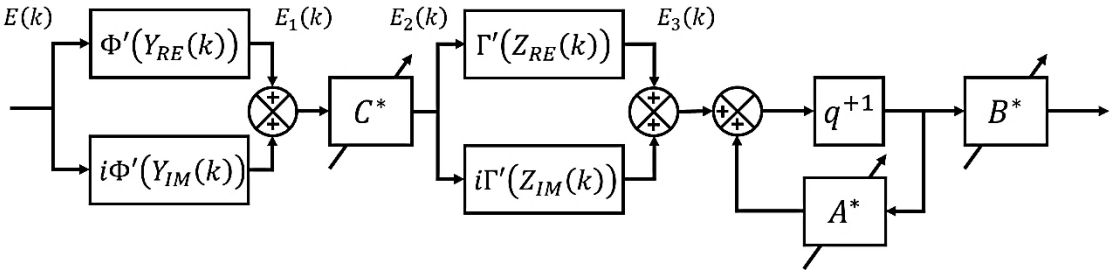


Figura 2.9 Red adjunta de una CVRNN, para el algoritmo CVBP.

La red adjunta que se desarrolló para el algoritmo CVBP se puede apreciar en la figura 2.9 y matemáticamente queda expresada por las siguientes ecuaciones:

$$E_1(k) = \Phi'(Y_{RE}(k))E_{RE}(k) + i\Phi'(Y_{IM}(k))E_{IM}(k) \quad (2.55)$$

$$E_2(k) = C^* E_1(k) \quad (2.56)$$

$$E_3(k) = \Gamma'(Z_{RE}(k))E_{2_{RE}} + i\Gamma'(Z_{IM}(k))E_{2_{IM}} \quad (2.57)$$

$$\Delta C = E_1(k)Z^*(k) \quad (2.58)$$

$$\Delta B = E_3(k)U^*(k) \quad (2.59)$$

$$\Delta A = E_3(k)X^*(k) \quad (2.60)$$

El error que se propaga por la red adjunta de la figura 2.9 se genera igual que en la ecuación (2.22), las operaciones $\Phi'(\cdot)$ y $\Gamma'(\cdot)$ se toman de la misma forma que se describió para el algoritmo BP y se conservan las mismas propiedades de estabilidad de éste.

Asimismo, para el algoritmo CVLM se tomó como base a la ecuación (2.29) y para la obtención de los términos del gradiente de la salida con respecto de los pesos sinápticos también se crea una red adjunta. Para este caso, el vector $H(k)$ se conforma por la suma de dos vectores; un vector de elementos igual a uno y un vector con elementos igual a uno multiplicados por la unidad

imaginaria. Los elementos de la red adjunta del algoritmo CVLM, figura 2.10, se describen a continuación:

$$H(k) = I + iI \tag{2.61}$$

$$H_1(k) = \Phi'\big(Y_{RE}(k)\big)H_{RE}(k) + i\Phi'\big(Y_{IM}(k)\big)H_{IM}(k) \tag{2.62}$$

$$H_2(k) = C^* H_1(k) \tag{2.63}$$

$$H_3(k) = \Gamma'\big(Z_{RE}(k)\big)H_{2_{RE}} + i\Gamma'\big(Z_{IM}(k)\big)H_{2_{IM}} \tag{2.64}$$

$$\nabla Y(C) = H_1 Z^*(k) \tag{2.65}$$

$$\nabla Y(B) = H_3 U^*(k) \tag{2.66}$$

$$\nabla Y(A) = H_3 X^*(k) \tag{2.67}$$

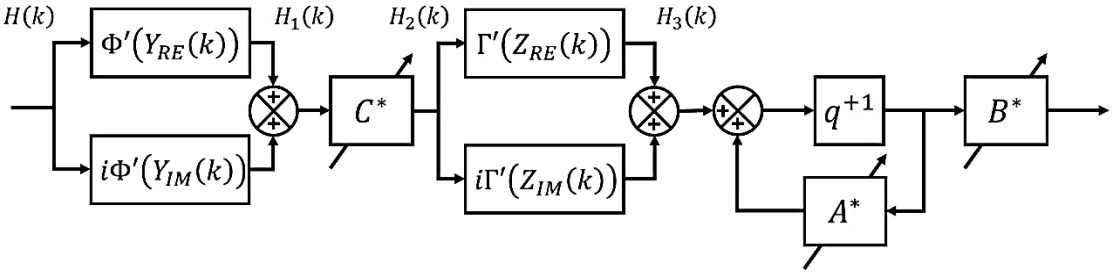


Figura 2.10 Red adjunta de una CVRNN, para el algoritmo CVLM.

Para el algoritmo CVLM se siguen las mismas consideraciones que para el algoritmo LM, además para obtener las matrices de covarianza se aplica el conjunto de ecuaciones (2.37)-(2.40), pero cambiando de la siguiente forma:

$$\Omega^*\big(W(k)\big) = \begin{bmatrix} \nabla Y^*\big(W(k)\big) & \\ 0 \quad \dots \quad 1 \quad \dots \quad 0 \end{bmatrix} \tag{2.68}$$

$$S\big(W(k)\big) = \alpha\Lambda(k) + \Omega^*\big(W(k)\big)P_W(k-1)\Omega\big(W(k)\big) \tag{2.69}$$

$$P_w(k) = \frac{1}{\alpha}\big[P_w(k-1) - P_w(k-1)\Omega\big(W(k)\big)S^{-1}\big(W(k)\big)\Omega^*\big(W(k)\big)P_w(k-1)\big] \tag{2.70}$$

La matriz $\Lambda(k)$ se toma como en (2.38). La única modificación que sufrieron las ecuaciones (2.68), (2.69) y (2.70) es que se sustituyó la operación transpuesta por la transpuesta conjugada.

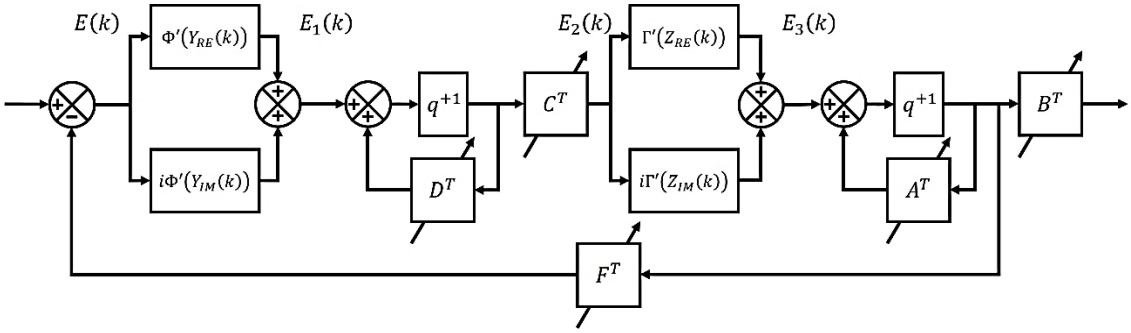


Figura 2.11 Red adjunta de una CVKFRNN, para el algoritmo CVBP.

Los algoritmos de entrenamiento para la CVKFRNN se desarrollaron de forma similar a como se hizo para la RNREE, pero tomando como base a lo establecido para los algoritmos CVBP y CVLM de la CVRNN. Para el algoritmo CVBP se sigue teniendo como base a la ecuación (2.21), la red adjunta correspondiente a este algoritmo se puede apreciar en la figura 2.11 y los términos de ésta, así como las matrices de incremento son iguales que en el conjunto de ecuaciones (2.55)-(2.60) con la incorporación de los siguientes términos:

$$\Delta D = E_1(k)V^*(k) \quad (2.71)$$

$$\Delta F = E_3(k)Y^*(k) \quad (2.72)$$

El error que se propaga por la red adjunta de la figura 2.11 sigue siendo el mismo que se expone en la ecuación (2.22).

Para el algoritmo CVLM se toma la ecuación (2.29) como base, la red adjunta correspondiente a éste se muestra en la figura 2.12, al igual que en los casos anteriores para el algoritmo LM se propaga un vector de elementos igual a uno por de la red adjunta y se deben de tener las mismas consideraciones que se mencionaron para el caso de la CVRNN. Se conservan las ecuaciones (2.61)-(2.67) para los términos de la red adjunta de la figura 2.12, pero se añaden a estas las siguientes expresiones:

$$\nabla Y(D) = H_1 V^*(k) \quad (2.73)$$

$$\nabla Y(F) = H_3 Y^*(k) \quad (2.74)$$

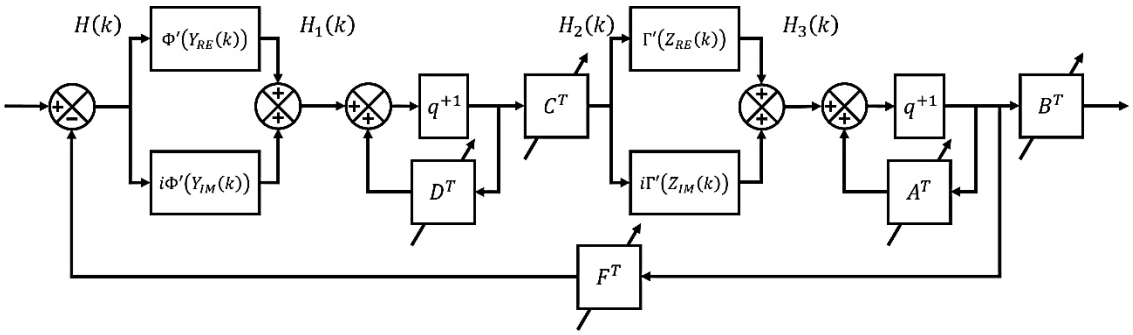


Figura 2.12 Red adjunta de una CVKFRNN, para el algoritmo CVLM.

La matriz de covarianza $P_w(k)$ se obtiene de la misma forma en que se describió anteriormente para el caso de la CVRNN.

Por último, todos los algoritmos de aprendizaje definidos en el dominio complejo conservan las mismas propiedades de estabilidad que se tienen para los algoritmos en el dominio real, por el hecho de que solamente se extendieron las estructuras originales al dominio complejo.

Tanto las topologías y los algoritmos de entrenamiento expuestos aquí, no son forzosamente la única manera de trabajar con la topología de una RNRE en el dominio complejo. Tal y como se demostró en [49], las topologías de RN pueden modificarse de acuerdo con el tipo de problema

con el que se trate, por lo que se pueden tener modificaciones a lo que en este capítulo se ha desarrollado, dependiendo del uso que se le quiera dar a las RNC. Además, se pueden llegar a crear estructuras que sean mejores, en el sentido de optimización de los parámetros, en especial para el algoritmo CVLM, debido a que este algoritmo es complejo porque tiene muchos más elementos que calcular en comparación con el algoritmo CVBP. No se mencionó antes, pero la función de costo que se busca minimizar para el caso de las extensiones realizadas se define como en la ecuación (2.75), ya que el error de estimación $E(k)$ (ecuación (2.22)), se encuentra en el dominio de los números complejos.

$$\zeta(k) = \frac{1}{2} E^{*}(k) E(k) \tag{2.75}$$

Esto solo quiere decir que el desempeño de las RNC se medirá tomando en cuenta a la parte real del error, en conjunto con la parte imaginaria de este mismo. Generalmente, se busca que la función (2.75) tenga valores que se encuentren en un intervalo aceptable, el cual entre más pequeño sea será mejor, porque quiere decir que la red es capaz de reproducir casi de manera exacta a la dinámica deseada. Esta función de costo es un factor importante al momento de establecer las constantes de los algoritmos de entrenamiento, ya que estas constantes influyen en gran medida en el valor de la ecuación (2.75); siempre se buscará que los valores de las constantes estén dentro de los intervalos establecidos por los algoritmos de entrenamiento, de tal manera que estos disminuyan al valor de la función de costo.

2.5. Conclusión

Se han definido las topologías de RNRE en el dominio complejo, aclarando que detalles faltaban en los trabajos anteriores a este es ese sentido. Asimismo, se han mencionado que consideraciones se deben de tomar en cuenta para trabajar con las topologías de RNC que se crearon en este capítulo. Para el proceso de aprendizaje de estas redes, se proponen dos algoritmos de entrenamiento en el dominio complejo, los cuales se basan en la teoría de flujos de señales, lo que implica que son más fáciles de implementar que los algoritmos basados en métodos convencionales.

Capítulo 3. Identificación de Sistemas con Redes Neuronales Recurrentes Entrenables con Parámetros en el Dominio Complejo

Uno de los problemas importantes dentro de la teoría de control es la identificación de sistemas, ya que si se quiere controlar a un sistema se requiere de la información de éste para el desarrollo de algoritmos de control, pero la mayoría de las veces esta información no está disponible o se dispone de parte de ella. En esta tesis se busca usar a las topologías de RNA desarrolladas en el capítulo anterior para resolver esta problemática, ya que de acuerdo con sus características se esperan buenos resultados para la estimación del comportamiento de cualquier sistema.

3.1. Identificación de Sistemas

Para abordar el problema de identificación de sistemas dentro de la teoría de control, se han desarrollado diversas técnicas a partir de la aproximación de funciones. Se ha demostrado que cualquier sistema se puede representar por medio de funciones matemáticas; al conjunto de funciones matemáticas que representan el comportamiento de un sistema, se le conoce como modelo matemático del sistema o simplemente como modelo del sistema. Un modelo de un sistema se compone generalmente por ecuaciones diferenciales o por ecuaciones en diferencias, en las cuales se sintetizan las principales características del sistema. La precisión en la representación las características de un sistema, dependerá de las consideraciones que se tomaron en cuenta al momento de la creación de su modelo.

En muchos casos, los modelos describen de manera aceptable al comportamiento de cualquier sistema, ya que se basan en leyes físicas (leyes de mecánica, hidráulica, termodinámica, etc.), las cuales se han desarrollado a lo largo del tiempo y son ampliamente aceptadas. El problema con esta forma de trabajar es que dichas leyes siempre están definidas con ciertas restricciones: las restricciones pueden ser desde las condiciones iniciales que debe tener el sistema hasta sus límites de operación, además no toman en cuenta todas las posibles interacciones causa-efecto que conforman un sistema, sino que se toman las interacciones más representativas de éste.

Si un sistema opera dentro de las restricciones establecidas al momento de derivar su modelo, se puede considerar a dicho modelo como una buena aproximación del sistema. Pero, si el sistema se llega a salir de esta zona o si los factores externos que no se tomaron en cuenta al momento del modelado son más significativos de lo esperado, la diferencia entre el comportamiento del sistema y el del modelo se irá incrementando de manera que el modelo puede llegar a no ser de utilidad para propósitos de control; ya que la información proporcionada por éste no logra captar al

comportamiento del sistema, lo que conlleva a hacer modificaciones en la estructura del modelo o en el peor de los casos a rehacer dicho modelo. Hacer esto, en la mayoría de las ocasiones incrementa la complejidad del modelo y por lo tanto también incrementa la dificultad para trabajar con él. Por lo que se tienen que buscar otros medios que sean más fáciles de implementar, como los algoritmos de identificación de sistemas.

3.2. Identificación de Sistemas usando RNA

En [49] se muestra un análisis de las técnicas que se han desarrollado para la identificación de sistemas, destacando que al menos se debe de contar con alguna información previa de los parámetros del sistema con el que se trabaja para poder aplicar éstas. Además, los algoritmos utilizados en estas técnicas de identificación, en muchas ocasiones son complicados de manejar y se deben de hacer muchas consideraciones para su aplicación.

Una de las técnicas de identificación de sistemas que ha destacado recientemente han sido las RNA, debido a sus propiedades de aprendizaje y generalización; una RNA puede aprender la dinámica de un sistema y después puede generar un comportamiento similar a éste, en especial ante otros parámetros diferentes a los usados durante su aprendizaje. Asimismo, como se mencionó en el primer capítulo de esta tesis, se tiene como justificación teórica al teorema de aproximación de funciones para RNA.

En el capítulo uno también se cita algunos trabajos en los que se muestran buenos resultados en cuanto a la identificación de sistemas usando RNA, ya sea para sistemas en el dominio real o en el dominio complejo. Con la única diferencia de que cuando el sistema está en el dominio complejo, la RNA tiene como ventaja que sus parámetros pueden manejar el doble de información que en el dominio real, independientemente del sistema y por ende se tiene un mejor desempeño en la tarea de identificación.

Un detalle importante al tener en cuenta al usar RNA para identificación de sistemas, es que la representación que se obtiene de un sistema puede no tener sentido físico o contener parámetros diferentes a los esperados. Ya que, la dinámica de un sistema puede ser similar a la de otro y por lo tanto se pueden tener diversas representaciones de determinada dinámica. El éxito en reproducir los parámetros de un sistema, por medio de una RNA, dependerá principalmente de las consideraciones y restricciones que se tengan en los parámetros de ésta y de su proceso de aprendizaje.

En específico, para las topologías basadas en RNRE se han tenido resultados favorables en la identificación de sistemas, destacando los resultados mostrados en [15], [19], [20] [52] y [59]. Entonces, en este trabajo se pretende verificar el desempeño de las topologías de RNC desarrolladas en el capítulo anterior para identificación del comportamiento de un sistema; ya que estas topologías, además de basarse en las RNRE con las que ya se han tenido buenos resultados,

éstas usan parámetros complejos, por lo que se prevé tener mejores resultados a comparación de los que ya se han presento anteriormente.

Para poder hacer uso ya sea de la topología de una CVRNN o de una CVKFRNN, se deben de hacer algunas consideraciones. En [50] y [51] se trata con las posibles formas en que se pueden aplicar a las RNA para la identificación de sistemas, las cuales son en serie-paralelo y en paralelo. Si se aplica una RNA en serie-paralelo, quiere decir que los parámetros de entrada de la RNA serán las entradas que estimulan al sistema a identificar y las salidas generadas por éste. Si se emplea la RNA en paralelo para la identificación de sistemas, la RNA únicamente tiene como parámetros de entrada a las señales que estimulan el sistema, en ambos casos se usa una señal de error definida como en la ecuación (2.22) para el entrenamiento de la red.

La mejor forma de usar las topologías basadas en RNRE para la identificación de sistemas es en paralelo, debido a que de esta forma se aprovechan al máximo las características de este tipo de red. Razón por la cual se utilizará este enfoque, el cual se puede apreciar en la figura 3.1, en donde se puede ver a una CVKFRNN para identificación de un sistema; se usará esta topología porque se supone que es la mejor opción ante perturbaciones externas y como se puede ver en la figura, se toman en cuenta los efectos de éstas tanto a la entrada como en la salida del sistema.

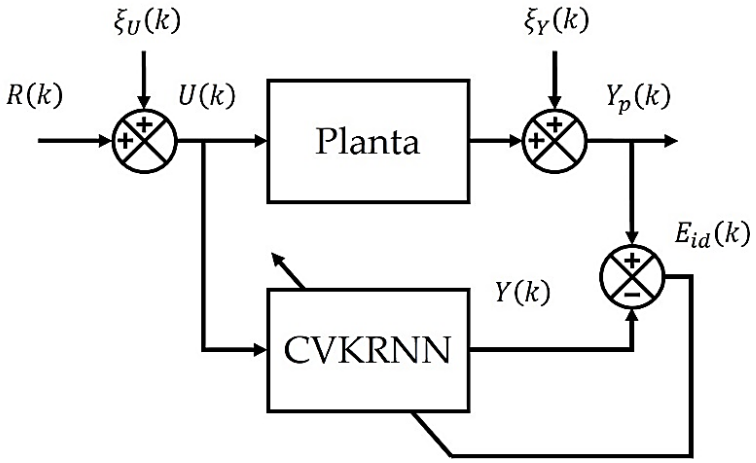


Figura 3.1 Modelo de identificación de sistemas usando una CVKRNN.

Se pueden apreciar los siguientes elementos en la figura 3.1:

- $R(k) \in \mathbb{C}^m$, es un vector que toma el valor de la señal de referencia $R(t)$ en el instante k, la referencia son los parámetros con los que se estimulara al sistema que se desea estimar.
- $\xi_U(k) \in \mathbb{R}^m$, es un vector que representa a las perturbaciones externas al sistema a la entrada de éste, usualmente toma valores dentro de una región acotada.
- $U(k)$, es la entrada de la CVKFRNN y se define igual que en el capítulo dos para esta topología, como se puede notar se conforma de la suma de los vectores $R(k)$ y $\xi_U(k)$.

- $\xi_Y(k) \in \mathbb{R}^l$, es un vector que representa a las perturbaciones externas que hay a la salida del sistema, de forma similar al otro vector de perturbaciones toma valores dentro de una región acotada.
- $Y_p(k)$, es la salida a que se usa para la estimación y se define como en el capítulo dos, se conforma tanto de la salida que generada por el sistema y el vector $\xi_Y(k)$.
- $E_{id}(k)$, es el vector de error entre el comportamiento de la planta y la salida $Y(k)$ generada por la red, se define igual que en la ecuación (2.22) y con este vector se ajustan los parámetros de la red.

Los subíndices m y l representan la al número de entradas y salidas respectivamente, tal y como se planteó con anterioridad.

En la siguiente sección se muestran resultados de una simulación, en donde se realizó la identificación de un sistema MIMO usando una CVKFRNN, presentando una etapa de aprendizaje y una etapa de generalización. Para evaluar el rendimiento de esta topología, se usará una variación del error medio cuadrático presentado en la ecuación (2.75), se trata de un promedio de este error el cual nos indica la efectividad de la red al tener la tendencia hacia cierto valor, tal y como se ha usado en todos los trabajos sobre topologías de RNRE. El error medio cuadrático que se planea usar se define de la siguiente forma:

$$\zeta_{pro}(k) = \frac{1}{N_k}\sum_{j=1}^{N_k}\zeta(j) \tag{3.1}$$

En donde N_k representa al número total de iteraciones k en el tiempo en el que el sistema opera, j es equivalente a k, dado que va incrementando su valor hasta alcanzar a N_k. Entre más pequeño sea el valor de $\zeta_{pro}(k)$, significa que la CVKFRNN se ha acercado a un comportamiento similar al del sistema que se está estimando.

3.3. Resultados de Simulaciones de Identificación de Sistemas usando la Topología de una CVKFRNN

En esta sección se muestran resultados de simulaciones echas en el software MatLab, con las cuales se pretende verificar el rendimiento de una CVKFRNN para la identificación de un sistema MIMO no lineal. Se espera un buen rendimiento en esta tarea, ya que esta topología fue desarrollada a partir de otra diseñada específicamente para identificación.

El sistema a identificar es un sistema MIMO, similar al que se utilizó en [46] y [47], esto con la finalidad de recalcar las diferencias que hay entre estos trabajos; se trata del modelo de un robot planar de dos grados de libertad, dicho modelo se puede consultar en el apéndice A de esta tesis.

Los parámetros que se tomaron en cuenta para la simulación de este sistema se pueden apreciar en la tabla 3.1, en esta tabla se incluyen tanto el valor y las unidades de las constantes significativas del modelo.

Tabla 3.1 Parámetros del sistema MIMO.

Parámetros	Valor
m_1	1 Kg
m_2	1 Kg
l_1	1 m
l_2	1 m
l_{c1}	0.5 m
l_{c2}	0.5 m
g	$9.81\ \frac{\text{m}}{\text{s}^2}$

El modelo del sistema requiere dos señales de entrada y genera dos señales de salida. Las señales de entrada deben de estar definidas en Newtons, ya que se tratan de las fuerzas que impulsan a los eslabones del robot. Mientras que las salidas generadas se definen en radianes, porque indican la posición angular de los eslabones del robot, que se producen a partir de las señales de entrada. Una característica importante de este modelo es que está definido en el dominio real, es decir, no contiene parte imaginaria, por lo que el problema de identificación se define de la siguiente forma: estimar el comportamiento de un modelo con parámetros en el dominio real, empleando una CVKFRNN, la cual está definida en el dominio complejo.

Se busca entonces estimar el comportamiento de las posiciones angulares θ_1 y θ_2 del robot planar, por lo que la salida del sistema se define de la siguiente forma:

$$Y_p(k) = \begin{bmatrix}\theta_1(k)\\ \theta_2(k)\end{bmatrix} + \xi_Y(k) \tag{3.2}$$

Como se puede notar en la ecuación (3.2), se toma en cuenta el efecto de las perturbaciones externas del sistema, las cuales se definen como:

$$\xi_U(k) = \begin{bmatrix}\xi_{U_1}(k)\\ \xi_{U_2}(k)\end{bmatrix} \tag{3.3}$$

$$\xi_Y(k) = \begin{bmatrix}\xi_{Y_1}(k)\\ \xi_{Y_2}(k)\end{bmatrix} \tag{3.4}$$

En donde cada elemento de (3.3) y (3.4) se encuentra acotado superiormente por 0.01, esto quiere decir que cada elemento de las anteriores ecuaciones tomara un valor diferente dentro del intervalo $(-0.01, 0.01)$, en cada iteración k de la simulación.

La referencia que se usara para estimular al sistema y así producir a $Y_p(k)$ se muestra en la ecuación (3.5); aunque ahí se ve que la referencia está definida en tiempo continuo, hay que recordar que esta señal se toma a partir de mediciones, es decir, se manejará en tiempo discreto como se estipulo en la descripción de $R(k)$ que se hizo anteriormente.

$$R(t) = \begin{bmatrix} 0.5\,\text{sen}\left(\frac{1}{25}t\right) + 0.3\,\text{sen}\left(\frac{1}{10}t\right) \\ 0.2\,\text{sen}\left(\frac{1}{15}t\right) + 0.6\,\text{sen}\left(\frac{1}{20}t\right) \end{bmatrix} \quad (3.5)$$

Por tanto, la señal que se usara en la entrada de la CVKFRNN se representa como:

$$U(k) = R(k) + \xi_U(k) \quad (3.6)$$

Entonces, a partir de las ecuaciones (3.2)-(3.6) se puede notar que $m = 2$ y $l = 2$, estos valores dependen totalmente del sistema. Por otro lado, los valores de este conjunto de ecuaciones se toman como números complejos con parte imaginaria igual a cero. Esto último es importante al momento de definir los parámetros de la CVKFRNN; ya que ninguna de las señales de entrada y salida del sistema tienen parte imaginaria, por lo que, en los parámetros de la red, al menos uno deberá tener condiciones iniciales con parte imaginaria diferente de cero, para que la CVKFRNN trabaje con parámetros en el dominio complejo.

A partir de la topología mostrada en la figura 2.8, ya se han definido los valores de las constantes m y l, para el número de neuronas en la primera capa recurrente de la red n, se fijó un valor de cuatro; si se toma con un valor inferior a éste la red corre el riesgo de volverse inestable, mientras que si toma un valor mayor aumenta la complejidad de cálculo de la red.

El proceso de aprendizaje de la CVKFRNN se hizo en línea, con un tiempo de cuatrocientos segundos para este proceso, se consideró que en este tiempo la red puede aprender de forma satisfactoria el comportamiento del sistema. El tiempo de muestreo T_0 para el sistema se fijó en 0.01, por lo tanto, se tienen cuarenta mil uno iteraciones en el tiempo que dura el entrenamiento de la red.

Las condiciones iniciales de las matrices de pesos sinápticos B, C, D y F se establecieron aleatoriamente en el intervalo de valores $(-0.1, 0.1)$, además para los vectores de la CVKFRNN se estableció que $X(0) = [0, 0, 0, 0]^T$ y que $V(0) = [0, 0]^T$. La matriz A se trato de manera diferente, sus condiciones iniciales para la parte real de sus parámetros se tomaron aleatoriamente en el intervalo de valores $(-0.1, 0.1)$, para la parte imaginaria de sus parámetros se tomo aleatoriamente en el intervalo de valores $(-0.01, 0.01)$.

Al ser los datos de las entradas y las salidas de la red números complejos con parte imaginaria igual a cero, se pueden considerar que estos son números definidos en el dominio real, al igual que las condiciones iniciales de los vectores y matrices de la CVKFRNN, a excepción de la matriz A. De esta forma se asegura que la CVKFRNN trabajara con parámetros complejos, ya que al menos uno de sus elementos tiene condiciones iniciales con parte imaginaria diferente de cero, como se mencionó antes.

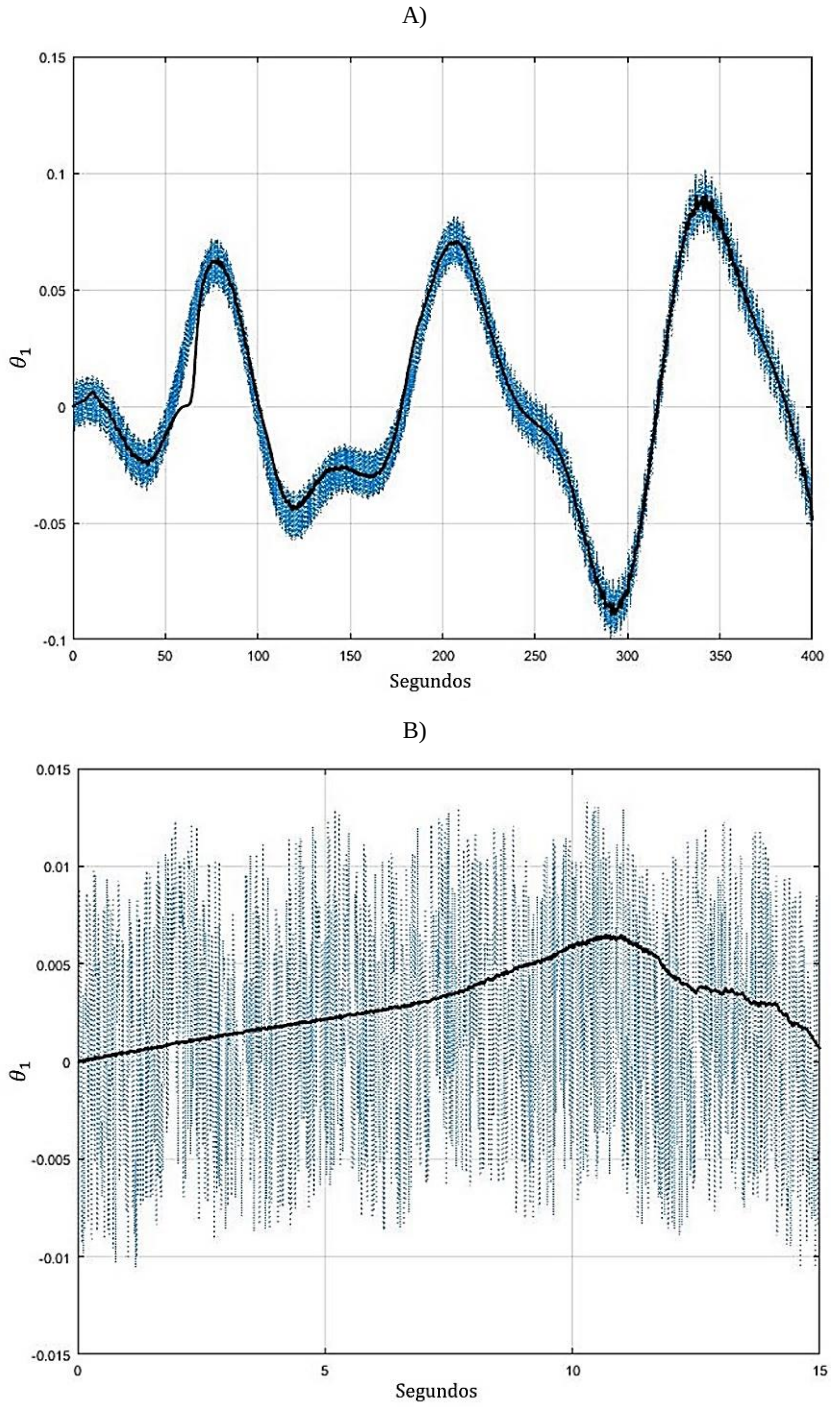
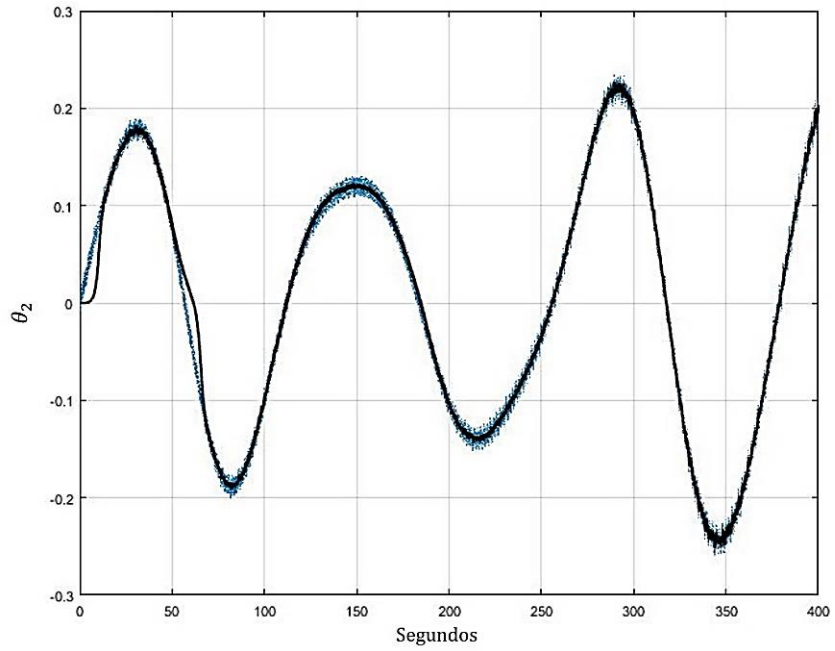


Figura 3.2 Comportamiento de θ_1 durante el proceso de entrenamiento de la CVKFRNN usando el algoritmo CVBP. A) Evolución de θ_1 durante todo el tiempo de simulación. B) Evolución de θ_1 durante los primeros 15 segundos de simulación. La línea punteada corresponde a la respuesta del sistema y la línea continua corresponde a la de la red.

A)



B)

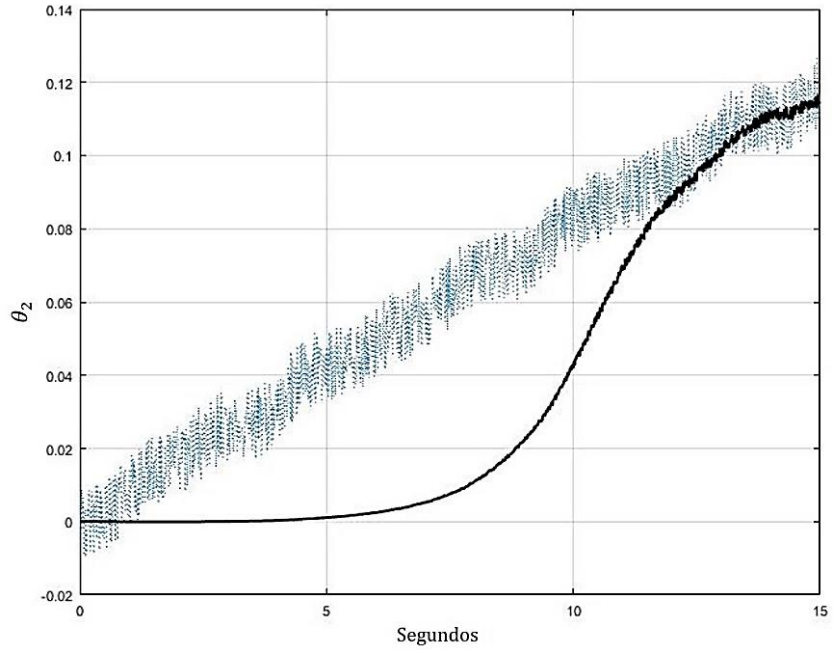


Figura 3.3 Comportamiento de θ_2 durante el proceso de entrenamiento de la CVKFRNN usando el algoritmo CVBP. A) Evolución de θ_2 durante todo el tiempo de simulación. B) Evolución de θ_2 durante los primeros 15 segundos de simulación. La línea punteada corresponde a la respuesta del sistema y la línea continua corresponde a la de la red.

Cabe aclarar que tanto las dimensiones de la red, así como las condiciones iniciales de los parámetros de la misma fueron escogidas mediante experimentación, ya que no se ha creado un método para seleccionar estos parámetros.

Las gráficas que se muestran a lo largo de la sección corresponden a los mejores resultados obtenidos en las simulaciones, en donde se entrenan los parámetros de la CVKFRNN con los

algoritmos CVBP y CVLM. Los valores de los parámetros libres de los algoritmos de entrenamiento, es decir, el valor de las contantes de las que se habló en el segundo capítulo se seleccionó similarmente por medio de experimentación y solo se muestran los valores con los que se obtuvieron los mejores resultados.

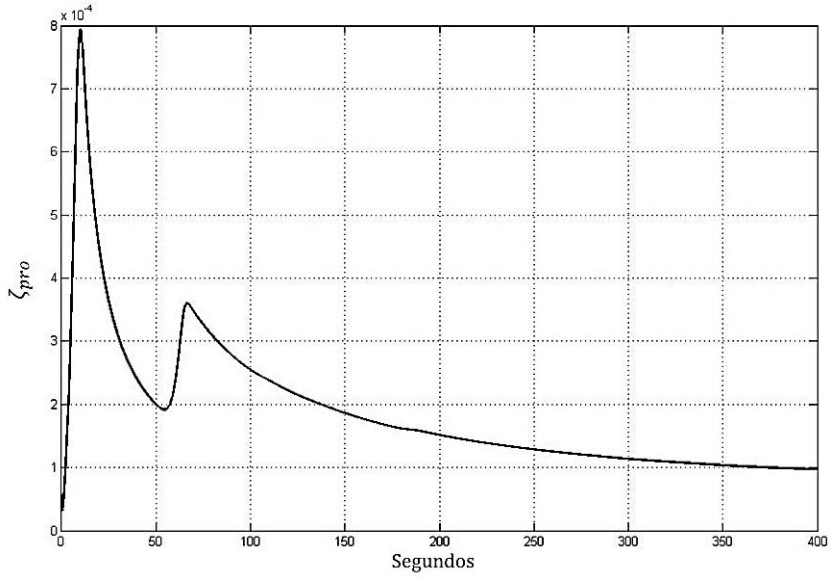


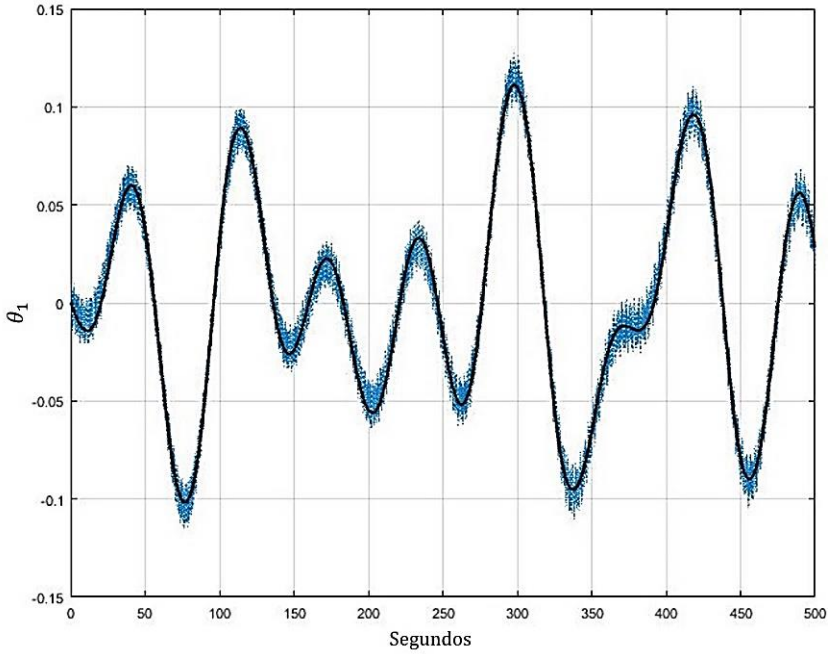
Figura 3.4 Tendencia de ζ_{pro} durante el proceso de entrenamiento de la CVKFRNN usando el algoritmo CVBP.

En principio se usó al algoritmo de entrenamiento CVBP, para el cual se fijaron los valores de sus parámetros libres en $\eta = 0.1$ y $\alpha = 0.001$. En la figura 3.2 se puede ver el comportamiento de θ_1 del sistema, en conjunto con la salida de la CVKFRNN, la cual va imitando el comportamiento de esa posición angular; en un inicio se puede observar que le cuesta algo de trabajo a la red ajustarse a la dinámica de esa salida, pero conforme pasa el tiempo, la respuesta por parte de la red se hace más semejante a la del sistema. En la figura 3.3 se observa algo similar, pero para la posición angular θ_2 del sistema.

Finalmente, en la figura 3.4, se puede ver la evolución del comportamiento del índice de desempeño ζ_{pro} para el algoritmo de entrenamiento CVBP. El valor de ζ_{pro} permanece en un rango de valores menores a 8×10^{-4}, que se considera muy bueno, ya que quiere decir que la red pudo aprender el comportamiento del sistema deseado casi a la perfección, lo cual se comprobara más adelante con el proceso de generalización.

Para el proceso de generalización a partir del aprendizaje usando al algoritmo CVBP, se sigue usando el esquema mostrado en la figura 3.1, con la diferencia de que se fijaron los últimos valores que se obtuvieron de las matrices de pesos sinápticos de la CVKFRNN durante el entrenamiento, en la última iteración. Con estos valores fijos, se alimenta de forma simultánea a la red y al sistema, con la finalidad de comparar el desempeño de la red para imitar el comportamiento del sistema ante otros parámetros de entrada y ver si realmente se aprendió la dinámica de este mismo.

A)



B)

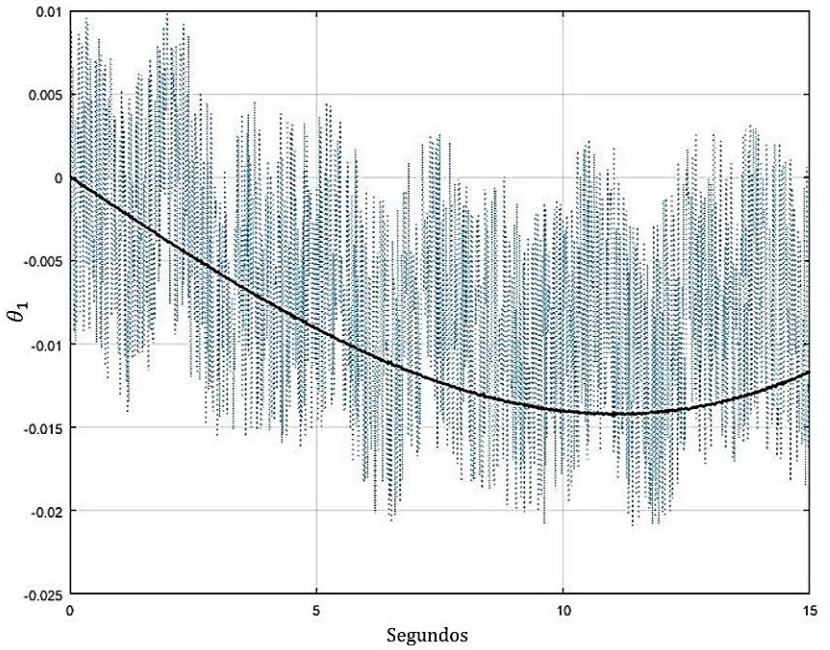


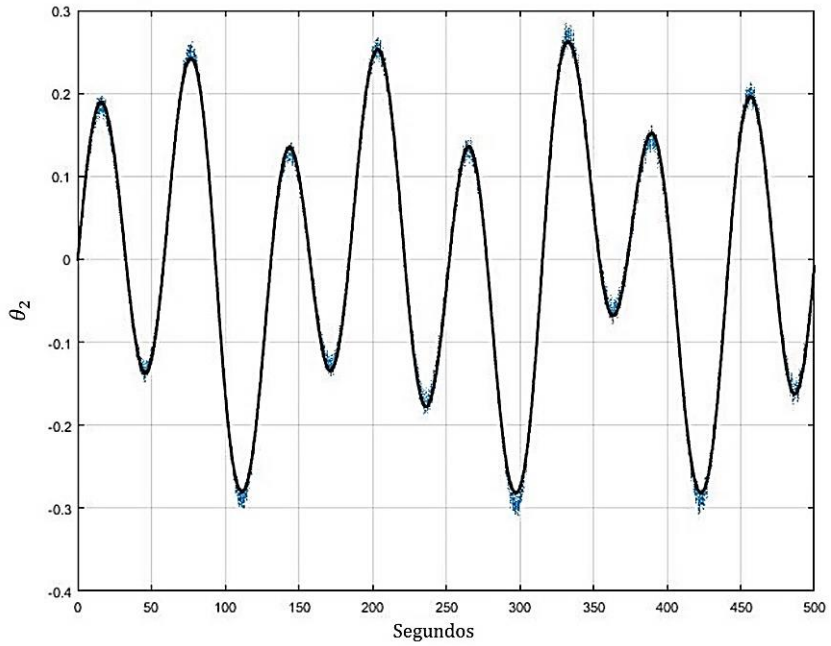
Figura 3.5 Comportamiento de θ_1 durante el proceso de generalización de la CVKFRNN usando el algoritmo CVBP. A) Evolución de θ_1 durante todo el tiempo de simulación. B) Evolución de θ_1 durante los primeros 15 segundos de simulación. La línea punteada corresponde a la respuesta del sistema y la línea continua corresponde a la de la red.

En seguida se hablan de las consideraciones que se siguieron para el proceso de generalización de la CVKFRNN. La expresión $R(t)$ mostrada en (3.5) debe de cambiar para el proceso de generalización, en (3.7) se muestra la señal de referencia para este proceso, se sigue usando una trayectoria similar a la usada en el proceso de aprendizaje. Además, las perturbaciones se siguen

considerando igual a como se describió en las expresiones (3.4) y (3.5), por lo que seguirán tomando valores aleatorios dentro del intervalo establecido.

$$R(t) = \begin{bmatrix} 0.5\,\text{sen}\left(\frac{1}{25}t\right) + 0.3\,\text{sen}\left(\frac{1}{10}t\right) \\ 0.2\,\text{sen}\left(\frac{1}{15}t\right) + 0.6\,\text{sen}\left(\frac{1}{20}t\right) \end{bmatrix} \quad (3.7)$$

A)



B)

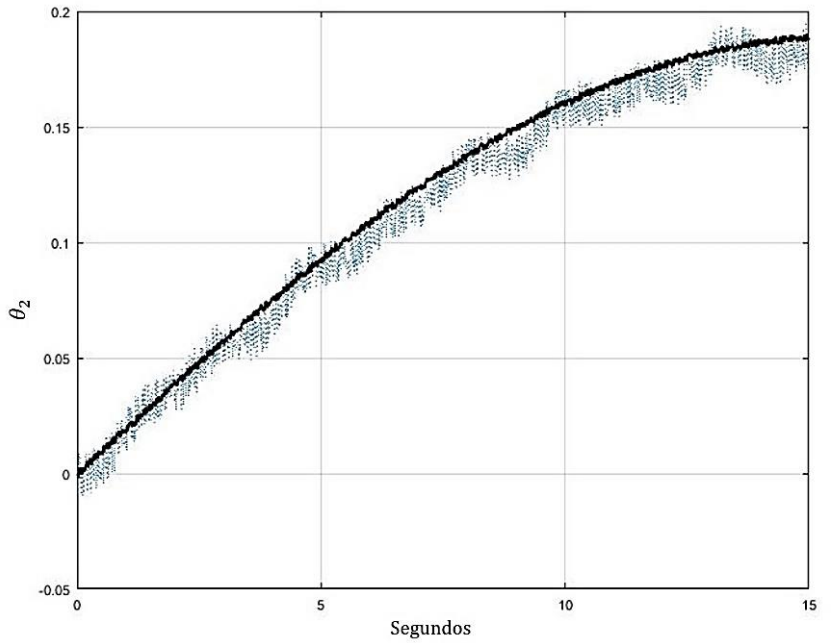


Figura 3.6 Comportamiento de θ_2 durante el proceso de generalización de la CVKFRNN usando el algoritmo CVBP. A) Evolución de θ_2 durante todo el tiempo de simulación. B) Evolución de θ_2 durante los primeros 15 segundos de simulación. La línea punteada corresponde a la respuesta del sistema y la línea continua corresponde a la de la red.

El tiempo que dura el proceso de generalización cambia a quinientos segundos, pero el periodo de muestreo T_0 se conserva igual que en el aprendizaje, por lo que ahora se tendrán cincuenta mil un iteraciones para este proceso y el sistema.

Los resultados del proceso de generalización se pueden apreciar a partir de la figura 3.5, en donde se ve la evolución de θ_1; se puede notar que la red genera un comportamiento similar a esta salida. En la figura 3.6 se muestra el comportamiento de θ_2, la cual describe un comportamiento similar a lo que se dijo para la figura 3.5, pero para la salida con la que se está tratando.

El desempeño de la red durante la generalización se sigue midiendo a partir de la expresión (3.1), en la figura 3.7 se ve la tendencia de ζ_{pro}, la cual se encuentra en valores menores a 6.5×10^{-4}. Esto significa que el comportamiento de la CVKFRNN, con los parámetros aprendidos a partir del CVBP es muy similar al sistema MIMO, que es lo que se buscaba.

Cabe aclarar que, en ambos casos, aprendizaje y generalización, no se grafica la parte imaginaria de la salida de la red $Y(k)$, puesto a que la salida del sistema $Y_p(k)$ no contiene ningún dato con números imaginarios. Se asume que la parte compleja es muy pequeña y solo sirve de apoyo para generar más rápido un comportamiento similar a la salida deseada. Las gráficas de ζ_{pro} confirman esto, puesto que si se tuviera un valor alto en la parte imaginaria de $Y(k)$, estas graficas tendrían valores en rangos mayores a los mostrados.

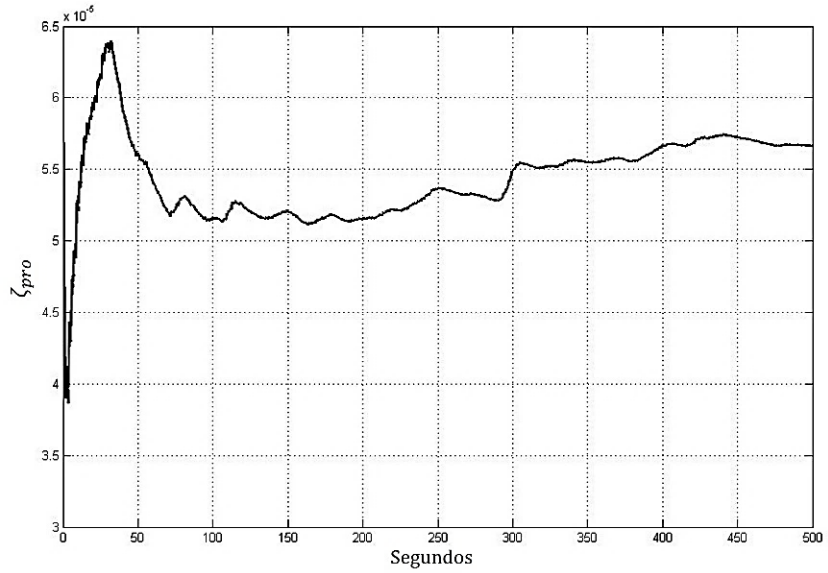


Figura 3.7 Tendencia de ζ_{pro} durante el proceso de generalización de la CVKFRNN usando el algoritmo CVBP.

Una vez que se Provo la eficiencia del algoritmo de entrenamiento CVBP, se hizo lo mismo para el algoritmo CVLM. Se establecieron los siguientes valores para los parámetros fijos de este algoritmo: $\alpha = 0.999$, $\rho = 1 \times 10^{-4}$, las condiciones iniciales de las matrices de covarianza se tomaron así $P_A(0) = P_B(0) = P_C(0) = P_D(0) = P_F(0) = 1 \times 10^3$. En general se siguió el mismo camino que se tomó para aplicar el algoritmo CVBP, aunque el algoritmo CVLM presento

mayor dificultad al momento de su sintonización. Pero al parecer se tuvo una mayor precisión en comparación al algoritmo CVBP.

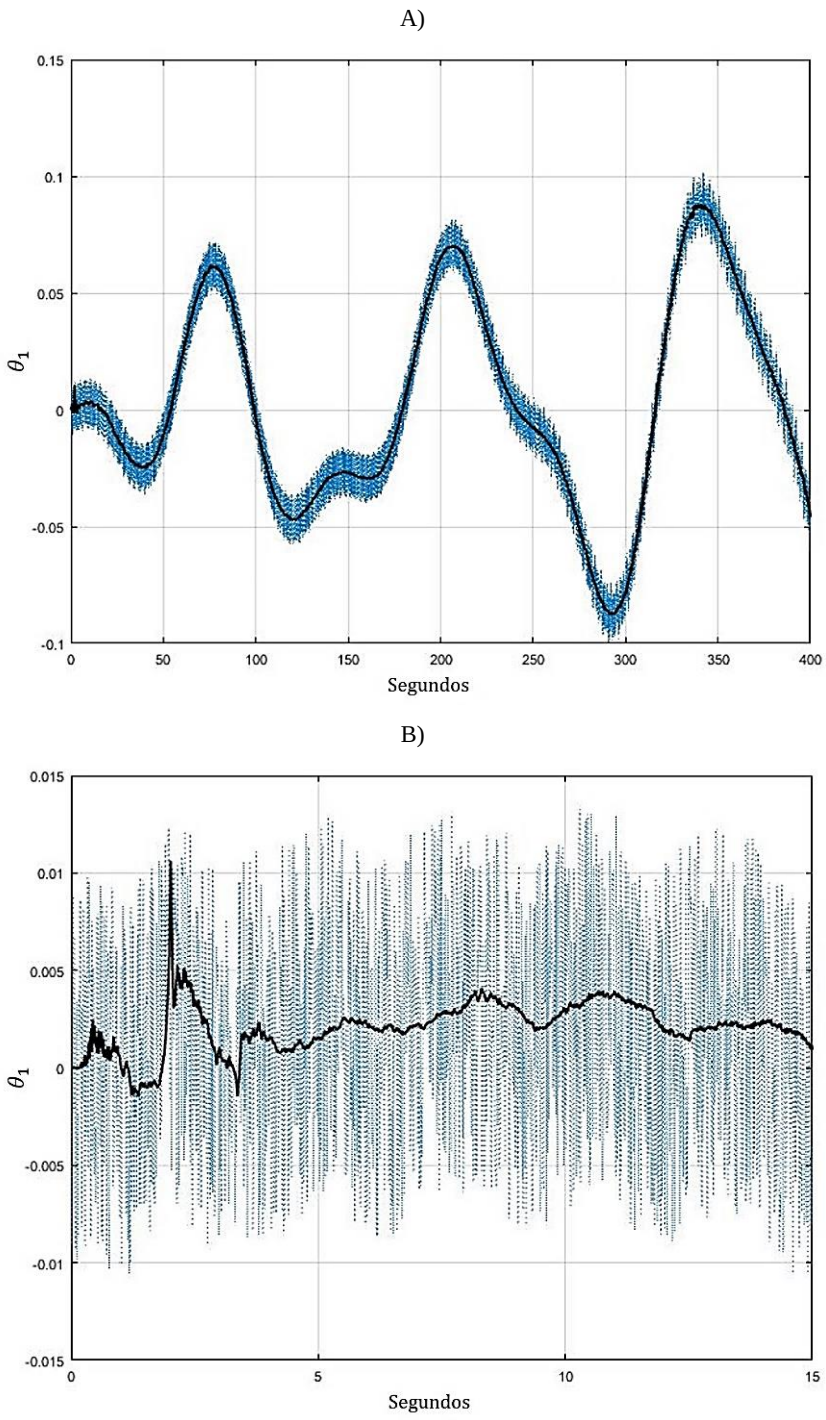
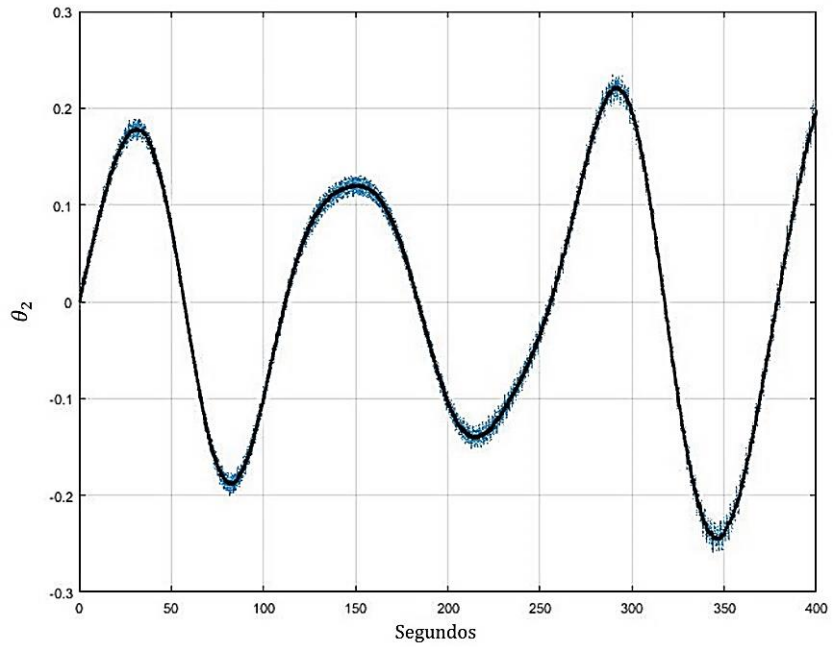


Figura 3.8 Comportamiento de θ_1 durante el proceso de entrenamiento de la CVKFRNN usando el algoritmo CVLM. A) Evolución de θ_1 durante todo el tiempo de simulación. B) Evolución de θ_1 durante los primeros 15 segundos de simulación. La línea punteada corresponde a la respuesta del sistema y la línea continua corresponde a la de la red.

En la figura 3.8, se muestra el comportamiento de θ_1 durante el proceso de aprendizaje empleando al algoritmo CVLM. Al comparar esta figura con la figura 3.2, se puede notar un mejor seguimiento de la salida del sistema por parte de la red al usar el algoritmo CVLM. Lo mismo

sucede para θ_2, comparando la figura 3.3 y la figura 3.9 respectivamente. Aquí es donde se muestra la diferencia al usar un algoritmo de aprendizaje de primer orden, contra un algoritmo que se derivó de un algoritmo de aprendizaje de segundo orden. Esta diferencia radica en la exactitud, pero a cambio de esto se hace más difícil el poder aplicar el algoritmo de aprendizaje. El escoger entre un algoritmo u otro depende de las necesidades que se tenga y de los estándares que se busquen cumplir.

A)



B)

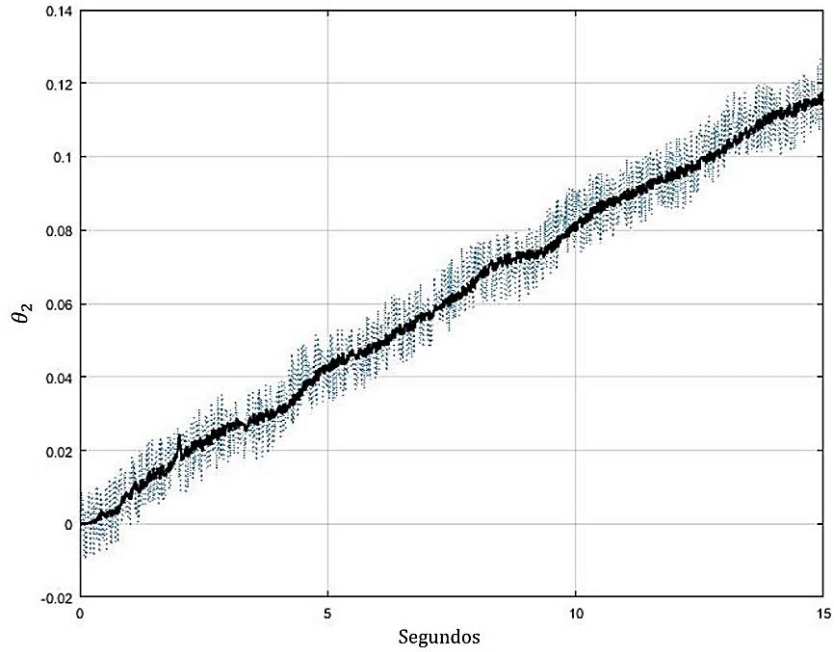
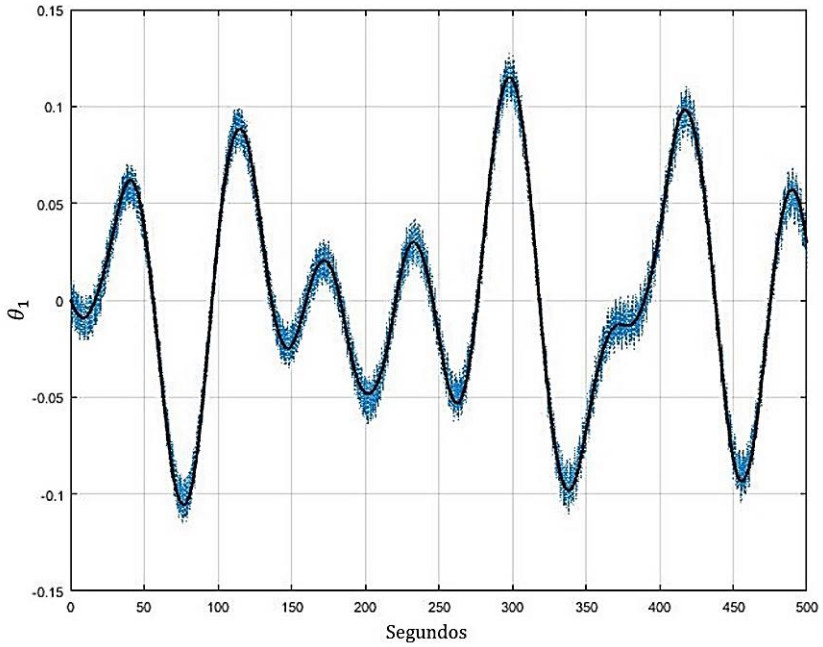


Figura 3.9 Comportamiento de θ_2 durante el proceso de entrenamiento de la CVKFRNN usando el algoritmo CVLM. A) Evolución de θ_2 durante todo el tiempo de simulación. B) Evolución de θ_2 durante los primeros 15 segundos de simulación. La línea punteada corresponde a la respuesta del sistema y la línea continua corresponde a la de la red.

A)



B)

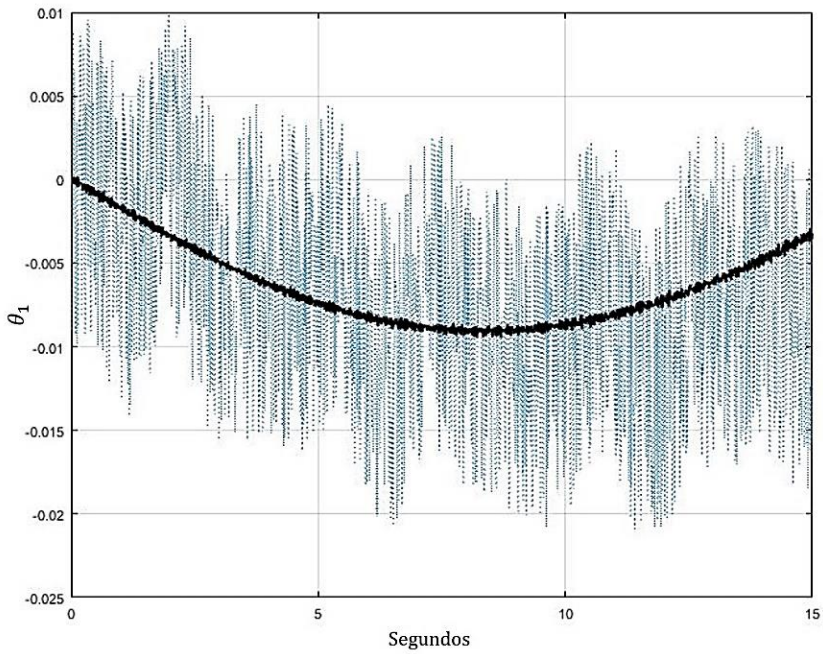


Figura 3.10 Comportamiento de θ_1 durante el proceso de generalización de la CVKFRNN usando el algoritmo CVLM.

A) Evolución de θ_1 durante todo el tiempo de simulación. B) Evolución de θ_1 durante los primeros 15 segundos de simulación. La línea punteada corresponde a la respuesta del sistema y la línea continua corresponde a la de la red.

La evaluación del proceso de aprendizaje usando el algoritmo CVLM se muestra en la figura 3.12, en la cual ζ_{pro} se encuentra en un rango de valores menor a 6×10^{-5} durante todo el proceso. Esto se traduce en que con este algoritmo se aprendió mejor la dinámica del sistema MIMO, aunque al final se observa una tendencia a crecer, esto no importa puesto que el índice de

desempeño aún se encuentra dentro de un rango de valores aceptables, sobre todo menor al que se obtuvo con el algoritmo CVBP.

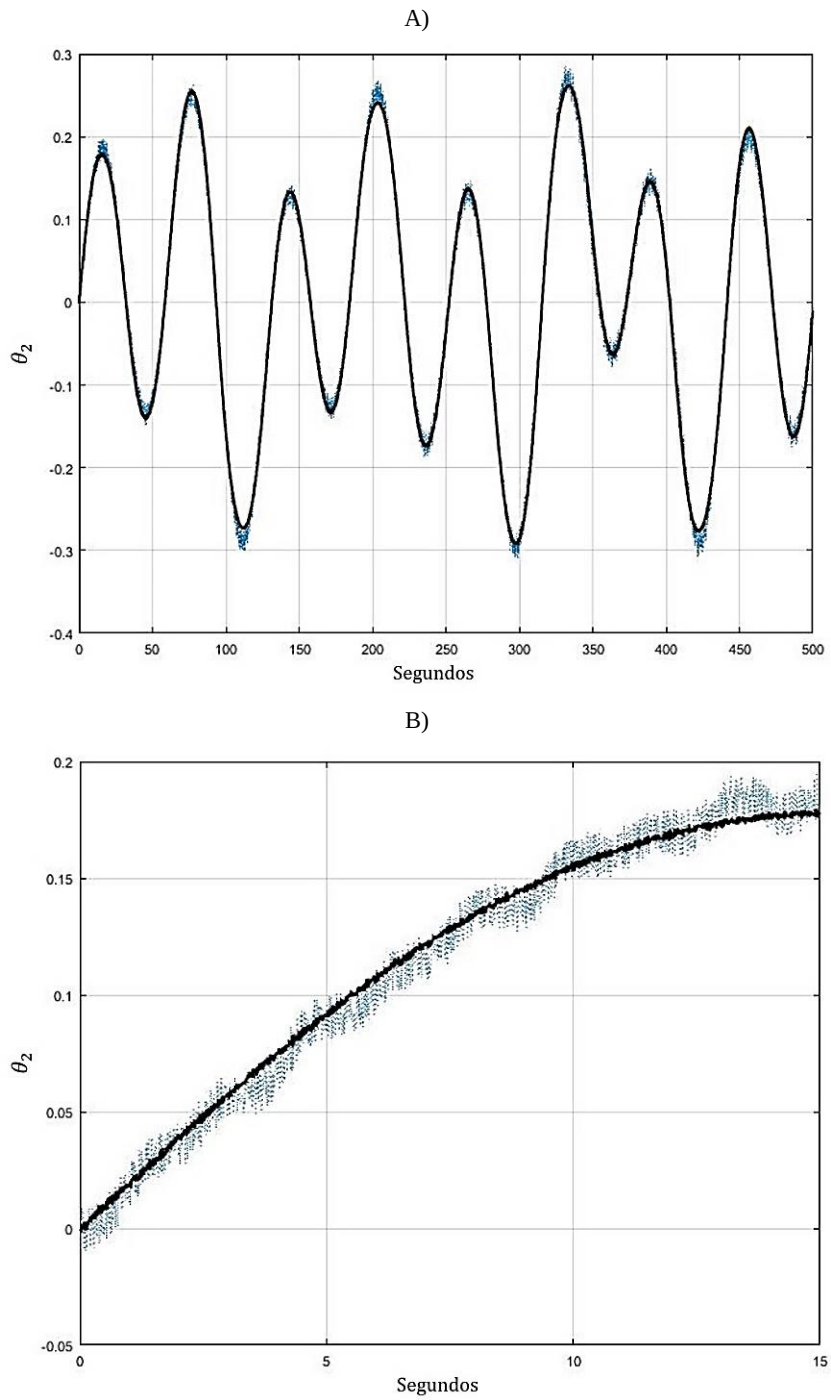


Figura 3.11 Comportamiento de θ_2 durante el proceso de generalización de la CVKFRNN usando el algoritmo CVLM.

A) Evolución de θ_2 durante todo el tiempo de simulación. B) Evolución de θ_2 durante los primeros 15 segundos de simulación. La línea punteada corresponde a la respuesta del sistema y la línea continua corresponde a la de la red.

El proceso de generalización de la CVKFRNN usando los parámetros de que se obtuvieron con el algoritmo CVLM, se hizo igual que el proceso de generalización a partir del algoritmo CVBP.

Los resultados de este proceso están en la figura 3.10 para θ_1 y en la figura 3.11 para θ_2, al compararlos con los resultados que se obtuvieron anteriormente, figura 3.5 y figura 3.6 respectivamente, se puede notar que no hay mucha diferencia entre ellos. El índice de desempeños obtenido por el proceso de generalización con el algoritmo CVLM, figura 3.13, es muy similar al que se obtuvo con el algoritmo CVBP; en este caso se tiene que ζ_{pro} se encuentra en un rango de valores menores a 3×10^{-4} durante todo el proceso.

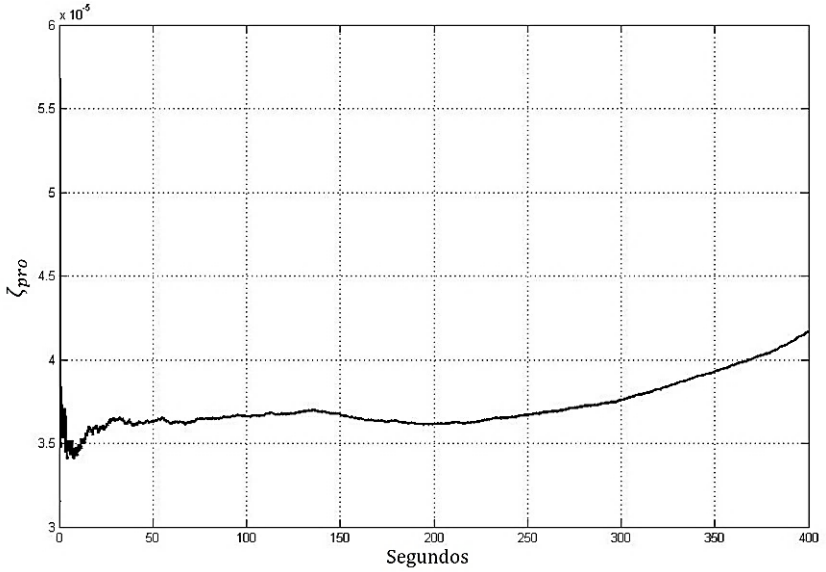


Figura 3.12 Tendencia de ζ_{pro} durante el proceso de aprendizaje de la CVKFRNN usando el algoritmo CVLM.

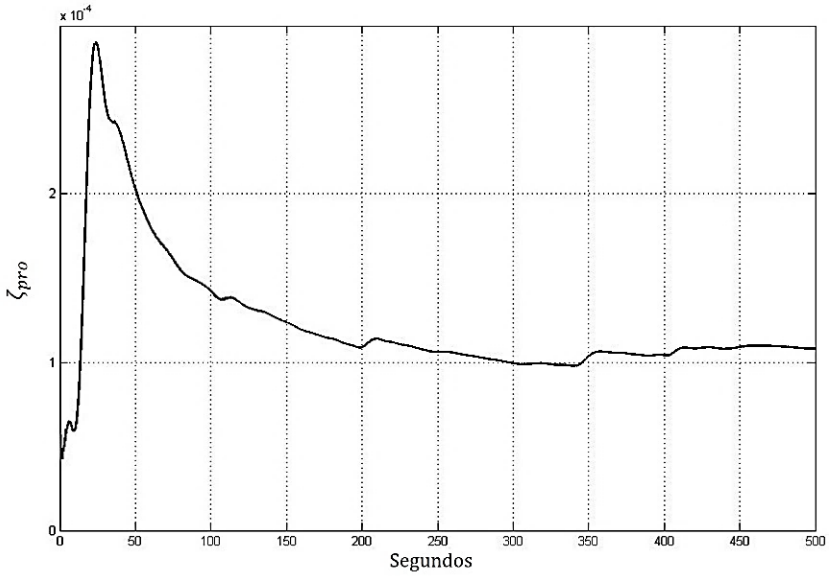


Figura 3.13 Tendencia de ζ_{pro} durante el proceso de generalización de la CVKFRNN usando el algoritmo CVLM.

Como comentarios finales para la comparación entre algoritmos de aprendizaje, se tiene que la gran diferencia entre ambos se da al momento de aprender la dinámica del sistema. Mientras que el algoritmo CVBP es más sencillo de aplicar, en un inicio no se tiene un buen seguimiento de la

señal deseada por parte de la red; no es hasta después de transcurrido un tiempo después de que se inició el proceso que la red adopta un comportamiento similar al sistema. Por otro lado, el algoritmo CVLM es mucho más difícil de implementar, pero desde el primer momento la red tiene un comportamiento similar al del sistema deseado.

Para escoger entre si aplicar un algoritmo u otro, todo dependerá de los requerimientos que se tengan. Algunas veces no es necesario que la red aprenda tan rápido la dinámica del sistema, mientras que en algunas aplicaciones entre más rápido se aprenda esta dinámica es mejor. Otro factor que influye en esta decisión es la capacidad de cómputo que se tenga, dado que para aplicar el algoritmo CVLM se notó que se ocupan más recursos computacionales en comparación con el CVBP. Hablando del proceso de generalización, en ambos casos con los parámetros aprendidos se obtuvieron comportamientos similares, por lo que ambos algoritmos son buenos para el ajuste de los parámetros de la CVKFRNN.

Tabla 3.2 Evaluación del desempeño de la CVKFRNN durante los procesos de entrenamiento y generalización.

Algoritmo	$\zeta_{pro}(k)$ final para el aprendizaje	$\zeta_{pro}(k)$ final para la generalización
CVBP	9.7204×10^{-5}	5.6616×10^{-5}
CVLM	4.1685×10^{-5}	1.0796×10^{-4}

Una comparación más clara, para el proceso de generalización y aprendizaje entre algoritmos de aprendizaje se puede hacer al observar la tabla 3.2, en esta tabla se muestra el valor final al que tiende el índice de desempeño ζ_{pro}. Se puede notar que al final de los procesos se tienen valores muy similares, aunque como se esperaba, se tiene un valor más pequeño para el índice de desempeño en el aprendizaje por parte del algoritmo CVLM, pero se tiene un mejor desempeño al final para la generalización usando el algoritmo CVBP. No obstante, todos estos valores mostrados en la tabla son muy buenos, aunque se pueden mejorar dependiendo de la elección de los parámetros libres de los algoritmos de entrenamiento.

3.4. Conclusión

Se ha comprobado que la CVKFRNN es una buena opción para la identificación de sistemas, ya sea que para el ajuste de parámetros de la red se haga uso del algoritmo CVBP o el algoritmo CVLM. En ambos casos, se observó un buen desempeño por parte de la CVKFRNN para imitar al sistema con el que se estaba trabajando, aunque se tiene la problemática de establecer el valor de las constantes (parámetros libres) de los algoritmos de entrenamiento y las dimensiones de la red, en específico el número de neuronas internas para la primera capa recurrente.

Capítulo 4. Control de Sistemas por Medio de Redes Neuronales Recurrentes Entrenables con Parámetros en el Dominio Complejo

Ya que se ha mostrado de la efectividad de las topologías desarrolladas en este trabajo para la identificación de sistemas, ahora se busca comprobar su desempeño para el control de sistemas. Los algoritmos de control que se esperan desarrollar se basan tanto en la información que se puede obtener de la estimación del comportamiento de un sistema, así como la simple aplicación de las RN directamente en el diseño del controlador, sin tomar en cuenta la información del sistema en cuestión.

4.1. Control de Sistemas

El control de sistemas implica buscar formas de alterar el comportamiento natural de determinado sistema, de manera que se genere un nuevo comportamiento que cumpla con determinados criterios de desempeño. Para poder controlar un sistema se sigue un proceso, el cual consiste en el diseño de un subsistema al que se le conoce como controlador. El controlador toma información del sistema a controlar y de la dinámica deseada para éste, de manera que cuando interactúen el controlador en conjunto con el sistema se genere el comportamiento esperado. Pero como se mencionó en el tercer capítulo, muchas veces no se dispone de información suficiente del sistema a controlar, para obtener dicha información se recurre a la identificación y modelado matemático de sistemas, dependiendo de la técnica que se use para estimar a los parámetros del sistema dependerá también la definición del control, además también se tienen que tener en cuenta a los requerimientos se buscan cumplir.

Una vez obtenidos los parámetros importantes del sistema, se pueden emplear técnicas de control, entre las que destacan recientemente se encuentra las técnicas de control robusto y control adaptable. En ocasiones un sistema se ve expuesto a factores externos que afectan en gran medida su desempeño, pero los algoritmos de control convencionales que se basan solo en la información del modelo del sistema no son muy efectivos ante estos fenómenos; por eso se ha dado auge de los algoritmos de control robusto, en donde se generan algoritmos de control que son capaces de compensar el efecto de los factores externos a un sistema sin que se afecte el desempeño que se busca. Asimismo, también se cuenta con el control adaptable, que se define como un subsistema que mide las características del sistema al cual se quiere controlar, las compara con las características de la dinámica deseada y usa la diferencia para variar los parámetros del sistema o

para generar una señal que alimente a dicho sistema, para que éste mantenga un desempeño optimo, independientemente de las condiciones de su entorno. Generalmente el control adaptable se usa para seguimiento de trayectorias, en donde también es complicado aplicar los algoritmos de control convencionales, ya que además de los efectos externos al sistema, el algoritmo de control debe de ir ajustando sus parámetros de acuerdo con la referencia deseada.

4.2. Control de Sistemas usando RNA

Debido a las características del proceso de aprendizaje de las RNA, en donde ajustan sus parámetros de acuerdo con determinado criterio, se han utilizado a éstas para el control de sistemas tal y como se mencionó en el primer capítulo de esta tesis.

Las RNA se pueden considerar por tanto dentro de los algoritmos de control adaptable, ya que han demostrado ser un medio por el cual se pueden resolver problemas de control, pero a diferencia de su uso en identificación de sistemas (en donde al menos se tienen datos de entrada y de salida para ajustar a la RNA), en los problemas de control casi siempre se desconoce del valor de la señal de control por anticipado, solo se tiene el valor de la referencia deseada y la condición actual del sistema. El objetivo, por tanto, será usar la información disponible para generar la señal de control que modifique el comportamiento del sistema de la forma deseada. Además, las RNA presentan propiedades de robustez y adaptación, aunque esto depende principalmente de la topología que se use, de las restricciones en sus parámetros y del tipo de entrenamiento con el que se ajustan estos.

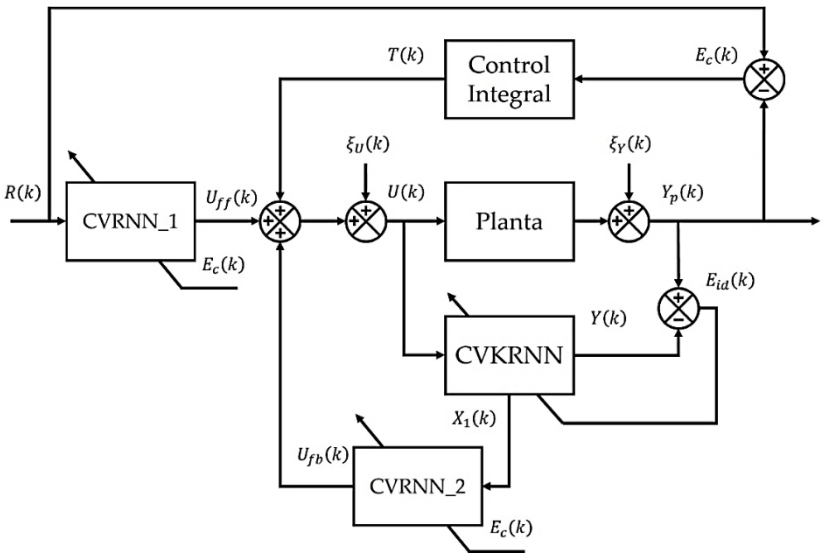


Figura 4.1 Modelo de control neuronal utilizando RNRE en el dominio complejo.

Para las topologías de RNREE en específico, se han generado diversos esquemas de control, que incluyen solamente el uso de RN para generar al control de un sistema, así como también la combinación de las RNRE con otros algoritmos de control adaptable y control robusto. El

esquema de control con el que se ha demostrado tener mayor éxito es el que se muestra en la figura 4.1, el cual fue desarrollado a lo largo de los trabajos con RNRE. La versión que aquí se exhibe está pensada para implementarse con las topologías de RNC que se definieron en el capítulo dos, pero se basa en el mismo concepto que el utilizado originalmente para las RNRE en el dominio real, puede consultarse más información al respecto en [60], [21] y [52].

El esquema de la figura 4.1 se compone de tres RNA y un control integral, en seguida se describe la función de cada elemento:

- CVRNN_1: esta RN se encarga de generar una señal de control $U_{ff}(k)$ conocida como "feedforward". Se alimenta directamente de la referencia $R(k)$ y ajusta sus parámetros con el error de control $E_C(k)$. A partir de esto se espera que la CVRNN genere una señal equivalente al modelo inverso del sistema que se busca controlar, de esta forma se cancelan las dinámicas no deseadas y se sustituyen por la dinámica de la referencia, que el objetivo principal. Esta red representa el control principal del esquema, las otras partes solo son apoyo de ésta.
- CVRNN_2: esta red genera una señal de control de retroalimentación de estados $U_{fb}(k)$, la cual se produce de los estados estimados del sistema a controlar y con los cuales se alimenta a esta red. Esta CVRNN ajusta sus parámetros con el error de control $E_C(k)$.
- CVKFRNN: se usa para estimar los parámetros del sistema a controlar, los estados de esta red se envían a la CVRNN_2 para general el control por retroalimentación de estados. Se alimenta con la señal de control total $U(k)$, al igual que el sistema, además esta red se entrena con el error de identificación $E_{id}(k)$, para lo cual se toman las mismas consideraciones que en la sección anterior.
- Control integral: como su nombre lo indica, es un control integral en tiempo discreto $T(k)$, el cual ayuda al esquema de control neuronal a alcanzar la referencia deseada más rápidamente. Se debe que tener cuidado al momento de su sintonización, ya que puede afectar de forma negativa a la estabilidad de todo el sistema en conjunto, dada su naturaleza oscilatoria.

Al igual que para el problema de identificación, para el problema de control también se toman en cuenta el efecto de perturbaciones a la entrada ya la salida del sistema con el que se trabaja.

El error de identificación $E_{id}(k)$ se define como en la ecuación (2.22), siguiendo las consideraciones hechas para la identificación de sistemas que se hicieron en el capítulo anterior. Para el error de control, se toma en cuenta siguiente ecuación:

$$E_C(k) = R(k) - Y_p(k) \tag{4.1}$$

Los elementos de la ecuación (4.1) se definen de la misma forma que para el esquema de la figura 3.1, por lo tanto $E_C(k)$ es un vector de dimensión l, similar al error de identificación.

Por otro lado, el control integral $T(k)$ se define como un vector y se obtiene conforme a la ecuación (4.2), en donde K_{int} es una matriz diagonal de dimensiones adecuadas, para la cual todos los elementos en la diagonal tienen el mismo valor. Mientras que T_0, se sigue considerando como el periodo de muestreo que se utiliza para el control del sistema.

$$T(k+1) = T(k) + T_0 K_{int} E_C(k) \quad (4.2)$$

Los vectores de las perturbaciones $\xi_U(k)$ y $\xi_Y(k)$ se toman de acuerdo con la descripción echa anteriormente para las ecuaciones (3.3) y (3.4), es decir, sus componentes se encuentran acotados dentro de cierta región.

Por último, la señal de control total que se aplicara al sistema se encuentra definida matemáticamente por (4.3), en donde $U_{fb}(k)$ y $U_{ff}(k)$ son vectores de dimensiones adecuadas.

$$U(k) = U_{ff}(k) + U_{fb}(k) + T(k) \quad (4.3)$$

En la siguiente sección, se muestran resultados de las topologías de RNC desarrolladas en este trabajo usando tal cual el esquema de la figura 4.1 para el control de sistemas. Hablando un poco más de la aplicación de este esquema, se pueden despreciar términos del mismo, con la finalidad de reducir el costo computacional del control. Generalmente, se quita ya sea la parte del control por retroalimentación de estados o el control integral, dependiendo de la aportación que tengan sobre la señal de control total.

Para finalizar esta sección, se tiene que aclarar que el esquema de control neuronal que se ha descrito hasta el momento no es el único con el que probaron las RNC del capítulo dos. También se probaron con un esquema de control indirecto, que se ha usado en trabajos como en [61], [18] y [52]. En estos trabajos se usa a las topologías de RNRE como identificadores de parámetros del sistema, los parámetros estimados con la red se emplean en un algoritmo de control por modos deslizantes. Pero en este trabajo no se obtuvieron buenos resultados usando ese esquema, dado que siempre los sistemas se hacían inestables al momento de aplicar este tipo de control. Esto puede ser causado porque los sistemas usados en el control están definidos en el dominio real, pero las topologías de redes se definieron en el dominio complejo, lo cual causa conflictos. No se descarta el uso del esquema de control indirecto usando RNC, tal vez resulte empleando otros sistemas o redefiniendo este esquema, cosas que quedaron fuera del alcance de este trabajo.

4.3. Resultados de Simulaciones para Control de Sistemas no Lineales usando CVRNN y CVKFRNN

Se muestran los resultados del control de dos sistemas a partir del esquema de la figura 4.1, el primer sistema del que se muestran resultados es el que se utilizó en el capítulo dos con una

referencia constante. El segundo sistema con el que se trabajo es uno similar al que se utilizó en [15] y [62], se empleó en este caso como referencia a una trayectoria. Para evaluar el desempeño de las redes, al igual que se hizo antes se utilizó a la ecuación (3.1), en donde el índice de desempeño se mide a partir de un promedio del error cuadrático, esto aplicara tanto paras redes en el control y la identificación.

4.3.1. Control de un Sistema MIMO usando una Referencia Constante

En esta sección se utilizará el sistema con el que se trabajó en el capítulo dos. Para definir los parámetros de las redes del esquema de control, se emplearán los subíndices uno, dos y tres para la CVRNN_1, la CVRNN_2 y la CVKFRNN respectivamente. La referencia con la que se usó se define en la ecuación (4.4), al ser constante quiere decir que $R(k)$ siempre tendrá el mismo valor.

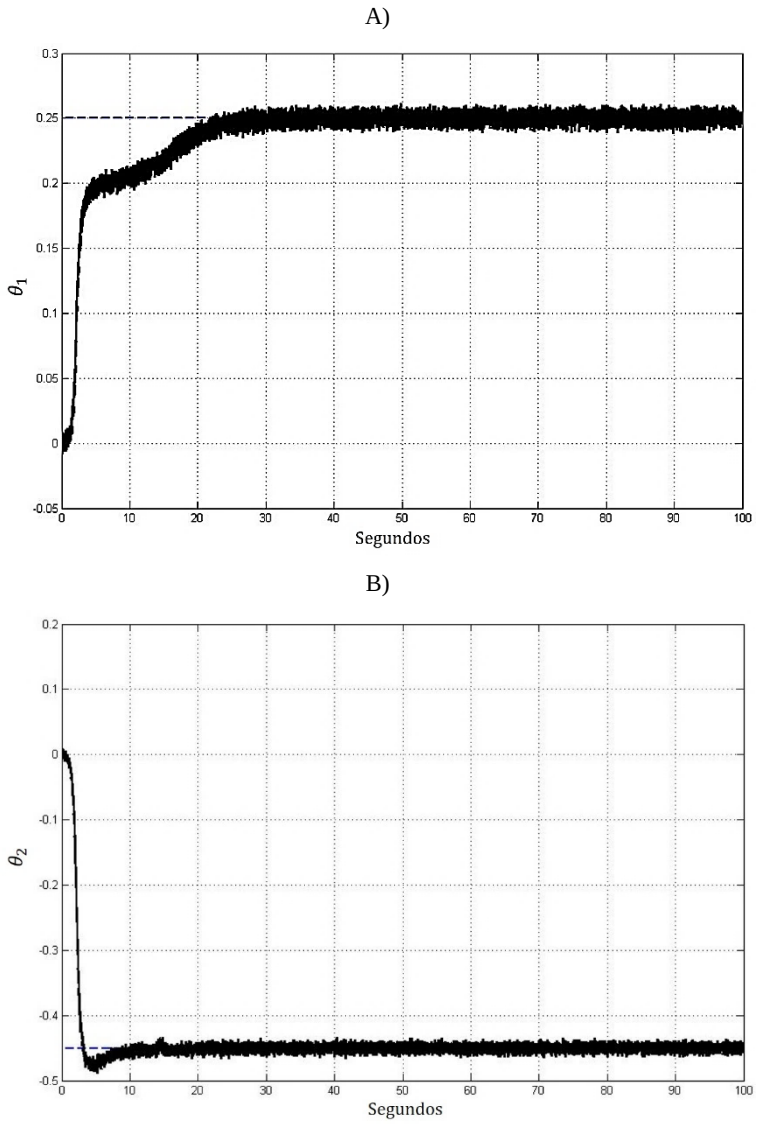


Figura 4.2 Resultado de control del sistema MIMO, usando el algoritmo CVBP.
A) Comportamiento de θ_1. B) Comportamiento de θ_2. La línea segmentada corresponde a la referencia, mientras que la línea continua corresponde a la salida del sistema.

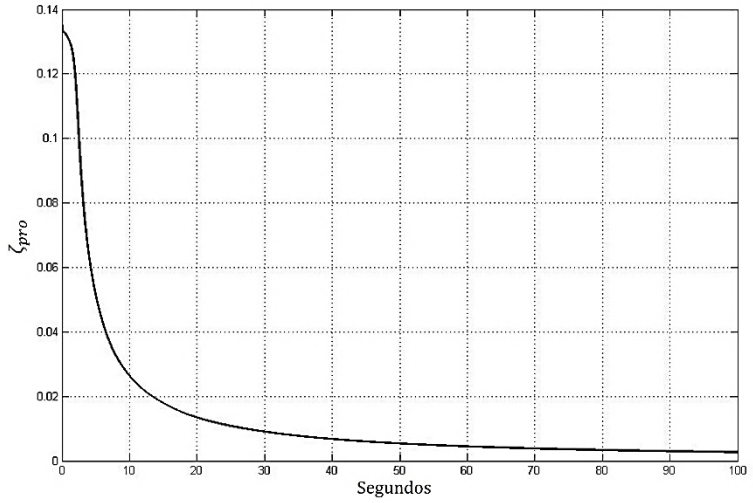
$$R(t) = \begin{bmatrix} 0.25 \\ -0.45 \end{bmatrix} \quad (4.4)$$

Las condiciones iniciales para parámetros de las redes se seleccionaron de la siguiente forma:

- Para las matrices de pesos sinápticos B_1, C_1, B_2, C_2, B_3, C_3, D_3 y F_3 se escogieron aleatoriamente en el intervalo de valores $(-0.1, 0.1)$.
- Para las matrices de pesos sinápticos A_1, A_2, y A_3 se escogieron para la parte real de sus parámetros valores en el intervalo $(-0.1, 0.1)$, para la parte imaginaria se escogieron valores en el intervalo $(-0.01, 0.01)$.
- Para los vectores de todas las redes se escogieron condiciones iniciales igual a cero.

Las dimensiones de las RNC se escogieron experimentalmente y de acuerdo con las dimensiones del sistema, resultando en $m_1 = l_1 = l_2 = m_3 = l_3 = 2$ y $n_1 = m_2 = n_2 = n_3 = 4$.

A)



B)

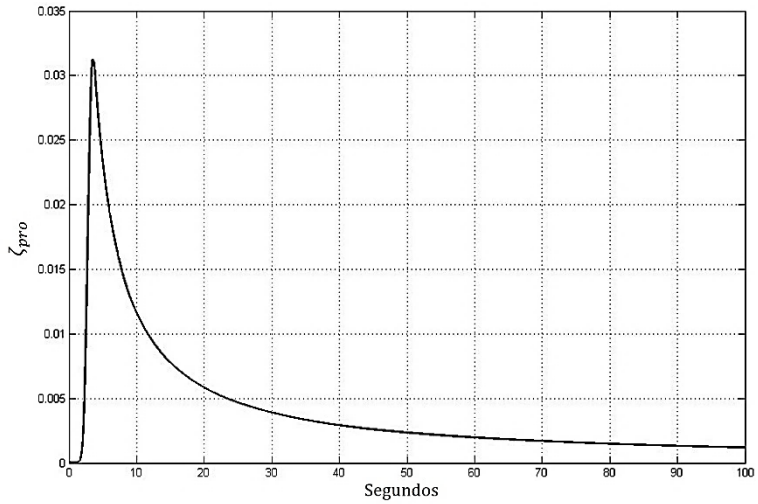
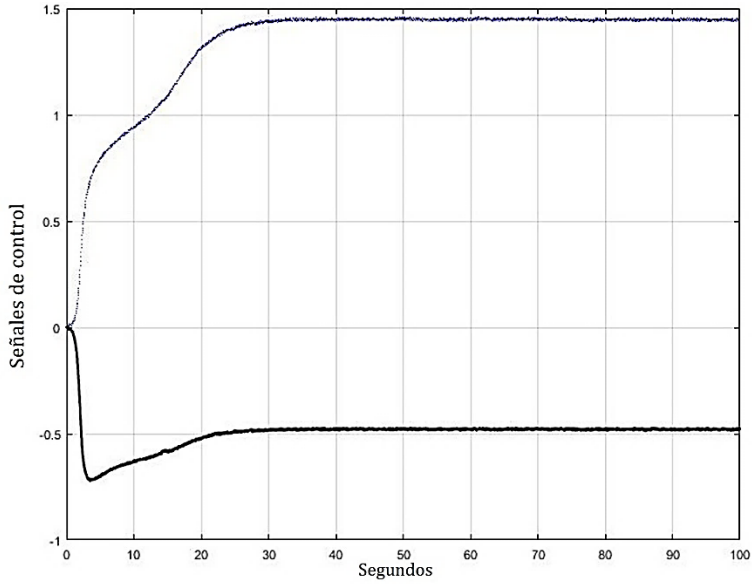


Figura 4.3 Rendimiento de las RNC para el control del sistema MIMO, usando el algoritmo CVBP. A) ζ_{pro} correspondiente al control. B) ζ_{pro} correspondiente a la identificación.

A)



B)

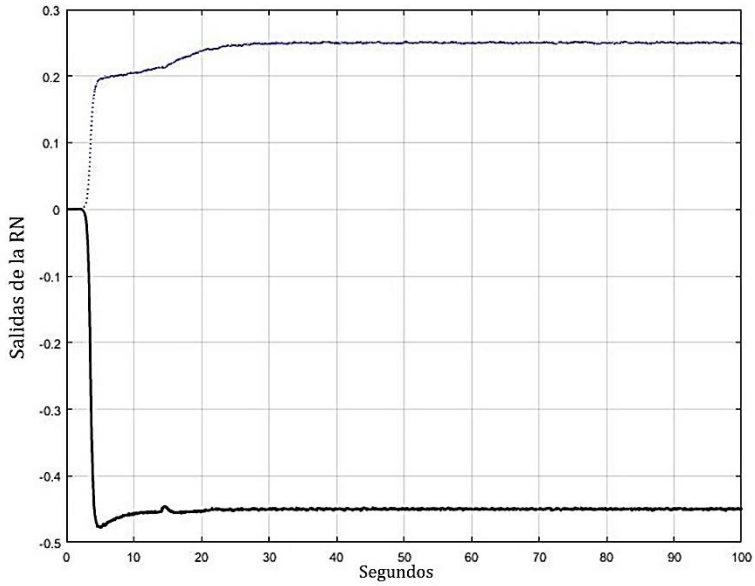


Figura 4.4 Señales generadas por las RNC en el control del sistema MIMO, usando el algoritmo CVBP. A) Señal de control $U(k)$. B) Salida de la CVKFRNN. Para A), la línea punteada corresponde a la primera entrada del sistema y la línea continua a la segunda entrada del sistema. Para B), la línea punteada corresponde a θ_1 y la línea continua a θ_2.

Teniendo en cuenta que el sistema a controlar está definido en el dominio real, quiere decir que $U(k)$ se debe de definir en ese dominio, por lo que su parte imaginaria debe ser igual a cero. Además, también $E_C(k)$ se encuentra definido en ese dominio, ya que la referencia tampoco contiene datos en el dominio complejo. Para resolver estos inconvenientes, se definió un error auxiliar que ayude a entrenar la parte compleja de CVRNN_1 y CVRNN_2; básicamente se generaron errores entre un vector de ceros de dimensiones adecuadas y la parte imaginaria de las salidas de cada red, así estos errores en combinación con $E_c(k)$ se usaron para el entrenamiento

de sus correspondientes redes. De esta forma, se asegura que la parte compleja de las salidas de las RNC afectadas tiende a cero y pueda no incluirse en las señales que se mandan al sistema.

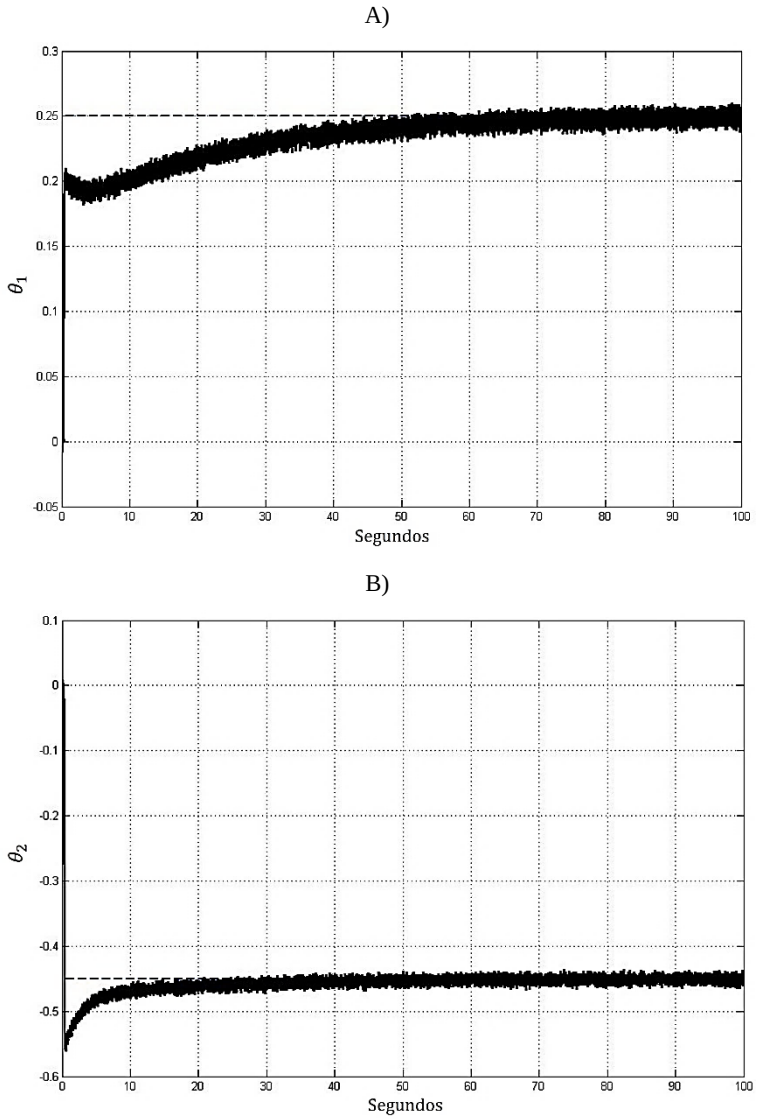


Figura 4.5 Resultado de control del sistema MIMO, usando el algoritmo CVLM.
A) Comportamiento de θ_1. B) Comportamiento de θ_2. La línea segmentada corresponde a la referencia, mientras que la línea continua corresponde a la salida del sistema.

Para la parte de identificación se siguió totalmente lo establecido en el capítulo tres. El periodo de muestreo T_0 para el control de este sistema fue de 0.01 y se observó el comportamiento del sistema con el control durante cien segundos, tiempo suficiente para observar la efectividad del esquema de control neuronal usando RNC. Las perturbaciones se acotaron superiormente en 0.1, es decir, tomaran valores positivos y negativos aleatoriamente menores a este valor.

En primer lugar, se entrenaron a las RNC involucradas en el control del sistema MIMO con el algoritmo CVBP, los parámetros de este algoritmo para cada red se definieron como sigue: se tiene $\eta_1 = \eta_2 = \eta_3 = 0.05$ y $\alpha_1 = \alpha_2 = \alpha_3 = 0.001$. Para el control integral, la constante de

integración se definió como $K_{int} = 0.005$ y por lo que se ve, su efecto no es muy grande en la señal de control $U(k)$.

Los resultados que se obtuvieron con el algoritmo de entrenamiento CVBP se muestran en las figuras 4.2-4.4; en la figura 4.2 se ve que el sistema converge de forma favorable a la referencia deseada, en la figura 4.3 se puede notar que el índice de desempeño tanto de identificación y control es muy bueno, finalmente en la figura 4.4 se puede ver que a las señales generadas por las RNC.

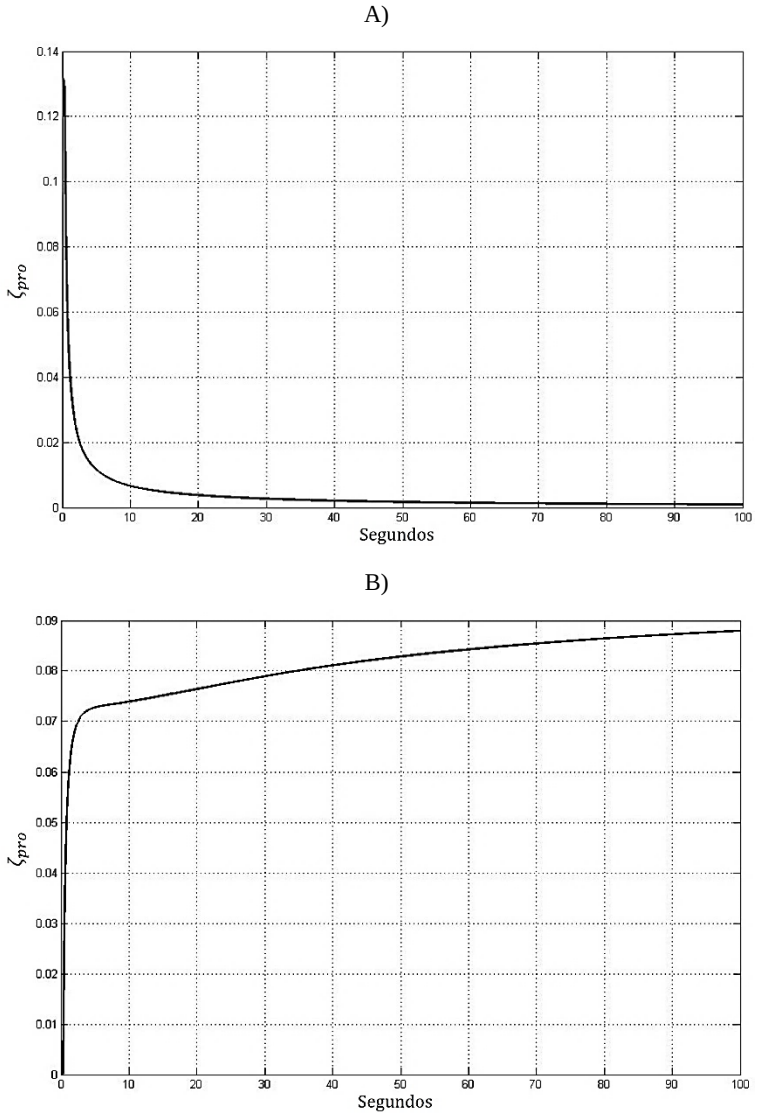


Figura 4.6 Rendimiento de las RNC para el control del sistema MIMO, usando el algoritmo CVLM. A) ζ_{pro} correspondiente al control. B) ζ_{pro} correspondiente a la identificación.

Después se hicieron simulaciones con el algoritmo CVLM para entrenar a las redes en el control del sistema, aunque con este algoritmo se tuvieron demasiados problemas al momento de sintonizarlo, debido a que es muy sensible y tiene muchos parámetros libres. Para asegurar el funcionamiento de las RNC con los parámetros obtenidos con este algoritmo, se tuvo que limitar

aún más a los valores de las matrices A_1, A_2, A_3 y D_3; ya que además de cumplir con la condición de estabilidad, que indica que sus valores propios deben de estar dentro del circulo unitario definido en el dominio complejo, se restringieron todavía más estos valores, resultando en que el módulo de estos no puede ser mayor a 0.2. De esta manera se aseguró que las RNC pudieran funcionar de forma estable. Si los valores propios de las matrices antes mencionadas eran mayor a la cota que se estableció, las redes se volvían inestables.

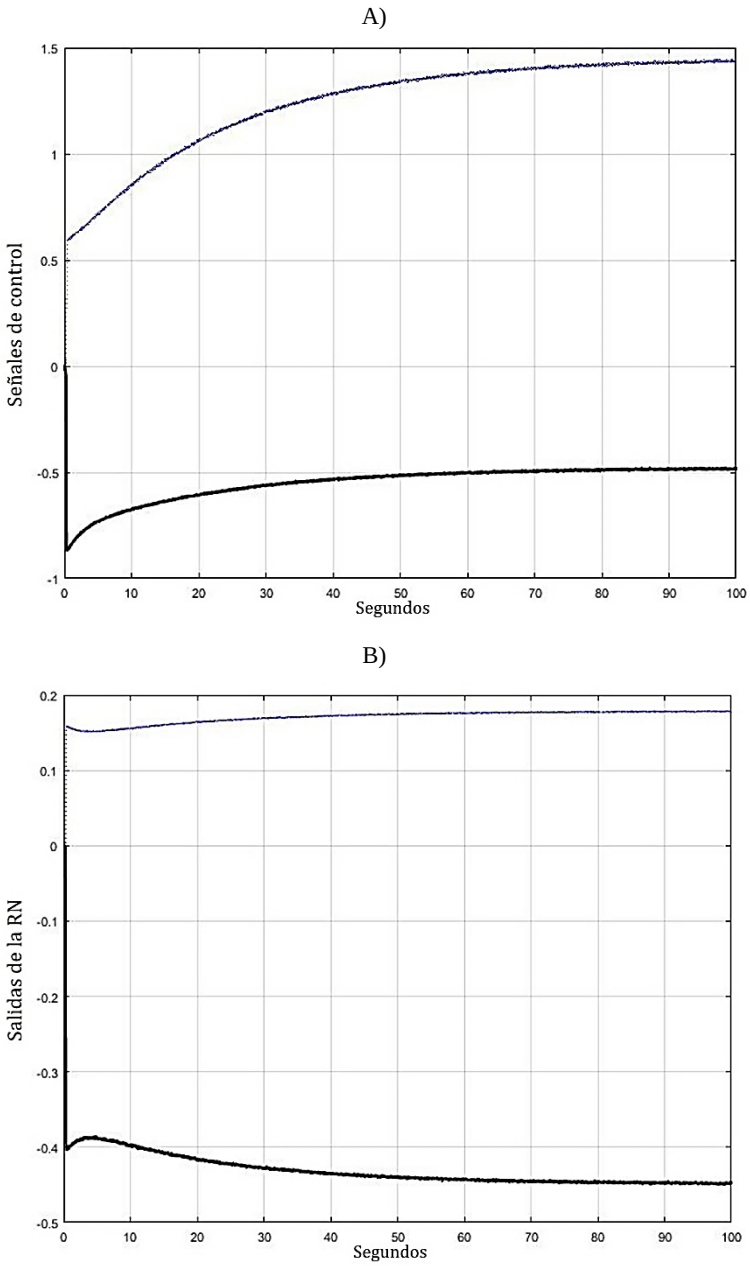


Figura 4.7 Señales generadas por las RNC en el control del sistema MIMO, usando el algoritmo CVLM. A) Señal de control $U(k)$. B) Salida de la CVKFRNN. Para A), la línea punteada corresponde a la primer entrada del sistema y la línea continua a la segunda entrada del sistema. Para B), la línea punteada corresponde a θ_1 y la línea continua a θ_2.

Los valores de los parámetros libres del algoritmo CVLM para este caso se fijaron de la siguiente forma: $\alpha_1 = \alpha_2 = \alpha_3 = 0.999$, $\rho_1 = \rho_2 = \rho_3 = 1 \times 10^{-4}$, además los valores iniciales de las matrices de covarianza se fijaron todos iguales como $P(0) = 1 \times 10^3$. El valor de la ganancia integral para generar a $T(k)$ con este algoritmo fue de $K_{int} = 0.5$, como se puede notar en este caso la ganancia integral fue mucho mayor que para el algoritmo CVBP. Esto porque se limitó el valor de los parámetros de las RNC, lo que da como consecuencia que la señal producida por las redes no sea suficiente para que el sistema alcance a la referencia, es decir, se tiene un error constante entre la referencia y la respuesta del sistema. Lo anterior, implica que la dependencia del control integral es mayor y por ende se tiene que usar una ganancia mayor.

En las figuras 4.5-4.7, se pueden ver los resultados de control con RNC utilizando para el ajuste de parámetros al algoritmo CVLM. La figura 4.5 muestra como converge el sistema a la referencia, en comparación con lo mostrado en la figura 4.2 correspondiente al algoritmo CVBP, se puede notar que al sistema le cuesta más alcanzar el comportamiento deseado. En la figura 4.6 se ve que se tiene un buen desempeño de ζ_{pro} para el control, pero no se tiene un buen desempeño de ζ_{pro} en la identificación; lo cual también es debido a que se limitó a los parámetros de la CVKFRNN, por lo que esta no puede alcanzar al sistema, teniendo siempre un error constante durante todo el tiempo que dura la simulación. Esto se puede ver con mayor claridad en la figura 4.7, en donde se ve que la salida de la CVKFRNN, a pesar de ser similar a la salida del sistema, nunca alcanza el mismo valor que ésta tiene, además, en esta figura se puede ver el comportamiento de la señal de control $U(k)$, que a diferencia del caso anterior es más suave y converge lentamente.

Tabla 4.1 Evaluación del desempeño de las RNC en el control del sistema MIMO.

Algoritmo	$\zeta_{pro}(k)$ final de identificación	$\zeta_{pro}(k)$ final de control
CVBP	0.0012	0.0027
CVLM	0.08795	0.0008487

Finalmente, al igual que para la identificación, en la tabla 4.1 se muestra el rendimiento de ζ_{pro} para la identificación y el control del sistema MIMO. De esta tabla, se puede concluir que el algoritmo CVBP es la mejor opción para ajustar a los parámetros de las RNC en el control de sistemas. Ya que con este algoritmo se obtuvieron los valores más bajos para el índice de desempeño, es decir, los mejores valores para ζ_{pro} sin tener que hacer muchas consideraciones en el funcionamiento de las RNC, a diferencia de lo expuesto para el algoritmo CVLM.

4.3.2. Control de un Sistema SISO usando una Trayectoria como Referencia

Ya que se presentaron resultados de control con el esquema de la figura 4.1 para una referencia constante, se decidió también probar con una referencia variante en el tiempo, tal y como se define en la ecuación (4.5), pero el sistema que se busca que siga a esta referencia es un sistema SISO, cuyo modelo se puede consultar en el apéndice B de esta tesis.

$$R(t) = 0.3\,\mathrm{sen}(\pi t) + 0.3\,\mathrm{sen}(0.5\pi t) \tag{4.5}$$

En esta ocasión el valor de $R(k)$ si variara en cada iteración en la que se tomen los valores de la referencia, los parámetros del sistema SISO se muestran en la tabla 4.2. Estos valores se tomaran en la simulaciones, las cuales tendrán una duración de dieciséis segundos, como se puede ver en la tabla, los parámetros cambian dependiendo del intervalo de tiempo en que este la simulación. La razón de esto, según lo establecido en los trabajos anteriores en los que se usó este sistema, es para que se noten más los efectos de la fricción en el sistema SISO.

Tabla 4.2 Parámetros del sistema SISO.

Parámetro	Valor en el tiempo (t)	
	$0 \leq t < 8$	$t \geq 8$
m	1 Kg	1.2 Kg
F^+	4.2 N	5.4 N
ΔF^+	1.8 N	2.5 N
$\dot{q}_{cr}^+$	$0.1\,\frac{\mathrm{m}}{\mathrm{s}}$	$0.2\,\frac{\mathrm{m}}{\mathrm{s}}$
b^+	$0.5\,\frac{\mathrm{Ns}}{\mathrm{m}}$	$0.4\,\frac{\mathrm{Ns}}{\mathrm{m}}$
F^-	-4 N	-5 N
ΔF^-	-1.7 N	-2 N
$\dot{q}_{cr}^-$	$0.1\,\frac{\mathrm{m}}{\mathrm{s}}$	$0.2\,\frac{\mathrm{m}}{\mathrm{s}}$
b^-	$0.5\,\frac{\mathrm{Ns}}{\mathrm{m}}$	$0.4\,\frac{\mathrm{Ns}}{\mathrm{m}}$
α	0.001	0.001

Para establecer las dimensiones de los parámetros de las RNC se usaran también los subíndices uno, dos y tres, para declarar los valores de CVRNN_1, CVRNN_2 y CVKFRNN respectivamente. Los parámetros iniciales considerados para las RNC se muestran a continuación:

- Para las matrices de pesos sinápticos B_1, C_1, B_2, C_2, B_3, C_3, D_3 y F_3 se escogieron aleatoriamente en el intervalo de valores $(-0.1, 0.1)$.
- Para las matrices de pesos sinápticos A_1, A_2, y A_3 se escogieron para la parte real de sus parámetros valores en el intervalo $(-0.1, 0.1)$, para la parte imaginaria se escogieron valores en el intervalo $(-0.01, 0.01)$.
- Para los vectores de la todas las redes se escogieron condiciones iniciales igual a cero.

Las dimensiones de las RNC se escogieron a experimentalmente y de acuerdo a las dimensiones del sistema, resultando en $m_1 = l_1 = l_2 = m_3 = l_3 = 1$ y $n_1 = m_2 = n_2 = n_3 = 4$.

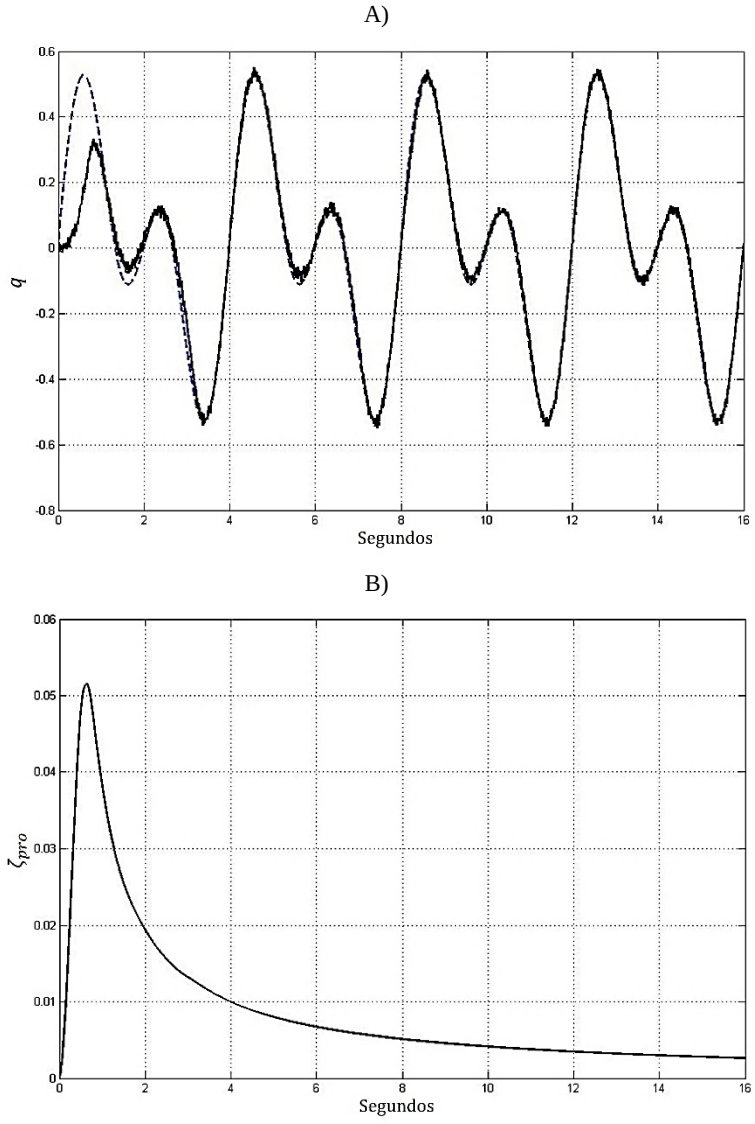


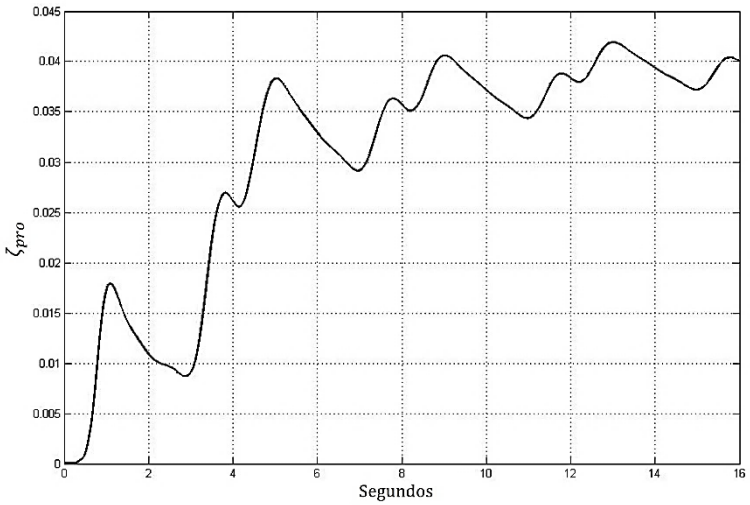
Figura 4.8 Resultados de control del sistema SISO, usando el algoritmo CVBP.
A) Comparación del comportamiento del sistema (línea continua) y la referencia (línea segmentada). B) Desempeño ζ_{pro} del control.

Las señales $U(k)$ y $E_C(k)$ que afectan a la CVRNN_1 y a la CVRNN_2, se trataron con las mimas medidas que en el caso anterior, pero teniendo cuidado con las dimensiones para el sistema SISO.

Las perturbaciones para el caso de la planta SISO se consideraron acotadas por 0.01, ya que si se usa una perturbación con valores más grandes no se lograran apreciar claramente las figuras con los resultados. Como se mencionó al inicio de la sección, las simulaciones tendrán una duración de dieciséis segundos, tiempo que se considera suficiente para que el control sea capaz de hacer que el comportamiento del sistema sea igual al de la referencia deseada.

A diferencia del sistema anterior, aquí se usa un tiempo mucho más pequeño, debido a que este sistema en comparación con el otro reacciona más rápido. Asimismo, se conserva el periodo de muestreo $T_0 = 0.01$, aunque en las simulaciones se vio que entre más pequeño es T_0 se tiene un mejor desempeño de las RNC, pero se tiene la desventaja de que se requieren más recursos computacionales para la aplicación de éstas.

A)



B)

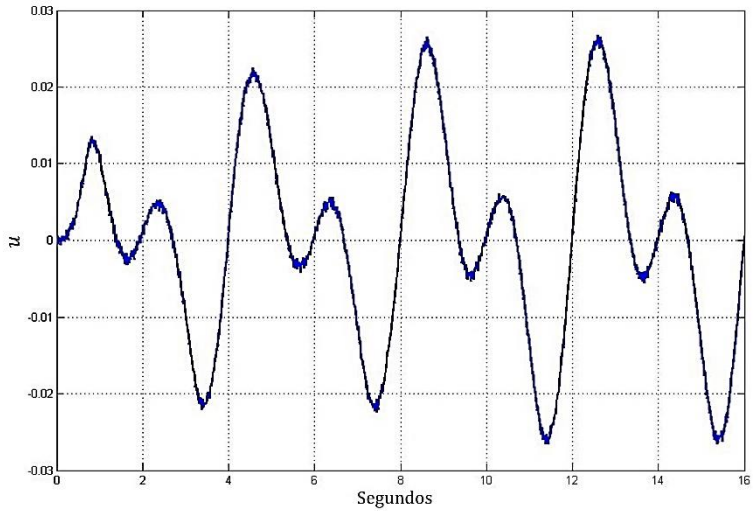
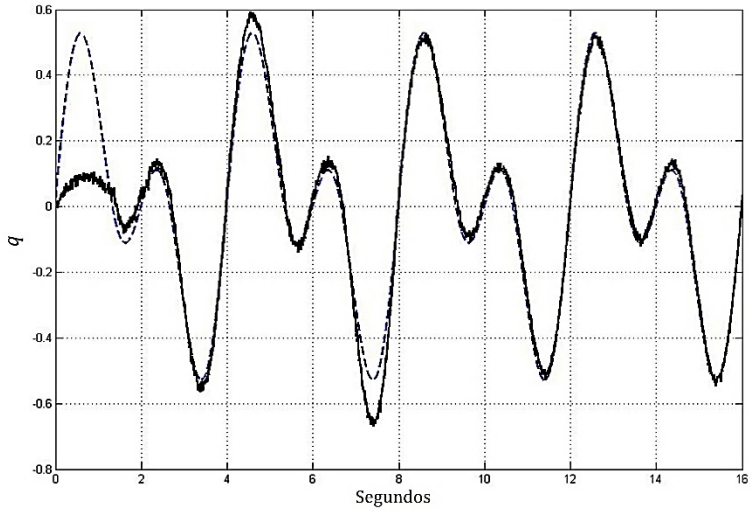


Figura 4.9 Continuación de los resultados del control del sistema SISO, usando el algoritmo CVBP. A) Desempeño ζ_{pro} en la identificación del sistema. B) Señal de control $U(k)$.

Al igual que se ha hecho a lo largo del desarrollo de este trabajo, primero se probó el control con RNC ajustando sus parámetros con el algoritmo CVBP. Los parámetros libres de este algoritmo de entrenamiento se tomaron como $\eta_1 = \eta_2 = \eta_3 = 0.1$ y $\alpha_1 = \alpha_2 = \alpha_3 = 0.04$. Para la ganancia del control integral, se usó un valor de $K_{int} = 0.005$, por lo que en esta ocasión tampoco es muy relevante este control. Los resultados del comportamiento del sistema se pueden ver en las figuras 4.8 y 4.9; la figura 4.8 muestra como la dinámica del sistema se va haciendo similar al de la referencia deseada conforme pasa el tiempo, además se muestra el índice de desempeño para la parte de control, el cual evoluciona de forma favorable. La figura 4.9 muestra el índice de desempeño para la parte de la identificación del sistema, no es tan bueno como el de la parte de control, pero se encuentra en rangos aceptables, aquí también se muestra la señal de control $U(k)$.

A)



B)

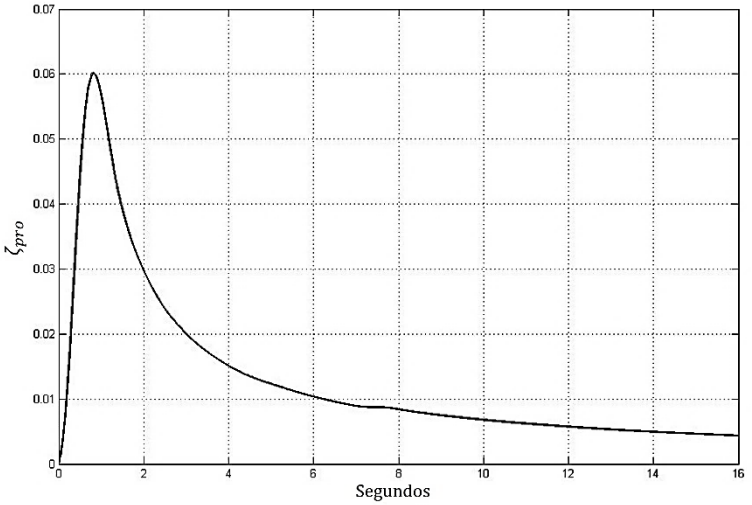


Figura 4.10 Resultados de control del sistema SISO, usando el algoritmo CVLM.
A) Comparación del comportamiento del sistema (línea continua) y la referencia (línea segmentada). B) Desempeño ζ_{pro} del control.

Para el control usando RNC a partir del ajuste de sus parámetros con el algoritmo CVLM, se fijaron los parámetros del este en $\alpha_1 = \alpha_2 = \alpha_3 = 0.999$, $\rho_1 = \rho_2 = \rho_3 = 1 \times 10^{-4}$ y las condiciones iniciales de las matrices de covarianza se fijaron en $P(0) = 1 \times 10^3$. Se empleó la misma ganancia integral para $T(k)$ usada con el algoritmo CVBP. Por lo que a diferencia del control del sistema MIMO, el control integral aquí no será muy relevante. Aunque se pudo aplicar el algoritmo CVLM sin tantas complicaciones como en el caso anterior, de igual forma se tuvieron que acotar los términos de las matrices A_1, A_2, A_3 y D_3, de la misma manera que en el caso del control del sistema MIMO.

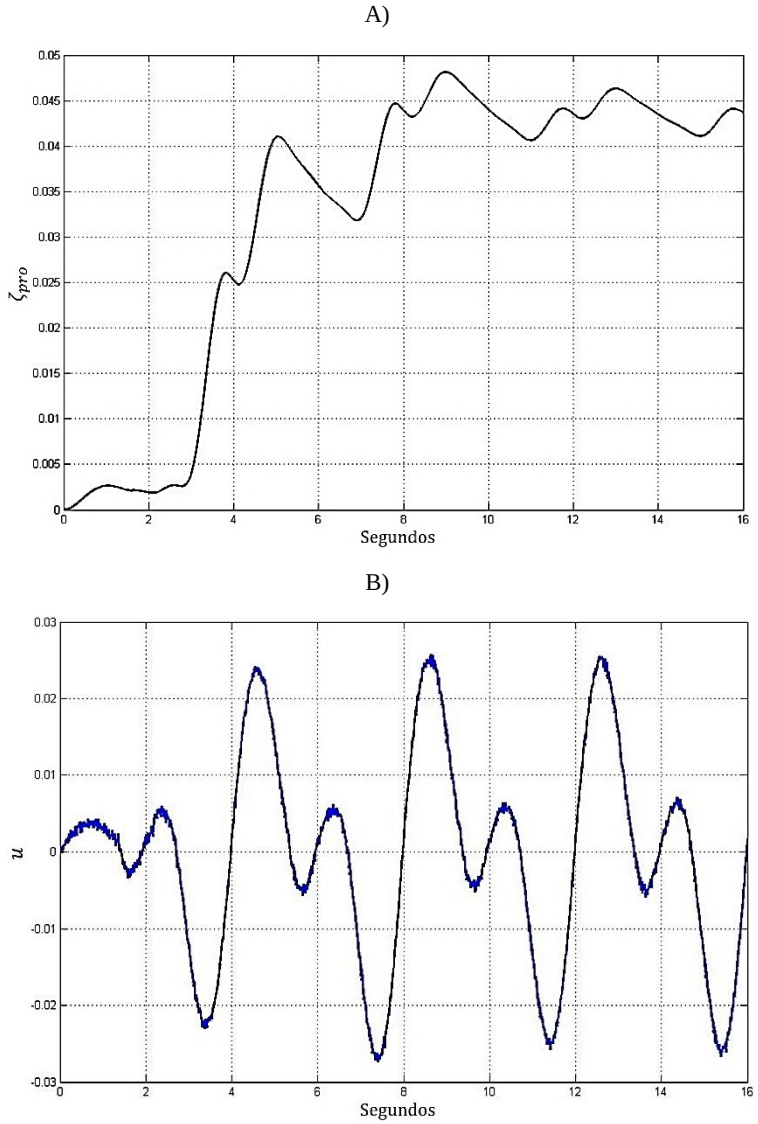


Figura 4.11 Continuación de los resultados del control del sistema SISO, usando el algoritmo CVLM. A) Desempeño ζ_{pro} en la identificación del sistema. B) Señal de control $U(k)$.

En las figuras 4.10 y 4.11 se pueden ver los resultados del control con RNC utilizando el algoritmo CVLM para su ajuste, como se puede notar no difieren mucho del comportamiento generado por

el algoritmo CVBP (figuras 4.8 y 4.9). En la tabla 4.3 se hace una comparación entre los índices de desempeño ζ_{pro} para ambos algoritmos, según lo mostrado en esta tabla la diferencia entre emplear un algoritmo u otro no es tan notoria como en el caso del control del sistema MIMO.

Tabla 4.3 Evaluación del desempeño de las RNC en el control del sistema SISO.

Algoritmo	$\zeta_{pro}(k)$ final de identificación	$\zeta_{pro}(k)$ final de control
CVBP	0.04	0.0026
CVLM	0.0437	0.0043

4.4. Conclusión

Se mostró que las RNC desarrolladas en el capítulo dos son útiles para el control de sistemas, aunque se tuvieron que hacer más consideraciones para su aplicación a diferencia de la identificación de sistemas. Esto debido a que los sistemas usados en esta sección están definidos en el dominio de los números reales, mientras que las RNA que se utilizaron están definidas en el dominio de los números complejos, lo cual genera conflictos al momento en que interactúan los sistemas en conjunto con las redes.

Conclusiones Generales

Se han logrado establecer las topologías correctas de RNRE en el dominio complejo, con la certeza de que los parámetros de estas RN están definidos en el dominio de los números complejos, todo esto según lo establecido en el segundo capítulo de este trabajo.

Asimismo, se probó el desempeño de las topologías aquí establecidas en el problema de identificación de sistemas, en específico de la CVKFRNN, dado que esta topología se desarrolló especialmente para esa tarea. Esto se hizo por medio de simulaciones, en las cuales se obtuvieron resultados favorables; se observó que los algoritmos de entrenamiento desarrollados aquí ajustan los parámetros de la CVKFRNN de manera efectiva, ya que por parte de ambos se tuvieron respuestas muy similares.

Aunque hubo algunas diferencias notorias empezando por el algoritmo CVBP, el cual demostró ser simple de aplicar, además de ser robusto ante las perturbaciones del sistema, pero con la desventaja de que tarda algún tiempo en aprender la dinámica deseada. Mientras que el algoritmo CVLM es más sensible y difícil de aplicar, debido a la gran cantidad de parámetros que tiene; a pesar de eso una vez que se ha logrado estabilizar tiene un desempeño mejor que el CVBP, en el sentido de que, desde las primeras iteraciones del proceso de aprendizaje las RNC entrenadas con este algoritmo ofrecen una señal similar a la señal deseada para éstas, por lo que se obtienen mejores resultados en este proceso en un menor tiempo. Para el proceso de generalización, se mostró que las redes entrenadas con ambos algoritmos de entrenamiento son igual de efectivas, por lo que la elección entre cualquiera de estos algoritmos dependerá de las características del sistema en el que se deseen emplear.

En comparación con los trabajos precedentes también se tuvo una gran mejora, ya que el entrenamiento que se utilizo fue en línea, a diferencia de los otros trabajos en los que para tener resultados medianamente aceptables se requería de hacer un entrenamiento fuera de línea, lo que demuestra además la gran ventaja de usar parámetros complejos en las RN.

Las topologías aquí desarrolladas igualmente se comprobaron para el control de sistemas, siguiendo un esquema de control neuronal en el que se combinan las topologías de CVRNN y CVKFRNN. Aunque, en esta problemática no se lograron resultados tan favorables en comparación con la identificación de sistemas, debido a que aquí la diferencia entre trabajar los sistemas en el dominio real y las RNA en el dominio complejo se hizo más notaria, ya que se tuvieron que tomar más consideraciones y restricciones.

El algoritmo CVBP para el ajuste de los parámetros de las redes en el control demostró ser la mejor opción, gracias a su robustez y su sencilla aplicación, así como a su naturaleza de tardar más en aprender un comportamiento determinado; esta característica queda a la perfección en la

tarea de control, porque las señales que producen las redes no son tan bruscas y los sistemas no se ven afectados negativamente por ellas. En cambio, el algoritmo CVLM resulto ser poco práctico para el control, ya que fue demasiado difícil sintonizar a las redes que conformaban el esquema de control neuronal con este algoritmo, tampoco ayuda mucho su naturaleza de seguir el comportamiento deseado desde las primeras iteraciones, porque se generan respuestas muy bruscas que afectan la estabilidad de los sistemas.

De lo anterior, se concluye que siempre es mejor trabajar los sistemas y las RN en el mismo dominio, al menos para el control de sistemas, tal y como se había desarrollado en los trabajos referentes al control con RNC que se nombraron en el primer capítulo.

Trabajo a Futuro

Hay diversas ideas de trabajos a futuro en base a lo que se hizo, entre las que más destacan se encuentra el desarrollar e implementar métodos que ajusten el valor de las constantes de los algoritmos de entrenamiento. Estos métodos pueden basarse en lógica difusa, en conjunto con los cambios en el error de estimación de la red.

Este ajuste de parámetros es necesario, en especial para el algoritmo CVLM, ya que como se mostró en los resultados de identificación y control de sistemas, este algoritmo es muy sensible en cuanto a sus parámetros, a pesar de presentar un mejor desempeño que el CVBP. Pero el algoritmo CVBP no requiere tanto de un ajuste automático de sus parámetros libres, aunque si se llegase a realizar, este algoritmo podría tener un rendimiento igual o mejor que el algoritmo CVLM; dado que el CVBP es fácil de implementar por no tener tantos parámetros libres, así como también es más robusto ante las perturbaciones presentes en los sistemas.

Se espera además, que en trabajos posteriores se desarrolle una prueba de estabilidad formal para los algoritmos de entrenamiento, ya que ésta podría ayudar en la selección de parámetros libres de las RN y de los propios algoritmos de entrenamiento. Aunque, se sabe que los algoritmos aquí desarrollados son extensiones de los algoritmos de entrenamiento en el dominio real, para los cuales ya se han desarrollado dichas pruebas y por tanto los algoritmos con los que se trabajó se consideran estables.

También, se puede buscar implementar las topologías de RN, así como sus algoritmos de entrenamiento desarrollados en esta tesis en simulaciones de sistemas que se definan en el dominio complejo, en donde se espera que se tengan un mejor desempeño, para la parte de control de sistemas principalmente. Similarmente, se busca que se haga la implementación de las RNC aquí desarrolladas en experimentos que involucren sistemas que existan físicamente, para poder apreciar claramente si en verdad estas topologías son buenas herramientas para poder aplicarse en la identificación y control de sistemas, aunque no se espera que estas implementaciones difieran mucho de las simulaciones mostradas en este trabajo.

Referencias

[1] W. McCulloch y W. Pitts, «A Logical Calculus of Ideas Inmmanent in Nervous Activity,» *Bulletin of Mathematical Biophysics*, pp. 115-133, 1943.

[2] F. Rosenblatt, «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain,» *Psychological Review*, pp. 386-408, 1958.

[3] P. De Wilde, Neural Networks Models, An Analysis, vol. 210, Lecture Notes in Control and Information Sciences, Springer, 1996.

[4] S. Haykin, Neural Networks. A Comprehensive Foundation, Segunda ed., Prentice Hall, 1999.

[5] J. R. Hilera González y V. J. Martinez Hernando, Redes Neuronales Artificiales: Fundamentos, Modelos y Aplicaciones, Segunda ed., Madrid, España: Alfaomega, 2010.

[6] R. Rojas, Neural Networks: A Systematic Introduction, Primera ed., Springer, 1996.

[7] B. Widrow, «An adaptive "ADALINE" neuron using chemical "memistors",» 1960.

[8] J. Hopfield, «Neural Networks and Physical Systems with Emergent Colective Comutational Abilities,» *Processing of the National Academy of Sciences*, nº 79, pp. 2554-2558, 1982.

[9] I. S. Baruch, I. P. Stoyanov y E. Gortcheva, «Topology and learning of a class RNN,» *ELEKTRIK (Turkish Journal of Electrical Engineering and Computer Sciences)*, vol. 4, nº 1, pp. 35-42, 1996.

[10] C. C. Ku y K. Y. Lee, «Diagonal recurrent neural networks for dynamic systems control,» *IEEE Transactions on Neural Networks*, vol. 6, nº 1, pp. 144-156, 1995.

[11] K. Ogata, Sistemas de Control en Tiempo Discreto, Segunda ed., Prentice Hall Inc, 1996.

[12] K. S. Narendra y K. Parthasarathy, «Identification and Control of Dynamical Systems Using Neural Networks,» *IEEE Transactions on Neural Networks*, vol. 1, nº 1, pp. 4-27, Marzo 1990.

[13] D. T. Pham y X. Liu, Neural Networks for Identification, Prediction and Control, Primera ed., Springer-Verlag, 1995.

[14] G. A. Rovithakis y M. A. Christodoulou, «Adaptive Control od Unknow Plants Using Dynamical Neural Networks,» *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, nº 3, pp. 400-412, 1994.

[15] I. S. Baruch y J. L. Olivares, «Implementación de un Multi-Modelo Neuronal Jerárquico Para Identificación y Control de Sistemas Mecánicos,» *Computación y Sistemas,* vol. 9, nº 1, pp. 28-40, 2005.

[16] C. T. Lin y C. G. Lee, Neural fuzzy systems, Prentice Hall, 1996.

[17] J. S. R. Jang, «ANFIS: adaptive-network-based fuzzy inference system,» *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 23, nº 3, pp. 665-685, 1993.

[18] I. S. Baruch, L. A. Hernandez y J. Barrera Cortez, «Recurrent Neural Identification and Sliding Mode Adaptive Control of an Aerobic Fermentation Plant,» *Cientifica ESIME-IPN,* pp. 55-62, 2007.

[19] I. S. Baruch, S. F. Escalante y C. R. Mariaca Gaspar, «Identification, Filtering and Control of Nonlinear Plants by Recurrent Neural Networks Using First and Second Order Algorithms of Learning,» *International Journal of Dynamics of Continuous, Discrete and Impulsive Systems, Series A: Mathematical Analysis, Special Issue on Advances in Neural Net-works Theory an Applications, Part 2,* pp. 512-521, 2007.

[20] I. S. Baruch y C. R. Mariaca Gaspar, «A Levenberg-Marquardt Learning Applied for Recurrent Neural Identification and Control of Wastewater Treatment Bioprocess,» *International Journal of Intelligent Systems,* nº 24, pp. 1094-1114, 2009.

[21] I. S. Baruch, J. M. Flores, F. Nava, I. R. Ramirez y B. Nenkova, «An Advanced Neural Network Topology and Learning, Applied for Identification and Control of a D.C. Motor,» *Proc. 1st Int. IEEE Symp. Intelligent Systems,* pp. 289-295, 2002.

[22] L. Ferariu, «Nonlinear System Identification Based on Evolutionary Dynamic Neural Netwok,» de *Proceedings of the European Control Conference*, Cambridge, U.K., 2003.

[23] S. S. Gee, T. H. Lee y C. J. Harris, Adaptive Neural Network Control of Robotic Manipulators, vol. 19, Worls Scientific Series in Robotics and Intelligent Systems, 1998.

[24] T. Fujiwara, Y. Maeda y H. Ito, «Learning of inverse-kinematics for a robot using high dimensional neural networks,» de *Proceedings of SICE Annual Conference*, Nagoya, Japan, 2013.

[25] R. Fletcher, Practical Methods of Optimization, Segunda ed., U.K.: John Wiley & Sons, 1987.

[26] M. T. Hagan y M. B. Menhaj, «Training Feedforward Networks With the Marqueardt Algorithm,» *IEEE Transactions on Neural Networks,* vol. 5, nº 6, pp. 989-993, 1994.

[27] V. S. Asirvadam, S. F. McLoose y W. G. Irwin, «Parallel and Separable Recursive Levenberg-Marquardt Training Algorithm,» *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing,* pp. 129-138, 2002.

[28] D. J. Tylavsky y G. R. L. Sohie, «Generalization of the Matriz Inversion Lemma,» *Proccedings of the IEEE,* vol. 74, nº 7, pp. 1050-1052, 1986.

[29] A. S. Householder, The Theory of Matrices in Numerical Analysis, Blais-dell, New York., 1964.

[30] A. Hirose, Complex-Valued Neural Networks, Segunda ed., Springer, 2012.

[31] H. G. Zimmermann y T. Ogawa, «Historical Consistent Complex-Valued Recurrent Neural Network,» *ICANN 2011,* pp. 185-192, 2011.

[32] H. G. Zimmermann, A. Minin y V. Kusherbaeva, «Historical Consistent Complex Valued Recurrent Neural Network,» *ICANN 2011,* pp. 185-192, 2011.

[33] H. G. Zimmermann, A. Minin y V. Kusherbaeva, «Comparison of the Complex Valued and Real Valued Neural Networks Trained with the Gradient Descent and Random Search Algorithms,» *ESANN 2011 Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence,* pp. 213-218, 2011.

[34] N. N. Aizenberg, Y. L. Ivaskiv y D. A. Pospelov, «A certain generalization of threshold functions,» *Dokrady Akademii Nausk SSSR 196,* pp. 1287-1290, 1971.

[35] B. Widrow, J. McCool y M. Ball, «The Complex LMS Algorithm,» *Proceedings of the IEEE,* vol. 63, pp. 719-720, 1975.

[36] H. Leung y S. Haykin, «The Complex BackPropagation Algoritm,» *IEEE Transactions on Signal Processing,* vol. 39, nº 9, pp. 2101-2104, 1991.

[37] N. Benvenuto y F. Piazza, «On the Complex Backpropagation Algorithm,» *Transactions on Signal Processing,* vol. 40, pp. 967-969, 1992.

[38] A. Hirose, «Motion Controls Using Complex-Valued Neural Networks with Feedback Loops,» *Proc, IEEE International Conference on Neural Networks,* vol. 1, pp. 156-161, 1993.

[39] A. Minin, Y. Chistyakov, E. Kholodova, H. G. Zimmermann y A. Knoll, «Complex-Valued Open Neural Network for Tower Transformer Modeling,» *International Journal of Applied Mathematics and Informatics,* vol. 6, nº 1, pp. 41-48, 2012.

[40] T. Kitajima y T. Yasuno, «Ouput Prediction of Wind Power Generation System Using Complex-Valued Neural Networks,» *Proceedings of SICE Annual Conference,* pp. 3610-3613, 2010.

[41] A. Luchetta, S. Manetti y M. C. Piccirilli, «Analog System Modeling Based on a Double Modified Complex-Valued Neural Network,» de *The 2013 International Joint Conference on Neural Networks,* Dallas TX, U.S.A, 2013.

[42] L. Zheng y Z. Wang, «desing of FIR Digital Filters in Complex Domain by Complex-Valued Neural Networks,» *Proceedings of the 2010 2nd International Conference on Information Science and Engineering*, pp. 4499-4502, 2010.

[43] K. Kawashima y T. Ogawa, «Complex-Valued Neural Network for Group-Movement Control of Mobile Robots,» *Proc. SICE Annual Conference 2012*, 2012.

[44] M. N. Szilagyi y B. Salik, «Neural Networks with Complex Activations and Connection Weights,» *Complex Systems*, vol. 2, pp. 115-126, 1994.

[45] A. Minin, A. Knoll y H. G. Zimmermann, «Complex-Valued Recurrent Neural Network: From Architecture to Training,» *Journal of Signal and Information Processing*, pp. 192-197, 2012.

[46] V. M. Arellano Quintanilla, *Identificación y Control de Sistemas Dinámicos Utilizando Redes Neuronales Complejas Recurrentes*, Ciudad de México: Tesis de Maestría, Departamento de Control Automático, CINVESTAV IPN, 2015.

[47] E. Perez Reynaud, *Identificación y Control de Sistemas No Lineales Utilizando el Algoritmo de Aprendizaje de Levenberg-Marquardt Para Redes Neuronales Complejas Recurrentes*, Ciudad de México: Tesis de Maestría, Departamento de Control Automático, CINVESTAV IPN, 2015.

[48] R. V. Churchill y J. W. Brown, Complex Variable and Applications, Octava ed., McGraw-Hill, 2009.

[49] J. M. Flores Albino, *Aprendizaje Neuronal para el control de Sistemas No Lineales*, Ciudad de México: Tesis de Doctorado, Departamento de Control Automatico, CINVESTAV IPN, 2004.

[50] S. F. Escalante Magaña, *Identificación, Filtraje y Control de Sistemas No Lineales Usando Redes Neuronales Recurrentes con el Aprendizaje Recursivo de Levenberg-Marquardt*, Ciudad de México: Tesis de Maestría, Departamento de Control Automático, CINVESTAV-IPN, 2006.

[51] C. R. Mariaca Gaspar, *Topologías, aprendizaje y estabilidad de redes neuronales híbridas con aplicación en procesos biotecnológicos no lineales*, Ciudad de México: Tesis de Doctorado, Departamento de Control Automático, CINVESTAV IPN, 2009.

[52] I. Baruch, C. Mariaca Gaspar y J. Barrera Cortes, «Recurrent Neural Network Identification and Adaptive Neural Control of Hydrocarbon Biodegradation Processes,» de *Recurrent Neural Networks*, In-Teh, 2008, pp. 61-88.

[53] A. P. Sage, Optimum Systems Control, Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1968.

[54] E. A. Wan y F. Beaufayas, «Diagramatic Methods for Deriving and Relating Temporal Neural Network Algorithms,» *Neural Computation,* pp. 182-201, 1996.

[55] F. Nava, I. S. Baruch, A. Poznyak y B. Nenkova, «Stability proofs of advanced recurrent neural networks topology and learning,» *Comptes Rendus (Proceedings of the Bulgarian Academy of Sciences),* vol. 57, nº 1, pp. 27-32, 2004.

[56] L. S. Ngia y J. Sjoberg, «Efficient Training of Neural Nets for Nonlinear Adaptive Filtering Using a Recursive Levenberg-Marquardt Algorithm,» *IEEE Transactions on Signal Processing,* vol. 48, pp. 1915-1927, 2000.

[57] C. R. Mariaca Gaspar, J. C. Tovar Rodriguez y F. Ortiz Rodriguez, «Recurrent Neural Control of a Continuous Bioprocess Using First and Second Order Learning,» *MICAI 2012,* pp. 211-222, 2013.

[58] I. S. Baruch, V. M. Arellano Quintana y E. Pérez Reynaud, «Complex-valued neural network topology and learning applied for identification and control of nonlinear systems,» *Neurocomputing,* vol. 233, pp. 104-115, 2017.

[59] I. S. Baruch, E. Gortcheva y R. Garrido, «Redes Neuronales para la Identificacion de Objetos No Lineales,» *Cientifica ESIME,* pp. 39-45, 1999.

[60] I. S. Baruch y R. Garrido, «A direct adaptive neural control scheme with integral terms,» *International Journal of Intelligent Systems (Special issue on soft computing for modelling, simulation and control of nonlinear dynamical systems, guest eds: O.Castillo, and P.Melin). WILEY Inter-Science,* pp. 213-224, 2005.

[61] I. S. Baruch, L. A. Hernandez, C. R. Mariaca Gaspar y B. Nenkova, «An Adaptive Sliding Mode Control with I-term Using Recurrent Neural Identifier,» *Cybernetics and Information Technologies, BAS,* pp. 21-32, 2007.

[62] K. H. Young y F. L. Lewis, «Reinforcement Adaptive Learning Neural Network Based Friction Compensation for High Speed and Precision,» de *Proceedings of the 37th IEEE Conference on Decision & Control*, Tampa, Florida USA, 1998.

[63] K. Ogata, Modern Control Engieneering, Cuarta ed., Pearson Education International, 2002.

[64] I. S. Baruch, P. Georgieva, J. Barrera Cortez y S. Feyo de Azevedo, «Adaptive Recurrent Neural Network Control of Biological Wastewater Treatment,» *International Journal of Intelligent Systems WILEY Inter-Science,* pp. 173-194, 2005.

Apéndice A. Modelado Matemático de un Robot Planar de dos Grados de Libertad

Como se establece en [23] y [47], el modelo del robot debe tener la siguiente forma:

$$D(\theta)\ddot{\theta} + C\left(\theta, \dot{\theta}\right)\dot{\theta} + G(\theta) = U$$

Con:

- $D(\theta)$, representado la matriz de inercia del robot.
- $C\left(\theta, \dot{\theta}\right)$, representando la matriz de términos de Coriolis y centrífugos del robot.
- $G(\theta)$, como la matriz de términos de gravedad del robot.
- $\ddot{\theta}$, aceleración angular en forma matricial.
- $\dot{\theta}$, velocidad angular en forma matricial.
- θ, posición angular en forma matricial.
- U, fuerza para el control.

En la figura A.1 se muestra una representación gráfica del robot que se quiere modelar, junto con todos los parámetros que se consideran importantes.

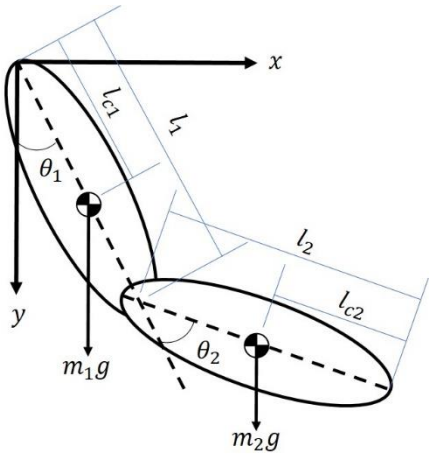


Figura A.1 Robot planar de dos grados de libertad.

Donde:

- θ_1, es la posición angular del primer eslabón del robot, expresada en radianes.
- θ_2, es la posición angular del segundo eslabón del robot, expresada en radianes.
- l_1, representa la longitud del primer eslabón, expresada en metros.
- l_2, representa la longitud del segundo eslabón, expresada en metros.

- l_{c1}, representa la distancia del punto de referencia del modelo al centro de masa del primer eslabón, expresada en metros.
- l_{c2}, representa la distancia del final del primer eslabón al centro de masa del segundo eslabón, expresada en metros.
- m_1, masa del primer eslabón, expresada en kilogramos.
- m_2, masa del segundo eslabón, expresada en kilogramos.
- g, constante de gravedad equivalente a $9.81\frac{m}{s^2}$
- x y y, son los ejes coordenados para representar las posiciones angulares.

Para empezar, se definen las posiciones de los centros de masa de cada eslabón.

$$\begin{bmatrix} x_{c1} \\ y_{c1} \end{bmatrix} = \begin{bmatrix} l_{c1}\,\mathrm{sen}(\theta_1) \\ -l_{c1}\cos(\theta_1) \end{bmatrix}$$
$$\begin{bmatrix} x_{c2} \\ y_{c2} \end{bmatrix} = \begin{bmatrix} l_1\,\mathrm{sen}(\theta_1) + l_{c2}\,\mathrm{sen}(\theta_1+\theta_2) \\ -l_1\cos(\theta_1) - l_{c2}\cos(\theta_1+\theta_2) \end{bmatrix}$$

Derivando con respecto del tiempo a las posiciones se puede obtener la velocidad de cada eslabón.

$$v_{c1} = \begin{bmatrix} l_{c1}\dot{\theta}_1\cos(\theta_1) \\ l_{c1}\dot{\theta}_1\,\mathrm{sen}(\theta_1) \end{bmatrix}$$
$$v_{c2} = \begin{bmatrix} l_1\dot{\theta}_1\cos(\theta_1) + l_{c2}\left(\dot{\theta}_1+\dot{\theta}_2\right)\cos(\theta_1+\theta_2) \\ l_1\dot{\theta}_1\,\mathrm{sen}(\theta_1) + l_{c2}\left(\dot{\theta}_1+\dot{\theta}_2\right)\mathrm{sen}(\theta_1+\theta_2) \end{bmatrix}$$

Con las velocidades se obtiene la energía cinética total del robot, quedando definida de la siguiente forma:

$$\mathcal{K} = \frac{1}{2}m_1 v_{c1}^T v_{c1} + \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}m_2 v_{c2}^T v_{c2} + \frac{1}{2}I_2\left(\dot{\theta}_1+\dot{\theta}_2\right)^2$$
$$\mathcal{K} = \frac{1}{2}m_1 l_{c1}^2\dot{\theta}_1^2 + \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}m_2 l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2 l_{c2}^2\left(\dot{\theta}_1+\dot{\theta}_2\right)^2 + \frac{1}{2}I_2\left(\dot{\theta}_1+\dot{\theta}_2\right)^2 + m_2 l_1 l_{c2}\dot{\theta}_1\left(\dot{\theta}_1+\dot{\theta}_2\right)\cos(\theta_2)$$

Simplificando la energía cinética se puede expresar de forma matricial como:

$$\mathcal{K} = \frac{1}{2}\dot{\theta}^T D(\theta)\dot{\theta}, \qquad \dot{\theta} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

Para continuar el desarrollo del modelo se definen las siguientes constantes:

$$p_1 = m_1 l_{c1}^2 + m_2 l_1^2 + I_1$$
$$p_2 = m_2 l_{c2}^2 + I_2$$
$$p_3 = m_2 l_1 l_{c2}$$

$$p_4 = m_1 l_{c2} + m_2 l_1$$
$$p_5 = m_2 l_{c2}$$

Ahora se puede definir correctamente a $D(\theta)$.

$$D(\theta) = \begin{bmatrix} p_1 + p_2 + 2p_3\cos(\theta_2) & p_2 + p_3\cos(\theta_2) \\ p_2 + p_3\cos(\theta_2) & p_2 \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$$

Para la matriz de términos de Coriolis se emplean los símbolos de Christofel.

$$c_{111} = \frac{1}{2}\frac{\partial d_{11}}{\partial \theta_1} = 0$$
$$c_{121} = c_{211} = \frac{1}{2}\frac{\partial d_{11}}{\partial \theta_2} = -p_3\,\mathrm{sen}(\theta_2)$$
$$c_{221} = \frac{\partial d_{12}}{\partial \theta_2} - \frac{1}{2}\frac{\partial d_{22}}{\partial \theta_1} = -p_3\,\mathrm{sen}(\theta_2)$$
$$c_{112} = \frac{\partial d_{21}}{\partial \theta_1} - \frac{1}{2}\frac{\partial d_{11}}{\partial \theta_2} = p_3\,\mathrm{sen}(\theta_2)$$
$$c_{122} = c_{212} = \frac{1}{2}\frac{\partial d_{22}}{\partial \theta_1} = 0$$
$$c_{222} = \frac{1}{2}\frac{\partial d_{22}}{\partial \theta_2} = 0$$

Organizando los términos anteriores se puede construir la matriz $C(\theta, \dot{\theta})$.

$$C(\theta, \dot{\theta}) = \begin{bmatrix} -p_3\dot{\theta}_2\,\mathrm{sen}(\theta_2) & -p_3(\dot{\theta}_1 + \dot{\theta}_2)\,\mathrm{sen}(\theta_2) \\ p_3\dot{\theta}_1\,\mathrm{sen}(\theta_2) & 0 \end{bmatrix}$$

Antes de definir la matriz de términos de gravedad del robot, se debe de definir la energía potencial de éste.

$$\mathcal{P} = m_1 g y_{c1} + m_2 g y_{c2}$$
$$\mathcal{P} = -m_1 g l_{c1}\cos(\theta_1) - m_2 g(l_1\cos(\theta_1) + l_{c2}\cos(\theta_1 + \theta_2))$$

Ahora se puede obtener a la matriz $G(\theta)$ de la siguiente forma:

$$G(\theta) = \frac{\partial \mathcal{P}}{\partial \theta} = \begin{bmatrix} m_1 g l_{c1}\,\mathrm{sen}(\theta_1) + m_2 g(l_1\,\mathrm{sen}(\theta_1) + l_{c2}\,\mathrm{sen}(\theta_1 + \theta_2)) \\ m_2 g l_{c2}\,\mathrm{sen}(\theta_1 + \theta_2) \end{bmatrix}$$
$$G(\theta) = \begin{bmatrix} p_4\,\mathrm{sen}(\theta_1) + p_5 g\,\mathrm{sen}(\theta_1 + \theta_2) \\ p_5 g\,\mathrm{sen}(\theta_1 + \theta_2) \end{bmatrix}$$

De esta forma se completa el modelo deseado.

Apéndice B. Modelado Matemático de un Sistema Mecánico de un Grado de Libertad con Términos de Fricción

El sistema mecánico de un grado de libertad que se consideró se retomó de [15] y [62] (ver figura B.1). Se trata de una masa que se mueve en una dirección, tomando en cuenta los efectos de la fuerza de fricción (fuerza de fricción debida al desplazamiento y fuerza de fricción a velocidad cero) que se presenta entre la masa y la superficie en la que se desplaza.

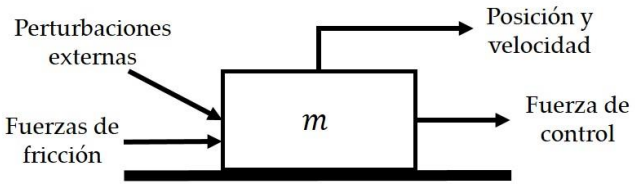


Figura B.1 Sistema mecánico de un grado de libertad.

La ecuación correspondiente al movimiento de la masa es la siguiente:

$$m\ddot{q}(t) + fr(\dot{q}, t) + d(t) = u(t)$$

Donde:

- m, es la masa del sistema.
- $q(t)$, representa al desplazamiento relativo.
- $fr(\dot{q}, t)$, representa a la fuerza de fricción.
- $u(t)$, representa a la fuerza de control.
- $d(t)$, es la perturbación a la que está expuesta el sistema.

La perturbación que se consideró para este sistema se encuentra acotada, representa el ruido de medición y el ruido de la fuente de poder y queda representada de la siguiente manera:

$$\|d(t)\| \leq b_d$$

Con b_d igual a una constante. Para modelar el efecto de la fricción, $fr(\dot{q}, t)$ se define de la siguiente forma:

$$fr(\dot{q}, t) = F_{slip}(\dot{q})\beta(\dot{q}) + F_{stick}(u)[1 - \beta(\dot{q})]$$

Con:

$$\beta(\dot{q}) = \begin{cases} 1, & |\dot{q}| > \alpha \\ 0, & |\dot{q}| \leq \alpha \end{cases} \qquad \alpha > 0$$

Donde

- F_{slip}, se refiere a la fuerza de fricción durante el desplazamiento.
- F_{stick}, se refiere a la fuerza de fricción a velocidad cero.

La ecuación que define a $F_{stick}(u)$ es:

$$F_{stick}(u) = \begin{cases} F_s^+, & 0 < F_s^+ \leq u(t) \\ u(t), & F_s^- < u(t) < F_s^+ \\ F_s^-, & 0 > F_s^- \geq u(t) \end{cases}$$

Donde:

- F_s^+, es el límite positivo de la fuerza de fricción estática.
- F_s^-, es el límite negativo de la fuerza de fricción estática.

Las ecuaciones que definen a $F_{slip}(\dot{q})$ son:

$$F_{slip}(\dot{q}) = F_d^+(\dot{q})\mu(\dot{q}) + F_d^-(\dot{q})\mu(-\dot{q})$$

Con:

$$\mu(\dot{q}) = \begin{cases} 1, & \dot{q} > 0 \\ 0, & \dot{q} \leq 0 \end{cases}$$

$$F_d^+ = F_s^+ - \Delta F^+ \left[1 - \exp\left(-\frac{\dot{q}}{\dot{q}_{cr}^+}\right)\right] + b^+\dot{q}$$

$$F_d^- = F_s^- - \Delta F^- \left[1 - \exp\left(-\frac{\dot{q}}{\dot{q}_{cr}^-}\right)\right] + b^-\dot{q}$$

Donde:

- ΔF^+ y ΔF^-, son los respectivos incrementos desde el nivel de fuerza estática a la fuerza cinética.
- $\dot{q}_{cr}^+$ y $\dot{q}_{cr}^-$, son las velocidades críticas de Stribeck.
- b^+y b^-, son los coeficientes de fricción viscosa.