CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE CONTROL AUTOMÁTICO

IDENTIFICACIÓN Y CONTROL DE SISTEMAS DINÁMICOS UTILIZANDO REDES NEURONALES COMPLEJAS RECURRENTES

T E S I S

QUE PRESENTA

Victor Manuel Arellano Quintana

PARA OBTENER EL GRADO DE
Maestro en Ciencias

EN LA ESPECIALIDAD DE
Control Automático

DIRECTOR DE TESIS
Dr. Ieroham Salomon Barouh

MÉXICO, D.F. FEBRERO DEL 2015

Agradecimientos

A mis padres y hermana, que en todo momento me han apoyado, alentado y aún más importante, han confiado en mí, a ustedes mi más profundo reconocimiento y gratitud.

A mi familia, por todas y cada una de sus enseñanzas, experiencias, pláticas y consejos brindados a lo largo de mi vida, y que indudablemente han formado la persona que soy hoy en día.

A mis amigos y compañeros de la vida, a todos ellos y en especial a los amigos que encontré durante esta etapa de mi vida, muchas gracias por su apoyo y por haber compartido sus vivencias, consejos, conocimientos y lo más importante, su tiempo conmigo.

A mi asesor de tesis, Dr. Ieroham S. Baruch por su orientación para la realización de esta tesis.

Al Dr. Carlos Mariaca y al Dr. Moisés Bonilla, por haber dedicado su tiempo en revisar esta tesis y brindarme sus comentarios y revisiones para que este trabajo fuera mejor.

A mi Alma Máter el Instituto Politécnico Nacional, por haberme brindado los conocimientos y valores que en lo personal considero invaluables, siempre estaré en deuda con ella.

Al pueblo de México, que a través del CONACyT se me otorgó una beca para realizar mis estudios de Maestría.

A mis Padres.

Índice General

Lista de Figuras

Lista de Tablas

Resumen

En el presente trabajo se realiza un estudio acerca de las redes neuronales en el dominio complejo. Primeramente se hace una reseña histórica acerca de ellas, demostrando así que su estudio aún está en constante desarrollo y la importancia por ampliar esta línea de investigación.

Posterior a esto, se introduce al lector a los preliminares matemáticos que se requieren para entender el desarrollo de este trabajo, dentro de estos preliminares se aborda el problema de tener una función de costo en el dominio complejo la cual se tiene que minimizar y al no ser una función analítica presenta complicaciones en el desarrollo de un algoritmo de aprendizaje basado en gradiente descendente.

Se mencionan los conceptos básicos de las redes neuronales, y después se desarrolla la contraparte en el dominio complejo. En la contraparte compleja se muestran las soluciones analíticas que se han reportado en la literatura más actual para dos tipos de funciónes de activación; en este trabajo se propone una extensión de un resultado basado en reglas diagramáticas para obtención de algoritmos de aprendizaje basadas en Backpropagation para el caso complejo.

Finalmente, se aplica la metodología antes descrita para el entrenamiento de redes neuronales complejas recurrentes con el fin de resolver tareas de identificación y control directo adaptable de dos robots con articulaciones flexibles: de uno y de dos grados de libertad. En el trabajo se presentan dos esquemas de control adaptable, el primer esquema contiene una prealimentación de la consigna y una realimentación de los estados, para lo cual se implementa una tercera red neuronal compleja con el objetivo de identificar el sistema dinámico. El segundo esquema incluye las tres redes neuronales del primer esquema de control y un controlador de término integral para reducir el error de la salida en estado estacionario. Los resultados obtenidos de las simulaciones de dos robots demuestran que la metodología propuesta utilizando reglas diagramáticas complejas para el aprendizaje Backpropagation de redes neuronales complejas funciona correctamente.

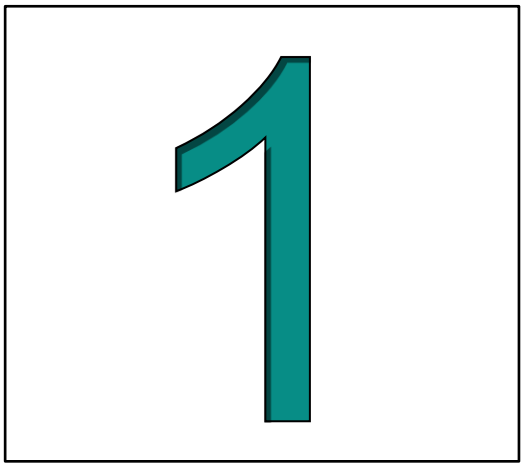
Abstract

In this work a study on neural networks in the complex domain is done. First of all, a brief background history about them is given, showing that the study is still under constant development and the importance to extend this line of research.

After this, the reader is introduced to the mathematical preliminaries required to understand the development of this work, within these preliminaries we address the problem of having a cost function in the complex domain which has to be minimized and because this function is not an analytical function, it presents complications for developing a learning algorithm based on gradient descent.

The basics concepts of neural networks are mentioned, and then we develop their counterpart in the complex domain. In the complex counterpart analytical solutions that have been reported in the current literature for two types of activation functions, are shown; in this work an extension of a system based on diagrammatic rules in order to obtain learning algorithms for Complex-Backpropagation is proposed.

Finally, we applied the methodology described above for complex training recurrent neural networks in order to solve tasks of identification and direct adaptive control of two robots with flexible arms: with one and two degrees of freedom. In this work two adaptive control schemes are presented, the first scheme, contains an reference feedforward control and a state feedback control, which is solved by a third complex valued neural network aimed to identify the dynamic system. The second scheme includes the three neural networks of the first scheme of control and an integral term controller to reduce the steady-state error. The good simulation results obtained for two robots show that using the proposed methodology based on complex diagrammatic rules for Backpropagation learning of complex neural networks works properly.



INTRODUCCIÓN

1. Introducción

Redes Neuronales para Identificación y Control de Sistemas Dinámicos

En la actualidad se tienen innumerables ejemplos de aplicación de las redes neuronales, demostrando así su gran capacidad para clasificación, predicción, identificación y control de sistemas dinámicos.

En [1], Narendra y Parthasarathy proponen lo que sería el primer trabajo en el cual se demuestra que las redes neuronales son útiles para tareas de identificación y control de sistemas dinámicos tanto lineales como no lineales, ya sea con redes neuronales hacia adelante o recurrentes; éstas últimas son un modelo computacional más potente que las clásicas redes neuronales hacia adelante. Ésta mayor potencia proviene que las redes recurrentes son capaces de procesar secuencias temporales gracias a la posibilidad de recordar la historia relevante de la secuencia por medio de una representación en forma de estado.

Otros autores como Pham y Liu [2], abordan problemas de identificación, predicción y control de sistemas dinámicos, proponiendo nuevas topologías para el control de robots manipuladores y diferentes plantas mecánicas, obteniendo buenos resultados.

Aunado a las redes neuronales se han implementado topologías de redes neuronales utilizando lógica difusa, creando así, multimodelos neuronales para la identificación y control de sistemas mecánicos, obteniendo resultados satisfactorios [3].

Las redes neuronales son capaces de aproximar funciones no lineales, lo que las hace una gran herramienta para la identificación de sistemas con dinámicas no lineales, siempre y cuando, tengan el suficiente número de pares entrada-salida para su entrenamiento. Una vez que se tiene identificada a la planta se implementa un control adaptable ya sea directo o indirecto.

Avances en Redes Neuronales en el Dominio Complejo

Por otro lado, el estudio de las redes neuronales en el dominio complejo ha tenido menos atención por parte de la comunidad científica, los resultados hasta el momento abarcan análi-

sis acerca de las funciones de activación, algoritmos de entrenamiento, diferentes topologías de redes, etc.

El estudio importante de las redes neuronales complejas comienza en la década de los setenta, cuando Aizenberg presenta un resultado extendiendo el valor de salida binario de "0" y "1" a múltiples valores en el plano complejo [4].

En esa misma década, en el problema de acondicionamiento de señales para radares, comunicaciones y otros sistemas, donde inevitablemente se utilizan números complejos, Widrow presenta el algoritmo de mínimos cuadrados en su versión compleja (LMS) [5]. Uno de los aspectos importantes de este resultado es que el algoritmo LMS es lineal, por lo que la dinámica está bien establecida y se puede decir lo siguiente, *se pueden sustituir los traspuestos de vectores y matrices en el dominio real por la Hermitiana conjugada (transpuesta conjugada).*

A principios de la década de los noventa, Leung y Haykin derivan un algoritmo de retropropagación del error en su versión compleja (*"Complex-Valued Backpropagation"*) utilizando una función de activación sigmoide con dominio en los números complejos; este algoritmo es implementado para una tarea de clasificación [6].

Al mismo tiempo, Benvenuto y Piazza consideran una red neuronal compleja con una función de activación que es expresada por separado, en su parte real y su parte imaginaria, que llamaron "*real-imaginary type activation function",* utilizando funciones que mapean de los reales a los reales para cada parte del número complejo; este tipo de función presenta ventajas tanto para derivar el algoritmo de entrenamiento como para evitar singularidades al ser evaluada [7]; otros investigadores utilizan este tipo de funciones de activación [8], [9], [10]; trabajos más recientes reportados en [11], proponen utilizar funciones de activación utilizando la magnitud y fase de los números complejos.

En cuanto a la implementación práctica, en la actualidad existen pocas aplicaciones que utilizan redes neuronales complejas. La mayoría de estas aplicaciones están relacionadas con sistemas oscilatorios o sistemas que por su naturaleza física, es conveniente tratarlos en el

dominio complejo, tales pueden ser: ondas electromagnéticas, ondas de luz, procesamiento de imágenes, sistemas eléctricos, etc. ([12], [13], [14]).

En [13], los autores aplican un tipo especial de red neuronal compleja para el modelado de un transformador de potencia, obteniendo buenos resultados. El inconveniente de este artículo es que no se menciona casi nada sobre la presencia de puntos de singularidad en la función de activación utilizada.

Por otro lado, en el campo de la identificación y control de sistemas mecánicos las redes neuronales complejas han tenido una presencia menor. A pesar de ello, algunos autores [14], [15] propusieron utilizar redes neuronales complejas para la identificación y control de plantas mecánicas, obteniendo buenos resultados. En [14], los autores aplicaron una red neuronal compleja para la identificación de un evaporador industrial utilizando un algoritmo evolutivo para el diseño de la red. En este mismo artículo utilizan funciones de base radial evitando así el cálculo de los términos de gradiente en el algoritmo de aprendizaje. Otros trabajos [15], utilizan redes neuronales complejas para este tipo de sistemas, obteniendo resultados satisfactorios; en [16], un tipo de red neuronal compleja se utiliza para el modelado del sistema de Lorenz, obteniendo de la misma manera resultados satisfactorios.

1.1 Motivación del tema de tesis

El estudio de las redes neuronales complejas ha despertado un especial interés en aplicaciones donde se requiere la manipulación de números complejos. Sin embargo, su estudio aún se puede extender a problemas donde que no los utilicen, pero que si presentan una ventaja clara sobre las redes neuronales ordinarias, [17]. Debido al poco estudio que han tenido, como ya se mencionó anteriormente, el desarrollo de algoritmos de aprendizaje para este tipo de redes es aún un problema abierto.

Uno de los principales problemas que presentan las redes neuronales ordinarias es el de generalizar el comportamiento de un sistema dinámico, requiriendo para cumplir con esta tarea, añadir capas ocultas a la red neuronal, o en su caso, añadir más neuronas a las capas que se utilizan, esto con el fin de tener una mayor profundidad de aprendizaje. En las redes neuronales complejas se tienen resultados que demuestran un mayor grado de profundidad en el aprendizaje utilizando menos neuronas que las que utilizaría una red neuronal ordinaria, esto debido a que al tener sus pesos sinápticos en el dominio complejo, existe un mayor intercambio de datos numéricos produciendo con esto un nivel extra de aprendizaje, esto es muy beneficioso si se consideran problemas donde se quiere aproximar una magnitud y una fase dentro de una misma red neuronal.

Sin embargo, como ya se mencionó anteriormente, los algoritmos de aprendizaje para el caso de las redes neuronales complejas son difíciles de obtener y además, éstos varían dependiendo la función de activación a considerar, es por esto que en este trabajo se propone una metodología para la obtención de un algoritmo basado en gradiente descendente utilizando reglas diagramáticas complejas, las cuales únicamente requieren manipulación de bloques y de flujo de señales.

Por último, se demuestra que este tipo particular de redes neuronales puede ser aplicado también a problemas de identificación y control de sistemas dinámicos y con esto exponer que este tipo de redes neuronales también pueden incursionar en esta área.

1.2 Planteamiento del problema

Los retos que presentan las redes neuronales complejas son de mayor complejidad a los que presentan las redes neuronales ordinarias, es decir, en el dominio real; esto es, la elección de la función de activación, la topología de la red, la deducción y programación del algoritmo de aprendizaje, entre otros. De todo lo anterior existe escasa literatura que trate estos problemas con fines de simplificar la obtención de dichos algoritmos.

Las redes neuronales complejas, se han utilizado principalmente en problemas que se pueden expresar en números complejos, sin embargo, en este trabajo se demostrará su funcionalidad en problemas de identificación y control de plantas mecánicas, esto con el fin de demostrar que este tipo de redes neuronales también se pueden utilizar en problemas que no se representan en números complejos.

Por otro lado, el estudio de las redes neuronales complejas ha sido restringido debido a la capacidad de computo que se tiene actualmente, al tener operaciones entre números complejos la capacidad de procesamiento computacional requerida es mayor, esta razón principalmente ha ocasionado que esta área haya quedado rezagada con respecto a las redes neuronales en el dominio real. Esta limitante algún día quedará superada y debido a esto se considera importante realizar mayor estudio en las redes neuronales complejas, que ya han demostrado ser superiores a las redes neuronales ordinarias [17], [13], además de que se puede ver como un intento de generalizar las redes neuronales.

1.3 Organización de la tesis

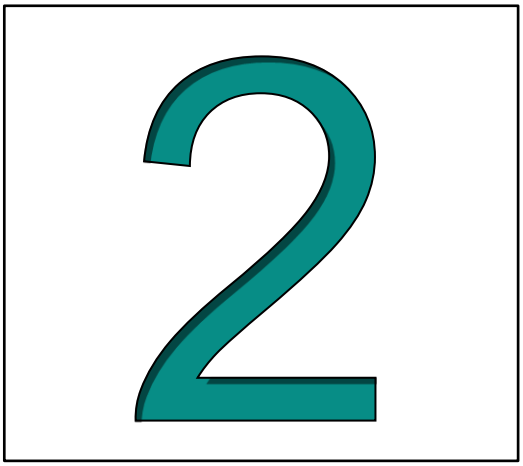
La tesis se dividió en cinco partes:

En el primer capítulo se menciona el estado del arte de las redes neuronales en el dominio real y de las redes neuronales complejas.

En el capítulo dos se abordan los conceptos básicos del análisis complejo que se requieren para entender el trabajo presentado, también se aborda el tema de la función de costo compleja y los problemas que se presentan al utilizarla en problemas de optimización.

En el capítulo tres se presentan los conceptos básicos de las redes neuronales en el dominio real, después se aborda la teoría de las redes neuronales complejas de manera que se puede hacer una comparación entre ambos tipos de redes neuronales.

Por último, en los capítulos cuatro y cinco se presentan los modelos de identificación y control directo adaptable utilizando redes neuronales complejas para plantas mecánicas de uno y dos grados de libertad, y los resultados obtenidos de estas simulaciones.



PRELIMINARES MATEMÁTICOS

2. Preliminares Matemáticos

En este capítulo se presentan los conceptos básicos del análisis complejo, una discusión acerca de la función de error en el dominio complejo y la introducción al cálculo de Wirtinger, conceptos que serán de utilidad en el desarrollo de este trabajo.

2.1 Conceptos de Análisis Complejo

Los números complejos pueden ser definidos como pares ordenados (x, y) de números reales que son interpretados como puntos en el plano complejo, con coordenadas rectangulares x e y, los puntos en x son considerados números reales y los puntos y son números imaginarios. De esto se sabe que si se consideran únicamente los pares $(x, 0)$, los números reales son un subconjunto de los números complejos [18].

Definición. El conjunto de los números complejos $\mathbb{C}$ es definido como:

$$\mathbb{C} = \{z = x + iy \mid x, y \in \mathbb{R}, i^2 = -1\} \tag{2.1}$$

En el análisis complejo es común representar un número complejo (x, y) con la letra z, por lo que un número complejo se puede representar por un par ordenado de la forma $z = (x, y)$, siendo $x = \mathbf{Re}(z)$ la parte real de z e $y = \mathbf{Im}(z)$ la parte imaginaria.

Dos números complejos z_1, z_2 son iguales siempre y cuando sus partes reales e imaginarias son iguales, si se cumple lo anterior lo siguiente es válido $z_1 = z_2$, y corresponden al mismo punto en el plano complejo; las operaciones básicas entre números complejos son definidas como sigue:

Sean dos números complejos $z_1 = (x_1, y_1)$, $z_2 = (x_2, y_2)$ se tiene lo siguiente:

$$(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2) \tag{2.2}$$

$$(x_1, y_1)(x_2, y_2) = (x_1x_2 - y_1y_2, y_1x_2 + x_1y_2) \tag{2.3}$$

Si se denota al número imaginario puro (0,1) con la letra i se tiene lo siguiente:

$$z = x + iy \tag{2.4}$$

Se define i^2 como:

$$i^2 = (0,1)(0,1) = -1$$
$$i^2 = -1 \tag{2.5}$$

De la definición anterior se tiene que las operaciones básicas antes mencionadas (2.2), (2.3) se pueden reescribir como:

$$(x_1 + iy_1) + (x_2 + iy_2) = (x_1 + x_2) + i(y_1 + y_2) \tag{2.6}$$
$$(x_1 + iy_1)(x_2 + iy_2) = (x_1x_2 - y_1y_2) + i(y_1x_2 + x_1y_2) \tag{2.7}$$

Se define el producto por cero:

$$z \cdot 0 = (x + iy) \cdot (0 + i0) = 0 + i0 = 0, \quad \forall\, z \in \mathbb{C}$$

Como caso particular se tiene el cuadrado de número complejo:

$$\begin{aligned} z^2 &= (x + iy) \cdot (x + iy) \\ &= x^2 + i2xy + (iy)^2 \\ &= x^2 - y^2 + 2xyi, \quad \forall\, z \in C \end{aligned}$$

Un número complejo se puede escribir utilizando la notación de Euler como sigue:

$$z = x + iy = re^{i\phi} \tag{2.8}$$

donde r es la magnitud, ϕ es el ángulo y $x, y \in \mathbb{R}$.

El conjugado de un número complejo $z = (x, y) = x + iy$ se define como:

$$\bar{z} = (x, -y) = x - iy \tag{2.9}$$

Sean dos números complejos $z_1 = (x_1, y_1)$ y $z_2 = (x_2, y_2)$ se tiene lo siguiente:

$$\overline{z_1 + z_2} = \bar{z_1} + \bar{z_2} = x_1 + x_2 - i(y_1 + y_2) = (x_1 - iy_1) + (x_2 - iy_2) \tag{2.10}$$

Cumpliendo con las siguientes operaciones:

$$z + \bar{z} = 2x = 2\Re(z) \tag{2.11}$$

$$z - \bar{z} = 2iy = 2\Im(z) \tag{2.12}$$

$$z\bar{z} = x^2 + y^2 \tag{2.13}$$

La división de dos números complejos es definida en términos del inverso multiplicativo:

$$\begin{aligned}\frac{z_1}{z_2} &= z_1 z_2^{-1} \\ &= (x_1, y_1)\left(\frac{x_2}{x_2^2 + y_2^2}, -\frac{y_2}{x_2^2 + y_2^2}\right) \\ &= \left(\frac{x_1x_2 + y_1y_2}{x_2^2 + y_2^2}, -\frac{y_1x_2 - x_1y_2}{x_2^2 + y_2^2}\right), \quad \forall z_1, z_2 \neq 0\end{aligned}$$

Definición. Una función compleja es un mapeo $f: \mathbb{C} \to \mathbb{C}$, tal que para $z = (x + iy) \in \mathbb{C}$, el resultado de la aplicación de f tiene la forma $f(x, y) = u(x, y) + iv(x, y)$ donde u y v son funciones de $\mathbb{R}^2$ en $\mathbb{R}$.

Definición. Una función $f: \mathbb{C} \to \mathbb{C}$, definida en una región (un subconjunto abierto de $\mathbb{C}$) se llama *holomorfa* si es derivable en toda ésta región.

Funciones que cumplen con esta última propiedad se llaman funciones analíticas. Éste es uno de los hechos que distinguen las funciones complejas de las funciones reales: para funciones complejas la propiedad de ser holomorfa coincide con la propiedad de ser analítica.

Definición. En análisis complejo, una función entera es una función en el dominio complejo que es *holomorfa* en todo el plano complejo.

Definición. Sea $f: \mathbb{C} \to \mathbb{C}$, la función compleja f se llama continua en $z \in \mathbb{C}$, si para todo $\epsilon > 0$, con $\epsilon \in \mathbb{R}$, existe $\delta > 0$, $\delta \in \mathbb{R}$, tal que

$$|z - z'| < \delta \text{ implica que } |f(z) - f(z')| < \epsilon$$

Para $A \subset \mathbb{C}$, se llama continua en A si f es continua en cada punto de A.

La propiedad de diferenciabilidad para el caso complejo, tiene restricciones más fuertes que en el caso real. Las funciones holomorfas son infinitamente diferenciables, esto no es cierto para funciones reales diferenciables.

Definición. Sea z un punto de cierto entorno de un punto fijo z_0, perteneciendo el entorno al dominio de definición de una función f. Se tiene $\Delta z = z - z_0$ y considérese a Δz como la variable compleja. La derivada f', ó df/dz, de f en z_0, se define por la ecuación

$$f'(z_0) = \lim_{\Delta z \to 0} \frac{f(z_0 + \Delta z) - f(z_0)}{\Delta z} \tag{2.14}$$

si el límite existe, entonces la función f es necesaramiente continua (cumpliendo con la definición de continuidad) en cualquier punto de z_0 en el cual su derivada exista. La función f se llama (complejo) derivable en z si este límite existe. La continuidad de una función no implica derivabilidad.

Las reglas de derivación para el caso complejo son iguales que el dominio real, siempre y cuando la derivada exista, por lo que se tiene que saber si una función es derivable antes de

aplicar fórmulas de derivación, una forma de saber esto es utilizando las condiciones de Cauchy-Riemann.

Condiciones de Cauchy-Riemann. Si la derivada $f'(z)$ de una funcion $f = u + iv$ existe en un cierto punto z, las derivadas parciales de primer orden, respecto a x y y, de cada uno de los componentes u y v han de existir en aquel punto y satisfacer las condiciones de Cauchy-Riemann, expresadas por

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \quad \text{y} \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{2.15}$$

Además, $f'(z)$ puede expresarse en función de estas derivadas parciales de la siguiente manera:

$$f'(z) = \frac{\partial u}{\partial x} + i\frac{\partial v}{\partial x} = \frac{\partial v}{\partial y} - i\frac{\partial v}{\partial y} \tag{2.16}$$

2.2 Funciones Analíticas

En esta sección se mencionan algunas definiciones necesarias para comprender la naturaleza de la función de activación en una red neuronal compleja. En la teoría de las redes neuronales, la función de activación desempeña un papel importante para el desarrollo del algoritmo de aprendizaje y para asegurar la estabilidad de dicho algoritmo y de toda la red. Las condiciones más comunes que se toman en cuenta para la elección de una función de activación en las redes neuronales en el dominio real, es que sea derivable y acotada, sin embargo, en el dominio complejo se requiere tomar en cuenta condiciones más fuertes para elegir una función de activación adecuada.

Definición. Una función $f(z)$ es analítica en un punto z_0, si su derivada $f'(z)$ existe no solamente en z_0 sino en cada punto z de cierto entorno de z_0, y es analítica en un dominio del plano complejo si es analítica en todo punto de aquel dominio.

Definición. Una función $f(z)$ entera es aquella que es analítica en cada punto del plano complejo, esto es, en la totalidad del plano.

Proposición. Las funciones enteras no son adecuadas para ser utilizadas como funciones de activación.

Una de las condiciones más fuertes para las funciones de variable compleja son las condiciones de Cauchy-Riemann:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y};\ \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{2.17}$$

si estas ecuaciones se cumplen se dice que la función compleja es analítica y $u(x, y)$ y $v(x, y)$ son harmónicas, ya que si (2.16) se cumple también se cumplirá (2.18).

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 v}{\partial x \partial y};\ \frac{\partial^2 u}{\partial y^2} = -\frac{\partial^2 v}{\partial y \partial x} \tag{2.18}$$

De lo anterior, se concluye que las segundas derivadas existen y que cuando una función compleja es analítica, las derivadas parciales de u y de v de todos los órdenes existen y son funciones continuas de x e y; juntando todo, se deduce que las dos derivadas en (2.18) son iguales y por tanto que en todo el dominio se cumple que:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{2.19}$$

La ecuación (2.19), es la ecuación diferencial de Laplace. Toda función que tiene derivadas parciales continúas de segundo orden y que satisfacen la ecuación de Laplace se llama función armónica, de esta ecuación se puede ver que ambas funciones reales son armónicas, algunos ejemplos de funciones analíticas son $\sin(z)$, $\cos(z)$, etc.

Más adelante se abordará el tema de las funciones de activación a utilizar para la red neuronal compleja, donde se volverá a discutir el tema de funciones analíticas.

2.3 Función Error

La discusión acerca de la función error es un aspecto importante en la deducción de algoritmos de aprendizaje de las redes neuronales; en este apartado se aborda la naturaleza de esta función en el dominio complejo; ésta debe de ser tratada con herramientas matemáticas diferentes a las que normalmente se utilizan en el cálculo de variables reales.

En el dominio complejo, la función error se define por ([19]):

$$\zeta(k) = \frac{1}{2}\sum_{j}\left[E_j(k)\right]\left[\overline{E_{j(k)}}\right], \quad j \in \mathrm{C}, \quad \zeta = \frac{1}{N_e}\sum_{j}\zeta(k) \tag{2.20}$$

donde $E_j(k) = y_j(w,x) - \hat{y}_j$ es el error entre la salida real de la red neuronal compleja y la salida de la planta; ζ es una función de la forma $f: \mathbb{C} \to \mathbb{R}$ y no cumple con las condiciones de Cauchy-Riemann, como se demostrará más adelante. En el caso de que la minimización de la función de error sea en línea se utilizará $\zeta(k)$, es decir, por cada paso se minimiza su función y se actualizan los pesos sinápticos de la red; en caso de que sea por lotes, se utilizará ζ, es decir, la función a minimizar es la suma de los errores de un lote completo.

La función de error (2.20), es una función no analítica, es decir, su derivada no está definida. Para que una función sea analítica debe de cumplir con las condiciones de Cauchy-Riemann (2.15), a continuación se utilizarán estas condiciones para demostrar que (2.20) no es una función analítica.

Se calcula la derivada del término $\overline{(y_t(w,x) - \hat{y}_t)} = \bar{z}$ de (2.20), que corresponde al complejo conjugado, utilizando la definición de límite expresada por (2.14) se demostrará que la derivada no está definida en todo el plano complejo.

Se define la función (2.21), que corresponde al complejo conjugado,

$$f(z) = \bar{z} \tag{2.21}$$

Aplicando la definición de límite (2.14) se tiene,

$$f'(z) = \lim_{\Delta z \to 0} \frac{f(z+\Delta z) - f(z)}{\Delta z} = \lim_{\Delta z \to 0} \frac{\overline{(z+\Delta z)} - \bar{z}}{\Delta z} = \lim_{\Delta z \to 0} \frac{\overline{\Delta z}}{\Delta z}$$

En este caso, Δz es un número complejo, se consideran dos casos particulares; si Δz es puramente real, esto es $\Delta z = x + i \cdot 0$, entonces si $\Delta z \to 0$ implica que $x \to 0$, y el límite converge a 1.

$$f'(z) = \lim_{\Delta z \to 0} \frac{\overline{\Delta z}}{\Delta z} = \lim_{\Delta x \to 0} \frac{x}{x} = \lim_{\Delta x \to 0} 1 = 1$$

El segundo caso corresponde que Δz es puramente imaginario, esto es $\Delta z = 0 + iy$, entonces si $\Delta z \to 0$ implica que $y \to 0$, y el límite converge a -1.

$$f'(z) = \lim_{\Delta z \to 0} \frac{\overline{\Delta z}}{\Delta z} = \lim_{\Delta y \to 0} \frac{-iy}{iy} = \lim_{\Delta y \to 0} -1 = -1$$

Con esto se demuestra que el límite no existe ya que $\Delta z \to 0$ no tiene el mismo comportamiento para los dos casos antes mencionados, por lo que se concluye que la función (2.21) no es derivable para cualquier punto $z \in \mathbb{C}$.

Utilizando las condiciones de Cauchy-Riemann (2.15), se llega a la misma conclusión de una manera más rápida, definiendo $u(x, y) = x, v(x, y) = -y$ se calculan las derivadas parciales:

$$\frac{\partial u}{\partial x} = 1, \qquad \frac{\partial v}{\partial x} = -1, \qquad \frac{\partial u}{\partial y} = \frac{\partial v}{\partial x} = 0$$

De lo anterior se concluye que la función (2.21) no es analítica y su derivada no está definida en cualquier punto $z \in \mathbb{C}$; en el siguiente apartado se introducirá una herramienta matemática que será de ayuda para encontrar las derivadas de funciones que no son analíticas.

2.4 Cálculo de Wirtinger

La optimización de parámetros de un sistema es un problema común en ingeniería. Sin embargo, muchos de estos problemas son tratados en el dominio complejo debido a su naturaleza, por ejemplo, problemas de comunicaciones; para resolver estos problemas de optimización se

requiere la derivación de funciones complejas, en los cuales el cálculo de Wirtinger es utilizado [20].

Retomando la definición de una función compleja se tiene,

$$w = f(z) = u(x,y) + iv(x,y), \quad u(x,y), v(x,y) \in \mathbb{R}, \qquad z = x + iy \in \mathbb{C} \tag{2.22}$$

Si se cumple que $v(x,y) = 0$, generalmente la función es no holomorfa. En este caso la función (2.22), queda como $f(z) = u(x,y)$ y puede ser vista como una función de dos variables reales. Por lo que la optimización puede ser resuelta para funciones multidimensionales reales.

El objetivo del problema de optimización tomando en cuenta lo antes mencionado, queda definido por lo siguiente:

$$f(z) \rightarrow \text{opt.} \quad \equiv \quad u(x,y) \rightarrow \text{opt.} \tag{2.23}$$

Lo que requiere las siguientes condiciones:

$$\frac{\partial u(x,y)}{\partial x} = 0, \qquad \frac{\partial u(x,y)}{\partial y} = 0 \tag{2.24}$$

Las ecuaciones (2.23) y (2.24), se pueden reescribir en una sola ecuación compleja,

$$a_1 \frac{\partial u(x,y)}{\partial x} + i\, a_2 \frac{\partial u(x,y)}{\partial y} = 0 + i0 = 0 \tag{2.25}$$

donde a_1 y a_2 son constantes reales diferentes de cero.

Escribiendo la parte real y la parte imaginaria de $z = x + iy$, como el par ordenado (x,y), se define el siguiente operador:

$$\frac{\partial}{\partial z} = a_1 \frac{\partial}{\partial x} + i\, a_2 \frac{\partial}{\partial y} \tag{2.26}$$

donde $a_1 = a_2 = 1/2$.

Definición. Las derivadas parciales de una función compleja $f(z)$ con variable compleja $z = x + iy \in \mathbb{C}, x, y \in \mathbb{R}$ con respecto a z y $\bar{z}$ se definen como sigue:

$$\frac{\partial f(z)}{\partial z} \triangleq \frac{1}{2}\left(\frac{\partial f}{\partial x} - i\frac{\partial f}{\partial y}\right) \tag{2.27}$$

$$\frac{\partial f(z)}{\partial \bar{z}} \triangleq \frac{1}{2}\left(\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y}\right) \tag{2.28}$$

Aplicando las ecuaciones (2.27) y (2.28) a la función (2.20) (siendo $(y_t(w,x) - \widehat{y_t}) = z$) se obtiene su derivada,

$$\frac{\partial z\bar{z}}{\partial z} = \frac{1}{2}\left(\frac{\partial(x^2+y^2)}{\partial x} - i\frac{\partial(x^2+y^2)}{\partial y}\right) = \frac{1}{2}(2x - i2y) = \bar{z} \tag{2.29}$$

$$\frac{\partial z\bar{z}}{\partial \bar{z}} = \frac{1}{2}\left(\frac{\partial(x^2+y^2)}{\partial x} + i\frac{\partial(x^2+y^2)}{\partial y}\right) = \frac{1}{2}(2x + i2y) = z \tag{2.30}$$

En la Tabla 2.1, se muestran algunas derivadas de funciones importantes utilizando el cálculo de Wirtinger.

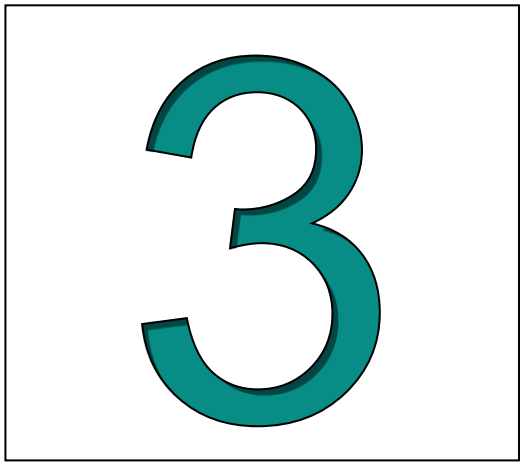
Tabla 2.1 Derivadas de Wirtinger de algunas funciones importantes.

$f(z)$	$\frac{\partial f(z)}{\partial z}$	$\frac{\partial f(z)}{\partial \bar{z}}$
cz	c	0
$c\bar{z}$	0	c
$z\bar{z}$	$\bar{z}$	z

De la misma manera se puede demostrar que las reglas de derivación de suma, producto y cociente se mantienen [20]; se puede decir que el cálculo de Wirtinger se encuentra entre la derivada real de una función real y la derivada compleja de una función compleja. Para el caso de que la función $f(z)$ es holomorfa, las condiciones de Cauchy-Riemann se mantienen. Para funciones holomorfas la derivada de Wirtinger coincide con la derivada normal de una función compleja. El cálculo de Wirtinger funciona para cualquier función compleja sea holomorfa o no lo sea, así como también para funciones reales.

2.5 Conclusiones

Los conceptos que se presentaron en este capítulo resultan fundamentales para entender el problema que existe al tratar las redes neuronales en el dominio complejo. Uno de los principales problemas es la elección de la función de activación que debe de cumplir con dos condiciones que en el análisis complejo son complicados de conseguir, éstos son, la función debe de ser acotada y a su vez analítica; por otro lado, la función error que será la función a minimizar en el proceso de optimización debe de ser una función analítica, que en el sentido del cálculo ordinario no cumple, es por eso la introducción del cálculo de Wirtinger el cual permite obtener sus gradientes evitando el problema de la no analiticidad de dicha función.



REDES NEURONALES

3. Redes Neuronales

En este capítulo se presenta una breve historia de la evolución, los fundamentos y los conceptos básicos de las redes neuronales en el dominio real y su comparación con las redes neuronales en el dominio complejo, se presentan topologías, algoritmos de aprendizaje, y el método diagramático desarrollado para la obtención de algoritmos de aprendizaje para redes neuronales en el dominio complejo.

3.1 Breve Historia de las Redes Neuronales Artificiales

En 1936, Alan Turing fue el primero en estudiar el cerebro como una forma que podría ser utilizada para el computo de datos, sin embargo fue hasta 1943 cuando los investigadores Warren McCulloch y Walter Pitts propusieron el primer modelo simple de la neurona [21].

Décadas posteriores, en 1957 el psicólogo Frank Rosenblatt construyó una máquina neuronal simple a la que llamó perceptrón [22]. El perceptrón es la más antigua máquina neuronal y era capaz de generalizar, es decir, después de haber aprendido una serie de patrones era capaz de reconocer otros similares aunque no se le hubieran presentado anteriormente, ésta máquina fue capaz de reconocer todas las letras del alfabeto [23].

En 1959, las redes neuronales fueron estudiadas por B. Widrow y M. E. Hoff, quienes desarrollaron una máquina llamada Adaline *(Adaptive Linear Element*), cabe mencionar que esta fue la primera red neuronal aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas) y se ha usado comercialmente durante varias décadas.

Para finales de los años setenta, Minsky y Papert demostraron que los perceptrones de Rosenblatt eran incapaces de realizar tareas simples como sintetizar la función lógica XOR. Por lo que el perceptrón demostraba ser un camino sin salida. Otros investigadores siguieron investigando las redes neuronales en los años siguientes; Grossberg junto con Carpenter propusieron un modelo de red neuronal llamado ART (*Adaptive Resonance Theory*).

Fue en la década de los ochenta cuando Mc Clelland y Rumelhart mostraron las ventajas y desventajas de las redes neuronales artificiales en un libro titulado: *Parallel Distributed*

Processing: Exploration in the Microstructure of Cognition, que junto con el trabajo de Hopfield en el que describe con claridad y rigor matemático una red la cual lleva su nombre, le dieron un impulso a la investigación de los sistemas neuronales.

3.2 Redes Neuronales Artificiales

En el campo de las redes neuronales artificiales, es común llamarlas únicamente "redes neuronales", por lo que se utilizará este nombre en todo el trabajo. Las redes neuronales tratan de reproducir el proceso de solución de problemas del cerebro. Así como los humanos aplican conocimiento adquirido de la experiencia y lo almacenan mediante un método de aprendizaje implícito en el cerebro con el fin de utilizarlo para tomar decisiones, una red neuronal artificial toma ejemplos de problemas resueltos para construir un sistema de toma de decisiones, esto las hace ideales para resolver problemas en los cuales los algoritmos computacionales resultan muy complejos de diseñar, como puede ser, el reconocimiento de imágenes [24].

Una red neuronal es un procesador masivo paralelo distribuido formado por unidades de procesamiento independientes, que tienen por naturaleza la capacidad de almacenar conocimiento experimental y utilizarlo posteriormente [25]; este tipo de redes se asemejan en dos cosas al cerebro:

1. El conocimiento del entorno de la red es adquirido por ésta mediante un proceso de aprendizaje.

2. Las conexiones entre neuronas llamadas pesos sinápticos son utilizadas para almacenar el conocimiento adquirido.

El procedimiento para llevar a cabo el proceso de aprendizaje se llama algoritmo de aprendizaje, que tiene como función modificar el valor de los pesos sinápticos de la red con el objetivo de conseguir un objetivo deseado, las ventajas principales de las redes neuronales son el poder de procesamiento alto debido a su estructura masiva paralela distribuida y su habilidad para aprender y después generalizar, es decir, producir salidas aproximadas a entradas que no estuvieron incluidas durante el proceso de entrenamiento (aprendizaje).

Algunas características y capacidades de las redes neuronales son las siguientes [25]:

No-linealidad. Una neurona puede ser lineal o no lineal. Una red neuronal construida con neuronas no lineales es una red neuronal no lineal.

Mapeo entrada-salida. Cada ejemplo de entrenamiento consiste en una única entrada que corresponde a una salida deseada, de esta manera la red es capaz de realizar un mapeo entrada-salida después del entrenamiento.

Adaptabilidad. Las redes neuronales tienen como característica intrínseca, la capacidad de adaptar sus pesos sinápticos a cambios en su entorno.

Tolerancia a fallas. La implementación de una red neuronal en hardware, es inherentemente tolerante a fallas, es decir, es robusta ante perturbaciones.

Como ya se mencionó antes, las redes neuronales son modelos inspirados de la biología, por lo que se abordará brevemente las redes neuronales biológicas para así tener un panorama de las funciones que intentan emular las redes neuronales artificiales.

3.3 Redes Neuronales Biológicas

Las redes neuronales artificiales tienen como motivación original, la estructura de sistemas biológicos, y constituyen un paradigma computacional. Millones de años de evolución han llevado a que el cerebro desarrolle soluciones muy sofisticadas en ambientes desconocidos [26].

En las redes neuronales biológicas, la información es almacenada en los puntos de contacto entre diferentes neuronas, llamadas sinapsis. Otras formas de almacenamiento conocida son las neuronas, ya que ellas mismas son sistemas complejos de auto-organización.

El sistema nervioso posee una arquitectura global de complejidad variable, constituida por bloques llamadas neuronas. Estas neuronas pueden realizar diferentes funciones y están conectadas en diferentes capas, cada capa con funciones específicas. Cada neurona recibe una señal y produce una respuesta.

En la Figura 3.1, se muestra el esquema general de una neurona biológica genérica. Las ramificaciones de la izquierda son los canales de trasmisión para información de entrada y son llamadas dendritas. Las dendritas reciben las señales de otras neuronas, estos puntos son las sinapsis. El núcleo, es la parte que suministra energía a la neurona, ya que genera las substancias necesarias para el funcionamiento de la neurona completa. Las señales de salida son trasmitidas a través del axón.

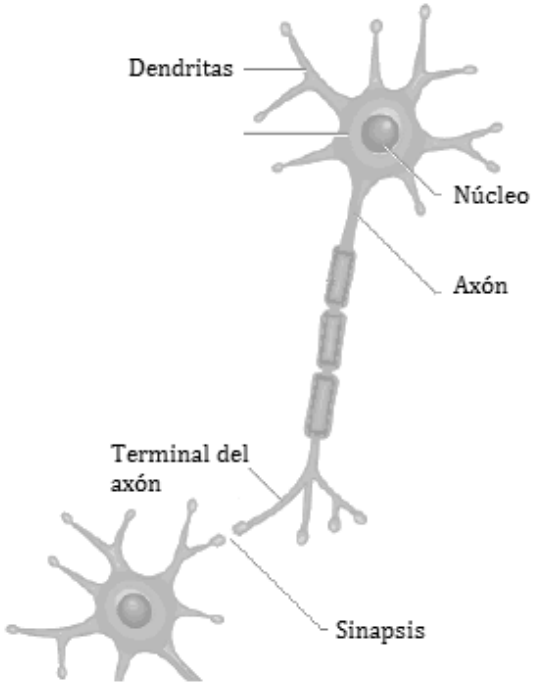


Figura 3.1 Esquema de una neurona biológica.

Estos elementos, las dendritas, la sinapsis, el núcleo y el axón son la estructura mínima que tiene el modelo biológico de una neurona.

Los biólogos han sabido desde hace más de cien años que la trasmisión de información entre neuronas es mediante señales eléctricas. La evolución ha llevado a cabo la tarea de dar solución a la trasmisión de información mediante señales químicas y eléctricas. Es decir, la combinación de señales eléctricas y químicas hace posible la comunicación entre neuronas.

La regulación entre las señales eléctricas y las señales químicas es llevada a cabo por la sinapsis. La sinapsis convierte una señal eléctrica presináptica en una señal química y después la

convierte en una señal eléctrica postsináptica, en general se sabe que la sinapsis provoca una excitación o inhibición de la señal eléctrica a trasmitir a través del axón a otras neuronas. Este juego interno entre las señales eléctricas y químicas en la neurona y su posterior conexión con otras neuronas es la base del procesamiento de la información; por otro lado, también se sabe que la información de la red neuronal es almacenada en la sinapsis; aunque existen otros tipos de almacenamiento de información que no se han llegado a conocer aún.

3.4 Redes Neuronales en el Dominio Real

Una neurona es una unidad de procesamiento de información que es fundamental para la operación de una red neuronal; en la Figura 3.2 se muestra el diagrama a bloques de una neurona, que es la base del diseño de una red neuronal.

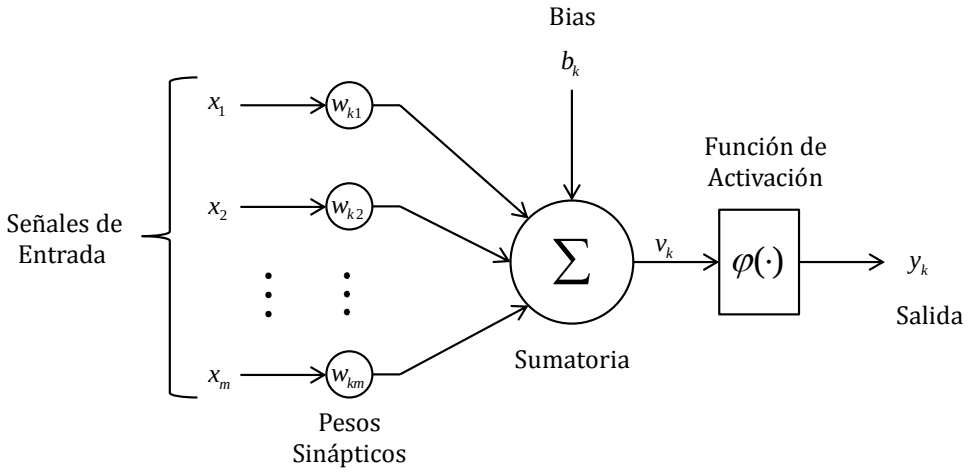


Figura 3.2 Diagrama de bloques del modelo no lineal de una neurona.

Se tienen tres elementos básicos de una neurona:

1. Un conjunto de "sinapsis" o enlaces de conexión asociados a un peso sináptico. Una señal de entrada x_j de la sinapsis j conectada a la neurona k es multiplicada por el peso sináptico w_{kj}.

2. Una sumatoria de las entradas multiplicadas por su pesos sináptico asociado, esto constituye una combinación lineal.

3. La función de activación que limita la salida de la neurona; su amplitud queda generalmente limitada en [0,1] o [−1,1].

En términos matemáticos, la neurona k se describe por las siguientes ecuaciones:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3.1}$$

$$v_k = u_k + b_k \tag{3.2}$$

$$y_k = \varphi(u_k + b_k) \tag{3.3}$$

donde x_m son las señales de entrada; w_{km} son los pesos sinápticos de la neurona k; u_k es una combinación lineal de las entradas a la neurona; b_k es el bias o umbral; $\varphi(\cdot)$ es la función de activación; e y_k es la salida de la neurona k.

3.4.1 Funciones de Activación

Las funciones de activación $\varphi(\cdot)$ definen la salida de la neurona, principalmente se tienen cuatro tipos de función de activación:

Función lineal. Este tipo de función de activación se define de la siguiente manera:

$$\varphi(v) = \begin{cases} 1 & si \quad -\frac{1}{2} > v \geq -\frac{1}{2} \\ v & si \quad -\frac{1}{2} > v > -\frac{1}{2} \\ 0 & si \quad -\frac{1}{2} > v \leq -\frac{1}{2} \end{cases} \tag{3.4}$$

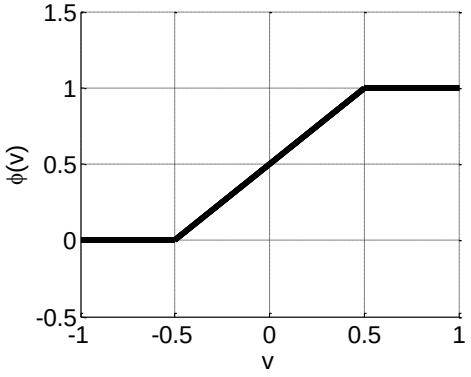


Figura 3.3 Función lineal.

Función escalón o función de Heaviside. Esta función se define de la siguiente manera:

$$\varphi(v) = \begin{cases} 1 & si \quad v \geq 0 \\ 0 & si \quad v < 0 \end{cases} \tag{3.5}$$

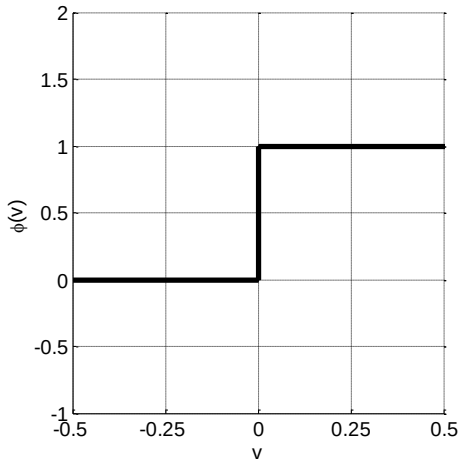


Figura 3.4 Función escalón o de Heaviside.

Función sigmoide. La función sigmoide, es por mucho la función de activación más utilizada en las redes neuronales, es estrictamente creciente, contiene una parte lineal y una parte no lineal, se define de la siguiente manera:

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{3.6}$$

donde a es el parámetro de pendiente, variando este parámetro se obtienen funciones con diferentes pendientes cada una; a diferencia de la función de Heaviside, la función sigmoide es diferenciable.

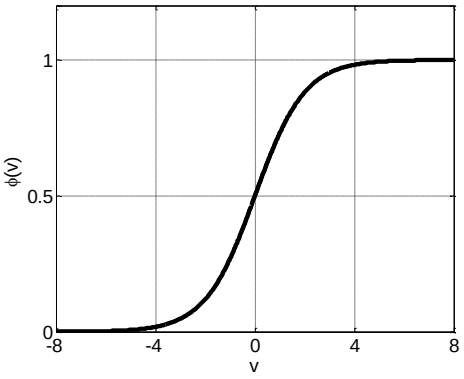


Figura 3.5 Función sigmoide.

Función tangente hiperbólica. Otra función parecida la sigmoide es la función tangente hiperbólica, en su forma general se define:

$$\varphi(v) = a\tanh(bv), \quad (a,b) > 0 \tag{3.7}$$

donde a y b son constantes, la función tangente hiperbólica es al igual que la función sigmoide, diferenciable.

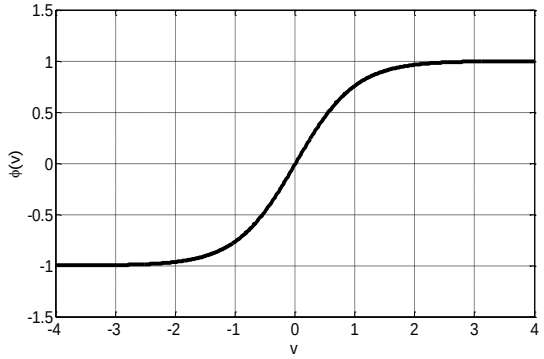


Figura 3.6 Función tangente hiperbólica.

3.4.2 Topologías en Redes Neuronales Reales

La topología de una red neuronal se refiere a la forma, estructura o patrón de conexión de las neuronas que la conforman. En una red neuronal los nodos se conectan por medio de sinapsis, esta estructura de conexiones sinápticas determina el comportamiento de la red. Las conexiones sinápticas son unidireccionales, es decir, la información solamente puede propagarse en un único sentido. En general, las neuronas se agrupan en unidades estructurales que se denominan capas; el conjunto de una o más capas constituye la *red neuronal*.
Se distinguen tres tipos de capas: de entrada, de salida y ocultas.

Capa de entrada. Una capa de entrada o sensorial está compuesta por neuronas que reciben datos o señales procedentes del entorno (por ejemplo, proporcionados por sensores).

Capa de salida. Es aquella cuyas neuronas proporcionan la respuesta de la red neuronal (sus neuronas pueden estar conectadas a efectores).

Capa oculta. Es aquella que no tiene una conexión directa con el entorno, es decir, que no se conecta directamente ni a órganos sensores ni a efectores. Este tipo de capa proporciona a la red neuronal grados de libertad adicionales, gracias a los cuales puede encontrar representaciones internas correspondientes a determinados rasgos del entorno, proporcionando una mayor riqueza computacional.

Se pueden definir principalmente tres tipos de topologías en las redes neuronales [25]:

Redes con conexión hacia adelante de una sola capa. Es la forma más simple de una red neuronal, se tiene una capa de entrada que conecta las entradas al sistema y se reflejan en la capa de salida. El flujo de señales siempre va de la capa de entrada a la de salida, es decir, hacia adelante. En la Figura 3.7, se tiene una red neuronal hacia adelante de una sola capa (la capa de entrada no se cuenta, ya que no se ha hecho ningún calculo en ese momento), se puede identificar la capa de entrada y la capa de salida, y las flechas denotan la dirección del flujo de señales.

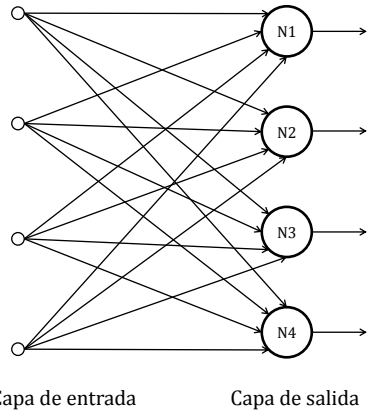


Figura 3.7 Topología de una red con conexión hacia adelante de una sola capa.

Redes con conexión hacia adelante multicapa. Esta topología se distingue por contener al menos una capa oculta, a las neuronas de esta capa se le llaman neuronas ocultas. La interacción de estas neuronas está ligada a las entradas externas de la red y la capa de salida, lo que hace más complejo el procesamiento de las señales y a la vez, puede almacenar información más profunda.

La red neuronal mostrada en la Figura 3.8, se dice totalmente conectada en el sentido de que cada neurona en cada capa está conectada a todos los nodos de la capa siguiente, si no es el caso se le llama parcialmente conectada.

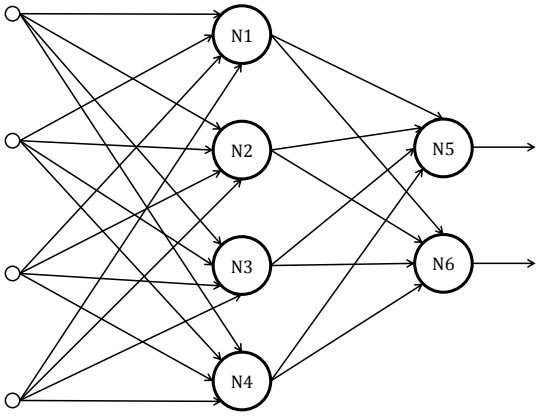


Figura 3.8 Topología de una red hacia adelante multicapa.

Redes recurrentes. Una red recurrente se distingue de una red hacia adelante, porque tiene al menos un lazo de realimentación. La existencia de lazos de realimentación tiene un profundo impacto en las capacidades de aprendizaje y desempeño de la red neuronal; estos lazos de realimentación se componen de retardos (denotados z^{-1}), que resultan en comportamiento de dinámicas no lineales, asumiendo que la red contiene neuronas no lineales.

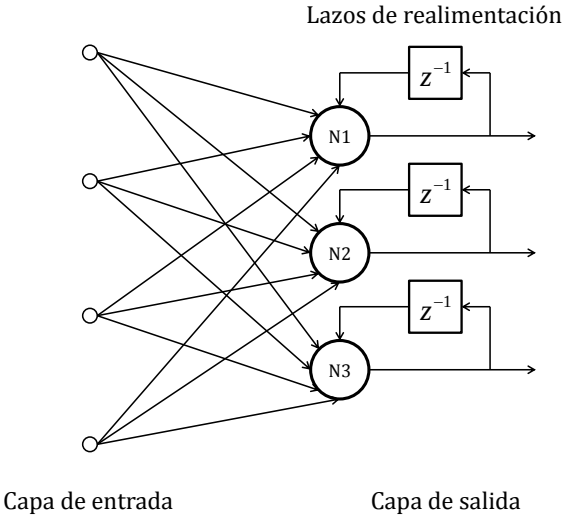


Figura 3.9 Topología de una red recurrente.

3.4.3 Algoritmos de Aprendizaje

Una de las propiedades fundamentales de las redes neuronales es la habilidad de aprendizaje a partir de información del entorno. Una red neuronal aprende a través de un proceso de ajuste

de sus pesos sinápticos. Se define el proceso de aprendizaje de las redes neuronales como sigue:

"Aprender es un proceso por el cual los parámetros libres (pesos simpáticos y umbrales) de una red neuronal son adaptados a través de un proceso de estimulación con su entorno en el cual la red neuronal está embebida. Este tipo de aprendizaje se determina por la manera en que los parámetros libres cambian."

De la definición anterior se pueden concluir tres cosas:

1. La red neuronal es estimulada por su entorno.

2. La red neuronal cambia sus parámetros libres como resultado de la estimulación.

3. La red neuronal responde de una manera diferente a su entorno por los cambios que han ocurrido en sus parámetros.

A un conjunto bien definido de reglas para la solución del problema de aprendizaje se le llama algoritmo de aprendizaje.

La mayoría de los métodos de entrenamiento utilizados en las redes neuronales con conexión hacia delante consisten en proponer una función de error que mida el rendimiento actual de la red en función de los pesos sinápticos. El objetivo del método de entrenamiento es encontrar el conjunto de pesos sinápticos que minimizan (o maximizan) la función. El método de optimización proporciona una regla de actualización de los pesos que en función de los patrones de entrada modifica iterativamente los pesos hasta alcanzar el punto óptimo de la red neuronal.

Aprendizaje por corrección del error.

Considérese una neurona k constituida por solo un nodo en la capa de salida de una red neuronal feedforward o realimentada descrita en la Figura 3.10. La señal de salida de la neurona k es denotada por $y_k(n)$. La señal de salida se compara con la señal deseada $d_k(n)$, conformando la señal de error $e_k(n)$.

$$e_k(n) = d_y(n) - y_k(n) \tag{3.8}$$

La señal $e_k(n)$ de error actúa como un mecanismo de control, cuyo propósito es aplicar los ajustes correspondientes a los pesos sinápticos de la neurona k. Estos ajustes se realizan con el objetivo de aproximar la señal de la salida a la deseada, minimizando el índice de desempeño:

$$\zeta(n) = \frac{1}{2} e_k^2(n) \tag{3.9}$$

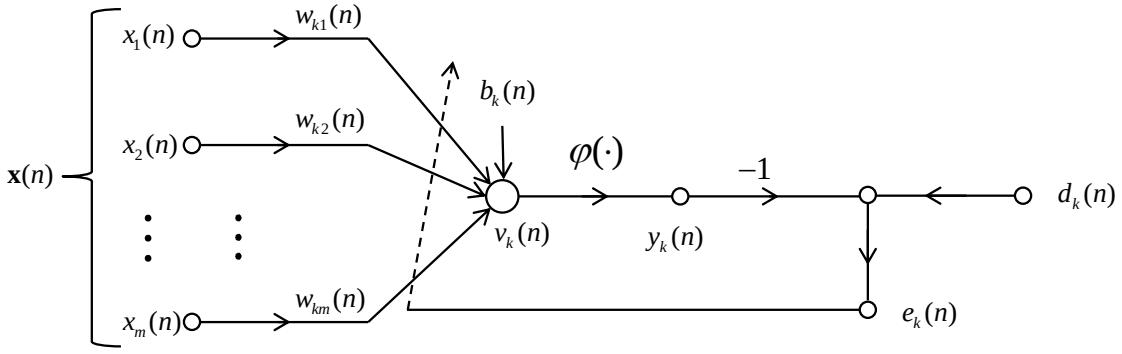


Figura 3.10 Gráfica de flujo de señal de la salida de una neurona.

Este tipo de aprendizaje utiliza la regla delta o de Widrow-Hoff. En la cual se aplica un ajuste $\Delta w_{kj}(n)$ al peso w_{kj} en el tiempo n definido por:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \tag{3.10}$$

donde η es la tasa de aprendizaje o parámetro de aprendizaje. Una vez obtenido el ajuste sináptico, entonces se actualizan los pesos de la siguiente manera:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \tag{3.11}$$

Métodos basados en optimización.

Considérese la función de costo $\zeta(\boldsymbol{w})$ que es una función continuamente diferenciable de algún vector de pesos desconocido $\boldsymbol{w}$. La función $\zeta(\boldsymbol{w})$ mapea los elementos de $\boldsymbol{w}$ en los números reales. El objetivo es encontrar un $\boldsymbol{w}^*$ como solución óptima que satisfaga la condición (3.12).

$$\zeta(\boldsymbol{w}^*) \leq \zeta(\boldsymbol{w}) \tag{3.12}$$

Es decir, se necesita resolver un problema de optimización sin restricciones, declarado como sigue:

"Minimizar la función de costo $\zeta(\boldsymbol{w})$ con respecto al vector de pesos $\boldsymbol{w}$."

La condición necesaria para la optimalidad es:

$$\nabla\zeta(\boldsymbol{w}*) = 0 \tag{3.13}$$

donde ∇ es el operador gradiente, así que $\nabla\zeta(\boldsymbol{w})$ es el gradiente de la función de costo:

$$\nabla\zeta(\boldsymbol{w}) = \left[\frac{\partial\zeta}{\partial w_1}, \frac{\partial\zeta}{\partial w_2}, \ldots, \frac{\partial\zeta}{\partial w_m}\right]^T \tag{3.14}$$

Una clase de algoritmos de optimización sin restricciones está basada en la idea del descenso iterativo local:

"Iniciando con la condición inicial denotada por $\boldsymbol{w}(0)$, generar una secuencia de vectores de pesos $\boldsymbol{w}(1), \boldsymbol{w}(2)$, ..., tal que la función de costo $\zeta(\boldsymbol{w})$ es reducida en cada iteración del algoritmo, como se muestra mediante,

$$\zeta[\boldsymbol{w}(n+1)] < \zeta[\boldsymbol{w}(n)] \tag{3.15}$$

donde $\boldsymbol{w}(n)$ es el valor del paso anterior del vector de pesos, y $\boldsymbol{w}(n+1)$ es el valor actualizado."

Se espera que el algoritmo converja eventualmente a la solución óptima $\boldsymbol{w}^*$.

Método del gradiente descendente.

El método de entrenamiento más utilizado es el método del gradiente descendente. Este método define una función $\zeta(\boldsymbol{w})$ que proporciona el error que comete la red neuronal en función del conjunto de pesos sinápticos $\boldsymbol{w}$. El objetivo del aprendizaje será encontrar la configuración de pesos que corresponda al mínimo global de la función de error; aunque en muchos casos es suficiente encontrar un mínimo local lo suficientemente bueno [27].

Este tipo de método funciona de la siguiente manera: se tiene un conjunto de pesos sinápticos iniciales $\boldsymbol{w}(0)$ en el tiempo cero, a continuación se calcula la dirección de máxima variación del error, que se define como el gradiente $\nabla\zeta(\boldsymbol{w})$ en $\boldsymbol{w}(0)$. Los pesos se actualizan siguiendo el sentido contrario indicado por el gradiente, es decir, la dirección de máximo descrecimiento, de esta manera, se va produciendo un descenso por la superficie del error hasta llegar a un mínimo, por lo general, local; la regla de aprendizaje queda expresada por:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \eta\nabla\zeta(\boldsymbol{w})$$
$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \eta\boldsymbol{g}(n) \tag{3.16}$$

donde η se le llama coeficiente de aprendizaje.

Para mostrar que la formulación del descenso más rápido satisface la condición (3.15) para el descenso iterativo, se utiliza la expansión en series de Taylor de primer orden alrededor de $\boldsymbol{w}(n)$ para aproximar $\zeta[\boldsymbol{w}(n+1)]$ como:

$$\zeta[\boldsymbol{w}(n+1)] \approx \zeta[\boldsymbol{w}(n)] + \boldsymbol{g}^T(n)\boldsymbol{\Delta w}(n) \tag{3.17}$$

donde $\boldsymbol{g}(n) = \nabla\zeta(\boldsymbol{w})$.

Si se define $\boldsymbol{\Delta w}(n) = \boldsymbol{w}(n+1) - \boldsymbol{w}(n)$, de la ecuación (3.16) se tiene que $\boldsymbol{\Delta w}(n) = -\eta\boldsymbol{g}(n)$, sustituyendo esta última definición en (3.17), se tiene:

$$\zeta[\boldsymbol{w}(n+1)] \approx \zeta[\boldsymbol{w}(n)] + \eta \boldsymbol{g}^T(n)\boldsymbol{g}(n)$$
$$\zeta[\boldsymbol{w}(n+1)] = \zeta[\boldsymbol{w}(n)] + \eta \left|\left|\boldsymbol{g}(n)\right|\right|^2 \quad (3.18)$$

De la ecuación (3.18), se muestra que, para un parámetro de velocidad de aprendizaje positivo η, la función de costo decrece cuando el algoritmo avanza de la iteración uno a la siguiente. El parámetro de velocidad de aprendizaje η tiene una profunda influencia en su comportamiento de convergencia, [25]:

- Cuando η es pequeña, la respuesta transitoria del algoritmo es *sobreamortiguada*, por lo que la trayectoria de $\boldsymbol{w}(n)$ sigue una trayectoria suave en el plano W.

- Cuando η es grande, la respuesta transitoria del algoritmo es *subamortiguada*, por lo que la trayectoria de $\boldsymbol{w}(n)$ sigue una trayectoria de zigzag (oscilatoria).

- Para obtener una buena precisión estadística, el parámetro η tiene que ser pequeño, del orden de 0.01, en ninguna aplicación este valor puede ser menor que cero. Por otro lado, cuando η sobrepasa cierto valor critico (determinado por el mismo algoritmo), el algoritmo puede volverse inestable, es decir, diverge.

Backpropagation.

El algoritmo Backpropagation es el método de entrenamiento más utilizado en redes con conexión hacia adelante. Es un método de aprendizaje supervisado basado en el gradiente descendente; su funcionamiento es relativamente sencillo, primero se le presenta a la red neuronal un patrón de entrada, este patrón se propaga por las distintas capas de la red neuronal hasta producir la salida de la misma; la salida de la red neuronal se compara con la salida deseada y se calcula el error en cada neurona de salida. Estos errores se propagan hacia atrás, desde la capa de salida hacia todas las neuronas de las capas intermedias. Cada neurona recibe un error que es proporcional a su contribución sobre el error total de la red. Basándose en el error recibido, se ajustan los valores de los pesos sinápticos de cada neurona.

Descripción del algoritmo de Backpropagation.

La señal del error en la salida de la neurona k en la iteración n está definida por:

$$e_k(n) = d_k(n) - y_k(n) \tag{3.19}$$

El valor instantáneo $\zeta(n)$ de la energía total del error se obtiene mediante:

$$\zeta(n) = \frac{1}{2}\sum_{k\in C} e_k^2\,(n) \tag{3.20}$$

donde C incluye a todas las neuronas de la capa de salida de la red. Sea N el número total de patrones que contiene el conjunto de entrenamiento. La energía del promedio del error cuadrático está dada por:

$$\zeta_{av}(n) = \frac{1}{N}\sum_{n=1}^{N} \zeta\,(n) \tag{3.21}$$

La energía del error instantáneo y la energía del promedio del error cuadrático están en función de todos los parámetros libres, es decir, de los pesos sinápticos y de los umbrales de la red. Para un conjunto de entrenamiento dado, ζ_{av} representa la función de costo como una medida del desempeño del aprendizaje.

El objetivo del proceso de aprendizaje es ajustar los parámetros libres de la red para minimizar ζ_{av}. Para lograr este objetivo, se considera un método simple de entrenamiento en el que los pesos se actualizan en una entrada por un vector base de entrada. Los ajustes son hechos de acuerdo con los respectivos errores calculados para cada vector de entrada presentado para la red neuronal. El promedio aritmético de estos cambios de peso individuales sobre el conjunto de entrenamiento es por consiguiente un estimado del verdadero cambio que habría resultado de modificar los pesos basados en minimizar la función del costo sobre el conjunto de entrenamiento entero.

Retomando la descripción matemática de una neurona, se tienen las siguientes ecuaciones:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$v_k = u_k + b_k$$

$$y_k = \varphi(u_k + b_k)$$

El algoritmo de propagación hacia atrás aplica una corrección $\Delta w_{kj}(n)$ al peso sináptico $w_{kj}(n)$, al igual que en la regla de Widrow-Hoff, el incremento a realizar a los pesos es proporcional al gradiente descendente:

$$\Delta w_{kj}(n) = \frac{\partial \zeta(n)}{\partial w_{kj}(n)} \tag{3.22}$$

De acuerdo a la regla de la cadena de cálculo, podemos expresar este gradiente como:

$$\frac{\partial \zeta(n)}{\partial w_{kj}(n)} = \frac{\partial \zeta(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial w_{kj}(n)} \tag{3.23}$$

La derivada $\partial \zeta(n) / \partial w_{kj}(n)$ representa un factor de sensibilidad; que determina la dirección de búsqueda en el espacio de pesos para los pesos sinápticos w_{kj}.

Diferenciando ambos lados de la ecuación (3.20) con respecto a $e_k(n)$, se obtiene:

$$\frac{\partial \zeta(n)}{\partial e_k(n)} = e_k(n) \tag{3.24}$$

Diferenciando ambos lados de la ecuación (3.19) con respecto a $y_k(n)$, se obtiene:

$$\frac{\partial e_k(n)}{\partial y_k(n)} = -1 \tag{3.25}$$

A continuación, diferenciando la ecuación (3.3) con respecto a $v_k(n)$, se obtiene:

$$\frac{\partial y_k(n)}{\partial v_k(n)} = \varphi_k{}'(v_k(n)) \tag{3.26}$$

Finalmente, diferenciando la ecuación (3.2) con respecto a $w_{kj}(n)$, se obtiene:

$$\frac{\partial v_k(n)}{\partial w_{kj}(n)} = x_j(n) \tag{3.27}$$

Por lo que la expresión (3.23), se reescribe de la siguiente manera:

$$\frac{\partial \zeta(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi_k'\big(v_k(n)\big)x_j(n) \tag{3.28}$$

La corrección $\Delta w_{kj}(n)$ aplicada a $w_{kj}(n)$ está definida por la regla delta:

$$\Delta w_{kj}(n) = -\eta \frac{\partial \zeta(n)}{\partial w_{kj}(n)} \tag{3.29}$$

donde η es el parámetro de velocidad de aprendizaje del algoritmo de propagación hacia atrás.

De acuerdo con las ecuaciones (3.28) y (3.29), se obtiene:

$$\Delta w_{kj}(n) = -\eta \delta_k(n) x_j(n) \tag{3.30}$$

donde el gradiente local $\delta_k(n)$ está definido por:

$$\delta_k(n) = -\frac{\partial \zeta(n)}{\partial v_k(n)} = -\frac{\partial \zeta(n)}{\partial e_k(n)}\frac{\partial e_k(n)}{\partial y_k(n)}\frac{\partial y_k(n)}{\partial v_k(n)} = e_k(n)\varphi_k{}'(v_k(n)) \tag{3.31}$$

Cuando la neurona k está localizada en la capa de salida de la red, la actualización de $w_{kj}(n)$ se hace utilizando las ecuaciones (3.30) y (3.31).

Cuando la neurona k se encuentra en una capa oculta, se redefine el gradiente local $\delta_j(n)$ para la neurona j de la capa oculta como:

$$\delta_j(n) = -\frac{\partial \zeta(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \zeta(n)}{\partial y_j(n)} \varphi_j{}'(v_j(n)) \tag{3.32}$$

Retomando la ecuación (3.20), y diferenciándola con respecto a $y_k(n)$ se obtiene:

$$\frac{\partial \zeta(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} \tag{3.33}$$

Haciendo uso de la regla de la cadena para la derivada parcial $\partial e_k(n)/\partial y_j(n)$, la ecuación (3.33) se reescribe de forma equivalente:

$$\frac{\partial \zeta(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{3.34}$$

Retomando la ecuación (3.19), y diferenciándola con respecto a $v_k(n)$ se tiene:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k{}'(v_k(n)) \tag{3.35}$$

De la ecuación (3.2) y diferenciándola con respecto a $y_j(n)$ se tiene:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{3.36}$$

Utilizando las ecuaciones (3.35) y (3.36) en (3.34) se obtiene la derivada parcial deseada:

$$\frac{\partial \zeta(n)}{\partial y_j(n)} = -\sum_k e_k(n)\, \varphi_k'\big(v_k(n)\big) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n) \tag{3.37}$$

donde $\delta_k(n)$ es el gradiente local usando la definición dada en la ecuación (3.31).

Finalmente, utilizando la ecuación en (3.37), se llega a la *fórmula de la propagación hacia atrás* para el gradiente local $\delta_j(n)$ descrito como:

$$\delta_j(n) = \varphi_j'\left(v_j(n)\right)\sum_k \delta_k(n)w_{kj}(n) \tag{3.38}$$

cuando la neurona j está oculta.

Finalmente la corrección $\Delta w_{ji}(n)$ del peso sináptico $w_{ji}(n)$ que conecta a la neurona i con la neurona j está definida por la regla delta:

$$\begin{pmatrix}\text{Corrección del peso}\\ \Delta w_{ji}(n)\end{pmatrix} = \begin{pmatrix}\text{Parametro de velocidad}\\ \text{de aprendizaje}\\ \eta\end{pmatrix}\cdot\begin{pmatrix}\text{Gradiente}\\ \text{local}\\ \delta_j(n)\end{pmatrix}\cdot\begin{pmatrix}\text{Entrada de la}\\ \text{neurona } j\\ x_i(n)\end{pmatrix} \tag{3.39}$$

donde el cálculo del gradiente local depende de si es de una neurona de la capa de salida (k) o de una neurona de alguna capa oculta (j).

Si es el gradiente local de una neurona de la capa de salida, se utiliza la ecuación (3.31) y si es el gradiente local de una neurona de alguna capa oculta, entonces se utiliza la ecuación (3.38).

Hay una modificación para la ecuación (3.39), en la que se agrega un término momento α como sigue, [25]:

$$\Delta w_{ji}(n) = \eta\delta_j(n)x_i(n) + \alpha\Delta w_{ji}(n-1) \tag{3.40}$$

3.4.3.1 Método Diagramático

La obtención del gradiente en los algoritmos para el entrenamiento de redes neuronales multicapa generalmente requiere de numerosas expansiones de la regla de la cadena. El método diagramático es un método que facilita este cálculo. Este método hace uso de la teoría de flujo gráfico para construir y manipular diagramas a bloques que representan a las redes neuronales. Este método consiste en simples reglas diagramáticas para representar redes y construir una red adjunta que representa la derivación del gradiente, por medio del flujo de señales del error hacia atrás [28].

En el aprendizaje supervisado, el objetivo es encontrar los pesos $\boldsymbol{w}$ que minimicen la función de costo:

$$J = \sum_{n=1}^{N} J_n \left[d(n), y(n) \right] \tag{3.41}$$

donde n es el índice del tiempo discreto, $y(n)$ es la salida de la red, $d(n)$ es la respuesta deseada, y J_n es la métrica del error genérico.

Las técnicas de optimización requieren el cálculo del gradiente de la función de costo (3.41) con respecto a sus pesos. Existen dos principales formas para calcular el gradiente. En el primer caso, la contribución a la actualización de pesos en cada paso de tiempo es:

$$\Delta W(n) = -\eta \frac{\partial J}{\partial W(n)} \tag{3.42}$$

La derivada parcial (3.42), lleva a una implementación fuera de línea (off-line), es decir, el sistema debe correr al final del tiempo N antes de que se realice la actualización del gradiente.

Por otro lado, el la derivada parcial (3.43), lleva a un algoritmo en línea en donde los pesos son actualizados en cada incremento

$$\Delta \widetilde{W}(n) = \eta \frac{\partial J_n}{\partial W} \tag{3.43}$$

donde la parcial es el cambio en la función de costo actual en el instante n debido al cambio de los pesos en todos los tiempos previos. Se utiliza la notación $\widetilde{W}(n)$ con tilde para resaltar que las actualizaciones no son equivalentes.

Si no hay memoria interna en la red (es decir, una red estática), entonces las dos ecuaciones son equivalentes. Hay que notar que los pesos se mantienen constantes durante todos los pasos hacia adelante y hacia atrás, los cambios totales de los pesos para los algoritmos en línea y fuera de línea son iguales, es decir,

$$\sum_{n=1}^{N} \frac{\partial J}{\partial W(n)} = \sum_{n=1}^{N} \frac{\partial J_n}{\partial W} \tag{3.44}$$

Sin embargo, en la práctica los pesos son actualizados en cada instante de tiempo n, resultando en diferencias en las propiedades de los dos enfoques.

Considérese el cálculo del gradiente en la ecuación (3.42) para algún subconjunto de los pesos $w_{ji} \in W$. A nivel arquitectónico se puede aislar este parámetro entre dos señales internas en la red $a_i(n)$ y $a_j(n)$, tal que

$$a_j(n) = F(w_{ji}, a_i(n)) \tag{3.45}$$

Aquí $F(\cdot)$ representa una función interna continua arbitraria para la representación de diagramas a bloques de la red. En el nivel más bajo, con señales escalares, w_{ji} puede corresponder a un solo peso sináptico, y se tiene simplemente $a_j(n) = w_{ji}a_i(n)$.

Utilizando la regla de la cadena, obtenemos

$$\frac{\partial J}{\partial w_{ji}(n)} = \frac{\partial a_j(n)}{\partial w_{ji}(n)} \frac{\partial J}{\partial a_j(n)} = G_w^F(n)\delta_j(n) \tag{3.46}$$

donde el gradiente del error se define como:

$$\delta_j(n) = \frac{\partial J}{\partial a_j(n)} \tag{3.47}$$

y el Jacobiano como

$$G_w^F = \frac{\partial a_j(n)}{\partial w_{ji}(n)} \tag{3.48}$$

Si el vector $a_j(n)$ tiene dimensiones $N \times 1$, y el vector de pesos $w_{ji}(n)$ tiene dimensiones $N_w \times 1$, las dimensiones de $\partial J/\partial w_{ji}(n)$, $\partial a_j(n)/\partial w_{ji}(n)$, y $\partial J/\partial a_j(n)$ son $N_w \times 1$, $N_w \times N$ y $N \times 1$, respectivamente.

Nótese que el gradiente del error $\delta_j(n)$ depende de la topología completa de la red, contrariamente al Jacobiano que depende sólo localmente de F.

En el caso escalar $G_w^F = a_i(n)$. La actualización de los pesos está dada por:

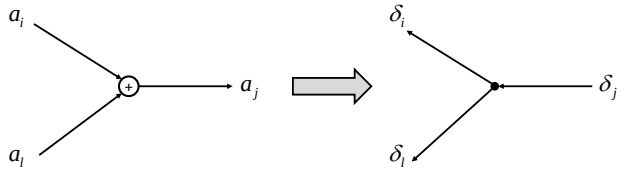
$$\Delta w_{ji}(n) = -\eta \frac{\partial J}{\partial w_{ji}(n)} = -\eta G_w^F(n)\delta_j(n) \tag{3.49}$$

Para completar la especificación del algoritmo se debe de encontrar una fórmula explícita para calcular los términos de delta. La propagación hacia atrás es por naturaleza un algoritmo que genera estos términos en una red alimentada hacia adelante.

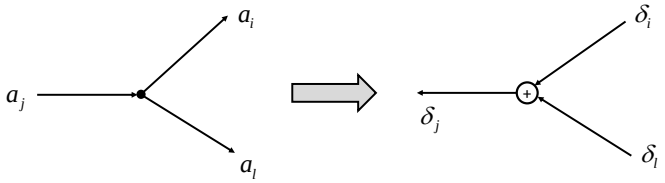
Ahora se desarrollará un método no algebraico simple para obtener los términos de la ecuación (3.49) asociados a alguna arquitectura o topología de alguna red.

Dada una representación de diagramas a bloques de una red y el objetivo de determinar el gradiente del error $\delta_j(n)$, sólo se necesita aplicar un conjunto de transformaciones de diagramas a bloques. Directamente del diagrama a bloques se puede construir la red adjunta invirtiendo la dirección del flujo de la red original, etiquetando todas las señales resultantes $\delta_j(k)$, y llevando a cabo las siguientes operaciones:

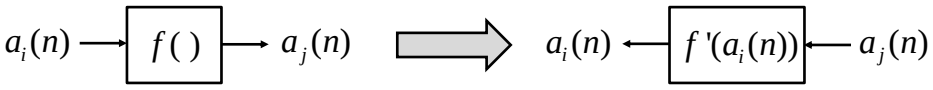
1. Las uniones de sumatoria son reemplazadas por puntos de bifurcación.



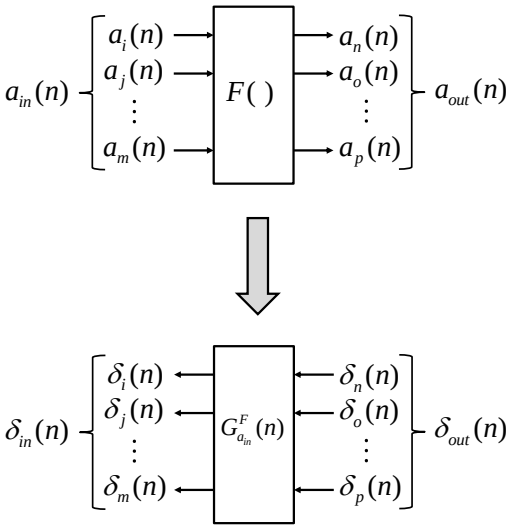
2. Los puntos de bifurcación son reemplazadas por uniones de sumatoria.



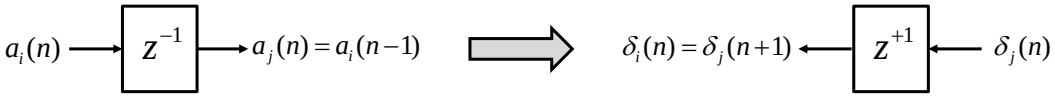
3. Las funciones univariables son reemplazadas por sus derivadas.



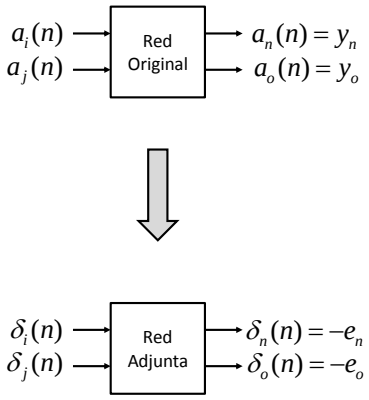
4. Las funciones multivariables son reemplazadas por sus Jacobianos.



5. Los operadores de retardo son reemplazados por operadores de adelanto.



6. Las salidas se convierten en entradas.



3.4.4 Redes Neuronales Recurrentes

Las redes recurrentes son redes neuronales con uno o más lazos de retroalimentación. La retroalimentación puede ser local o global.

Básicamente, hay dos usos funcionales de las redes recurrentes:

- Memorias asociativas
- Mapeo entrada-salida

Existen diferentes topologías de redes recurrentes, aunque todas ellas comparten las siguientes dos características:

1. Todas ellas incluyen un perceptrón multicapa estático o partes de éste.
2. Todas ellas explotan la capacidad de mapeo no lineal del perceptrón multicapa.

La Figura 3.11 muestra la arquitectura de una red recurrente genérica que tiene naturalmente un perceptrón. El modelo tiene una sola entrada que es aplicada a un conjunto de retardos en línea de q unidades.

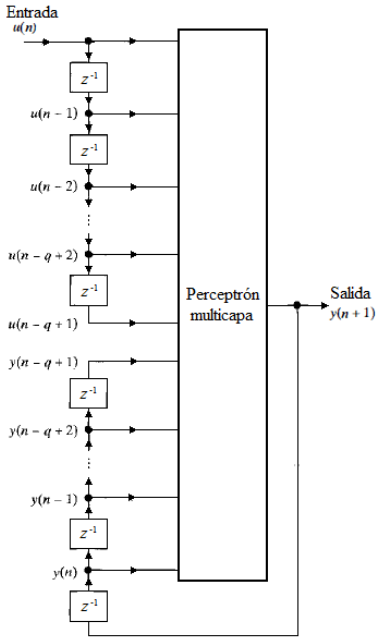


Figura 3.11 Modelo autoregresivo no lineal con entradas externas (NARX), [25].

El vector de la señal en la capa de entrada consiste de un conjunto de datos de los valores originados del exterior de la red presentes y pasados, y los valores retardados de la propia salida de la red neuronal, a continuación se describen:

Valores presentes y pasados de la entrada. Los valores que representan las entradas externas que se originan en el exterior de la red, $(u(n), u(n-1), \dots, u(u-q+1)$.

Valores retardados de la salida. Los valores que se originan a la salida de la red neuronal y se regresan a la capa de entrada de la red neuronal, $y(n), y(n-1), \dots, y(n-q+1)$.

Así que la red recurrente de la se refiere a un modelo autoregresivo no lineal con entradas externas (NARX). El comportamiento dinámico del modelo NARX está descrito por la siguiente ecuación:

$$y(n+1) = F\big(y(n), \dots, y(n-q+1), u(n), \dots, u(n-q+1)\big) \tag{3.50}$$

donde F es una función no lineal de su argumento.

Modelo Espacio-Estado.

La Figura 3.12 muestra un diagrama a bloques de otra red recurrente genérica, llamado modelo espacio-estado. Las neuronas ocultas definen el estado de la red. La salida de la capa oculta es retroalimentada a la entrada mediante un banco de unidades de retardos. La entrada de la capa consiste de una concatenación de nodos retroalimentados y nodos fuente. La red está conectada al exterior mediante los nodos fuente. El número de unidades de retardo que alimenta la entrada de la capa oculta determina el orden del modelo. Sea $\boldsymbol{u}(n)$ un vector de $m\times 1$ que denota la entrada, y $\boldsymbol{x}(n)$ un vector de $q\times 1$ que denota la salida de la capa oculta en el tiempo n. Podemos describir el comportamiento dinámico de la por el par de ecuaciones:

$$\boldsymbol{x}(n+1) = \boldsymbol{f}(\boldsymbol{x}(n), \boldsymbol{u}(n)) \tag{3.51}$$

$$\boldsymbol{y}(n) = \boldsymbol{C}\boldsymbol{x}(n) \tag{3.52}$$

donde $\boldsymbol{f}(\cdot)$ es una función no lineal que describe a la capa oculta, y $\boldsymbol{C}$ es la matriz de los pesos sinápticos que describe a la capa de salida. La capa oculta es no lineal, pero la capa de salida es lineal.

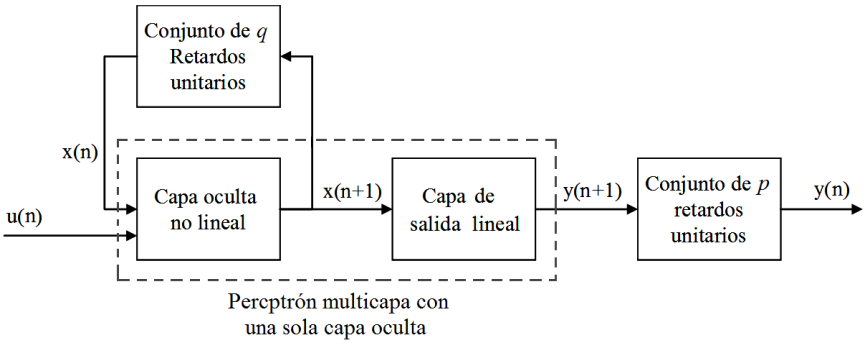


Figura 3.12 Modelo Espacio-Estado.

3.4.4.1 Topología de una Red Neuronal Recurrente en la forma canónica de Jordan.

Ahora se define una topología especial de red recurrente conocida como Red Neuronal Recurrente Entrenable (RNRE) en el modelo espacio-estado y en la forma canónica de Jordan [29], [30], [31]. Este tipo de red está descrito por:

$$\mathbf{x}(\mathrm{n}+1) = \mathbf{J}\mathbf{x}(\mathrm{n}) + \mathbf{B}\mathbf{u}(\mathrm{n}) \tag{3.53}$$

$$\mathbf{z}(\mathrm{n}) = \boldsymbol{\Gamma}[\mathbf{x}(\mathrm{n})] \tag{3.54}$$

$$\mathbf{y}(\mathrm{n}) = \boldsymbol{\Phi}[\mathbf{C}\mathbf{z}(\mathrm{k})] \tag{3.55}$$

$$\mathbf{J}\text{: Diagonal por bloques; } |\mathrm{J_i}| < 1 \tag{3.56}$$

donde: $\mathbf{x}(\mathrm{n})$ es un vector de N estados de la RNRE, $\mathbf{u}(\mathrm{n})$ es un vector de M entradas, $\mathbf{y}(\mathrm{n})$ es un vector de L salidas, $\mathbf{z}(\mathrm{n})$ es un vector de dimensión N de la salida de la capa oculta, $\boldsymbol{\Gamma}$ y $\boldsymbol{\Phi}$ son funciones de activación con dimensiones apropiadas y ambas son tangentes hiperbólicas, $\mathbf{J}$ es una matriz de pesos del estado y es diagonal por bloques con dimensión $N \times N$, J_i son los elementos de $\mathbf{J}$, $\mathbf{B}$ es la matriz de pesos de la entrada con dimensión $N \times M$, $\mathbf{C}$ es la matriz de pesos de la salida de dimensión $L \times N$. La ecuación (3.56) es una condición para la estabilidad.

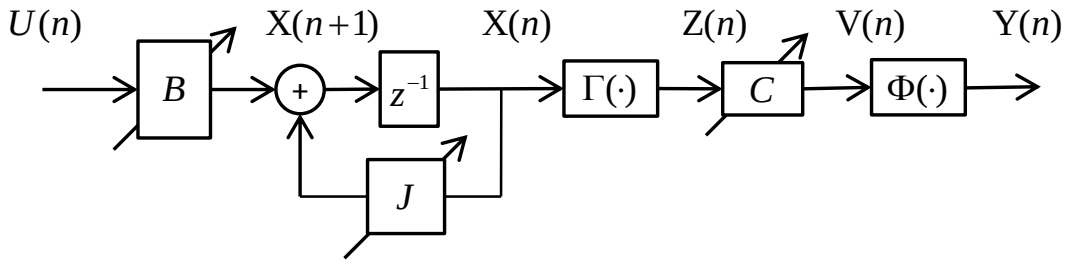


Figura 3.13 Diagrama a bloques de la topología de la RNRE.

Algoritmo de aprendizaje utilizando las reglas diagramáticas.

La función de costo a minimizar está dado por:

$$\zeta(k) = \frac{1}{2}\sum_{j}[E_j(n)]^2, \quad j \in \mathrm{C}, \qquad \zeta = \frac{1}{N_e}\sum_{j}\zeta(n)$$

donde el Error Medio Cuadrático (EMC) $\zeta(n)$ es minimizado en aplicaciones en tiempo real y el total del EMC ζ es minimizado para una época N_e en aplicaciones fuera de línea. El algoritmo de aprendizaje basado en backpropagation con término momento para aplicaciones en tiempo real se expresa en la ecuación siguiente:

$$W(n+1) = W(n) + \eta\Delta W(n) + \alpha\Delta W(n-1)$$
$$|W_{ij}| < W_o \qquad (3.57)$$

donde: $W(\cdot)$ es una matriz de pesos (J, B, C), $\Delta W(\cdot)$ es la modificación de $\Delta W(\cdot)$; η es una matriz diagonal que contiene los coeficientes de aprendizaje; α es una matriz diagonal de coeficientes del termino momento; W_o es una zona restringida para los pesos. Utilizando el diagrama de bloques mostrado en la Figura 3.13, y la red adjunta mostrada en la Figura 3.14, se puede obtener el algoritmo de aprendizaje para la actualización de los pesos sinápticos. Utilizando la red adjunta mostrada en la Figura 3.14, y los vectores obtenidos de las ecuaciones (3.53)-(3.56), se puede obtener el algoritmo de actualización de pesos sinápticos para las matrices J, B, C.

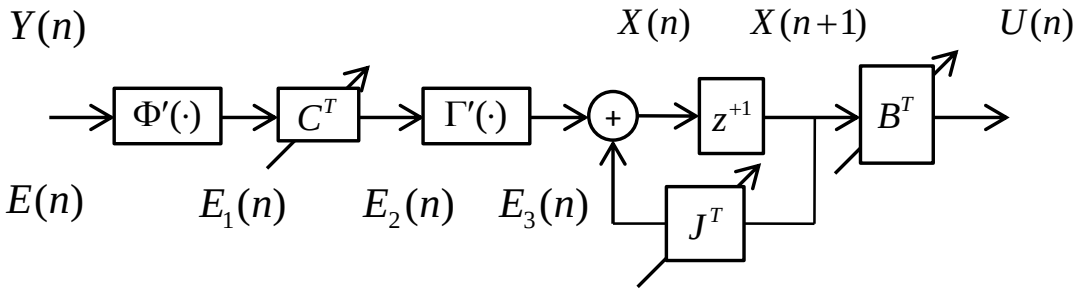


Figura 3.14 Diagrama de bloques de la red adjunta de la topología de la RNRE.

Para la capa de salida:

$$\Delta C(n) = E_1(n)Z^T(n) \tag{3.58}$$

$$E_1(n) = \Phi'[Y(n)]E(n) \tag{3.59}$$

$$E(n) = Y_p(n) - Y(n) \tag{3.60}$$

Para la capa oculta:

$$\Delta J(n) = E_3(n)X^T(n) \tag{3.61}$$

$$E_3(n) = \Gamma'[Z(n)]E_2(n) \tag{3.62}$$

$$E_2(n) = C^T(n)E_1(n) \tag{3.63}$$

$$\Delta vJ(n) = E_3(n) \otimes X(n) \tag{3.64}$$

$$\Delta B(n) = E_3(n)U^T(n) \tag{3.65}$$

donde: $\Delta C, \Delta J, \Delta B$ son las correcciones a los pesos sinápticos de las matrices C, J, B; El error de la salida está dado por (3.60) y con la red adjunta se obtienen los vectores de error E_1, E_2, E_3; Φ', Γ' son derivadas de la función de activación utilizada por la red neuronal.

La ecuación (3.64) da la solución para la actualización de la matriz J como un producto punto entre dos vectores. Las ecuaciones (3.58)-(3.65) representan el algoritmo completo de back-propagation para la red neuronal mostrada en la Figura 3.13. Esta solución puede ser aplicada para cualquier topología de la red neuronal y no requiere el cálculo analítico de derivadas parciales. Utilizando las reglas diagramáticas para el caso complejo pueden ser derivadas de las presentadas en esta sección con el objetivo de obtener algoritmos de aprendizaje de una manera más sencilla.

3.5 Redes Neuronales en el Dominio Complejo

En esta sección se desarrolla el tema central de este trabajo que es la descripción de las redes neuronales en el dominio complejo, que por simplicidad se les llamará redes neuronales complejas.

Las redes neuronales complejas son redes neuronales que trabajan con información en el dominio complejo utilizando variables y parámetros complejos [12].

Una neurona compleja tiene sus entradas, pesos sinápticos y salidas complejas, su función de activación es evaluada en el dominio complejo. Esto presenta ventajas sobre las neuronas en el dominio real. En [32], se demuestra que el problema que se presenta para aproximar la función XOR utilizando sólo una neurona en el dominio real, es resuelto en el dominio complejo utilizando también solo una neurona. Una neurona compleja se puede describir casi siempre, y más adelante se verá por qué, con el mismo diagrama de bloques de una neurona real, en la Figura 3.15 se muestra el esquema de una neurona compleja genérica; las entradas $x_1, x_2, \dots, x_m \in \mathbb{C}$, los pesos sinápticos $w_{k1}, w_{k2}, \dots, w_{km} \in \mathbb{C}$, el umbral o bias $b_k \in \mathbb{C}$; la función de activación $\varphi(\cdot)$ es evaluada en el dominio complejo; la salida $y_k \in \mathbb{C}$.

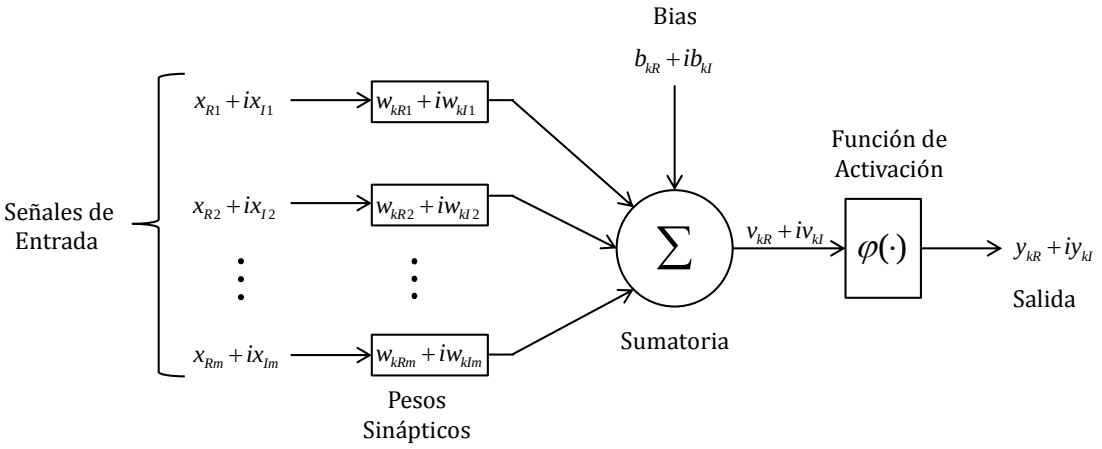


Figura 3.15 Diagrama de bloques del modelo de una neurona en el domino complejo.

La topología de la neurona no siempre será la misma, en el caso complejo la topología de las neuronas queda definida por la función de activación $\varphi(\cdot)$, por lo que la topología de la red neuronal completa queda también definida por la forma que tenga la función de activación, así

como también el algoritmo de aprendizaje será completamente diferente dependiendo de la forma de la función de activación; es por esta razón que se abordará el tema de funciones de activación y posteriormente las topologías resultantes al utilizar cada una de las funciones.

3.5.1 Funciones de Activación

La selección de la función de activación en las redes neuronales es de suma importancia. Las condiciones que se consideran para seleccionar estas funciones en el caso real, es que sean funciones continuas y diferenciables, también es deseable que sean acotadas. Estas funciones ya fueron mencionadas en la sección anterior; sin embargo, para el caso complejo las condiciones son más fuertes, [33], [34].

Para que una función compleja se considere analítica, es decir, que su derivada exista, se deben de cumplir las condiciones de Cauchy-Riemann (2.15),

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \quad \text{y} \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$$

Las propiedades de la derivación compleja son las siguientes:

- $f'(z)$ existe y es continua.
- $f(z)$ es holomorfa.
- $f(z)$ cumple con las condiciones de Cauchy-Riemann.

Algunos ejemplos de funciones analíticas son: z^n, e^z, $\sin(z)$, $\cos(z)$; algunos ejemplos de funciones no analíticas son: $\bar{z}$, $|z|^2$.

Por otro lado, se requiere que la función compleja sea acotada, con esto se cumpliría las condiciones para seleccionar una función de activación compleja. El teorema de Liouville da las condiciones para definir si una función compleja es acotada.

Teorema de Liouville. Para toda función compleja holomorfa f, si existe un número positivo M tal que $|f(z)| \leq M$ para toda $z \in \mathbb{C}$ entonces es constante.

El teorema anterior es falso para las funciones reales, por contraejemplo se tiene la función $\cos(x)$ la cual está acotada y no es constante; por otro lado, existen condiciones más fuertes que están dadas por el "pequeño" teorema de Picard.

"Pequeño" teorema de Picard. Si una función compleja $f(z)$ es entera y no constante, entonces el conjunto de valores que $f(z)$ puede tener es todo el plano complejo o todo el plano menos un punto.

Singularidades en las funciones de activación complejas.

Otro de los problemas en las funciones de activación complejas es el de las singularidades. A continuación se mencionan algunas definiciones que tienen que ver con las singularidades de las funciones complejas.

Definición. Si $f(z)$ tiene sólo un punto singular en z_0, la singularidad es llamada removible si $\lim_{z \to z_0} f(z)$ existe.

Definición. Si $\lim_{z \to z_0} |f(z)| \to \infty$ y $f(z)$ es analítica en una vecindad de z_0, entonces la singularidad no es removible y es llamada aislada.

Definición. Si una singularidad no es removible ni aislada, entonces es conocida como una singularidad esencial o severa.

Este tipo de problemas pueden hacer que el algoritmo no converja y que los cálculos tiendan a hacer cosas incoherentes. Es por esta razón que no se debe de permitir que la función de activación sea evaluada en estos puntos singulares. Se grafica la función definida en (3.66) con el propósito de mostrar su comportamiento ante los puntos singulares.

$$f(z) = \tanh(z), \qquad z \in \mathbb{C} \tag{3.66}$$

En la Figura 3.16, se muestra el comportamiento de la magnitud ($|f(z)|$) de la función tangente hiperbólica en el plano complejo.

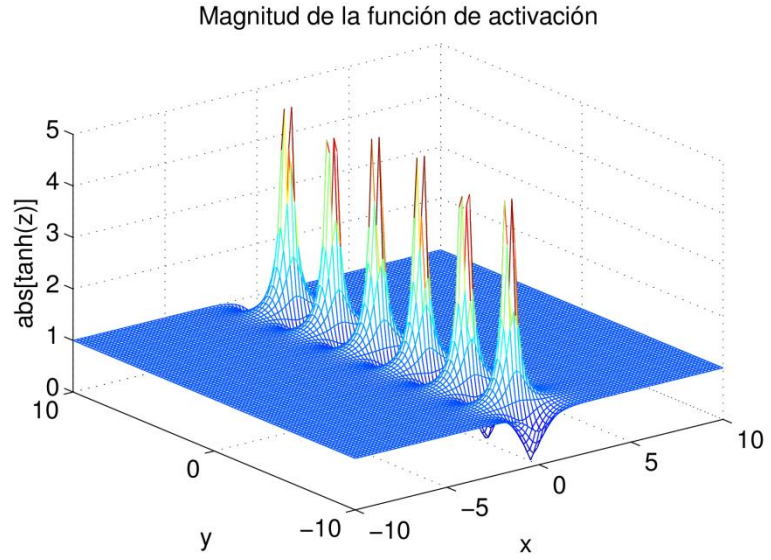


Figura 3.16 Comportamiento de la magnitud de la función tangente hiperbólica compleja.

En la Figura 3.17, se muestra el comportamiento de la fase ($ang(f(z))$) de la función tangente hiperbólica en el plano complejo.

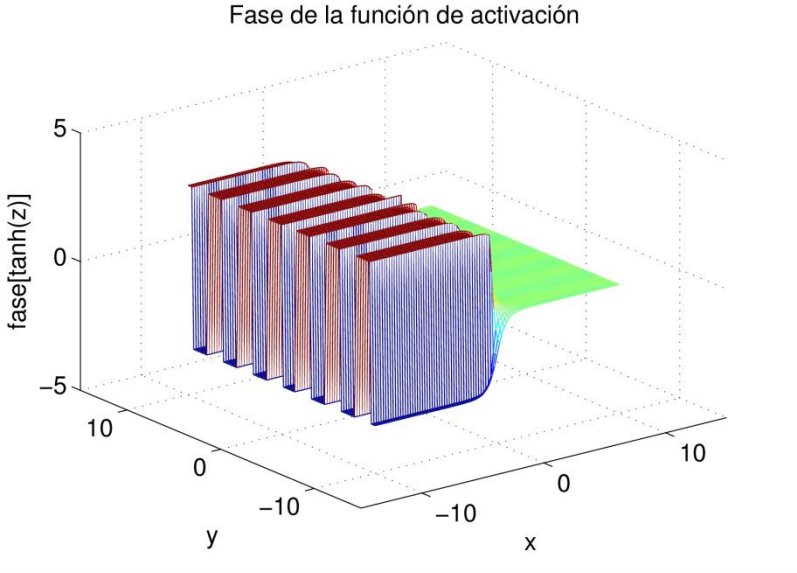


Figura 3.17 Comportamiento de la fase de la función tangente hiperbólica compleja.

En la Figura 3.18, se muestra el comportamiento de la parte real ($Re(f(z))$) de la función tangente hiperbólica en el plano complejo.

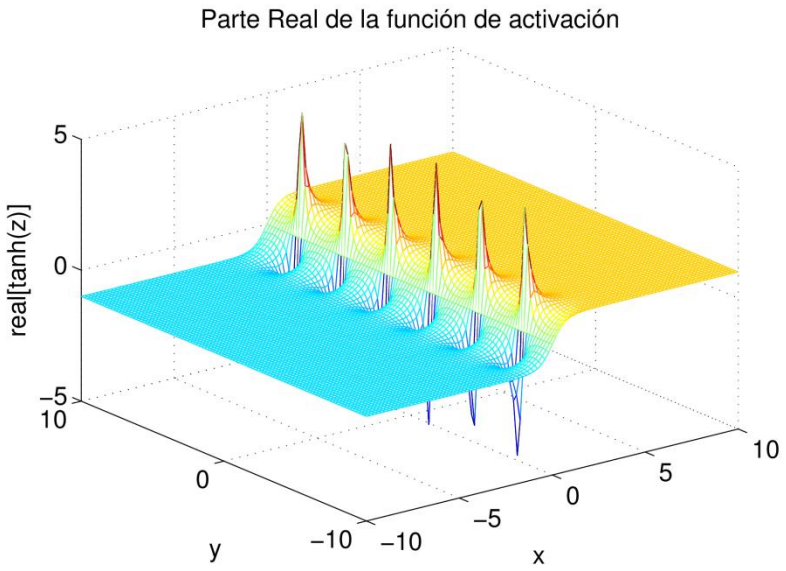


Figura 3.18 Comportamiento de la parte real de la función tangente hiperbólica compleja.

En la Figura 3.19, se muestra el comportamiento de la parte imaginaria ($Im(f(z))$) de la función tangente hiperbólica en el plano complejo.

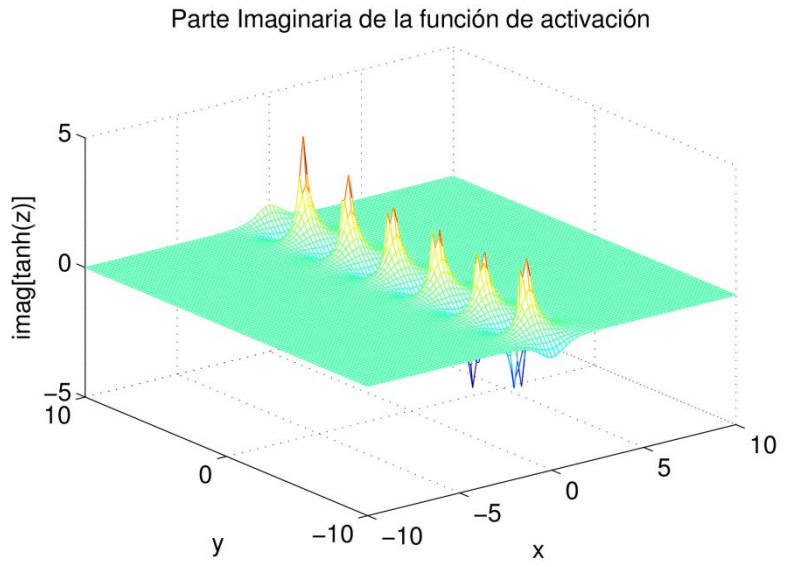


Figura 3.19 Comportamiento de la parte imaginaria de la función tangente hiperbólica compleja.

En las figuras anteriores se puede ver que los puntos singulares son periódicos, estos puntos pueden hacer divergir a la red neuronal, por lo que se utilizan funciones fabricadas que más adelante se presentan.

Habiendo definido las propiedades de la derivación compleja y las condiciones para que una función compleja sea acotada, se concluye que si una función es holomorfa no será acotada y viceversa, si la función es acotada no es holomorfa.

De lo anterior se puede decir que no existe una función que cumpla con las condiciones para ser derivable y acotada al mismo tiempo. Regresando al teorema de Liouville, éste sólo es válido para todo el plano complejo, por lo que si se restringe el domino de la función compleja el teorema ya no es válido.

La siguiente proposición [33], presenta una solución a este problema de funciones de activación en el dominio complejo.

Proposición. En un dominio acotado del plano complejo $\Im$, una función de activación compleja no lineal $f(z)$ necesita ser analítica y acotada.

Por lo que si se considera una región del plano complejo restringido, es posible encontrar una función que sea analítica y esté acotada. La solución a lo mencionado, son las llamadas "*engineered complex functions*" o funciones complejas construidas; estas funciones son necesarias ya que los algoritmos de optimización utilizados en el desarrollo de este trabajo no considera restricciones, por lo que no se puede limitar el crecimiento de los valores en las neuronas.

Funciones complejas construidas.

Para enfrentar el problema antes descrito se recurren a las funciones construidas. La idea principal de estas funciones es deshacerse de las propiedades que hacen que una función no sea analítica y acotada al mismo tiempo, se sustituye la singularidad de la función por otra función, es decir, se unen las funciones al borde de la singularidad. Algunas de estas funciones introducidas en [35], son las siguientes:

$$\begin{cases} f_1(z) = tanh(z), & z \in \mathbb{C}\backslash\left[-\frac{\pi}{2}-\epsilon; -\frac{\pi}{2}+\epsilon\right] \cup [-\epsilon;\epsilon] \\ f_2(z) = u\big(Re(z)\big) + iv\big(Im(z)\big), & u(\cdot), v(\cdot) \in \mathbb{R} \\ f_3(z) = f\big(abs(z)\big)exp^{i\phi(z)} & \end{cases} \tag{3.67}$$

Las funciones de activación que se consideran en este trabajo son $f_1(z)$, $f_2(z)$; para conocer la importancia de estas funciones.

La primera función de activación que se estudiará está definida en (3.68). Esta función tiene singularidades en algunas partes del plano complejo (singularidades que resultan ser periódicas con periodo de $\pi/2$) y es por eso que esta función es evaluada únicamente en regiones del plano complejo donde se eviten estas singularidades.

$$f(z) = tanh(z), \qquad z \in \mathbb{C}\backslash[-\frac{\pi}{2} - \epsilon; -\frac{\pi}{2} + \epsilon] \cup [-\epsilon; \epsilon] \tag{3.68}$$

La segunda función de activación que se estudiará está definida en (3.68). Esta función no presenta singularidades por lo que puede ser evaluada en todo el plano complejo.

$$f(z) = tanh\big(Re(z)\big) + i\ tanh(Im(z)), \qquad z \in \mathbb{C} \tag{3.69}$$

Es importante mencionar que la función de activación define directamente la topología de la red como se verá más adelante.

3.5.2 Algoritmos de Aprendizaje

El algoritmo que se tratará en este trabajo está basado en un método de optimización llamado gradiente descendente. Este algoritmo de optimización minimiza una función de costo (2.20). Sin embargo, como ya se vio en el capítulo dos, esta función de costo no es analítica por lo que no cumple las condiciones de Cauchy-Riemann. Este es un problema que será abordado haciendo uso del llamado cálculo de Wirtinger detallado en el capítulo dos, el cual se utiliza para obtener derivadas de funciones no analíticas.

Se desarrollará el algoritmo para ambas funciones de activación (3.68), (3.69). Más adelante, se demostrará que utilizando reglas diagramáticas en su versión compleja, desarrolladas más adelante, resultan eficientes para obtener algoritmos de aprendizaje sin necesidad de manipulaciones de términos analíticos.

Para la primera función de activación (3.68), se desarrollará el algoritmo de Backpropagation en su versión compleja reportado en [6].

La relación entrada salida de una neurona se representa por la siguiente ecuación:

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj}x_j + b_k\right) \tag{3.70}$$

El error que se requiere para realizar la adaptación se define por la siguiente ecuación:

$$e_k(n) = d_k(n) - y_k(n) \tag{3.71}$$

donde $d_k(n)$ es el vector deseado y $y_k(n)$ es la salida de la red de la neurona k.

La suma de los errores al cuadrado se puede expresar de la siguiente forma:

$$\zeta(k) = \frac{1}{2}\sum_{j}\left[E_j(n)\right]\left[\overline{E_j(n)}\right], \quad j \in \mathrm{C}, \quad \zeta = \frac{1}{N_e}\sum_{j}\zeta(n) \tag{3.72}$$

Como ya se mencionó antes, el algoritmo Backpropagation está basado en la minimización de la funcional de costo $\zeta(n)$ alterando los valores de w_{kj} basado en la técnica del gradiente.

Lo primero que se debe de hacer es encontrar la derivada parcial de $\zeta(k)$ con respecto a los pesos sinápticos correspondientes a la capa de salida y extender los coeficientes a la capa oculta. Como ya se mencionó en el capítulo dos, la función (3.72) no es analítica, por lo que se tiene que derivar con respecto a la parte real y a la parte imaginaria por separado.
Los pesos sinápticos en su forma compleja se pueden escribir de la siguiente forma:

$$w_{kj}(k) = wr_{kj}(n) + i\, wi_{kj}(n)$$

donde $wr_{kj}(k)$ es la parte real de $w_{kj}(k)$ y $wi_{kj}(k)$ es la parte imaginaria de $w_{kj}(n)$.

El objetivo es encontrar las expresiones,

$$\frac{\partial \zeta(n)}{\partial wr_{kj}(n)} \quad \text{y} \quad \frac{\partial \zeta(n)}{\partial wi_{kj}(n)}$$

y deducir el algoritmo de aprendizaje basado en gradiente descendente para la capa de salida y para la capa oculta, se comenzará con deducir el algoritmo para la capa de salida; la regla de adaptación para la capa de salida está definida por las siguientes ecuaciones:

$$wr_{kj}(n+1) = wr_{kj}(n) - \frac{1}{2}\mu \frac{\partial \zeta(n)}{\partial wr_{kj}(n)} \tag{3.73}$$

$$wi_{kj}(n+1) = wi_{kj}(n) - \frac{1}{2}\mu \frac{\partial \zeta(n)}{\partial wi_{kj}(n)} \tag{3.74}$$

Combinando (3.73) y (3.74), se obtiene:

$$w_{kj}(n+1) = w_{kj}(n) - \frac{1}{2}\mu \left(\frac{\partial \zeta(n)}{\partial wr_{kj}(n)} + i \frac{\partial \zeta(n)}{\partial wi_{kj}(n)} \right) \tag{3.75}$$

Para la obtención del primer término que se encuentra entre paréntesis, es decir la derivada parcial del error con respecto a la parte real de los pesos sinápticos, se aplica la regla de la cadena como sigue:

$$\begin{aligned} \frac{\partial \zeta(n)}{\partial wr_{kj}(n)} &= \frac{\partial \zeta(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial wr_{kj}(n)} + \frac{\partial \zeta(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial \overline{v_k(n)}} \frac{\partial \overline{v_k(n)}}{\partial wr_{kj}(n)} \\ &= + \frac{\partial \zeta(n)}{\partial \overline{y_k(n)}} \frac{\partial \overline{y_k(n)}}{\partial \overline{v_k(n)}} \frac{\partial \overline{v_k(n)}}{\partial wr_{kj}(n)} + \frac{\partial \zeta(n)}{\partial \overline{y_k(n)}} \frac{\partial \overline{y_k(n)}}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial wr_{kj}(n)} \end{aligned} \tag{3.76}$$

donde $y_k(n) = \varphi(u_k + b_k)$ y $v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$.

En la ecuación (3.76), se pueden identificar los términos $\partial y_k\,(n)/\partial \overline{v_k\,(n)}$ y $\partial \overline{y_k\,(n)}/\partial v_k\,(n)$, los cuales debido a la naturaleza de la función de activación son iguales a cero, ya que la función $\phi(\cdot)$ hace un mapeo del dominio real al dominio real y del dominio complejo al dominio

complejo, pero nunca del dominio real al complejo; de esto que la ecuación (3.76) queda reescrita en (3.77).

$$\frac{\partial \zeta(n)}{\partial wr_{kj}(n)} = \frac{\partial \zeta(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial wr_{kj}(n)} + \frac{\partial \zeta(n)}{\partial \overline{y_k(n)}} \frac{\partial \overline{y_k(n)}}{\partial \overline{v_k(n)}} \frac{\partial \overline{v_k(n)}}{\partial wr_{kj}(n)} \tag{3.77}$$

El siguiente paso es calcular las derivadas parciales de (3.77):

$$\frac{\partial \zeta(n)}{\partial wr_{kj}(n)} = -\left(\overline{d_k(n)} - \overline{y_k(n)}\right)\varphi_k'\left(v_k(n)\right)x_j(n) - \left(d_k(n) - y_k(n)\right)\varphi_k'\left(\overline{v_k(n)}\right)\overline{x_j(n)} \tag{3.78}$$

De manera similar se puede obtener la expresión para la derivada parcial $\partial\zeta(n)/\partial wi_{kj}\,(n)$:

$$\frac{\partial \zeta(n)}{\partial wi_{kj}(n)} = -i\,\left(\overline{d_k(n)} - \overline{y_k(n)}\right)\varphi_k'\left(v_k(n)\right)x_j(n) - i\,\left(d_k(n) - y_k(n)\right)\varphi_k'\left(\overline{v_k(n)}\right)\overline{x_j(n)} \tag{3.79}$$

Por lo que sumando (3.78) y (3.79) se tiene:

$$\frac{\partial \zeta(k)}{\partial wr_{kj}(k)} + i\frac{\partial \zeta(k)}{\partial wi_{kj}(k)} = -2\left(d_k(n) - y_k(n)\right)\varphi_k'\left(\overline{v_k(n)}\right)\overline{x_j(n)} \tag{3.80}$$

Sustituyendo (3.80) en (3.75) se obtiene la regla de actualización de pesos sinápticos para la capa de salida:

$$w_{ji}(n+1) = w_{ji}(n) - \eta\left(d_k(n) - y_k(n)\right)\varphi_k'\left(\overline{v_k(n)}\right)\overline{x_j(n)} \tag{3.81}$$

Para las neuronas de la capa oculta se tiene la misma regla de actualización de pesos sinápticos que la capa de salida (3.75), las derivadas parciales presentan un cambio, ya que se debe de obtener el gradiente del error con respecto a los pesos de la neurona oculta, la generalización para varias capas ocultas es presentada al final del desarrollo.

$$\frac{\partial \zeta(n)}{\partial wr_{ji}(n)} = \frac{\partial \zeta(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial wr_{ji}(n)} + \frac{\partial \zeta(n)}{\partial \overline{y_j(n)}} \frac{\partial \overline{y_j(n)}}{\partial \overline{v_j(n)}} \frac{\partial \overline{v_j(n)}}{\partial wr_{ji}(n)} \tag{3.82}$$

El cálculo de las derivadas parciales en (3.82) no es directo, por lo que es necesario utilizar la regla de la cadena nuevamente.

Considérese la siguiente ecuación para describir la derivada parcial del error con respecto a las neuronas de la capa de salida.

$$\begin{aligned}\frac{\partial \zeta(n)}{\partial y_j(n)} &= -\sum_k \left(\overline{d_k(n)} - \overline{y_k(n)}\right) \frac{\partial y_k(n)}{\partial y_j(n)} \\ &= -\sum_k \left(\overline{d_k(n)} - \overline{y_k(n)}\right) \varphi_k'\left(v_k(n)\right) w_{kj}(n)\end{aligned} \tag{3.83}$$

De la misma manera se calcula la derivada parcial $\partial \zeta(n)/\partial y_j\ (n)$:

$$\frac{\partial \zeta(n)}{\partial \overline{y_j(n)}} = -\sum_k \left(d_k(n) - y_k(n)\right) \varphi_k'\left(\overline{v_k(n)}\right) \overline{w_{kj}(n)} \tag{3.84}$$

Sustituyendo (3.83) y (3.84) en (3.82) se tiene:

$$\begin{aligned}\frac{\partial \zeta(n)}{\partial w r_{kj}(n)} &= -\sum_k \left(\overline{d_k(n)} - \overline{y_k(n)}\right) \varphi_k'\left(v_k(n)\right) w_{kj}(n) \varphi_j'\left(v_j(n)\right) x_j(n) \\ &= +\left[-\sum_k \left(d_k(n) - y_k(n)\right) \varphi_k'\left(\overline{v_k(n)}\right) \overline{w_{kj}(n)}\right] \varphi_j'\left(\overline{v_j(n)}\right) \overline{x_j(n)}\end{aligned} \tag{3.85}$$

La derivada parcial con respecto a la parte imaginaria se define como:

$$\begin{aligned}\frac{\partial \zeta(n)}{\partial w r_{kj}(n)} &= i\left[-\sum_k \left(\overline{d_k(n)} - \overline{y_k(n)}\right) \varphi_k'\left(v_k(n)\right) w_{kj}(n)\right] \varphi_j'\left(v_j(n)\right) x_j(n) \\ &= -i\left[-\sum_k \left(d_k(n) - y_k(n)\right) \varphi_k'\left(\overline{v_k(n)}\right) \overline{w_{kj}(n)}\right] \varphi_j'\left(\overline{v_j(n)}\right) \overline{x_j(n)}\end{aligned} \tag{3.86}$$

Sumando (3.85) y (3.86) se obtiene la regla de aprendizaje:

$$w_{ji}(n+1) = w_{ji}(n) - \eta \left[\sum_{k} (d_k(n) - y_k(n))\, \varphi_k'\big(\overline{v_k(n)}\big) \overline{w_{kJ}(n)} \right] \varphi_j'\big(\overline{v_j(n)}\big) \overline{x_J(n)} \tag{3.87}$$

Las ecuaciones (3.81) y (3.87) son las dos ecuaciones básicas de actualización de pesos sinápticos del algoritmo Backpropagation.

El algoritmo de aprendizaje queda definido por el tipo de función de activación que se utilice; por lo que se necesita desarrollar un algoritmo diferente para una función de activación diferente.

Para la segunda función de activación que se considera en este trabajo (3.69), se desarrolla un algoritmo de aprendizaje basado en gradiente descendente, es decir, el algoritmo sigue siendo backpropagation pero adaptado a la segunda función de activación.

Retomando la ecuación (3.75),

$$w_{kj}(k+1) = w_{kj}(k) - \frac{1}{2}\mu \left(\frac{\partial \zeta(k)}{\partial wr_{kj}(k)} + i \frac{\partial \zeta(k)}{\partial wi_{kj}(k)} \right)$$

se puede obtener el algoritmo de aprendizaje para la función de activación (3.69),

$$f(z) = tanh\big(Re(z)\big) + i\, tanh(Im(z)), \qquad z \in \mathbb{C}$$

Se comienza obteniendo la expresión de la derivada parcial del error (3.72) con respecto a la parte real de los pesos sinápticos.

$$\frac{\partial \zeta(n)}{\partial wR_{kj}(n)} = -\frac{1}{2}\left(\overline{d_k(n)} - \overline{y_k(n)}\right)\left[\frac{\partial y_k[v_kR(n)](n)}{\partial v_kR(n)}\frac{\partial v_kR(n)}{\partial wr_{kj}(n)} + i\,\frac{\partial y_k[v_kI(n)](n)}{\partial y_kI(n)}\frac{\partial y_kI(n)}{\partial \overline{v_kR(n)}}\right]$$
$$= -\frac{1}{2}\left(d_k(n) - y_k(n)\right)\left[\frac{\partial y_k[v_kR(n)](n)}{\partial v_kR(n)}\frac{\partial v_kR(n)}{\partial wr_{kj}(n)} - i\,\frac{\partial y_k[v_kI(n)](n)}{\partial y_kI(n)}\frac{\partial y_kI(n)}{\partial \overline{v_kR(n)}}\right] \quad (3.88)$$

Calculando las derivadas parciales de la ecuación (3.88), se tiene:

$$\frac{\partial \zeta(n)}{\partial wR_{kj}(n)} = -\frac{1}{2}\left(\overline{d_k(n)} - \overline{y_k(n)}\right)\left[\varphi_k'\,(v_kR(n))x_jR(n) + i\,\varphi_k'\,(v_kI(n))x_jI(n)\right]$$
$$= -\frac{1}{2}\left(d_k(n) - y_k(n)\right)\left[\varphi_k'\,(v_kR(n))x_jR(n) - i\,\varphi_k'\,(v_kI(n))x_jI(n)\right] \quad (3.89)$$

De manera similar se encuentra la derivada parcial para la parte imaginaria del peso sináptico:

$$\frac{\partial \zeta(n)}{\partial wR_{kj}(n)} = -\frac{1}{2}\left(\overline{d_k(n)} - \overline{y_k(n)}\right)\left[\varphi_k'\,(v_kR(n))\left(-x_jI(n)\right) + i\,\varphi_k'\,(v_kI(n))x_jR(n)\right]$$
$$= -\frac{1}{2}\left(d_k(n) - y_k(n)\right)\left[\varphi_k'\,(v_kR(n))\left(-x_jI(n)\right) - i\,\varphi_k'\,(v_kI(n))x_jR(n)\right] \quad (3.90)$$

Sumando las ecuaciones (3.89) y (3.90) se tiene:

$$\frac{\partial \zeta(n)}{\partial wr_{kj}(n)} + i\frac{\partial \zeta(n)}{\partial wi_{kj}(n)}$$
$$= -\frac{1}{2}\left[\left(d_kR(n) - y_kR(n)\right)\varphi_k'\left(v_kR(n)\right) + i\left(d_kI(n) - y_kI(n)\right)\varphi_k'\,(v_kI(n))\right]\overline{x_j(n)} \quad (3.91)$$

Se define el gradiente local como sigue:

$$\delta_k(n) = \left(d_kR(n) - y_kR(n)\right)\varphi_k'\left(v_kR(n)\right) + i\left(d_kI(n) - y_kI(n)\right)\varphi_k'\,(v_kI(n)) \quad (3.92)$$

De esto, que la regla de aprendizaje para los pesos sinápticos de la capa de salida se puede escribir de la forma:

$$w_{kj}(n+1) = w_{kj}(n) - \eta\delta_k(n)\overline{x_J(n)} \tag{3.93}$$

Para la capa oculta se desarrolla el algoritmo teniendo en cuenta que el gradiente local será con respecto a los pesos de la capa oculta.

El gradiente de $\zeta(n)$ con respecto a los pesos de la capa oculta se define de la siguiente manera:

$$\begin{aligned}
&\frac{\partial\zeta(n)}{\partial wr_{ji}(n)} + i\frac{\partial\zeta(n)}{\partial wi_{ji}(n)} \\
&\qquad = -\sum_k \Big[\Big(\delta_k R(n) wr_{kj}(n) + \delta_k I(n) wi_{kj}(n)\Big)\varphi_k'\left(v_k R(n)\right) \\
&\qquad + i\,\Big(\delta_k I(n) wr_{kj}(n) + \delta_k R(n) wi_{kj}(n)\Big)\varphi_k'\left(v_k I(n)\right)\Big]\overline{x_\iota(n)} \\
&= -\left[\varphi_k'\left(v_k R(n)\right) Re\left(\sum_k \delta_k \overline{w_{kJ}(n)}\right) + \varphi_k'\left(v_k I(n)\right) Im\left(\sum_k \delta_k \overline{w_{kJ}(n)}\right)\right]\overline{x_\iota(n)}
\end{aligned} \tag{3.94}$$

Si se define δ_j como sigue:

$$\delta_j(n) = \varphi_k'\left(v_k R(n)\right) Re\left(\sum_k \delta_k \overline{w_{kJ}(n)}\right) + \varphi_k'\left(v_k I(n)\right) Im\left(\sum_k \delta_k \overline{w_{kJ}(n)}\right) \tag{3.95}$$

La regla de aprendizaje para la capa oculta se expresa por la siguiente ecuación:

$$w_{ji}(n+1) = w_{ji}(n) - \eta\delta_j(n)\overline{x_\iota(n)} \tag{3.96}$$

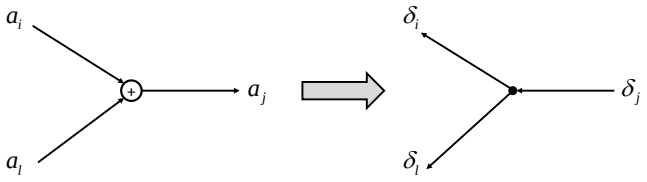
De esta manera, se tiene el algoritmo de aprendizaje para la función de activación (3.69).

3.5.2.1 Método Diagramático Complejo

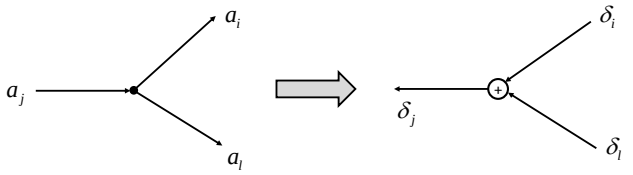
La obtención del algoritmo de aprendizaje basado en el gradiente descendente implica el cálculo de derivadas parciales que son calculadas utilizando la regla de la cadena. Como ya se estudió en la sección anterior, este algoritmo es matemáticamente difícil de obtener, es por eso que se introduce el método diagramático en su versión compleja, como una generalización que abarca las redes neuronales reales y complejas.

En seguida se mencionan las reglas que se obtuvieron para la construcción de la red adjunta para el caso del método diagramático complejo.

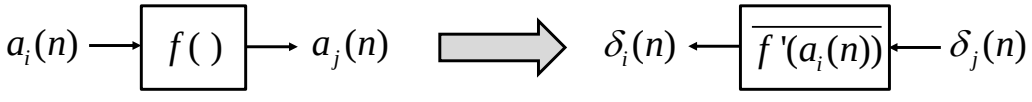
1. Las uniones de sumatoria son reemplazadas por puntos de bifurcación.



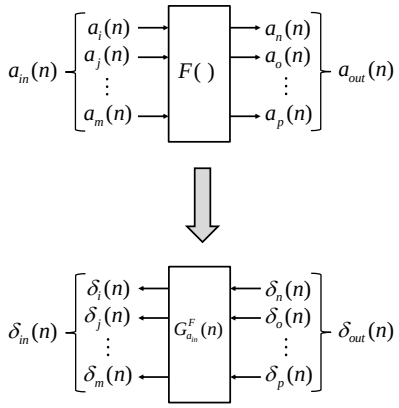
2. Los puntos de bifurcación son reemplazadas por uniones de sumatoria.



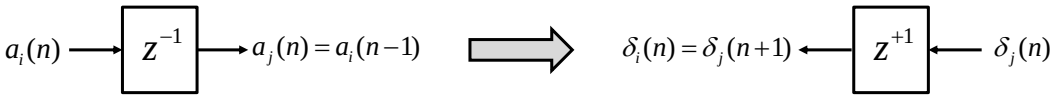
3. Las funciones univariables son reemplazadas por sus derivadas.



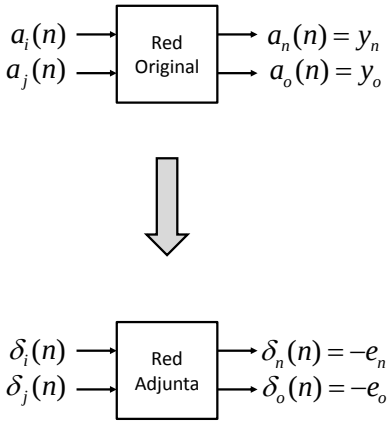
4. Las funciones multivariables son reemplazadas por sus Jacobianos.



5. Los operadores de retardo son reemplazados por operadores de adelanto.



6. Las salidas se convierten en entradas.



3.5.3 Topologías en Redes Neuronales Complejas Recurrentes

Las redes recurrentes complejas tienen la misma topología que las redes neuronales recurrentes en el dominio real, las entradas, las salidas, los pesos y las funciones de activación son complejas. Debido a lo anterior, únicamente se tratará el modelo espacio estado en la forma de diagrama de bloques ya mencionado; se desarrolla su algoritmo de aprendizaje considerando ambas funciones de activación utilizando las reglas diagramáticas propuestas, [36], [37].

Se abordará el modelo de espacio estado para la función de activación siguiente:

$$f(z) = tanh(z), \qquad z \in \mathbb{C}\backslash[-\frac{\pi}{2} - \epsilon; -\frac{\pi}{2} + \epsilon] \cup [-\epsilon; \epsilon] \tag{3.97}$$

La topología especial de red recurrente conocida como Red Neuronal Recurrente Entrenable (RNRE) en el modelo espacio-estado para la función de activación considerada en la forma canónica de Jordan [29], [30], [31]. Este tipo de red está descrito por:

$$\mathbf{x}(\mathrm{n}+1) = \mathbf{J}\mathbf{x}(\mathrm{n}) + \mathbf{B}\mathbf{u}(\mathrm{n}) \tag{3.98}$$

$$\mathbf{z}(\mathrm{n}) = \boldsymbol{\Gamma}[\mathbf{x}(\mathrm{n})] \tag{3.99}$$

$$\mathbf{y}(\mathrm{n}) = \boldsymbol{\Phi}[\mathbf{C}\mathbf{z}(\mathrm{n})] \tag{3.100}$$

$$\mathbf{J}\text{: Diagonal por bloques; } |\mathrm{J_i}| < 1 \tag{3.101}$$

donde: $\mathbf{x}(\mathrm{n})$ es un vector de N estados de la RNRE, $\mathbf{u}(\mathrm{n})$ es un vector de M entradas, $\mathbf{y}(\mathrm{n})$ es un vector de L salidas, $\mathbf{z}(\mathrm{n})$ es un vector de dimensión N de la salida de la capa oculta, $\boldsymbol{\Gamma}$ y $\boldsymbol{\Phi}$ son funciones de activación con dimensiones apropiadas y ambas son tangentes hiperbólicas, $\mathbf{J}$ es una matriz de pesos del estado y es diagonal por bloques con dimensión $N \times N$, J_i son los elementos de $\mathbf{J}$, $\mathbf{B}$ es la matriz de pesos de la entrada con dimensión $N \times M$, $\mathbf{C}$ es la matriz de pesos de la salida de dimensión $L \times N$. La ecuación (3.56) es una condición para la estabilidad.

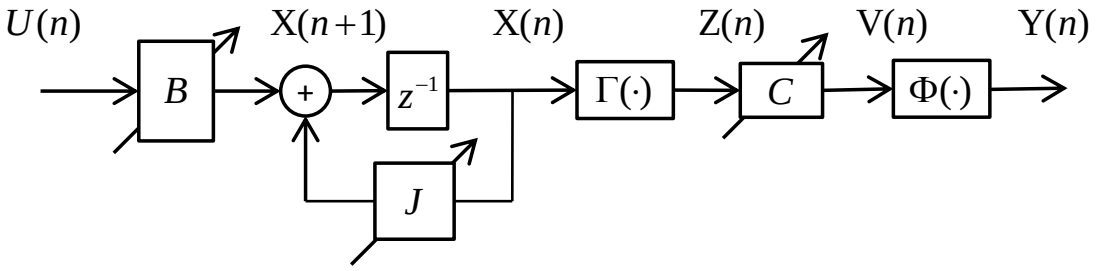


Figura 3.20 Diagrama a bloques de la topología de la RNRE.

3.5.4 Algoritmo de Aprendizaje

El algoritmo de aprendizaje para las redes recurrentes se vuelve más complicado de obtener en el caso complejo, por lo que se utilizarán las reglas diagramáticas propuestas para su desarrollo. Se utilizarán ambas funciones de activación en la red neuronal recurrente propuesta.

3.5.4.1 Primera función de activación

La primera función de activación que será considerada está expresada por (3.68). Esta función de activación tiene singularidades en algunas regiones del plano complejo y por esta razón, la función es evaluada evitando estas regiones con singularidades, [35].

La red neuronal que utiliza esta función de activación está representada en su diagrama de bloques en la Figura 3.21. Este tipo de representación permite utilizar las reglas diagramáticas para describir en bloques la red adjunta y obtener de esta ultima el algoritmo de aprendizaje para toda la red.

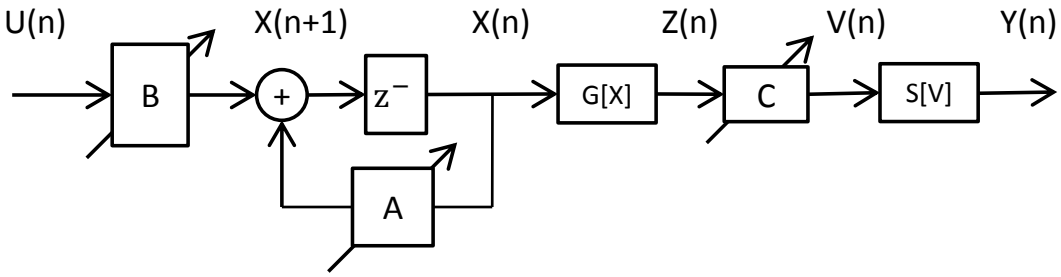


Figura 3.21 Diagrama de bloques de la red neuronal recurrente compleja con la primera función de activación

La descripción matemática de la red neuronal recurrente compleja es la siguiente:

$A \epsilon \mathbb{C}^{nxn}$: Matriz de realimentación.;

$B \epsilon \mathbb{C}^{nxm}$: Matriz de entrada;

$C \epsilon \mathbb{C}^{pxn}$: Matriz de salida;

$X(k) \epsilon \mathbb{C}^{nx1}$: Vector de estado;

$U(k) \epsilon \mathbb{C}^{mx1}$: Entradas a la red;

$Y(k) \epsilon \mathbb{C}^{px1}$: Salida de la red;

$G[\cdot], S[\cdot]$: Vectores evaluados con la función de activación considerada.;

m: Número de entradas;

n: Número de neuronas en la capa oculta;

p: Número de neuronas en la capa de salida

La matriz de realimentación A es diagonal a bloques y con la siguiente restricción $|A_i| < 1$.

Aplicando las reglas diagramáticas a la red neuronal en diagrama de bloques se obtiene la red adjunta, que se presenta en la Figura 3.22.

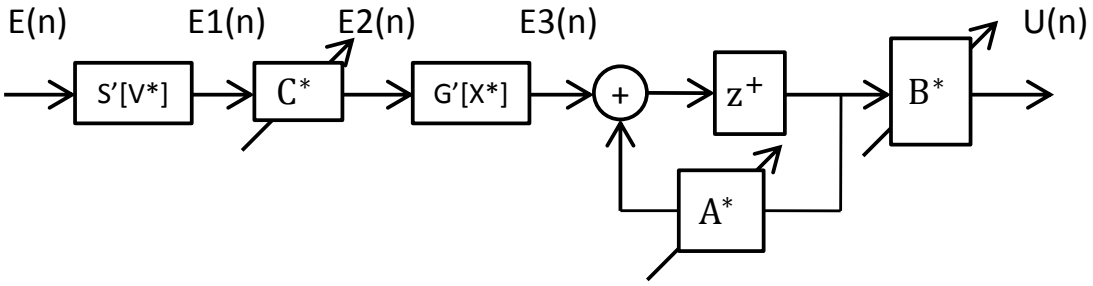


Figura 3.22 Red adjunta dela primera función de activación

Aplicando la regla de Backpropagation se puede obtener el algoritmo de aprendizaje utilizando la red adjunta.

Para la capa de salida:

$$\Delta C(n) = E_1(n)Z^*(n) \tag{3.102}$$

$$E_1(n) = S'[Y[V^*(n)](k)]E(n) \tag{3.103}$$

$$E(n) = T(n) - Y(n) \tag{3.104}$$

Para la capa oculta:

$$\Delta A(n) = E_3(n)X^*(n) \tag{3.105}$$
$$E_3(n) = G'[Z[X^*(n)](k)]E_2(n) \tag{3.106}$$
$$E_2(n) = C^*(n)E_1(n) \tag{3.107}$$
$$\Delta vA(n) = E_3(n) \otimes X^*(n) \tag{3.108}$$
$$\Delta B(n) = E_3(n)U^*(n) \tag{3.109}$$

donde:

$G'[\cdot], S'[\cdot]$: Derivada de la función de activación;

a^*: Transpuesto conjugado de a;

$T(n)$: Vector de salida deseado.

Como se puede ver, la aplicación de las reglas diagramáticas para obtener la red adjunta y posteriormente el algoritmo de aprendizaje es un método simplificado para basado en el gradiente descendente en el dominio complejo.

3.5.4.2 Segunda función de activación

La segunda función de activación expresada en (3.69) no singularidades, por lo que puede ser evaluada en todo el plano complejo.

La topología de la red neuronal queda definida por la función de activación, por lo que la descripción matemática de esta red neuronal con la segunda función de activación queda definida por las siguientes ecuaciones:

$$\mathrm{X}(n+1) = AX(n) + BU(n) \tag{3.110}$$
$$Z(n) = \mathrm{G}[X\_Re\ (n)] + iG[X\_Im\ (n)] \tag{3.111}$$
$$V(n) = Z_{Re}C_{Re} + iZ_{Im}C_{Im} \tag{3.112}$$
$$Y(n) = \ \mathrm{S}[V_{Re}(n)] + iS[V_{Im}(n)] \tag{3.113}$$

La red neuronal compleja recurrente puede ser representada en diagrama de bloques como se muestra en la Figura 3.23.

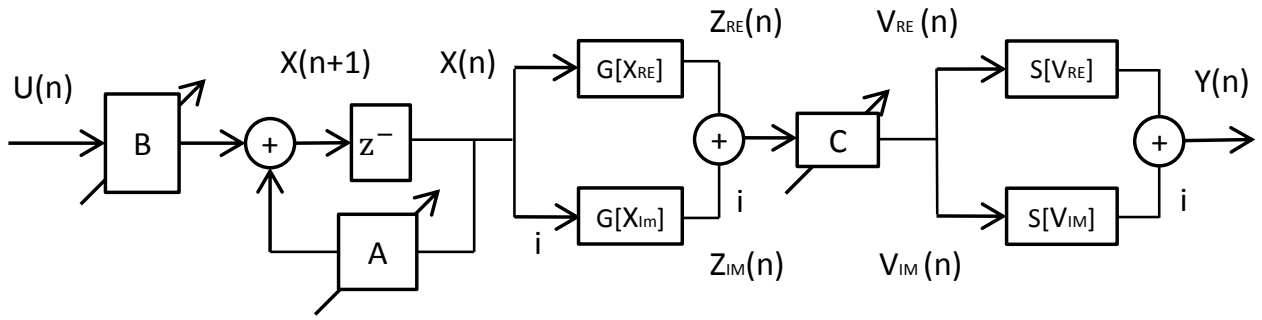


Figura 3.23 Diagrama de bloques de la red neuronal compleja recurrente con la segunda función de activación

De la figura anterior es claro que la parte real e imaginaria son separadas antes de ser evaluadas en las funciones de activación.

Aplicando las reglas diagramáticas a la red neuronal en diagrama de bloques se obtiene la red adjunta, que se presenta en la Figura 3.24.

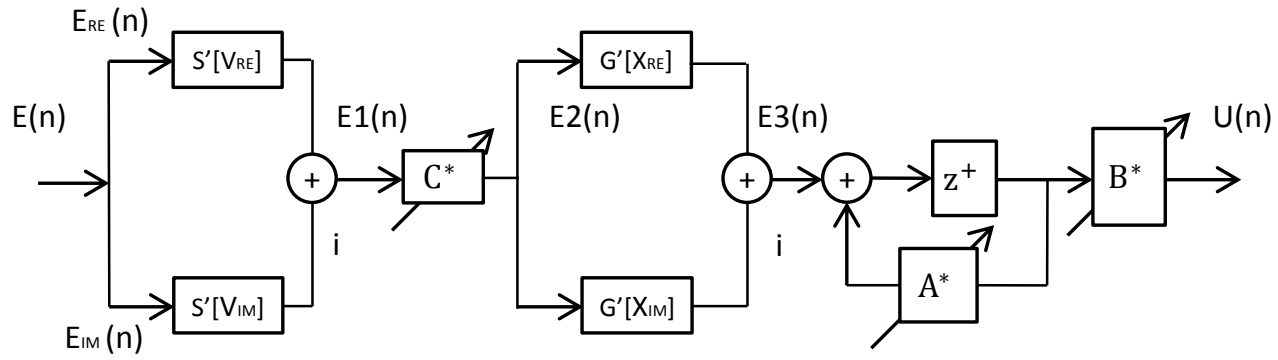


Figura 3.24 Red adjunta de la segunda función de activación

Aplicando la regla de Backpropagation se puede obtener el algoritmo de aprendizaje utilizando la red adjunta.

Para la capa de salida:

$$\Delta C(n) = E_1(n)Z^*(n) \tag{3.114}$$

$$E_1(n) = \mathrm{E_{Re}S'}\left[Y[V_{Re}(n)]\right] + i\mathrm{E_{Im}S'}\left[Y[V_{Im}(n)]\right] \tag{3.115}$$

$$E(n) = T(n) - Y(n) \tag{3.116}$$

Para la capa oculta:

$$\Delta A(n) = E_3(n)X^*(n-1) \tag{3.117}$$

$$E_3(n) = \mathrm{E}_{2\mathrm{Re}}\mathrm{G}'\left[Z[X_{Re}(n)]\right] + i\mathrm{E}_{2\mathrm{Im}}\mathrm{G}'\left[Z[X_{Im}(n)]\right] \tag{3.118}$$

$$E_2(n) = C^*(n)E_1(n) \tag{3.119}$$

$$\Delta \mathrm{v}A(n) = E_3(n) \otimes X^*(n-1) \tag{3.120}$$

$$\Delta \mathrm{B}(n) = E_3(n)U^*(n) \tag{3.121}$$

donde:

$G'[\cdot], S'[\cdot]$: Derivada de la función de activación;

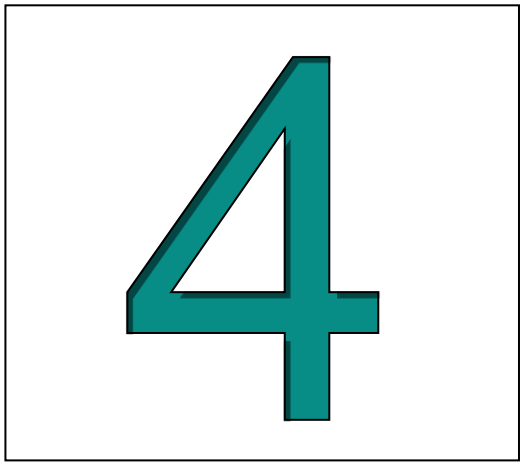
a^*: Transpuesto conjugado de a;

$T(n)$: Vector de salida deseado.

Como se puede ver, la aplicación de las reglas diagramáticas para obtener la red adjunta y posteriormente el algoritmo de aprendizaje es un método simplificado para basado en el gradiente descendente en el dominio complejo.

3.6 Conclusiones

En este capítulo se introdujo el tema de redes neuronales, primeramente en el dominio real, presentando diferentes topologías de redes neuronales, funciones de activación, y algoritmos de aprendizaje, la introducción de las reglas diagramáticas en el dominio real. En la segunda parte del capítulo se abordó el tema principal de la tesis que son las redes neuronales en el dominio complejo, presentando los problemas que éstas exigen, tanto para la selección de la función de activación como para el desarrollo de los algoritmos de aprendizaje. Se realizó una extensión de las reglas diagramáticas en el dominio real al dominio complejo y se dedujeron algoritmos de aprendizaje para redes neuronales complejas recurrentes con dos funciones de activación diferentes.



IDENTIFICACIÓN CON REDES NEURONALES COMPLEJAS RECURRENTES

4. Identificación con Redes Neuronales Complejas Recurrentes

En este capítulo se aborda el problema de identificación de sistemas dinámicos utilizando las redes neuronales complejas con dos funciones de activación diferentes; se presentan las arquitecturas de identificación propuestas en este trabajo y se utilizan para la identificación de dos plantas mecánicas.

4.1 Problema de Identificación

Existen muchos procesos en los cuales es difícil obtener un modelo matemático que describa el comportamiento de dichos procesos, debido a su complejidad, al número de variables que involucran, por esta razón se requiere de algunas técnicas para obtener un modelo matemático que describa el comportamiento de dicho proceso, en las cuales solo se requiere información de entrada y salida del sistema a identificar.

Al proceso para la obtención del modelo que describa la dinámica de un sistema se conoce como identificación. En la identificación, puede ser que sólo se desconozcan los parámetros de algún modelo conocido, o que se desconozca todo el modelo. En el segundo caso es posible obtener un modelo a partir del conocimiento de las entradas y salidas del proceso que se desea modelar, cabe mencionar que aunque el modelo obtenido tiene el mismo comportamiento que el proceso que se desea modelar, puede ser que este modelo no tenga significado físico. Una técnica muy poderosa para la identificación de alguna planta desconocida son las redes neuronales. Como se vio en el capítulo tres las redes neuronales pueden ser entrenadas a partir de entradas y salidas, y el entrenamiento se encarga de ajustar los pesos hasta que se logre una buena aproximación entre la salida de la planta y la salida de la red neuronal.

4.2 Identificación con redes neuronales

La identificación de sistemas no lineales desconocidos utilizando redes neuronales es una herramienta importante que no requiere de un modelo analítico, y se basa en las ventajas de aproximación de las redes neuronales.

El empleo de redes neuronales es una herramienta muy efectiva para identificar sistemas no lineales muy complejos cuando no se tiene suficiente información del modelo de la planta, o cuando se considera a la planta como una caja negra. Los identificadores neuronales pueden clasificarse dependiendo del tipo de red neuronal que se utiliza para su construcción, de aquí que existan identificadores neuronales estáticos y dinámicos. Las redes neuronales dinámicas tienen la capacidad de representar sistemas no lineales y un buen comportamiento en presencia de dinámicas no modeladas gracias a que su estructura incorpora realimentación [38].

El procedimiento para entrenar una red neuronal con la finalidad de representar la dinámica de la planta se le conoce como modelado directo [1]. La estructura para obtener este modelo se ilustra en la Figura 4.1.

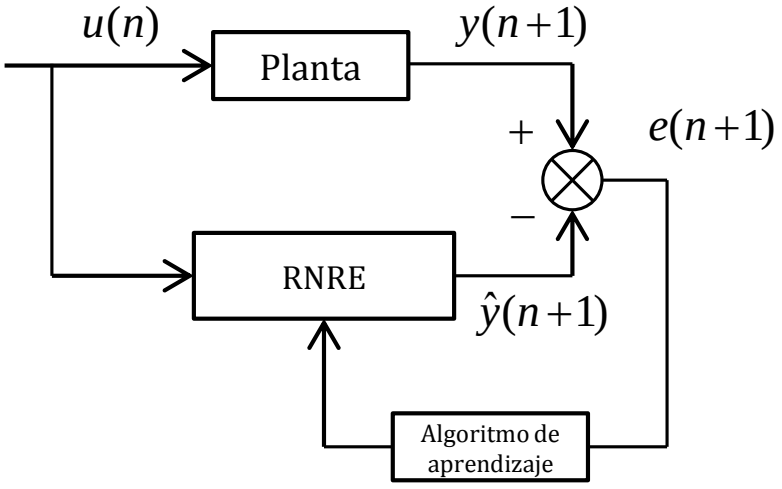


Figura 4.1 Esquema de identificación con la RNRE modular.

En este esquema de identificación la red neuronal se conecta en paralelo con la planta, el error de identificación es la señal que se utiliza en el algoritmo de aprendizaje de la red para ajustar los pesos. Esto se hace con la finalidad de que la salida de la red aproxime a la salida de la planta, minimizando el error medio cuadrático (3.72).

Una de las ventajas del enfoque de identificación de sistemas desconocidos, es que las incertidumbres pueden considerarse dentro de la dinámica de la red neuronal, de tal manera que el algoritmo sin modificación alguna puede tener propiedades de robustez ante variaciones en la dinámica de la planta e incertidumbres.

4.3 Identificación en el caso complejo

El problema de identificación para el caso complejo es posible debido a las siguientes razones. Primeramente, se debe de establecer una función de costo (3.72), que en el caso complejo debe de ser analítica, es decir, su derivada debe estar definida. Por otro lado, la minimización de una función evaluada en el plano complejo no puede ser definida debido a que las relaciones de "mayor que" y "menor que" no son válidas. Es debido a esto que se recurre al llamado cálculo de Wirtinger para dar solución a al problema de minimización de una función de costo compleja.

La función de costo a considerar es la propuesta en (3.72), que es una función que realiza un mapeo del plano complejo al conjunto de los números reales, la expresión de la función de costo compleja es la siguiente:

$$\zeta(n) = \frac{1}{2}\sum_{j}\left[E_j(n)\right]\left[\overline{E_j(n)}\right], \quad j \in \mathrm{C}, \quad \zeta = \frac{1}{N_e}\sum_{j}\zeta(n)$$

Haciendo uso del cálculo de Wirtinger se pueden obtener las derivadas de la función de costo compleja considerada. Sin embargo, en este trabajo se utilizaron las reglas diagramáticas en su caso complejo para la obtención de algoritmos de aprendizaje basados en la minimización de la función de costo compleja.

A continuación, se presentan los resultados de simulación del problema de identificación en dos plantas mecánicas, utilizando una red neuronal compleja recurrente con dos funciones de activación diferentes. Primero se define la dinámica de cada una de las plantas a identificar y después se detalla la red neuronal compleja que se utilizó.

Los resultados de simulación se dividen en dos partes: aprendizaje y generalización. El primero corresponde a la etapa en la que los pesos sinápticos son aproximados a los valores óptimos tal que la función de costo sea mínima; una vez que la etapa de aprendizaje finaliza y los pesos sinápticos convergen, éstos se fijan, se cambia la señal de entrada y se verifica que la red neu-

ronal compleja recurrente hay efectivamente aprendido la dinámica de la planta, a esta etapa se le llama generalización.

Como medida para evaluar el desempeño del aprendizaje y de la generalización se utilizó el Error Medio Cuadrático (EMC), al término de las simulaciones se presenta una tabla con la comparativa entre ambas funciones de activación.

El esquema de identificación que se utilizó para todas las simulaciones se muestra en la Figura 4.2.

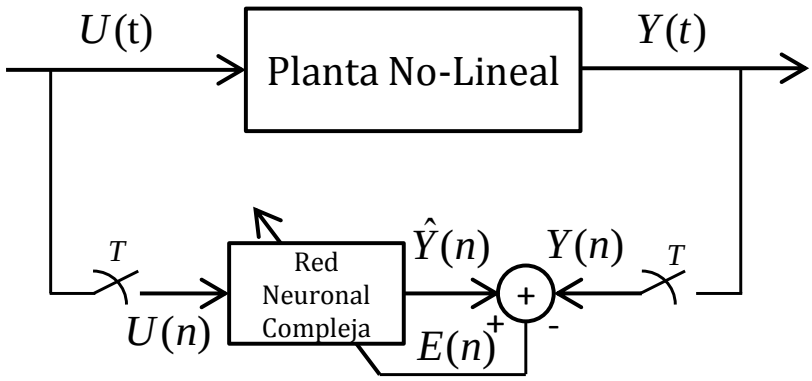


Figura 4.2 Diagrama de bloques del modelo de identificación utilizado

El modelo de identificación está en tiempo discreto y en tiempo continuo debido a la naturaleza de la planta, por lo que se muestrea la señal de entrada y de salida de la planta con el objetivo de utilizar la red neuronal compleja recurrente en tiempo discreto.

4.4 Identificación de un robot flexible de un grado de libertad

Descripción de la planta No-lineal.

El sistema a identificar es un modelo no lineal idealizado de un robot con articulación flexible, en la Figura 4.3. La flexibilidad en la articulación de un robot es generalmente al uso de motores del tipo *harmonic drive*, que es un tipo de motor con un mecanismo de engranes para la transmisión del par y de tamaño compacto.

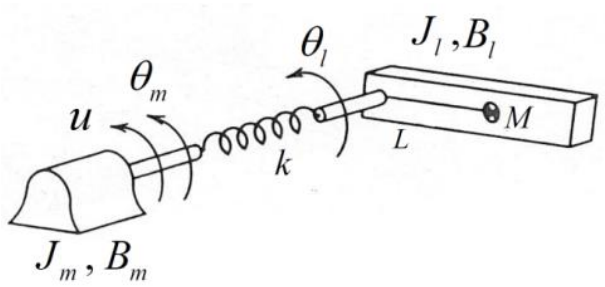


Figura 4.3 Modelo idealizado de un robot con articulación flexible

El modelo idealizado del robot con articulación flexible consiste en un actuador conectado a un eslabón a través de un resorte torsional con el objetivo de representar la flexibilidad de la articulación. Se toma el par del motor como la entrada u. Las ecuaciones que describen la dinámica de este sistema son las siguientes:

$$\begin{aligned} J_l\ddot{\theta}_l + B_l\dot{\theta}_l + Mgl\sin\theta_l + k(\theta_l - \theta_m) &= 0 \\ J_m\ddot{\theta}_m + B_m\dot{\theta}_m - k(\theta_l - \theta_m) &= u \end{aligned} \tag{4.1}$$

donde: J_l, J_m son las inercias del eslabón y del motor respectivamente; B_l y B_m son las constantes de amortiguamiento del eslabón y del motor respectivamente; u es el par de entrada aplicada a la flecha del motor; M y L son las masa del eslabón y la distancia entre la flecha del motor al el centro de masa del eslabón respectivamente; k representa la constante de rigidez del resorte torsional en el mecanismo del motor harmonic drive.

El sistema dinámico es oscilatorio, descrito por dos ecuaciones diferenciales de segundo orden, teniendo un sistema de dos grados de libertad con una sola entrada, por lo que se considera un sistema subactuado.

Las señales de entrada para la etapa de aprendizaje $u_l(n)$ y para la etapa de generalización $u_g(n)$ son las siguientes:

$$u_l(t) = \sin(t/10) + 0.5\sin(2t/50) \tag{4.2}$$

$$u_g(t) = 0.5\sin(t/10) + 0.8\sin(t/30) \tag{4.3}$$

4.4.1 Resultados de Simulación

Resultados con la primera función de activación. Etapa de aprendizaje.

Los resultados de la identificación de la planta utilizando la primera función de activación se muestran en la Figura 4.4; se muestra la salida de la red neuronal compleja recurrente y la salida de la planta.

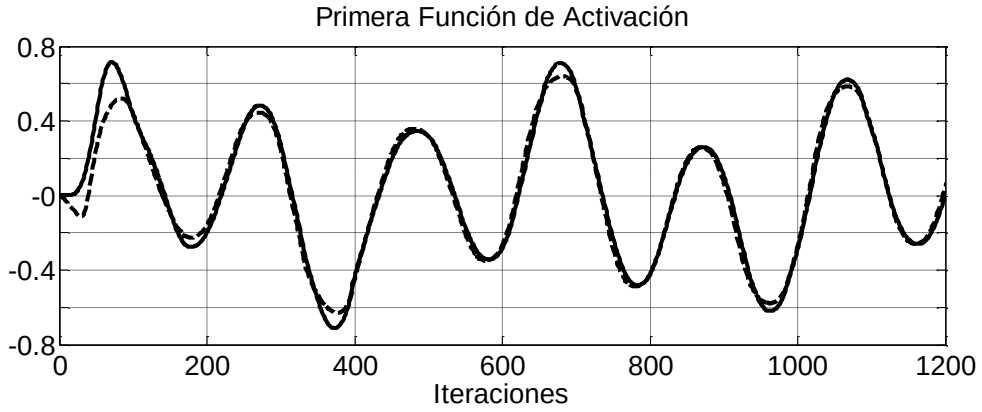


Figura 4.4 Resultados de simulación con la primera función de activación utilizando Backpropagation en la etapa de aprendizaje; La salida de la red neuronal (línea discontinua) y la salida de planta (línea continua).

Resultados con la primera función de activación. Etapa de generalización.

Los resultados de la etapa de generalización son mostrados en la Figura 4.5; en esta etapa los valores a los cuales convergieron los pesos sinápticos en la etapa de entrenamiento son fijados y la señal de entrada es cambiada, esto con el fin de verificar que la red neuronal haya aprendido la dinámica de la planta. La salida de la red neuronal compleja recurrente es comparada con la salida de la planta.

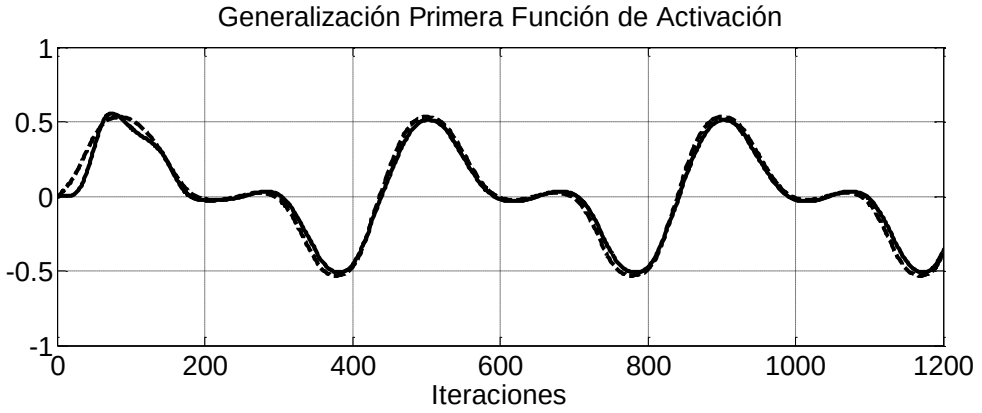


Figura 4.5 Resultados de simulación con la primera función de activación utilizando Backpropagation en la etapa de generalización; La salida de la red neuronal (línea discontinua) y la salida de planta (línea continua).

Resultados con la segunda función de activación. Etapa de aprendizaje.

Los resultados de la identificación de la planta utilizando la segunda función de activación se muestran en la Figura 4.6; se muestra la salida de la red neuronal compleja recurrente y la salida de la planta.

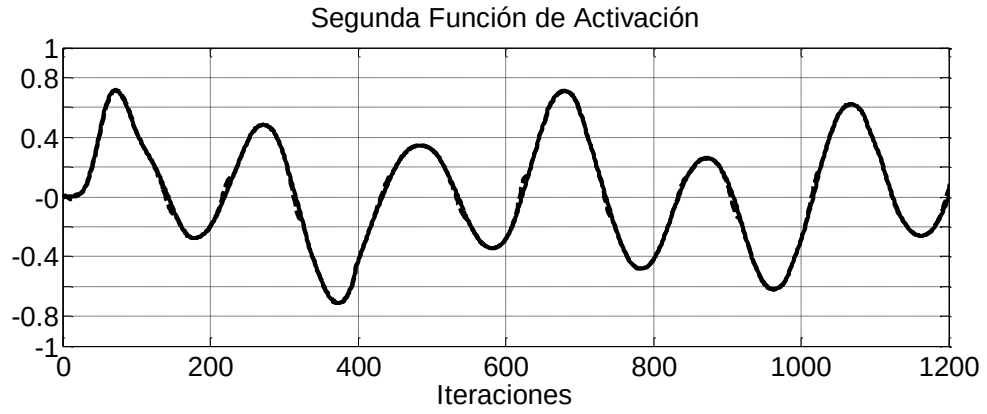


Figura 4.6 Resultados de simulación con la segunda función de activación utilizando Backpropagation en la etapa de aprendizaje; La salida de la red neuronal (línea discontinua) y la salida de planta (línea continua).

Resultados con la segunda función de activación. Etapa de generalización.

Los resultados de la etapa de generalización son mostrados en la Figura 4.7; en esta etapa los valores a los cuales convergieron los pesos sinápticos en la etapa de entrenamiento son fijados y la señal de entrada es cambiada, esto con el fin de verificar que la red neuronal haya aprendido la dinámica de la planta. La salida de la red neuronal compleja recurrente es comparada con la salida de la planta.

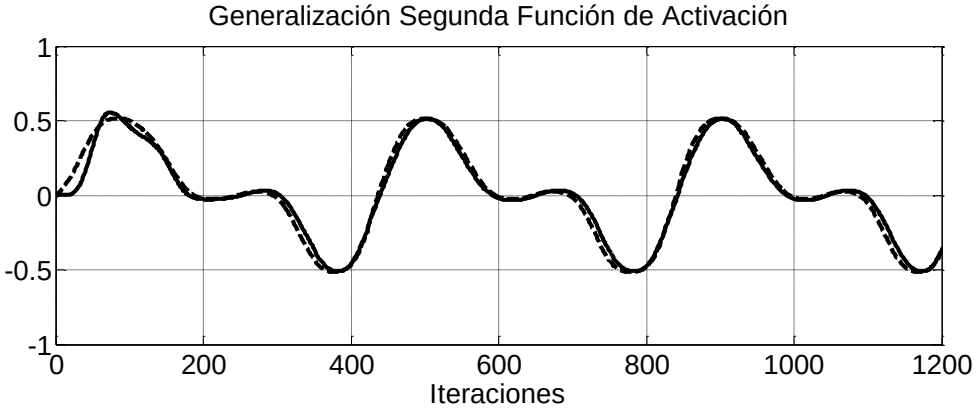


Figura 4.7 Resultados de simulación con la segunda función de activación utilizando Backpropagation en la etapa de generalización; La salida de la red neuronal (línea discontinua) y la salida de planta (línea continua).

Comparación final entre EMC de ambas funciones de activación, para la etapa de aprendizaje y generalización.

Los valores finales de EMC obtenidos durante las simulaciones utilizando ambas funciones de activación y para ambas etapas se muestran en la Tabla 4.1, se puede ver que la segunda función de activación logra tener un EMC menor en la etapa de aprendizaje, sin embargo, en la etapa de generalización se puede ver que la primera función de activación presenta un EMC menor.

Tabla 4.1 Valores de EMC para la tarea de identificación con ambas funciones de activación

Etapa de aprendizaje		
Número de paso	Primera función de activación	Segunda función de activación
1000	0.0264	0.0045
2000	0.0178	0.0029
3000	0.0097	0.0028
Etapa de Generalización		
Número de paso	Primera función de activación	Segunda función de activación
1000	0.0031	0.0048
2000	0.0026	0.0032
3000	0.0018	0.0021

De la tabla anterior se concluye que es mejor utilizar la segunda función de activación en tareas de identificación en línea, ya que al no presentar singularidades, es más confiable el proceso de identificación que utilizando la primera función de activación.

4.5 Identificación de un robot flexible de dos grados de libertad

Descripción de la planta No-lineal.

Es el caso multivariable del modelo del robot con articulación flexible, cada una de sus articulaciones son flexibles, lo que provoca que los grados de libertad totales del sistema sea $2n$

siendo n el numero de articulaciones. La articulación i y $i-1$ son modeladas como se muestra en la Figura 4.8.

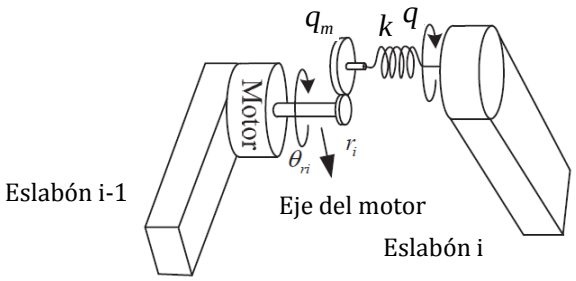


Figura 4.8 Modelo idealizado de la junta flexible, representando la articulación i.

El sistema en general es un robot de dos grados de libertad como se muestra en la Figura 4.9; cada una de sus articulación son del tipo que se muestra en la Figura 4.8.

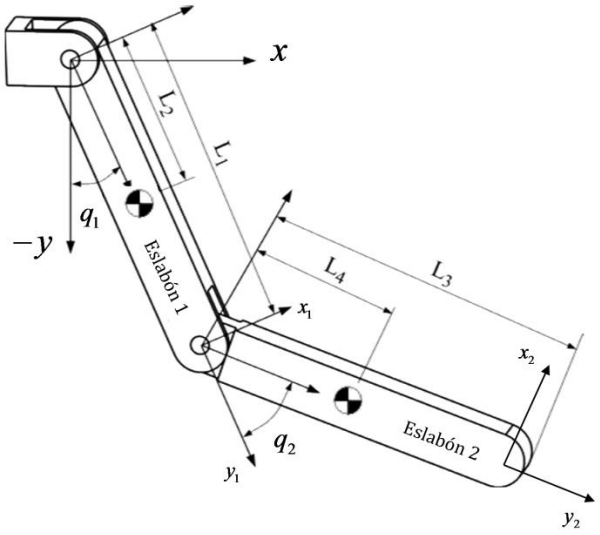


Figura 4.9 Modelo idealizado de un robot de dos grados de libertad con junta flexible.

Con el fin de simplificar el modelo del robot flexible se consideran las siguientes suposiciones, [39]:

- La energía cinética del rotor es debida a su propia rotación. De la misma manera, el movimiento del rotor es rotación pura con respecto al marco inercial.

- La inercia del rotor es simétrica con respecto al eje de rotación, por o tanto, la energía potencial del sistema y la velocidad del centro de masa del rotor son ambas independientes de la posición del rotor.

Bajo estas condiciones, las ecuaciones que describen la dinámica del robot de dos grados de libertad con articulación flexible son las siguientes:

$$\begin{aligned} D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + K(q - q_m) &= 0 \\ J\ddot{q_m} - K(q - q_m) &= u \end{aligned} \tag{4.4}$$

donde: q y $q_m \in \mathbb{R}^n$ denotan el desplazamiento angular de los eslabones y del motor, respectivamente; $D(q)\colon \mathbb{R}^n \to \mathbb{R}^{nxn}$ es la matriz de inercia, simétrica definida positiva; $C(q,\dot{q})\colon \mathbb{R}^n x \mathbb{R}^n \to \mathbb{R}^n$ representa las fuerzas de Coriolis; $G(q)\colon \mathbb{R}^n \to \mathbb{R}^n$ es el vector de fuerzas gravitacionales de los eslabones; $K \epsilon \mathbb{R}^{nxn}$ es una matriz diagonal positiva definida con las constantes de rigidez de los resortes de las juntas flexibles; $J \epsilon \mathbb{R}^{nxn}$ es la matriz de momentos de inercia de los motores, y $u \in \mathbb{R}^n$ es el vector de entradas de los pares de los motores.

Si se define un vector extendido $q_f = [q^T q_m^T]^T$, se pueden reescribir las dos ecuaciones en la siguiente forma matricial:

$$D_f(q)\ddot{q_f} + C_f(q,\dot{q})\dot{q_f} + G_f(q) + K_f q_f = u_f \tag{4.5}$$

donde:

$$D_f(q) = \begin{bmatrix} D(q) & 0 \\ 0 & J \end{bmatrix}, \qquad C_f(q,\dot{q}) = \begin{bmatrix} C(q,\dot{q}) & 0 \\ 0 & 0 \end{bmatrix}$$

$$G_f(q) = \begin{bmatrix} G(q) \\ 0 \end{bmatrix}, \qquad K_f = \begin{bmatrix} K & -K \\ -K & K \end{bmatrix}, \qquad u_f = \begin{bmatrix} 0 \\ I \end{bmatrix} u$$

Las propiedades del modelo son mencionadas en [39]. Como se puede ver, la planta es un sistema oscilatorio, descrito por cuatro ecuaciones diferenciales de segundo orden, de esto, que el sistema tiene 4 grados de libertad pero solo dos entradas de control, por lo que se considera un sistema subactuado.

4.5.1 Resultados de Simulación

Resultados con la primera función de activación. Etapa de aprendizaje.

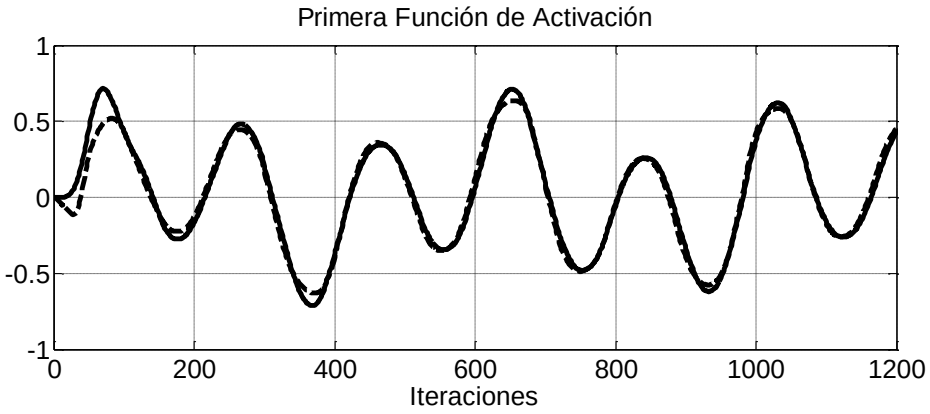


Figura 4.10 Resultados de simulación con la primera función de activación utilizando Backpropagation en la etapa de aprendizaje; La salida de la red neuronal (línea discontinua) y la salida de la planta (línea continua), (Caso multivariable)

Resultados con la primera función de activación. Etapa de generalización.

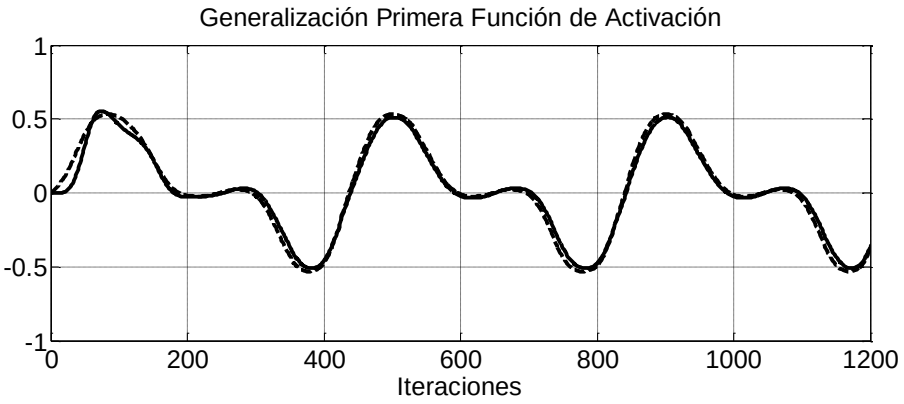


Figura 4.11 Resultados de simulación con la primera función de activación utilizando Backpropagation en la etapa de generalización; La salida de la red neuronal (línea discontinua) y la salida de la planta (línea continua), (Caso multivariable)

Resultados con la segunda función de activación. Etapa de aprendizaje.

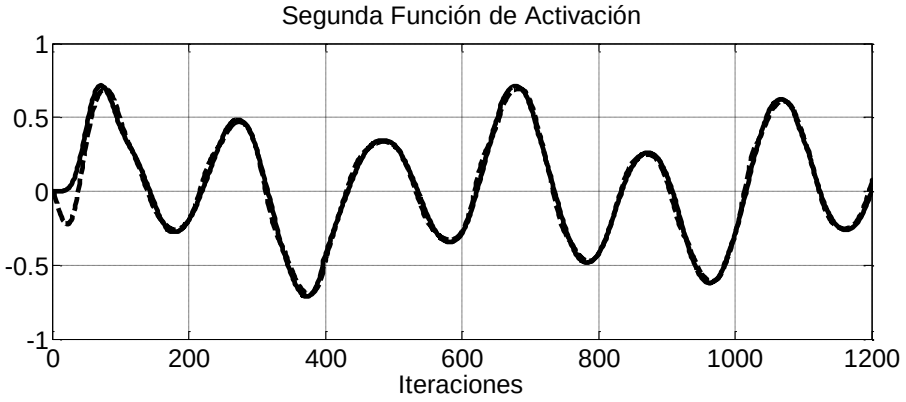


Figura 4.12 Resultados de simulación con la segunda función de activación utilizando Backpropagation en la etapa de aprendizaje; La salida de la red neuronal (línea discontinua) y la salida de la planta (línea continua), (Caso multivariable)

Resultados con la segunda función de activación. Etapa de generalización.

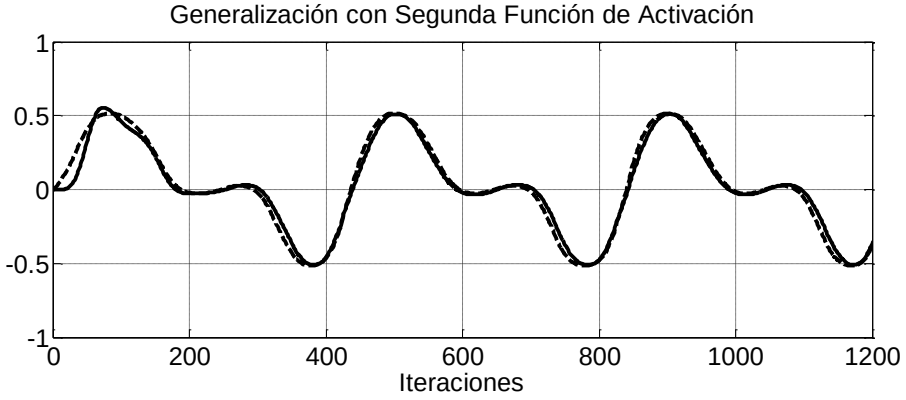


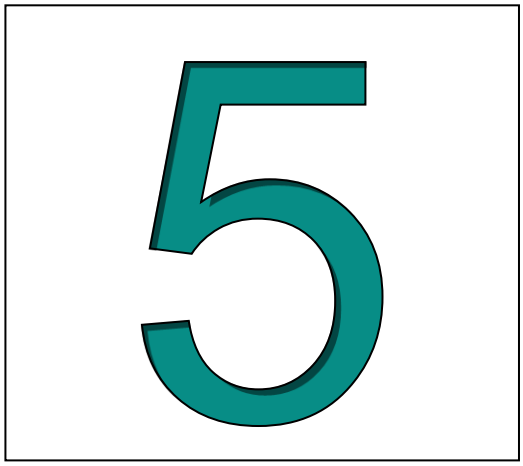
Figura 4.13 Resultados de simulación con la segunda función de activación utilizando Backpropagation en la etapa de generalización; La salida de la red neuronal (línea discontinua) y la salida de la planta (línea continua), (Caso multivariable)

Tabla 4.2 Valores de EMC para la tarea de identificación con ambas funciones de activación, caso multivariable

Etapa de aprendizaje		
Número de paso	Primera función de activación	Segunda función de activación
1000	0.0251	0.0126
2000	0.0121	0.0069
3000	0.0105	0.0048
Etapa de Generalización		
Número de paso	Primera función de activación	Segunda función de activación
1000	0.0056	0.0048
2000	0.0038	0.0023
3000	0.0024	0.0015

4.6 Conclusiones

En el presente trabajo se utilizaron las reglas diagramáticas para el caso complejo y se aplicaron para obtener el algoritmo de entrenamiento basado en Backpropagation. Los resultados de la simulación que se compararon entre las dos etapas de simulación identificación y la generalización de una planta oscilatoria no lineal confirman la calidad del algoritmo de aprendizaje. La comparación del desempeño de ambas funciones de activación, se concluye que es mejor utilizar funciones de activación sin puntos singulares que utilizar funciones de activación con puntos singulares y tratar de evitarlos.



CONTROL CON REDES NEURONALES COMPLEJAS RECURRENTES

5. Control con Redes Neuronales Complejas Recurrentes

En este capítulo se presentan las arquitecturas de control utilizando redes neuronales complejas con dos tipos de funciones de activación; estos esquemas de control fueron implementados en los modelos dinámicos descritos en el capítulo anterior.

5.1 Diseño de Sistemas de Control Neuronal Adaptable

Un sistema de control adaptable es aquel en el que en forma continua o discreta y automática mide características dinámicas de la planta, las compara con las características de la dinámica deseada y usa la diferencia para variar los parámetros ajustables del sistema o generar una señal actuante de modo que se mantenga un desempeño óptimo, independientemente de las condiciones en su entorno, [40], [41].

Un esquema de control que utilice redes neuronales por definición es un control adaptable debido a que la red neuronal siempre ajusta sus pesos tratando de minimizar el índice de desempeño. Recientemente las redes neuronales se han dado a conocer como una poderosa herramienta en el aprendizaje de la dinámica de sistemas no lineales. Debido a su masivo paralelismo, rápida adaptación e inherente capacidad de aproximación, [23].

Se utilizaron dos plantas no lineales oscilatorias de uno y dos grados de libertad, descritas por las ecuaciones (4.1) y (4.5), respectivamente; ambos modelos son expresados en ecuaciones diferenciales de segundo orden en tiempo continuo, por lo que se requiere muestrear las señales de entrada y salida de la planta con un periodo T.

Las redes neuronales utilizadas son redes neuronales complejas recurrentes y fueron entrenadas utilizando la metodología descrita en este trabajo. Las dimensiones de cada red son definidas en cada uno de los esquemas de control adaptable.

5.2 Control Neuronal Directo con Realimentación de Estados

El esquema de control adaptable se muestra en el diagrama de bloques en la Figura 5.1. El esquema contiene tres redes neuronales complejas: RCVNN-1 es una red neuronal es dedicada a la identificación de la planta y a la estimación de los estados; RCVNN-2 es un control neuronal directo; RCVNN-3 es un control neuronal realimentado que tiene como entradas los estados estimados por la primera red neuronal compleja recurrente. Para la identificación del sistema, el vector deseado es la salida de la planta. El objetivo de la identificación es ajustar los pesos complejos de la red neuronal de tal manera que la red neuronal compleja recurrente siga a la planta con un EMC mínimo.

La RCVNN-1 es el identificador de la planta y estimador de estados, esta red es entrenada con el error de identificación: $E_i = Y_p - Y$. La RCVNN-2 y RCVNN-3 son la realimentación y el control neuronal directo, respectivamente, ambas son entrenadas con el error de control: $E_c = R - Y_p$.

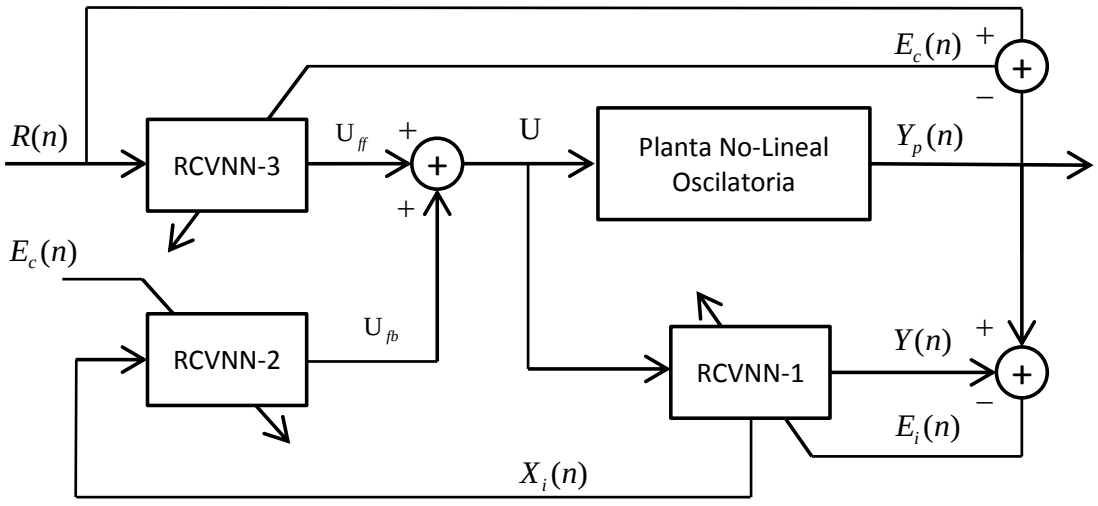


Figura 5.1 Diagrama de bloques del esquema de control adaptable directo con realimentación de estado

El vector de control es la suma de ambas acciones de control U_{fb}, U_{ff} (salidas que corresponden a las salidas de las redes neuronales complejas).

5.2.1 Robot Flexible de Un Grado de Libertad

Para este caso, se alimenta al sistema de control con una referencia constante, y se espera que el eslabón converja a ella en tiempo finito, teniendo como objetivo reducir el EMC producido por el error de control.

5.2.1.1 Resultados de simulación

Las simulaciones se realizaron con ambas funciones de activación (3.68) y (3.69); se presentan resultados de la etapa de aprendizaje y de generalización. Al final se realiza una comparación de desempeño de ambas funciones de activación.

Primera Función de Activación.

La simulación se ejecutó con las siguientes configuraciones: RCVNN-1 tiene dimensiones: $m = 1,\ n = 2,\ p = 1$; el número de las neuronas ocultas de RCVNN-1 fue elegido a prueba y error; RCVNN-2 tiene dimensiones de $m = 2,\ n = 4,\ p = 1$, el número de las neuronas ocultas de RCVNN-2 fue elegido a prueba y error; RCVNN-3 tiene dimensiones de $m = 1,\ n = 2,\ p = 1$, el número de las neuronas ocultas de RCVNN-3 fue elegido a prueba y error.

Los resultados gráficos de las salidas de la planta de las etapas de aprendizaje y generalización se muestran en la siguiente figura; en la Figura 5.2 para la etapa de aprendizaje (a) y para la etapa de generalización (b); la salida de la planta se compara con una referencia constante; la referencia es diferente para cada una de las etapas.

Se puede observar que la salida de la planta tiende a converger a la referencia bastante lento, y presenta un sobretiro en las primeras iteraciones; en la parte de generalización, converge en un tiempo menor, sin embargo, la respuesta es muy lenta, más adelante se mostrará que se puede hacer converger la respuesta del sistema de una manera más rápida utilizando un término integral.

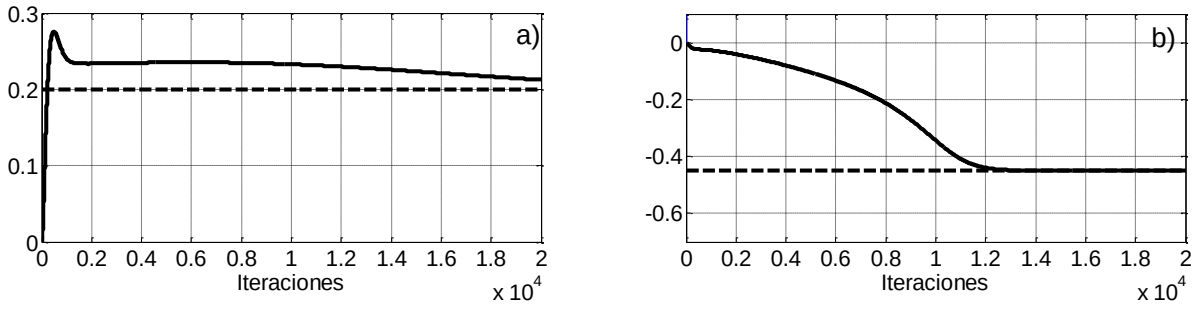


Figura 5.2 Resultados de simulación para la etapa de aprendizaje (a) y generalización (b) con el primer esquema de control utilizando la primera función de activación y la planta no-lineal de un grado de libertad; Señal de referencia (línea discontinua) y la salida de planta (línea continua).

Segunda Función de Activación.

La simulación se ejecutó con las siguientes configuraciones: RCVNN-1 tiene dimensiones: $m = 1,\ n = 2,\ p = 1$; el número de las neuronas ocultas de RCVNN-1 fue elegido a prueba y error; RCVNN-2 tiene dimensiones de $m = 2,\ n = 4,\ p = 1$, el número de las neuronas ocultas de RCVNN-2 fue elegido a prueba y error; RCVNN-3 tiene dimensiones de $m = 1,\ n = 2,\ p = 1$, el número de las neuronas ocultas de RCVNN-3 fue elegido a prueba y error.

Los resultados gráficos de las salidas de la planta de las etapas de aprendizaje y generalización se muestran en la siguiente figura; en la Figura 5.3 para la etapa de aprendizaje (a) y para la etapa de generalización (b); la salida de la planta se compara con una referencia constante; la referencia es diferente para cada una de las etapas.

Se puede observar que la salida de la planta tiende a converger a la referencia bastante más lento que utilizando la primera función de activación, presenta un sobretiro en las primeras iteraciones; en la parte de generalización, converge en un tiempo menor, sin embargo, con un error en estado estacionario muy grande, en este caso resulta superior la utilización de la primera función de activación.

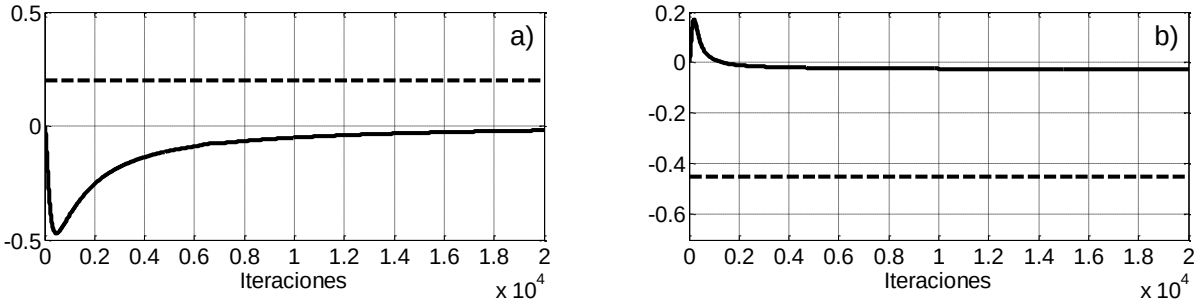


Figura 5.3 Resultados de simulación para la etapa de aprendizaje (a) y generalización (b) con el primer esquema de control utilizando la segunda función de activación y la planta no-lineal de un grado de libertad; Señal de referencia (línea discontinua) y la salida de planta (línea continua).

En la Figura 5.4, se pueden ver los resultados de identificación con ambas funciones activación.

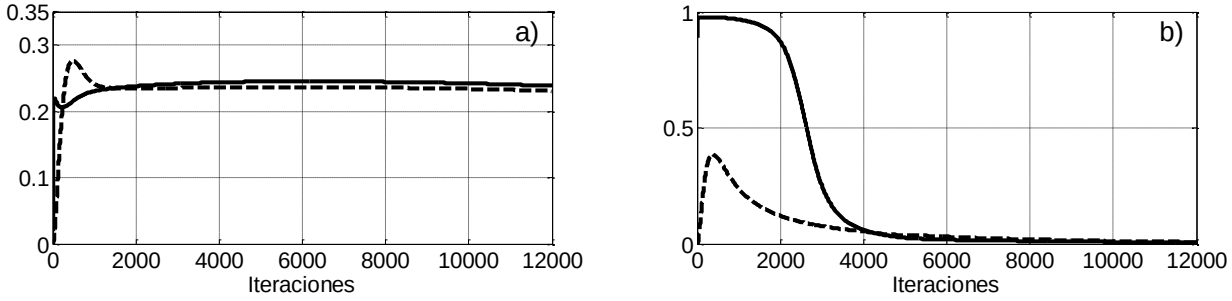


Figura 5.4 Resultados de simulación para la etapa de identificación utilizando la primera (a) y la segunda función de activación (b) con el primer esquema de control y la planta no-lineal de un grado de libertad; Señal de la planta referencia (línea discontinua) y la salida de la red neuronal de identificación (línea continua).

5.2.1.2 Comparación del EMC entre ambas funciones de activación

Los valores obtenidos de EMC al final de la simulación (20,000 pasos de iteración) con ambas funciones de activación para el primer esquema de control neuronal adaptable se muestran en la Tabla 5.1. Se puede observar que la primera función de activación presenta un mejor desempeño en comparación con la segunda función de activación, tanto para la etapa de aprendizaje como para la generalización.

Tabla 5.1 EMC para ambas funciones de activación y el primer esquema de control.

Etapa de aprendizaje		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0018	0.1400
15000	0.0015	0.1124
20000	0.0012	0.0690
Etapa de Generalización		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.1125	0.1954
15000	0.0754	0.1906
20000	0.0516	0.1880

5.2.2 Robot Flexible de Dos Grados de Libertad

Para este caso, se alimenta al sistema de control con una referencia constante diferente para cada articulación, y se espera que ambos eslabones converjan a la referencia deseada, reduciendo el EMC producido por el error de control.

5.2.2.1 Resultados de simulación

Las simulaciones se realizaron con ambas funciones de activación; se presentan resultados de la etapa de aprendizaje y de generalización. Al final se realiza una comparación del desempeño de ambas funciones de activación.

Primera Función de Activación.

La simulación se ejecutó con las siguientes configuraciones: RCVNN-1 tiene dimensiones: $m = 2,\ n = 4,\ p = 2$; el número de las neuronas ocultas de RCVNN-1 fue elegido a prueba y error; RCVNN-2 tiene dimensiones de $m = 2,\ n = 6,\ p = 2$, el número de las neuronas ocultas de RCVNN-2 fue elegido a prueba y error; RCVNN-3 tiene dimensiones de $m = 2,\ n = 4,\ p = 2$, el número de las neuronas ocultas de RCVNN-3 fue elegido a prueba y error.

Los resultados gráficos de las salidas de la planta para la parte de aprendizaje y generalización se muestran en la siguiente figura; en la Figura 5.5 se muestran los resultados para la etapa de aprendizaje y generalización para ambas articulaciones q_1 y q_2.

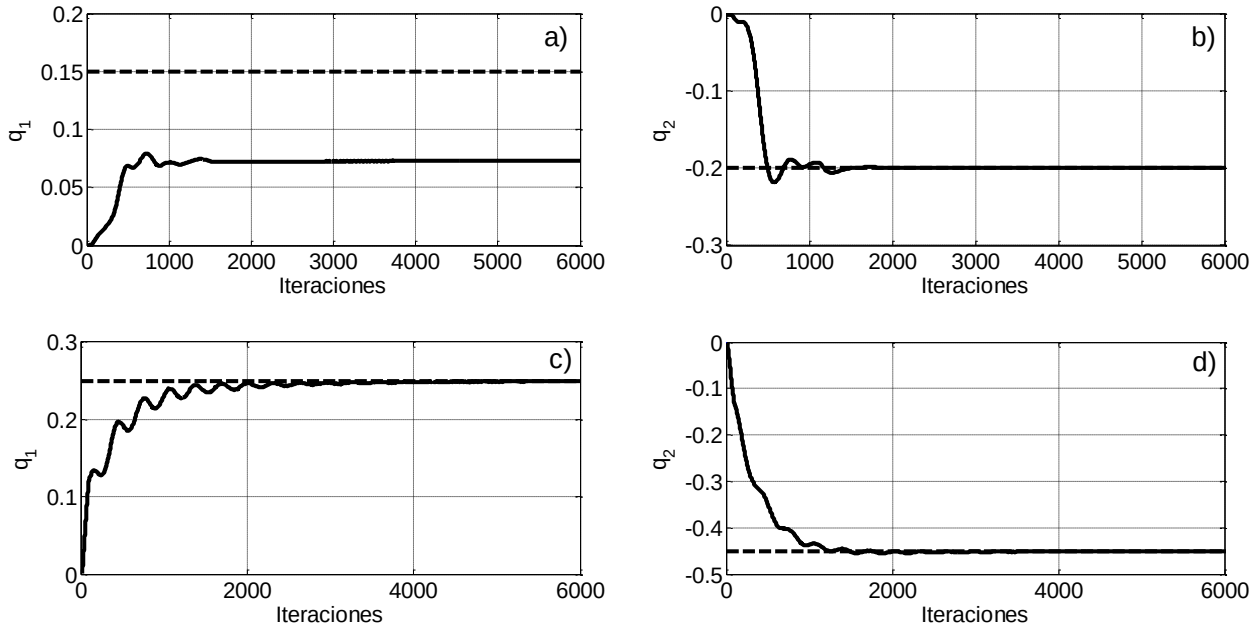


Figura 5.5 Resultados de simulación para la etapa de aprendizaje (a),(b) y generalización (c), (d) con el primer esquema de control utilizando la primera función de activación y la planta no-lineal de dos grados de libertad; Señal de referencia (línea discontinua) y la salida de planta (línea continua).

Segunda Función de Activación.

La simulación se ejecutó con las siguientes configuraciones: RCVNN-1 tiene dimensiones: $m = 2,\ n = 4,\ p = 2$; el número de las neuronas ocultas de RCVNN-1 fue elegido a prueba y error; RCVNN-2 tiene dimensiones de $m = 2,\ n = 6,\ p = 2$, el número de las neuronas ocultas de RCVNN-2 fue elegido a prueba y error; RCVNN-3 tiene dimensiones de $m = 2,\ n = 4,\ p = 2$, el número de las neuronas ocultas de RCVNN-3 fue elegido a prueba y error.

Los resultados gráficos de las salidas de la planta para la parte de aprendizaje y generalización se muestran en la siguiente figura; en la Figura 5.6, se muestran los resultados para la etapa de aprendizaje y generalización para ambas articulaciones q_1 y q_2.

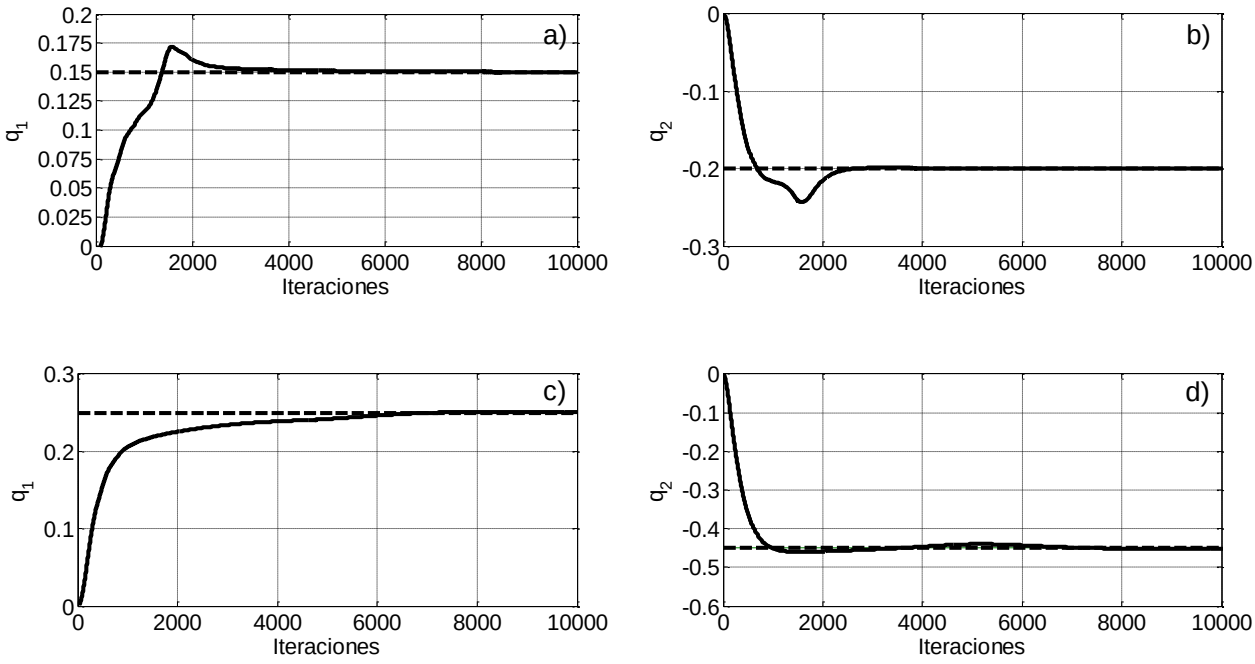


Figura 5.6 Resultados de simulación para la etapa de aprendizaje (a), (b) y generalización (c), (d) con el primer esquema de control utilizando la segunda función de activación y la planta no-lineal de dos grados de libertad; Señal de referencia (línea discontinua) y la salida de planta (línea continua).

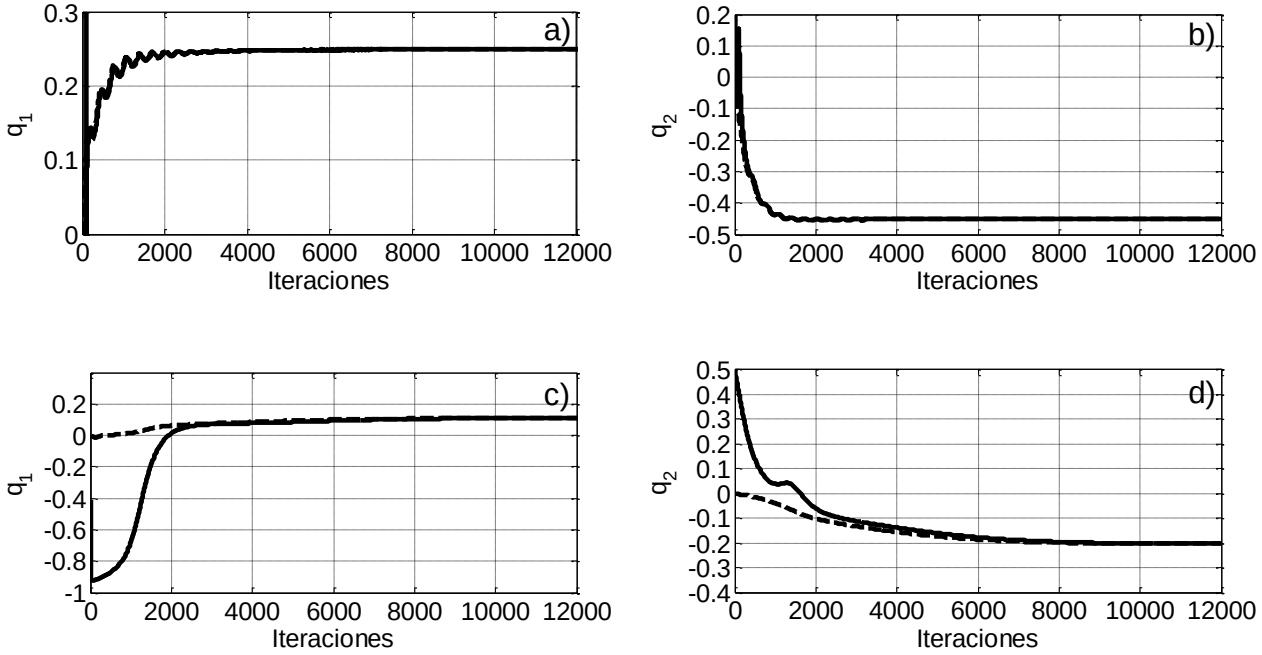


Figura 5.7 Resultados de simulación para la etapa de identificación utilizando la primera (a), (b) y la segunda función de activación (c), (d) con el primer esquema de control y la planta no-lineal de dos grados de libertad; Señal de la planta referencia (línea discontinua) y la salida de la red neuronal de identificación (línea continua).

5.2.2.2 Comparación del EMC entre ambas funciones de activación

Los valores obtenidos de EMC al final de la simulación (20.000 pasos de iteración) con ambas funciones de activación para las dos articulaciones se muestran en la Tabla 5.2 y en la Tabla 5.3. Los resultados son un tanto controvertidos, ya que para la etapa de generalización la primera función de activación tiene un mejor desempeño, sin embargo, la segunda función de activación presenta un mejor desempeño en la etapa de aprendizaje.

Tabla 5.2 EMC de q_1 para ambas funciones de activación y el primer esquema de control.

Etapa de aprendizaje		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0062	0.0059
15000	0.0048	0.0045
20000	0.0042	0.0037
Etapa de Generalización		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0009	0.0020
15000	0.0006	0.0013
20000	0.0004	0.0009

Tabla 5.3 EMC de q_2 para ambas funciones de activación y el primer esquema de control.

Etapa de aprendizaje		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0070	0.0063
15000	0.0051	0.0042
20000	0.0043	0.0031
Etapa de Generalización		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0032	0.0042
15000	0.0021	0.0028
20000	0.0016	0.0020

5.3 Control Neuronal Directo con Realimentación de Estados y Término Integral

El diagrama de bloques del esquema de control se muestra en la Figura 5.8. El esquema contiene tres redes neuronales complejas: una red neuronal es dedicada a la identificación de la planta y a la estimación de los estados; la segunda es un control directo; y la tercera es un control realimentado que tiene como entradas los estados estimados por la primera red neuronal compleja recurrente. Para la identificación del sistema, el vector deseado es la salida de la planta. El objetivo de la identificación es ajustar los pesos complejos de la red neuronal de tal manera que la red neuronal compleja recurrente siga a la planta con un EMC mínimo.

Para eliminar el error en estado estacionario el término integral es agregado, expresado por:

$$V(k+1) = V(k) + T_o k_i E_c \tag{5.1}$$

donde:

T_o: Periodo de muestreo.

k_i: Ganancia de acción integral.

E_c: Error de control.

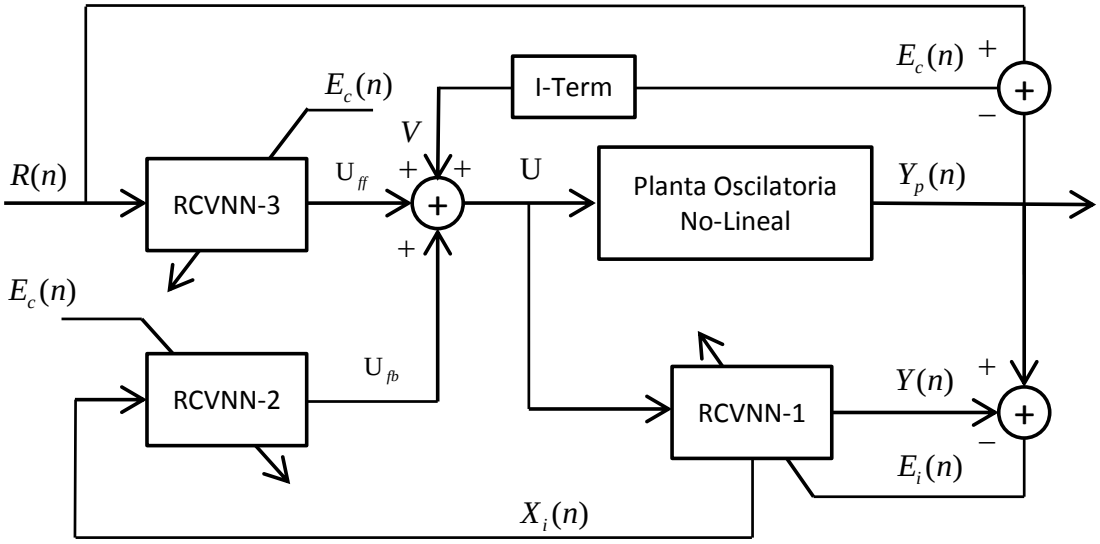


Figura 5.8 Diagrama de bloques del esquema de control adaptable directo con realimentación de estado con término integral.

El vector de control es la suma de ambas acciones de control U_{fb}, U_{ff} (salidas que corresponden a las salidas de las redes neuronales complejas) y V que es el término integral.

5.3.1 Robot Flexible de Un Grado de Libertad

5.3.1.1 Resultados de simulación

Las simulaciones se realizaron con ambas funciones de activación; se presentan resultados de la etapa de aprendizaje y de generalización. Al final se realiza una comparación de desempeño de ambas funciones de activación.

Primera Función de Activación.

La simulación se ejecutó con las siguientes configuraciones: RCVNN-1 tiene dimensiones: $m = 1,\ n = 2,\ p = 1$; el número de las neuronas ocultas de RCVNN-1 fue elegido a prueba y error; RCVNN-2 tiene dimensiones de $m = 2,\ n = 4,\ p = 1$, el número de las neuronas ocultas de RCVNN-2 fue elegido a prueba y error; RCVNN-3 tiene dimensiones de $m = 1,\ n = 2,\ p = 1$, el número de las neuronas ocultas de RCVNN-3 fue elegido a prueba y error, se agregó el término integral, la ganancia del termino integral fue seleccionada a prueba y error.

Los resultados gráficos de las salidas de la planta de las etapas de aprendizaje y generalización se muestran en las siguientes figuras; en la Figura 5.9, se muestran resultados para la etapa de aprendizaje y generalización; la salida de la planta se compara con una referencia constante; la referencia es diferente para cada una de las etapas. Se puede observar que la salida de la planta en la etapa de aprendizaje converge sin sobretiro a diferencia de la etapa de generalización; el tiempo en que convergen ambas etapas es mucho menor a comparación del esquema de control que no implementa el término integral.

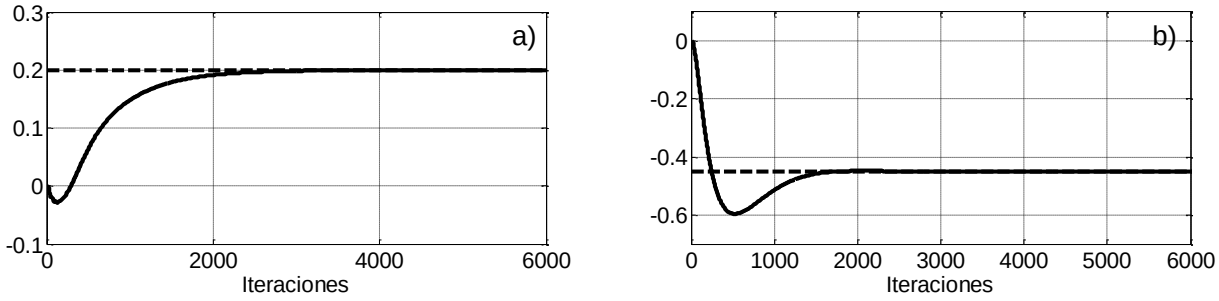


Figura 5.9 Resultados de simulación para la etapa de aprendizaje (a) y generalización (b) con el segundo esquema de control utilizando la primera función de activación y la planta no-lineal de un grado de libertad; Señal de referencia (línea discontinua) y la salida de planta (línea continua).

Segunda Función de Activación.

La simulación se ejecutó con las siguientes configuraciones: RCVNN-1 tiene dimensiones: $m = 1,\ n = 2,\ p = 1$; el número de las neuronas ocultas de RCVNN-1 fue elegido a prueba y error; RCVNN-2 tiene dimensiones de $m = 2,\ n = 4,\ p = 1$, el número de las neuronas ocultas de RCVNN-2 fue elegido a prueba y error; RCVNN-3 tiene dimensiones de $m = 1,\ n = 2,\ p = 1$, el número de las neuronas ocultas de RCVNN-3 fue elegido a prueba y error, se agregó el término integral, la ganancia del termino integral fue seleccionada a prueba y error.

Los resultados gráficos de las salidas de la planta de las etapas de aprendizaje y generalización se muestran en la siguiente figura; en la Figura 5.10, para la etapa de aprendizaje (a) y en la generalización (b); la salida de la planta se compara con una referencia constante; la referencia es diferente para cada una de las etapas.

Se puede observar que la salida de la planta tiende a converger a la referencia bastante más rápido que en el primer esquema de control, al igual que la primera función de activación, se presenta un sobretiro en las primeras iteraciones; de la gráfica Figura 5.10 se concluye que se tiene una buena generalización de las redes neuronales complejas recurrentes.

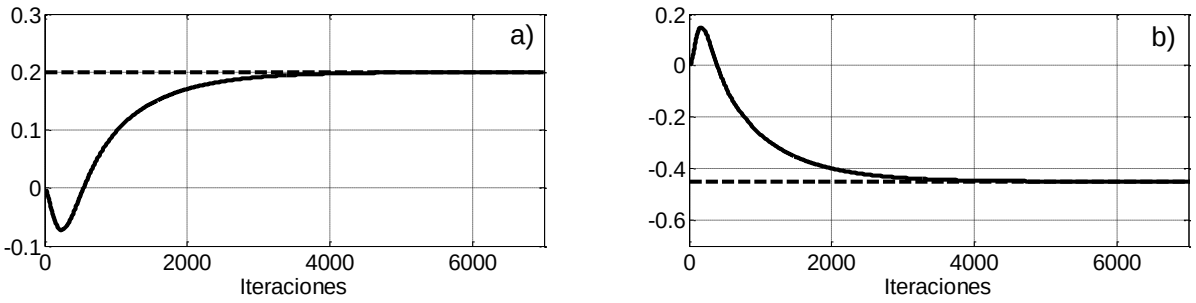


Figura 5.10 Resultados de simulación para la etapa de aprendizaje (a) y generalización (b) con el segundo esquema de control utilizando la segunda función de activación y la planta no-lineal de un grado de libertad; Señal de referencia (línea discontinua) y la salida de planta (línea continua).

En la Figura 5.11, se pueden ver los resultados de identificación con ambas funciones activación.

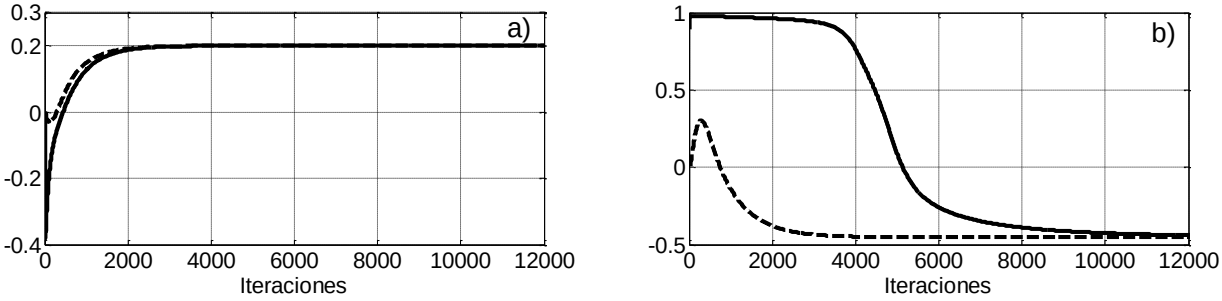


Figura 5.11 Resultados de simulación para la etapa de identificación utilizando la primera (a) y la segunda función de activación (b) con el segundo esquema de control y la planta no-lineal de un grado de libertad; Señal de la planta referencia (línea discontinua) y la salida de la red neuronal de identificación (línea continua).

5.3.1.2 Comparación del EMC entre ambas funciones de activación

Los valores obtenidos de EMC al final de la simulación (20,000 pasos de iteración) con ambas funciones de activación para el segundo esquema de control neuronal adaptable se muestran en la Tabla 5.4. Se puede observar que la primera función de activación presenta un mejor desempeño en comparación con la segunda función de activación, tanto para la etapa de aprendizaje como para la generalización.

Tabla 5.4 EMC para ambas funciones de activación y el segundo esquema de control.

Etapa de aprendizaje		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0025	0.0047
15000	0.0016	0.0031
20000	0.0012	0.0023
Etapa de Generalización		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0030	0.0181
15000	0.0020	0.0121
20000	0.0015	0.0091

De la Tabla 5.4, podemos concluir que el término integral acelera la convergencia de la planta hacia la referencia, y elimina el error en estado estacionario; de la misma manera se concluye que el mejor desempeño tanto para la etapa de aprendizaje como para la etapa de generalización lo tiene la primera función de activación.

5.3.2 Robot Flexible de Dos Grados de Libertad

5.3.2.1 Resultados de simulación

Las simulaciones se realizaron con ambas funciones de activación; se presentan resultados de la etapa de aprendizaje y de generalización. Al final se realiza una comparación de desempeño de ambas funciones de activación.

Primera Función de Activación.

La simulación se ejecutó con las siguientes configuraciones: RCVNN-1 tiene dimensiones: $m = 2,\ n = 4,\ p = 2$; el número de las neuronas ocultas de RCVNN-1 fue elegido a prueba y error; RCVNN-2 tiene dimensiones de $m = 2,\ n = 6,\ p = 2$, el número de las neuronas ocultas de RCVNN-2 fue elegido a prueba y error; RCVNN-3 tiene dimensiones de $m = 2,\ n = 4,\ p = 2$, el número de las neuronas ocultas de RCVNN-3 fue elegido a prueba y error. Las ganancias del término integral fueron elegidas a prueba y error.

Los resultados gráficos de las salidas de la planta para la parte de aprendizaje y generalización se muestran en la siguiente figura; en la Figura 5.12, se muestran los resultados para la etapa de aprendizaje y generalización para ambas articulaciones q_1 y q_2.

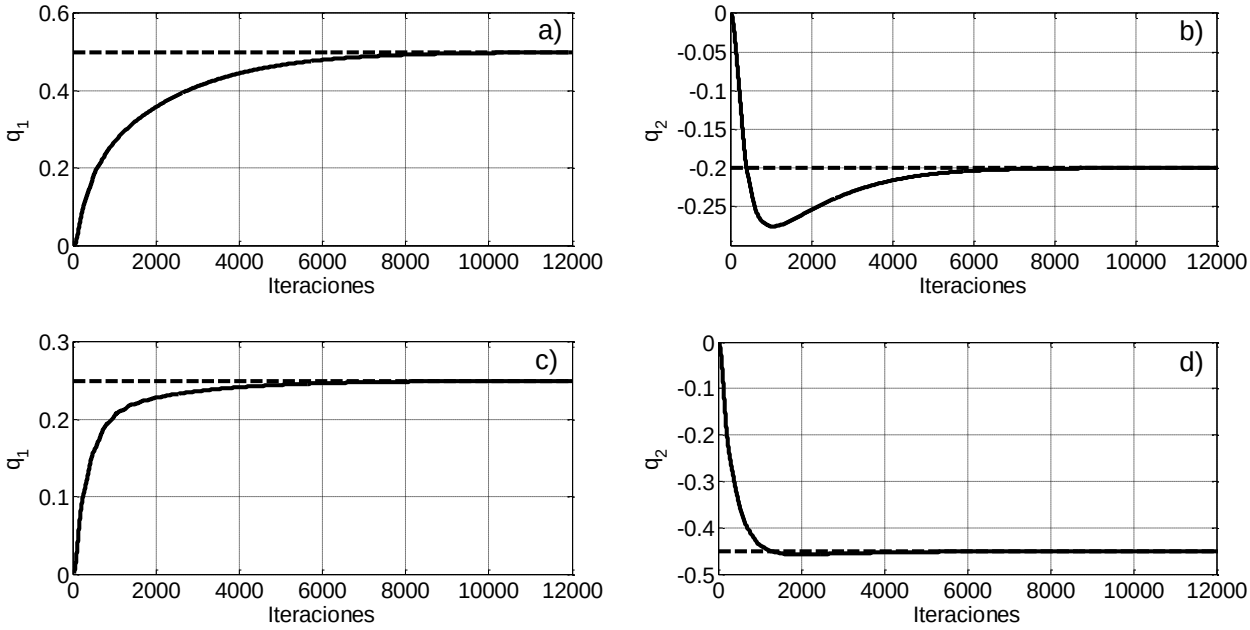


Figura 5.12 Resultados de simulación para la etapa de aprendizaje (a), (b) y generalización (c), (d) con el segundo esquema de control utilizando la primera función de activación y la planta no-lineal de dos grados de libertad; Señal de referencia (línea discontinua) y la salida de planta (línea continua).

Segunda Función de Activación.

La simulación se ejecutó con las siguientes configuraciones: RCVNN-1 tiene dimensiones: $m = 2,\ n = 4,\ p = 2$; el número de las neuronas ocultas de RCVNN-1 fue elegido a prueba y error; RCVNN-2 tiene dimensiones de $m = 2,\ n = 6,\ p = 2$, el número de las neuronas ocultas de RCVNN-2 fue elegido a prueba y error; RCVNN-3 tiene dimensiones de $m = 2,\ n = 4,\ p = 2$, el número de las neuronas ocultas de RCVNN-3 fue elegido a prueba y error. Las ganancias del término integral fueron elegidas a prueba y error.

Los resultados gráficos de las salidas de la planta para la parte de aprendizaje y generalización se muestran en la siguiente figura; en la Figura 5.13, se muestran los resultados para la etapa de aprendizaje y generalización para ambas articulaciones q_1 y q_2.

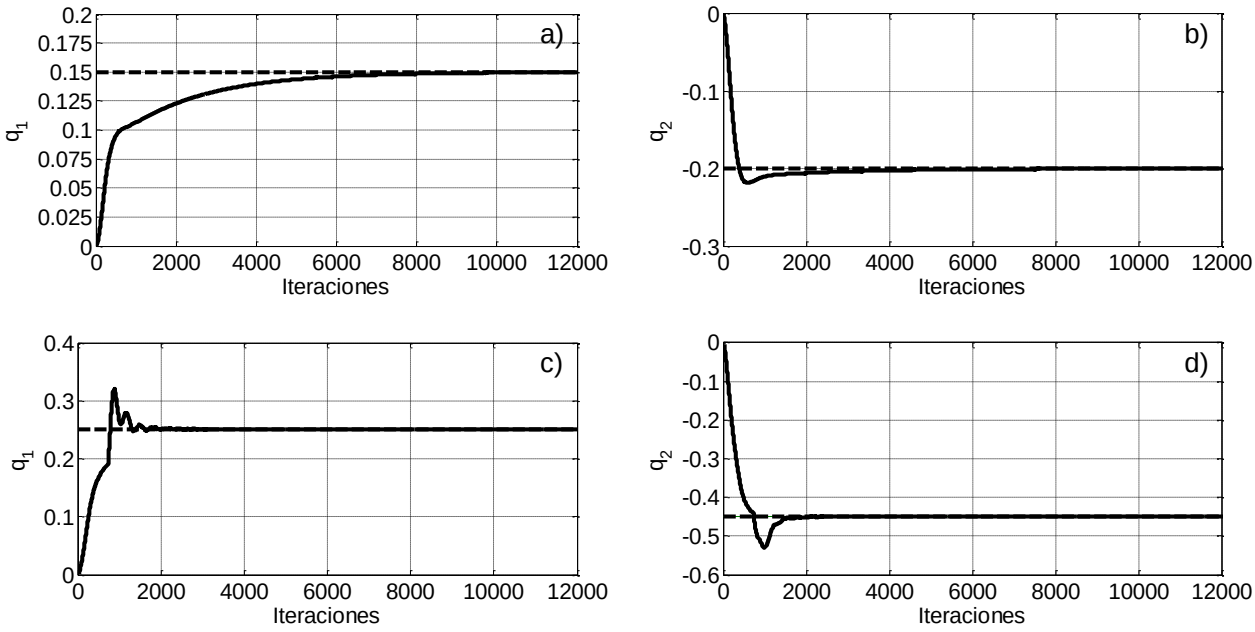


Figura 5.13 Resultados de simulación para la etapa de aprendizaje (a), (b) y generalización (c), (d) con el segundo esquema de control utilizando la segunda función de activación y la planta no-lineal de dos grados de libertad; Señal de referencia (línea discontinua) y la salida de planta (línea continua).

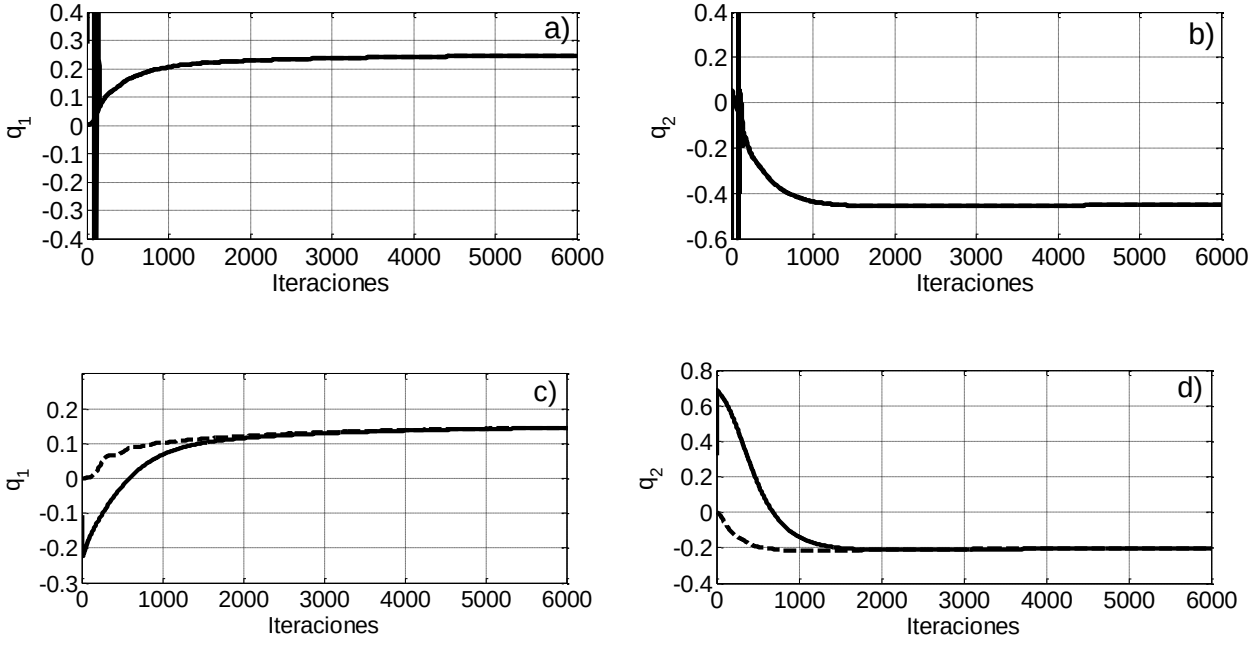


Figura 5.14 Resultados de simulación para la etapa de identificación utilizando la primera (a), (b) y la segunda función de activación (c), (d) con el segundoesquema de control y la planta no-lineal de dos grados de libertad; Señal de la planta referencia (línea discontinua) y la salida de la red neuronal de identificación (línea continua).

5.3.2.2 Comparación del EMC entre ambas funciones de activación

Los valores obtenidos de EMC al final de la simulación (20.000 pasos de iteración) con ambas funciones de activación para las dos articulaciones se muestran en la Tabla 5.5 y en la Tabla 5.6. Se puede observar que la implementación del término integral aceleró la convergencia independientemente de la función de activación utilizada. Por otro lado, la función la segunda función de activación mostro tener un desempeño superior con respecto a la primera, tanto en la etapa de aprendizaje como en la etapa de generalización.

Tabla 5.5 EMC de q_1 para ambas funciones de activación y el segundo esquema de control.

Etapa de aprendizaje		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0010	0.0008
15000	0.0007	0.0005
20000	0.0005	0.0004
Etapa de Generalización		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0018	0.0017
15000	0.0012	0.0011
20000	0.0009	0.0008

Tabla 5.6 EMC de q_2 para ambas funciones de activación y el segundo esquema de control.

Etapa de aprendizaje		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0008	0.0005
15000	0.005	0.0003
20000	0.0004	0.0002
Etapa de Generalización		
Número de paso	Primera función de activación	Segunda función de activación
10000	0.0040	0.0036
15000	0.0027	0.0024
20000	0.0020	0.0018

5.4 Conclusiones.

Del capítulo anterior se concluye que es posible utilizar redes neuronales complejas recurrentes para resolver problemas de control en plantas mecánicas oscilatorias, aunque la naturaleza del sistema no se expresa estrictamente en el dominio complejo, la red mostró tener resultados satisfactorios, y nuevamente se puede concluir que la metodología propuesta en este trabajo para el entrenamiento de redes neuronales complejas funciona correctamente. En la comparación de ambos esquemas control, se puede concluir que la implementación del término integral reduce notablemente el tiempo de convergencia de la planta a la referencia y elimina al mismo tiempo el error en estado estacionario.

Conclusión General

En este trabajo se estudiaron dos tipos de función de activación adecuadas para ser utilizadas en redes neuronales complejas. Se explicaron diferentes topologías de redes neuronales complejas. Se introdujo un método para la obtención de algoritmos de aprendizaje basado en reglas diagramáticas. En las redes neuronales complejas el algoritmo de aprendizaje está fuertemente relacionado con el tipo de función de activación que se utiliza, es por esta razón que el método diagramático desarrollado en este trabajo fue utilizado con ambas funciones de activación.

Una vez que este algoritmo de aprendizaje fue obtenido mediante el método diagramático, se implementó para el entrenamiento de redes neuronales complejas con tareas primeramente de identificación de plantas mecánicas, y posteriormente en problemas de control directo con realimentación de estado con y sin termino integral, en ambas casos las redes neuronales complejas tuvieron un desempeño adecuado, por lo que se concluye que el algoritmo de aprendizaje obtenido mediante la metodología propuesta es correcto.

Por último, se concluye que las redes neuronales complejas requieren una mayor capacidad de computo comparadas con las redes neuronales convencionales, debido a que los cálculos con números complejos requieren de una mayor cantidad de operaciones, sin embargo, este tipo de redes neuronales pueden ser en algún futuro una generalización en la cual las redes neuronales convencionales queden incluidas.

Trabajo Futuro

1. Como trabajo futuro se considera el desarrollo del algoritmo de Levenberg-Marquardt en su versión fuera de línea y en su versión recursiva y realizar una comparación con el algoritmo desarrollado en este trabajo.

2. Demostrar matemáticamente mediante una función de energía de Lyapunov la estabilidad de la red neuronal compleja.

Bibliografía

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

[2] D. T. Pham and X. Liu, *Neural Networks for Identification, Prediction and Control*, Primera ed. Springer, 1995.

[3] I. S. Baruch and J. L. Olivares, "Implementacion de un Multimodelo Neuronal Jerarchico oara Identificación y Control de Sistemas mecanicos," *Computación y Sistemas*, vol. 9, no. 1, pp. 28-40, 2005.

[4] N. N. Aizenberg, Y. L. Ivaskiv, and D. A. Pospelov, "A certain generalization of threshold functions," in *Dokrady Akademii Nauk SSSR 196*, Rusia, 1971, pp. 1287-1290.

[5] B. Widrow, J. McCool, and M. Ball, "The complex LMS algorithm," in *Proceedings of the IEEE*, vol. 63, 1975, p. 719–720.

[6] H. Leung and S. Haykin, "The Complex Backpropagation Algorithm," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2101-2104, 1991.

[7] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," *Transactions on Signal Processing*, vol. 40, p. 967–969, 1992.

[8] T. Nitta, *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*. IGI Global, 2009.

[9] G. Georgiou and C. Koutsougeras, "Complex Domain Backpropagation," *IEEE Transactions on Circuits and Systems-11: Analog and Digital Signal Processing*, vol. 39, no. 5, pp. 330-334, 1992.

[10] C. Woo and D. S. Hong, "Adaptive equalization using the complex backpropagation algorithm," in *Proc. of IEEE International Conference on Neural Networks*, vol. 4, Washington, DC, 1996, pp. 2136-2141.

[11] A. Hirose, "Motion Controls Using Complex-Valued Neural Networks with Feedback Loops," in *Proc. IEEE International Conference on Neural Networks*, vol. 1, San Francisco, CA, 1993, pp. 156-161.

[12] A. Hirose, *Complex-Valued Neural Networks*, 2nd ed., J. Kacprzyk, Ed. Polonia: Springer-Studies in Computational Intelligence, 2012, vol. 400.

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

[14] L. Ferariu, "Nonlinear System Identification Based on Evolutionary Dynamic Neural," in *Proc. of European Control Conference*, Cambridge, UK, 2003.

[15] K. Kawashima and T. Ogawa, "Complex-Valued Neural Network for Group-Movement Control of Mobile Robots," in *Proc. SICE Annual Conference 2012*, Japan, 2012.

[16] H. G. H.G. Zimmermann, A. Minin, and V. Kusherbaeva, "Historical Consistent Complex Valued Recurrent Neural Network," in *ICANN 2011*, Espoo, Finland, 2011, pp. 185-192.

[17] H. G. Zimmermann, A. Minin, and V. Kusherbaeva, "Comparison of the Complex Valued and Real Valued Neural Networks Trained with Gradient Descent and Random Search Algorithms," in *ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence.*, Brujas, Bélgica, 2011, pp. 213-218.

[18] J. Ward and R. Churchill, *Complex Variables and Applications*, Octava ed. EE.UU.: McGraw-Hill, 2009.

[19] D. H. Brandwood, "A complex gradient operator and its application in adaptive array theory," *Communications, Radar and Signal Processing, IEE Proceedings F*, vol. 130, no. 1, pp. 11-16, Feb. 1983.

[20] R. F. H. Fischer, *Precoding and Signal Shaping for Digital Transmission*, Primera ed. John Wiley & Sons, Inc, 2002.

[21] W. McCulloch and W. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.

[22] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, no. 65, pp. 386-408, 1958.

[23] J. R. Hilera González and V. J. Martínez Hernando, *Redes Neuronales Artificiales: Fundamentos, Modelos y Aplicaciones*, Segunda ed. Madrid, España: Alfaomega, 2000.

[24] P. Ponce Cruz, *Inteligencia Aritificial con Aplicaciones a la Ingeniería* , Primera ed., A. Herrera, Ed. México, México: Alfaomega, 2010.

[25] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Segunda ed., M. Horton, Ed. EE.UU.: Prentice-Hall, Inc., 1999.

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

[27] G. Cauwenberghs, "A fast stochastic Error-Descent Algorithm for supervised learning and optimization.," *Advances in Neural Information Processing Systems*, no. 5, pp. 244-251, 1993.

[28] E. Wan and F. Beaufays, "Diagrammatic method for deriving and relating temporal neural networks algorithms," *Neural Compute*, pp. 182-201, 1996.

[29] I. S. Baruch, J. Barrera-Cortés, and L. A. Hernandez, "A Fed-Batch Fermentation Process Identification and Direct Adaptive Neural Control with Integral Term," in *MICAI 2004: Advances in Artificial Intelligence*. Springer-Verlag, Berlin Heidelberg New York, 2004, pp. 764-773.

[30] I. S. Baruch, P. Georgieva, J. Barrera-Cortés, and S. Feyo de Azevedo, "Adaptive Recurrent Neural Network Control of Biological Wastewater Treatment," *International Journal of Intelligent Systems (Special Issue on Soft Computing for Modeling, Simulation and Control of Nonlinear Dynamical Systems, Guest Eds: Castillo, O., Melin, P.)*, vol. 20, no. 2, pp. 73-194, 2005.

[31] I. S. Baruch and C. R. Mariaca-Gaspar, "A Levenberg–Marquardt Learning Applied for Recurrent Neural Identification and Control of a Wastewater Treatment Bioprocess," *International Journal of Intelligent Systems*, vol. 24, pp. 1094-1114, 2009.

[32] I. Aizenberg, *Complex-Valued Neural Networks with Multi-Valued Neurons*, Primera ed., J. Kacprzyk, Ed. Springer-Studies in Computational Intelligence, 2011.

[33] A. S. Minin, "Modeling of Dynamical Systems with Complex-Valued Recurrent Neural Networks," Technische Universität München - Fakultät für Informatik Tesis Doctoral, 2012.

[34] N. Miklos and B. Salik, "Neural Networks with Complex Activations and Connection Weights," *Complex Systems*, vol. 8, pp. 115-126, 1994.

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

[36] V. M. Arellano-Quintana and I. S. Baruch, "Identification of Dynamical Systems Using Recurrent Complex-Valued Neural Networks," in *18th International Conference on Circuits, Systems, Communications and Computers*, Santorini, Grecia, 2014, pp. 74-79.

[37] I. S. Baruch and V. M. Arellano-Quintana, "Identification and Control of Oscillatory Dynamical Systems Using Recurrent Complex-Valued Neural Neworks," in *18th International Conference on Systems, Communications and Computers*, Santorini, Grecia, 2014, pp. 534-539.

[38] A. G. Parlos, T. C. Kil , and F. A. Amir, "Application of the Recurrent Multilayer Perceptron in Modeling Complex Process Dynamics," *IEEE Trans. On NN*, vol. 5, no. 2, pp. 225-265, Mar. 1994.

[39] S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators*, C. J. Harris, Ed. UK: World Scientific Publishing Co. Pte. Ltd., 1998, vol. 19.

[40] I. Laundau, R. Lozano, and M. M'Saad, *Adaptive Control*. Gran Bretaña: Springer, 1998.

[41] K. Narendra and S. Mukhopandhyay, "Adaptive Control of Nonlinear Multivariale Systemas Using Neural Networks," *Neural Networks*, vol. 7, no. 5, pp. 732-752, 1999.

ANEXO
ARTÍCULOS PUBLICADOS

Anexo-Artículos Publicados

- V. M. Arellano-Quintana y I. S. Baruch, "*Identification of Dynamical Systems Using Recurrent Complex-Valued Neural Networks*," In: Nikos Mastorakis et al., Eds, Latest Trends on Systems, Vol. I, Recent Adavances in Electrical Engineering Series – 37, Proc. of the *18th International Conference on Circuits, Systems, Communications and Computers*, Santorini, Greece, 2014, pp. 74-79, ISSN: 1790-5117, ISBN: 978-1-61804-243-9.

- I. S. Baruch y V. M. Arellano-Quintana, "*Identification and Control of Oscillatory Dynamical Systems Using Recurrent Complex-Valued Neural Neworks*," In: Nikos Mastorakis et al., Eds, Latest Trends on Systems, Vol. II, Recent Adavances in Electrical Engineering Series – 38,Proc. of the *18th International Conference on Systems, Communications and Computers*, Santorini, Greece, 2014, pp. 534-539, ISSN: 1790-5117, ISBN: 978-1-61804-244-6.

Latest Trends on Systems - Volume I

Identification of dynamical systems using recurrent complex-valued neural networks

VíctorM. Arellano-Quintana and Ieroham S. Baruch

***Abstract*—**The existing in the literature backpropagation algorithms for training Complex-Valued Neural Networks (CVNN) are very complicated, especially where we deal with a Recurrent CVNN (RCVNN). For that reason the paper proposed to use diagrammatic rules so to construct an adjoined RCVNN and propagate the complex output error through it so to perform the weight adjustment. Two different RCVNN topologies with different activation functions avoiding singularity are considered and their CV backpropagation learning algorithms are derived using the proposed learning methodology. Finally, some comparative simulation results of RCVNN identification of flexible-joint robot are given and discussed, and then a validation stage is presented in order to confirm the good quality of the proposed learning methodology.

***Keywords*—**Diagrammatic rules, learning algorithms, recurrent complex-valued neural networks, systems identification

I. Introduction

UNTIL now there are few applications using Recurrent Complex-Valued Neural Networks (RCVNN). Most of them deal with oscillatory systems which by their physical nature it is convenient to be treated in the complex domain, such as electromagnetic waves, light waves, images processing, electric power systems etc. (see [1], [2], [3]). In [2] the authors apply a special type of a RCVNN for modeling of power transformer, obtaining good results. The drawback here is that nothing is mentioned about the presence of singularity points due to the activation function nature.

In the field of mechanical systems identification the RCVNN have had a minor presence. In spite of that, some papers like [3], [4] proposed to use RCVNN for mechanical plants identification and control, obtaining good results. In [3] the authors applied a CVNN for an industrial evaporator system identification using an evolutionary algorithm to design the network. They use also radial basis functions NN avoiding the gradient terms computation in the learning algorithm. Other papers like [4], [5] used CVNN for these kind of systems, obtaining satisfactory results.

In [6], a type of RCVNN is used for modeling of Lorenz system. In [7], Leung and Haykin derived a Complex Value Backpropagation (CVBP) algorithm used for pattern classification. However, this learning algorithm presented some problems because of the activation function singularity. For that reason some authors (see [8]-[11]) proposed different activation functions that avoid activation function singularity.

So, to simplify the backpropagation learning of the RCVNN, the present paper proposed to use diagrammatic rules so to construct an adjoined RCVNN and propagate the complex output error through it so to obtain the weight adjustment. Two different RCVNN topologies with different activation functions avoiding singularity are considered and their CV backpropagation learning algorithms are derived using the diagrammatic rules and the constructed by them adjoined RCVNN. Finally, some comparative simulation results of RCVNN identification of flexible-joint robot are given and discussed, and then a validation stage is presented in order to confirm the good quality of the proposed learning methodology.

V.M. Arellano-Quintana is MS student in the Department of Automatic Control, CINVESTAV-IPN, D.F., Mexico, e-mail: varellano@ctrl.cinvestav.mx.
I.S. Baruch is with the Department of Automatic Control, CINVESTAV-IPN, D.F., Mexico, e-mail: baruch@ctrl.cinvestav.mx.

II. Topology and Backpropagation Learning of Real-Valued Recurrent Neural Network

In Fig.1 it is shown the block-diagram of a Real Value Recurrent Neural Network (RVRNN) with Jordan Canonical topology, [12]. The canonical RNN topology possesses the controllability, observability, reachability and identifiability-properties, and minimum number of parameters, subject to learning. The RVRNN model is described by the following equations:

$$\mathrm{X(k+1)} = \mathrm{JX(k)} + \mathrm{BU(k)}$$
$$J = \text{block-diag}(J_i);\ |J_i| < 1, i = 1, \ldots, N \quad (1)$$

$$E(k) = Y_p(k) - Y(k) \quad (2)$$

$$\mathrm{Z}(k) = \Gamma[X(k)] \quad (3)$$

$$\mathrm{Y(k)} = \Phi[CZ(k)] \quad (4)$$

Where: $X(\cdot)\epsilon\mathbb{R}^N$- State vector; $U(\cdot)\epsilon\mathbb{R}^M$- Input vector; $Y(\cdot)\epsilon\mathbb{R}^L$- Output vector; $Z(\cdot)\epsilon\mathbb{R}^L$- Output vector of the hidden layer; $\Gamma(\cdot), \Phi(\cdot)$- Vector valued activation functions with compatible dimensions; J- Weight state diagonal matrix with elements J_i. The inequality in (1) is a stability preserving condition, imposed on the weights J_i; B and C are weight input and output matrices with compatible dimensions. The RVRNN

ISBN: 978-1-61804-243-9

model is completely parallel, capable to issue weights J, B, C, and states X.

Applying the diagrammatic rules, [13], to the RVRNN topology, given on Fig.1 we could obtain the adjoined RVRNN model, given on Fig.2. The adjoined RVRNN is used for the backward pass of the backpropagation algorithm to pass the output error through it so to train RVRNN weights. The performance index to be minimized is given by:

$$\zeta(k) = \frac{1}{2}\sum_j [E_j(k)]^2, \quad j \in \mathbb{C}, \qquad \zeta = \frac{1}{N_e}\sum_j \zeta(k) \tag{5}$$

Where the instantaneous Means Squared Error (MSE) $\zeta(k)$is minimized in real-time applications and the total MSE ζ is minimized for one epoch $N_e$in off-line applications. The general RVRNN real-time backpropagation learning algorithm with momentum term is given by the following vector-matricial equation:

$$W(k+1) = W(k) + \eta\Delta W(k) + \alpha\Delta W(k-1)$$
$$|W_{ij}| < W_o \tag{6}$$

Where: $W(\cdot)$ is a general weight matrix (in fact J, B, C); $\Delta W(\cdot)$ is the modification of $\Delta W(\cdot)$; η is a diagonal constant matrix of learning; α is a diagonal momentum term matrix; W_o is a restricted region for the weight W_{ij}. Using the specified in Fig.2 errors and the obtained in the forward pass (1)-(3) intermediate vectors, we could obtain the following weight update algorithm for the matrices J, B, C.

For the output layer:

$$\Delta C(k) = E_1(k)Z^T(k) \tag{7}$$
$$E_1(k) = \Phi'[Y(k)]E(k) \tag{8}$$
$$E(k) = Y_p(k) - Y(k) \tag{9}$$

For the hidden layer:

$$\Delta J(k) = E_3(k)X^T(k) \tag{10}$$
$$E_3(k) = \Gamma'[Z(k)]E_2(k) \tag{11}$$
$$E_2(k) = C^T(k)E_1(k) \tag{12}$$
$$\Delta vJ(k) = E_3(k) \otimes X(k) \tag{13}$$
$$\Delta B(k) = E_3(k)U^T(k) \tag{14}$$

Where: $\Delta C, \Delta J, \Delta B$are weight corrections of C, J, B; the output error $E$is given by (9) and passed through the adjoined RVRNN so to obtain the intermediate error vectors E_1, E_2, E_3; $\Phi', \Gamma'$are derivatives of the respective activation functions expressed with respect to its correspondent outputs. The equation (13) gives the learning solution for the diagonal matrix J as a dot product of two vectors. Equations (7)-(14) represented the complete backpropagation learning algorithm for RVRNN.

This solution could be applied for any NN topology and does not require computations of partial derivatives. Based on the diagrammatic rules we could derive a similar learning solution for the complex value case.

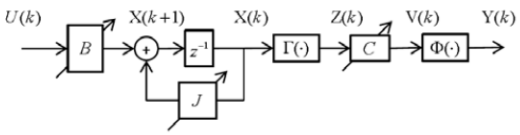


Fig. 1 Block-diagram of the RVRNN topology

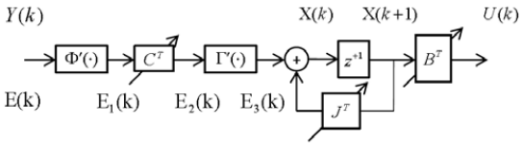


Fig. 2 Block-diagram of the adjoined RVRNN topology, obtained from the RVRNN topology using the diagrammatic rules

III. Topology and Backpropagation Learning of Recurrent Complex-Valued Neural Network

The general Complex Valued Neural Network topology below consideration has complex valued input, output, and state vectors, and complex A, B, C weight matrices. In this part we consider two particular Recurrent CVNNs (RCVNNs) with different activation functions. In the same manner as in the real value case we could apply complex valued diagrammatic rules so to derive an adjoined RCVNN for each CVNN case.

The performance index to be minimized is given by:

$$\zeta(k) = \frac{1}{2}\sum_j [E_j(k)][E_j^*(k)], \quad j \in \mathbb{C}, \quad \zeta = \frac{1}{N_e}\sum_j \zeta(k) \tag{15}$$

This function $\zeta(k)$ is a mapping of the form $f: \mathbb{C} \to \mathbb{R}$, so it is not analytic in the sense that it does not have derivative and also it does not satisfy the Cauchy-Riemann equations. This complicates the use of the gradient descendent algorithm, because we have to use the so-called Wirtinger's calculus. Using diagrammatic rules we avoid this complicated problem.

A. RCVNN Topology with First Type Activation Function

The first activation function that we consider is given by (16).This activation function has singularities in some parts of the complex domain and because of that, this function is evaluated in a region of the complex plane avoiding singularity points, [14]. The activation function is defined as:

$$f(z) = \tanh z, \quad z \in \mathbb{C} \setminus \left[-\frac{\pi}{2} - \epsilon; -\frac{\pi}{2} + \epsilon\right] \cup [-\epsilon; \epsilon] \tag{16}$$

The NN using this activation function is described in the given in Fig.3 block diagram. This type of representation allows us to apply diagrammatic rules so to derive the adjoined RCVNN and use it for NN learning. In the present

ISBN: 978-1-61804-243-9

paper we propose a new approach of the CVBP for the given in Fig. 3 RCVNN extending the real-time real value NN algorithm of learning, [12], to the complex value case.The topology of this RCVNN is given on Fig.3.

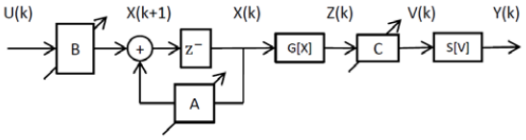


Fig. 3 Topology of the first type RCVNN

The mathematical description of that topology is the same as (1)-(4) for the real valued case but the variables are complex. The vectors and matrices of the RCVNN topology are given as follows:

$A \in \mathbb{C}^{nxn}$: Feedback Matrix;
$B \in \mathbb{C}^{nxn}$: Input matrix;
$C \in \mathbb{C}^{nxn}$: Output matrix;
$X(k) \in \mathbb{C}^{nxn}$: State vector;
$U(k) \in \mathbb{C}^{nxn}$: Network input;
$Y(k) \in \mathbb{C}^{nxn}$: Network output;
$G[\cdot], S[\cdot]$: Complex- valued vector-*tanh* - activation functions, given by (16);
m: Number of inputs;
n: Number of neurons in the hidden layer;
p: Number of neurons in the output layer.

The complex state feedback matrix A is defined as diagonal with the same weight restriction as the given in (1) for J.

As we could see the used *tanh* - activation function has singularity in some parts of the complex domain. To overcome that, the activation function is evaluated in regions outside these points, as it could be seen in(16).

Applying the complex valued diagrammatic rules we could obtain the adjoined RCVNN, given on Fig.4.

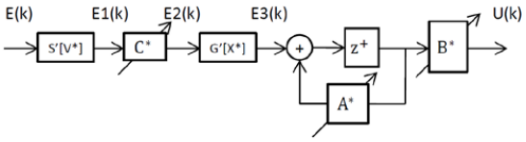


Fig. 4 Adjoined topology of the first type RCVNN

Now, the backpropagation learning rule (6) could be defined in a complex domain with the same significance of the participating variables. Using the adjoined RCVNN topology we could derive the following weight update algorithm:

For the output layer:

$$\Delta C(k) = E_1(k) Z^*(k) \quad (17)$$

$$E_1(k) = S'[Y[V^*(k)](k)]E(k) \quad (18)$$

$$E(k) = T(k) - Y(k) \quad (19)$$

For the hidden layer:

$$\Delta A(k) = E_3(k) X^*(k) \quad (20)$$

$$E_3(k) = G'[Z[X^*(k)](k)]E_2(k) \quad (21)$$

$$E_2(k) = C^*(k) E_1(k) \quad (22)$$

$$\Delta vA(k) = E_3(k) \otimes X^*(k) \quad (23)$$

$$\Delta B(k) = E_3(k) U^*(k) \quad (24)$$

Where:

$G'[\cdot], S'[\cdot]$: Derivatives of the activation functions G, S;
a^*: Transpose and conjugate of the complex number a;
$T(k)$: Desired output vector.

As it could be seen, the application of the diagrammatic rules and the adjoined RCVNN topology simplified the learning with respect to the classical gradient descent learning in complex domain, [7].

B. RCVNN Topology with Second Type Activation Function

The second type activation function, [9], does not have singularity points. It is given by the next equation:

$$f(z) = \tanh Re(z) + i \tanh Im(z) \quad (25)$$

Unlike the RVNN, the topology of the CVNN will be defined by the second type activation function. In this case, the RCVNN is given by the following equations:

$$X(k+1) = AX(k) + BU(k) \quad (26)$$

$$Z(k) = G[X\_Re\ (k)] + iG[X\_Im\ (k)] \quad (27)$$

$$V(k) = Z_{Re} C_{Re} + iZ_{Im} C_{Im} \quad (28)$$

$$Y(k) = S[V_{Re}(k)] + iS[V_{Im}(k)] \quad (29)$$

The topology of the RCVNN in this case is given on Fig. 5.

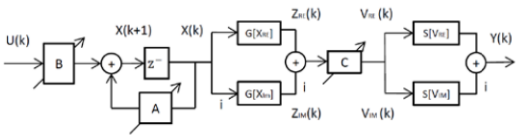


Fig. 5 Topology of the second type RCVNN

From Fig.5 and equation (25) it is clear that this topology separates the real and imaginary parts of the CV activation functions in order to simplify the CV learning algorithm.

The vectors and matrices of the RCVNN topology are defined as follows:

$G[\cdot], S[\cdot]$ are complex valued vector-*tanh*-activation functions given by (25).

ISBN: 978-1-61804-243-9

$A \in \mathbb{C}^{nxn}$: Feedback Matrix;
$B \in \mathbb{C}^{nxn}$: Input matrix;
$C \in \mathbb{C}^{nxn}$: Output matrix.
$X(k) \in \mathbb{C}^{nxn}$: State vector;
$U(k) \in \mathbb{C}^{nxn}$: Network input;
$Y(k) \in \mathbb{C}^{nxn}$: Network output;
m: Number of inputs;
n: Number of neurons in the hidden layer;
p: Number of neurons in the output layer.

The complex state feedback matrix A is defined as diagonal with the same restriction as (1) for J.

Applying the complex valued diagrammatic rules we could obtain the adjoined RCVNN, given on Fig.6.

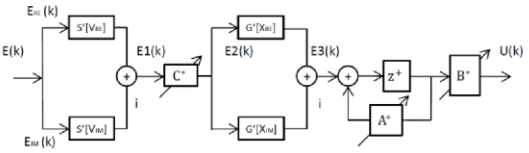


Fig. 6 Adjoined topology of the second type RCVNN

Now, the backpropagation learning rule (6) could be defined in a complex domain with the same significance of the participating variables. Using the adjoined RCVNN topology we could derive the following weight update algorithm:

For the output layer:

$$\Delta C(k) = E_1(k)Z^*(k) \tag{30}$$

$$E_1(k) = \mathrm{E_{Re}}S'\left[Y[V_{Re}(k)]\right] + i\mathrm{E_{Im}}S'\left[Y[V_{Im}(k)]\right] \tag{31}$$

$$E(k) = T(k) - Y(k) \tag{32}$$

For the hidden layer:

$$\Delta A(k) = E_3(k)X^*(k-1) \tag{33}$$

$$E_3(k) = \mathrm{E_{2Re}}G'\left[Z[X_{Re}(k)]\right] + i\mathrm{E_{2Im}}G'\left[Z[X_{Im}(k)]\right] \tag{34}$$

$$E_2(k) = C^*(k)E_1(k) \tag{35}$$

$$\Delta vA(k) = E_3(k) \otimes X^*(k) \tag{36}$$

$$\Delta B(k) = E_3(k)U^*(k) \tag{37}$$

Where:

$G[\cdot], S[\cdot]$: The derivatives of the activation functions G, S;
a^*: Transpose and conjugate of the complex number a;
$T(k)$: Desired output vector;
E_{Re}: Real part of E(k);
E_{Im}: Imaginary part of E(k).

As it could be seen, the application of the complex diagrammatic rules in the design of the adjoined RCVNN topology simplified the learning with respect to the classical gradient descent learning in complex domain.

IV. Complex -Valued Neural Solution of Nonlinear Identification Problem

This part of the paper illustrates the application of the RCVNN for nonlinear oscillatory plant identification. The nonlinear oscillatory plant model under investigation is a flexible – joint robot arm. The model, the output and the input of the plant are given in continuous time. In order to use a recurrent neural network for its identification, the output/input signals of the plant are discretized with sampling period T_0.

A. Description of the Nonlinear Plant Model

The identified system is an idealized nonlinear model of a flexible – joint robot arm, illustrated by Fig. 7. The flexibility of the robot joint is caused by a harmonic drive, which is a type of robot gear mechanism with high torque transmission, low backlash and compact size.

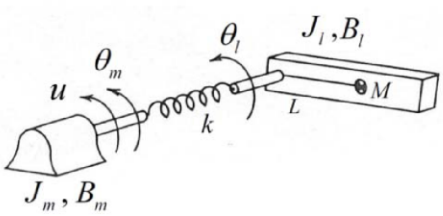


Fig. 7 Idealized model representing robot joint flexibility

The robot joint model consists of an actuator connected to a load through a torsional spring representing the joint flexibility. We take the motor torque as an input u.The equations of motion of the flexible - joint robot are given as follows.

$$\begin{aligned} J_l\ddot{\theta}_l + B_l\dot{\theta}_l + Mgl\sin\theta_l + k(\theta_l - \theta_m) &= 0 \\ J_m\ddot{\theta}_m + B_m\dot{\theta}_m - k(\theta_l - \theta_m) &= u \end{aligned} \tag{378}$$

Where: J_l, J_m are load and motor inertias; $B_l$and B_m are load and motor damping constants; u is the input torque applied to the motor shaft; M and L are the mass of the link and the length between the shaft and the center of mass of the link; k represents the torsional stiffness constant of the harmonic drive gear.

As we can see, the plant is an oscillatory system, described by two second order differential equations, representing a system with two degrees of freedom but only one input which makes the system sub-actuated.

B. Plant Identification

The plant identification using the given up two RCVNN models is illustrated by Fig.8. Here the desired complex target vector is the output of plant. The identification objective is that the complex weight parameters of the RCVNN are adjusted in such manner that the RCVNN output follows the plant output with minimum MSE.

ISBN: 978-1-61804-243-9

The RCVNN used has three neurons in the hidden layer, one input and one output neurons. The RCVNN dimensions are as follows: $m = 1;\ p = 1;\ n = 3$ and a sampling time $T_0 = 0.01$.

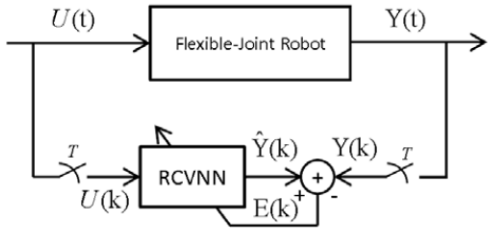


Fig. 8 Block diagram of RCVNN plant identification

C. Simulation Results

This section described the simulation results obtained using both type RCVNNs for nonlinear oscillatory plant identification. The simulation is done in two stages: stage of system identification and stage of NN generalization. During the first stage the RCVNN weights are learned until convergence and after that they are frozen and the input signal is changed so to validate (generalized) the result of identification. As a measure of comparison we use the final MSE% of identification and generalization. The input plant signals used for system identification $u(t)$and generalization $u_v(t)$are given by:

$$u(t) = \sin\left(\frac{t}{10}\right) + 0.5\sin\left(\frac{2t}{50}\right) \quad (38\ 9)$$

$$u_v(t) = 0.5\sin\left(\frac{t}{10}\right) + 0.8\sin\left(\frac{t}{30}\right) \quad (40)$$

Simulation results obtained with the first type activation function RCVNN-1 topology. The graphical results of plant identification for the first case are given on Fig. 9 where the output of the plant is compared with the output of the RCVNN-1. The final MSE of RCVNN-1 convergence during system identification is given on Table 1 for 1000, 2000, and 3000 learning iterations. The results show a constant MSE decreasing, exhibiting a good BP NN convergence.

The graphical results of RCVNN-1 generalization are given on Fig. 10, where the NN weights are frozen and input signal is changed. The output of the plant is compared with the output of the RCVNN-1 exhibiting a good generalization. The final results of RVCNN-1 generalization are given on Table 1 for 3000 steps. The validation results show a constant MSE decreasing exhibiting a good RCVNN-1 generalization.

Simulation results obtained with the second type activation function RCVNN-2 topology. The graphical results of plant identification for the first case are given on Fig. 11 where the output of the plant is compared with the output of the RCVNN-2.

The final MSEs of RCVNN-2 convergence during system identification is given on Table 1 for 1000, 2000, and 3000 learning iterations. The results show a constant MSE decreasing exhibiting a good BP NN convergence.

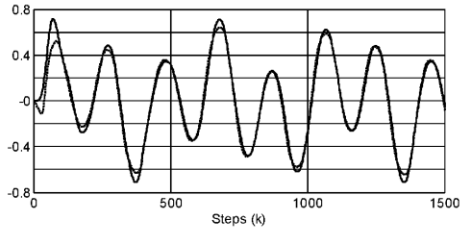


Fig. 9 Simulation results of RCVNN-1 BP learning; RCVNN-1 output (dotted line); plant output (continue line)

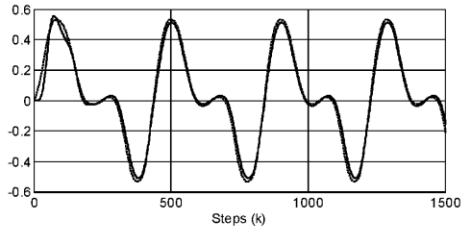


Fig.10 Simulation results of RCVNN-1 generalization; RCVNN-1 output (dotted line); plant output (continue line)

The graphical results of RCVNN-2 generalization are given on Fig. 12, where the NN weights are frozen and input signal is changed. The output of the plant is compared with the output of the RCVNN-2 exhibiting a good generalization. The final results of RVCNN-2 generalization are given on Table 1 for 3000 steps. The validation results show a constant MSE decreasing exhibiting a good RCVNN-2 generalization.

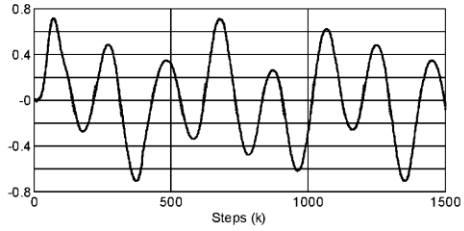


Fig. 11 Simulation results of RCVNN-2 BP learning; RCVNN-2 output (dotted line); plant output (continue line)

Comparative final MSE simulation results of nonlinear oscillatory plant identification and generalization using RCVNN-1 and RVCNN-2. The final MSE values obtained during identification and generalization experiments with both RCVNN-1,2 (see Table 1) show that the RCVNN-2 outperformed the RCVNN-1 because the RCVNN-1 activation func-

ISBN: 978-1-61804-243-9

tion possess singular points that affects the BP learning and the RCVNN-2 activation function does not possess such singular points so its learning is perfect. In generalization stage the result are opposite but still good so the identification results are dominating.

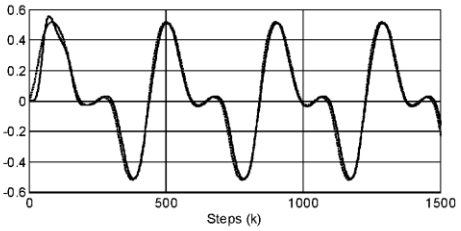


Fig.12. Simulation results of RCVNN-2 generalization; RCVNN-2 output (dotted line); plant output (continue line)

Table I. Final MSE of learning and generalization

Learning		
Number of steps	First type of activation function	Second type of activation function
1000	0.0264	0.0045
2000	0.0178	0.0029
3000	0.0097	0.0028
Generalization		
Number of steps	First type of activation function	Second type of activation function
3000	0.0014	0.0025

Finally we could make the conclusion that it is better to use RCVNN-2 which does not have singular points then to use RCVNN-1 which have singular points and try to overcome them.

V. Conclusions

In the present paper we use the diagrammatic rules, [13], for the complex case and applied them to obtain the backpropagation training algorithm. We studied two different RCVNNs topologies with two different activation functions. Using the diagrammatic rules we obtain the RCVNNs adjoined and derived the backpropagation algorithm for both cases in a simpler manner that coincides with the actual training algorithm using Wirtinger's calculus. The obtained comparative simulation results for identification and generalization of a nonlinear oscillatory plant confirm the quality of the BP learning algorithm. The comparison of both RCVNN-1, and RCVNN-2 topologies for both activation functions give some priority to the second case in the sense that it is better to use activation functions without singular points then to use activation functions with singular points and try to avoid them.

References

[1] A. Hirose, *Complex-Valued Neural Networks*, 2nd ed., S. i. C. Intelligence, Springer-Verlag, 2012, vol. 400.

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

[3] L. Ferariu, "Nonlinear System Identification Based on Evolutionary Dynamic Neural Network," in *Proc. of European Control Conference*, Cambridge, UK, paper no. 432, 2003.

[4] K. Kawashima, and T. Ogawa, "Complex-Valued Neural Network for Group-Movement Control of Mobile Robots," in *Proc. SICE Annual Conference 2012*, Japan, 2012, pp. 1806 - 1809.

[5] A. Hirose, "Motion Controls Using Complex-Valued Neural Networks with Feedback Loops," in *Proc. IEEE International Conference on Neural Networks*, vol. 1, San Francisco, CA, 1993, pp. 156-161.

[6] H. G. H.G. Zimmermann, A. Minin, and V. Kusherbaeva, "Historical Consistent Complex Valued Recurrent Neural Network," in *ICANN 2011*, Espoo, Finland, 2011, pp. 185-192.

[7] H. Leung, and S. Haykin, "The Complex Backpropagation Algorithm," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2101-2104, 1991.

[8] T. Nitta, *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*. IGI Global, 2009.

[9] C. Woo, and D. S. Hong, "Adaptive equalization using the complex backpropagation algorithm," in *Proc. of IEEE International Conference on Neural Networks*, vol. 4, Washington, DC, 1996, pp. 2136-2141.

[10] N. Miklos, and B. Salik, "Neural Networks with Complex Activations and Connection Weights," *Complex Systems*, vol. 8, pp. 115-126, 1994.

[11] G. Georgiou, and C. Koutsougeras, "Complex Domain Backpropagation," *IEEE Transactions on Circuits and Systems-11: Analog and Digital Signal Processing*, vol. 39, no. 5, pp. 330-334, 1992.

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

[13] E. Wan, and F. Beaufays, "Diagrammatic method for deriving and relating temporal neural networks algorithms," *Neural Computation*, pp. 182-201, 1996.

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

ISBN: 978-1-61804-243-9

Latest Trends on Systems - Volume II

Identification and control of oscillatory dynamical systems using recurrent complex-valued neural networks

Ieroham S. Baruch and Víctor M. Arellano-Quintana

Abstract—The present paper used the obtained results of Recurrent Complex Valued Neural Network (RCVNN) identification and extend these results to direct complex value control of nonlinear oscillatory plants. After the introduction, the paper gives a short description of the used RCVNN topology and BP learning using diagrammatic rules to obtain it. Than a RCVNN version of direct control schemes are derived. First, an inverse model control scheme is designed. Second, an extension of the first scheme with an integral term is done. Third a RCVNN for plant identification is added and the obtained state estimation is used for feedback control in order to perform a composite feedback-feed-forward control with I-term. Finally, comparative simulation results of flexible-joint robot model using the three schemes of direct complex value control are obtained. The obtained comparative simulation results confirmed the good quality of the proposed control methodology.

Keywords—Direct adaptive feedforward/feedback neural control with integral term**,** **d**iagrammatic rules, backpropagation learning algorithm, recurrent complex-valued neural network, systems identification and control

I. Introduction

IN last decade there are some applications using Recurrent Complex-Valued Neural Networks (RCVNN). Most of them deal with oscillatory systems which by their physical nature it is convenient to be treated in the complex domain, such as electromagnetic waves, light waves, images processing, electric power systems, evaporator system, mechanical systems etc. (see [1], [2], [3], [4]). In [2] the authors apply a special type of a RCVNN for modeling of power transformer, obtaining good results. Some other papers like [3], [4] proposed to use RCVNN for mechanical plants identification and control, obtaining good results. In [3] the authors applied a CVNN for an industrial evaporator system identification using an evolutionary algorithm to design the network. They use radial basis functions NN avoiding the gradient terms computation in the learning algorithm. Other papers like [4], [5] used CVNN for these kind of systems, obtaining satisfactory results.

In [6], Leung and Haykin derived a Complex Value Backpropagation (CVBP) algorithm used for pattern classification. However, this learning algorithm presented some problems because of activation function singularity. For that reason some papers (see [7], [8], [9], [10], [11]) proposed different activation functions that avoid activation function singularity. The paper [11] considered two type of activation functions avoiding singularity, applied for RCVNN identification of nonlinear oscillatory mechanical plant.

The present paper is based on the obtained in [11] results and extends these results to direct control using complex value control algorithms. First the paper gives a short description of the used RCVNN topology and BP learning using diagrammatic rules to obtain it. Than a complex value versions of direct control algorithms are derived, using a complex valued neural network (NN) to model the inverse of the plant. Further an extension of the first control scheme with an integral term is proposed. Than the control scheme is extended using RCVNN for plant identification. Further the estimated states are used for feedback control in order to obtain a composite feedback-feedforward control. Finally, comparative simulation results of flexible-joint robot plant model using three control schemes of direct complex value control are given, obtaining good comparative results.

II. Topology and Backpropagation Learning of Recurrent Complex-Valued Neural Network

The considered general Recurrent Complex Valued Neural Network topology, [11], has complex valued input, output, and state vectors, and complex $\boldsymbol{A,B,C}$ weight matrices. In [11], the authors consider a new approach based on diagrammatic rules to obtain the training algorithm. The performance index to be minimized is given by:

$$\zeta(k) = \frac{1}{2}\sum_j [E_j(k)][E_j^*(k)],\ j \in \mathbb{C},\ \zeta = \frac{1}{N_e}\sum_j \zeta(k) \qquad (1)$$

The function $\zeta(k)$is a mapping of the form $f: \mathbb{C} \to \mathbb{R}$, so it is not analytic in the sense that it does not have derivative and also it does not satisfy the Cauchy-Riemann equations. This complicates the use of the gradient descendent algorithm, because we have to use the so-called Wirtinger's calculus. Using diagrammatic rules, [11], we avoid this complicated problem. The activation function that we consider has separate real and imaginary parts so it does not have singular points. It is given by the next equation, [11]:

Ieroham S. Baruch is with the Department of Automatic Control, CINVESTAV-IPN, Mexico City, Mexico, e-mail: baruch@ctrl.cinvestav.mx
Víctor Manuel Arellano-Quintana is MS student in the Department of Automatic Control, CINVESTAV-IPN, Mexico City, Mexico, e-mail: varellano@ctrl.cinvestav.mx

ISBN: 978-1-61804-244-6

$$f(z) = \tanh \mathrm{Re}(z) + i \tanh \mathrm{Im}(z), \quad z \in \mathbb{C} \tag{2}$$

This type of activation function representation allows us to apply diagrammatic rules so to derive the adjoined RCVNN and use it for NN learning. The topology of the RCVNN is given on Fig.1.

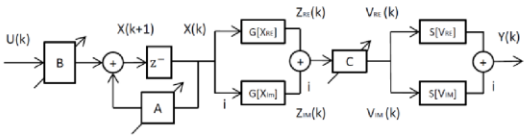


Fig.1 Topology of the RCVNN

The mathematical description of that topology is given by equations (3)-(6).

$$\begin{gathered} \mathrm{X}(k+1) = AX(k) + BU(k) \\ A = \text{block-diag}(A_i); |A_i| < 1, i = 1, \dots, n \end{gathered} \tag{3}$$

$$Z(k) = \mathrm{G}[X\_Re\ (k)] + i\mathrm{G}[X\_Im\ (k)] \tag{4}$$

$$V(k) = Z_{Re}C_{Re} + iZ_{Im}C_{Im} \tag{5}$$

$$Y(k) = \mathrm{S}[V_{Re}(k)] + i\mathrm{S}[V_{Im}(k)] \tag{6}$$

The vectors and matrices of the RCVNN topology are as follows:

$A \in \mathbb{C}^{nxn}$: Feedback Matrix;
$B \in \mathbb{C}^{nxm}$: Input matrix;
$C \in \mathbb{C}^{pxn}$: Output matrix;
$X(k) \in \mathbb{C}^{nx1}$: State vector;
$U(k) \in \mathbb{C}^{mx1}$: Network input;
$Y(k) \in \mathbb{C}^{px1}$: Network output;
$G[\cdot], S[\cdot]$: Complex- valued vector-*tanh* - activation functions, given by(2);
m: Number of inputs;

The Means Squared Error (MSE), (1), is minimized in real-time applications and the total MSE ζ is minimized for one epoch $N_e$in off-line applications. The general RCVNN real-time backpropagation learning algorithm with momentum term is given by the following vector-matricial equation:

$$\begin{gathered} W(k+1) = W(k) + \eta \Delta W(k) + \alpha \Delta W(k-1) \\ |W_{ij}| < W_o \end{gathered} \tag{7}$$

Where: $W(\cdot)$ is a general weight matrix (in fact *A, B, C*); $\Delta W(\cdot)$ is the learning modification of $\Delta W(\cdot)$; η is a diagonal constant matrix of learning; α is a diagonal momentum term matrix; W_o is a restricted region for the common W_{ij} weight. Applying the complex valued diagrammatic rules we could obtain the adjoined RCVNN, given on Fig.2. Using the specified in Fig.2 errors and the obtained in the forward pass (3)-(6) intermediate vectors, we could obtain the following weight update algorithm for the matrices *A, B, C.*

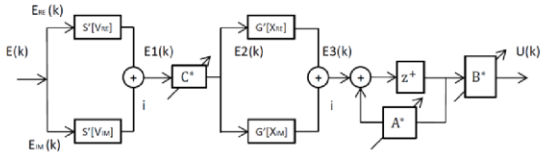


Fig. 2 Adjoined topology of the RCVNN

Now, the BP learning rule (7) could be defined in a complex domain using the adjoined RCVNN topology. The following weight update algorithm is defined as:

For the output layer:

$$\Delta C(k) = E_1(k)Z^*(k) \tag{8}$$

$$E_1(k) = \mathrm{E}_{\mathrm{Re}}\mathrm{S}'\left[Y[V_{Re}(k)]\right] + i\mathrm{E}_{\mathrm{Im}}\mathrm{S}'\left[Y[V_{Im}(k)]\right] \tag{9}$$

$$E(k) = T(k) - Y(k) \tag{10}$$

For the hidden layer:

$$\Delta A(k) = E_3(k)X^*(k-1) \tag{11}$$

$$E_3(k) = \mathrm{E}_{2\mathrm{Re}}\mathrm{G}'\left[Z[X_{Re}(k)]\right] + i\mathrm{E}_{2\mathrm{Im}}\mathrm{G}'\left[Z[X_{Im}(k)]\right] \tag{12}$$

$$E_2(k) = C^*(k)E_1(k) \tag{13}$$

$$\Delta \mathrm{v}A(k) = E_3(k) \otimes X^*(k) \tag{14}$$

$$\Delta \mathrm{B}(k) = E_3(k)U^*(k)$$

Where:

$G'[\cdot], S'[\cdot]$: The derivatives of the activation functions;
a^*: Transpose conjugate of the complex number a;
$T(k) \in \mathbb{C}^{px1}$:Desired output vector;
E_{Re}: Real part of E(k);
E_{Im}: Imaginary part of E(k).

As it could be seen, the application of the diagrammatic rules and the adjoined RCVNN topology simplified the learning with respect to the classical gradient descent learning in complex domain, [6].

III. Adaptive Neural Control Systems Design

This part of the paper illustrates the application of the RCVNN for direct I-Term adaptive neural control scheme of a nonlinear oscillatory plant. The nonlinear oscillatory plant model under investigation is a flexible – joint robot of two degree of freedom (DOF) that actually is a system with four DOF due to the flexible-joints. The model, the output and the input of the plant are given in continuous time. In order to use a recurrent neural network to control, the output/input signals of the plant are discretized with a sampling time T_o.

All neural networks used for system identification and control follow the topology mentioned above. The dimensions of each NN will be defined for the three different control schemes. The first control scheme under consideration is the direct feedforward adaptive control scheme with I-term.

ISBN: 978-1-61804-244-6

A. Direct Feedforward Adaptive Neural Control with I-Term

The control scheme is depicted in Fig.3. The control objective is that the complex weight parameters of the RCVNN are adjusted in such manner that the RCVNN converges to the inverse of the plant in order that the output of the plant follows de reference vector.

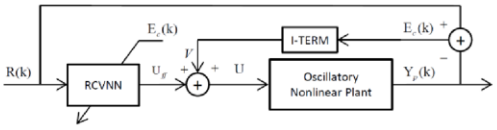


Fig.3 Block-diagram of the direct feedforward adaptive neural control system with I-Term

The RCVNN converges to the inverse model of the plant, trained by the control error: $E_c = Y_p - R$. Here output of the RCVNN is the vector control U so, if the RCVNN converges to the inverse model of the plant, the system output will follow the reference. In order to eliminate the steady-state error an I-Term is added to the control. The I-Term of the error is defined as:

$$V(k+1) = V(k) + T_o k_i E_c \tag{15}$$

Where:
T_o: Sampling time.
k_i: Integral action gain.
E_c: Control error.

So, the total control is defined by the sum of the feedforward and the I-Term parts, as it is:

$$U = U_{ff} + V \tag{16}$$

The stability of the whole system is assured by the boundedness of the activation function, and at the same time by the weight restricted condition for A, given in (3).

B. Direct Feedforward Adaptive Neural Control with State Feedback

The block-diagram of the control system is given in Fig.4. The control scheme contained three RCVNNs of the given up type: one RCVNN of plant identification and state estimation, one Feedforward control RCVNN, and one Feedback control RCVNN. The topologies of the three RCVNN are equal but they possessed different dimensions. For system identification, the desired complex target vector is the output of plant. The identification objective is that the complex weight parameters of the RCVNN-1 are adjusted in such manner that the RCVNN-1 output follows the plant output with minimum MSE.

The RCVNN-1 plant identifier and state estimator is learned by the identification error $(E_i = Y_p - Y,)$. The RCVNN-2 and RCVNN-3 are feedback and feedforward neural controllers, respectively, both learnt by the control error $(E_c = R - Y_p)$.

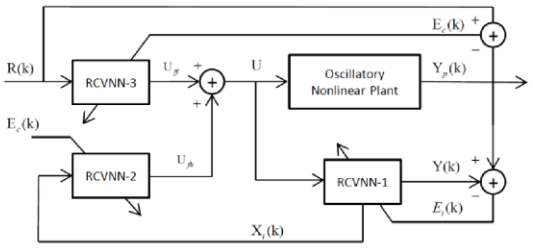


Fig.4 Block-diagram of the direct adaptive neural control system with state feedback

The control vector is sum of both control RCVNN tions U_{fb}, U_{ff}, generated by the corresponding RCVNN-2, 3 controllers.

C. Direct Feedforward Adaptive Neural Control with I-Term and State Feedback

The block-diagram of the control system is given on Fig.5. The control scheme contained three RCVNNs of the given up type: one RCVNN of plant identification and state estimation, one Feedforward control RCVNN, and one Feedback control RCVNN. To eliminate steady-state errors, an I-Term of the control error, (15), is added.

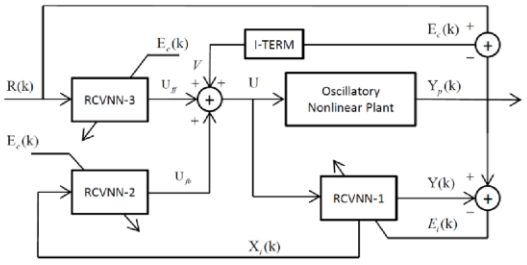


Fig. 5 Block-diagram of the direct adaptive neural control system with I-Term and state feedback

The control vector is sum of both control RCVNN tions $(U_{fb}, U_{ff} -$ outputs of the corresponding RCVNN controllers) and V- the integral term, respectively.

D. Description of the Nonlinear Plant Model

The dynamic model of flexible-joint robot was developed due to the use of harmonic drives, which is a type of robot gear mechanism with high torque transmission, low backlash and compact size. The elastic coupling of the *i-th* joint is modeled as a linear torsional spring of constant stiffness K_i, The *i-th* elastic joint of a revolute robot is schematically shown in Fig.6.

The system is a robot of two DOF showed in Fig.7. It is considered that each joint is flexible of the type illustrated by Fig.6. The flexible joint robot model consists of an actuator

ISBN: 978-1-61804-244-6

connected to a load through a torsional spring representing the joint flexibility.

To simplify, the model of the flexible joint robot, it is derived under the following assumptions, [12]:

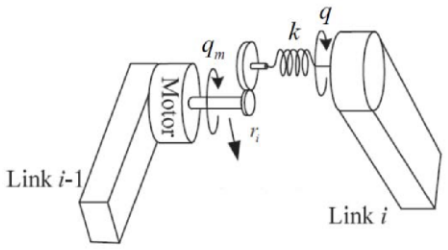


Fig. 6 Idealized model representing *i-th* elastic joint flexibility of a revolute robot

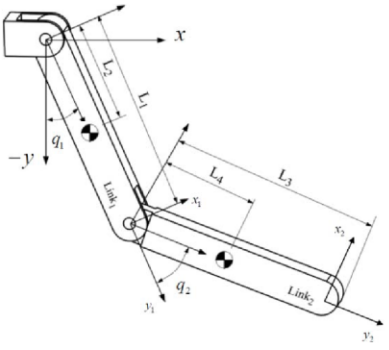


Fig. 7 Two- DOF- robot with flexible joints

a) The Kinetic energy of the rotor is due to its own rotation. Equivalently, the motion of the rotor is a pure rotation with respect to an inertial frame.
b) The rotor/gear inertia is symmetric about the rotor axis of rotation so that the gravitational potential of the system and the velocity of the rotor center of the mass are both independent of the rotor position.

Under these assumptions, the equations of motion of the flexible - joint robot are given as follows.

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + K(q - q_m) = 0$$
$$J\ddot{q_m} - K(q - q_m) = u \qquad (17)$$

Where: q and $q_m \epsilon \mathbb{R}^n$ denote the angular displacement of the links and the motor shaft, respectively; $D(q)$: $\mathbb{R}^n \rightarrow \mathbb{R}^{nxn}$ is the symmetric positive definite inertia trix; $C(q,\dot{q})$: $\mathbb{R}^n x \mathbb{R}^n \rightarrow \mathbb{R}^n$ represents the Coriolis and centrifugal forces; $G(q)$: $\mathbb{R}^n \rightarrow \mathbb{R}^n$ is the gravitational force vector of the rigid links; $K \epsilon \mathbb{R}^{nxn}$ is the diagonal positive definite spring constant matrix of the flexible joints; $J \epsilon \mathbb{R}^{nxn}$ is the moment of inertia matrix of the motor, and $u \epsilon \mathbb{R}^n$ is the exogenous input torque vector.

If we define an extended vector $q_f = [q^T q_m^T]^T$ we can rewrite the two equations above in the following matrix form:

$$D_f(q)\ddot{q_f} + C_f(q,\dot{q})\dot{q_f} + G_f(q) + K_f q_f = u_f \qquad (18)$$

Where:

$$D_f(q) = \begin{bmatrix} D(q) & 0 \\ 0 & J \end{bmatrix}, \quad C_f(q,\dot{q}) = \begin{bmatrix} C(q,\dot{q}) & 0 \\ 0 & 0 \end{bmatrix}$$

$$G_f(q) = \begin{bmatrix} G(q) \\ 0 \end{bmatrix}, \quad K_f = \begin{bmatrix} K & -K \\ -K & K \end{bmatrix}, \quad u_f = \begin{bmatrix} 0 \\ I \end{bmatrix} u$$

The properties of this model are mentioned in [12]. As we can see, the plant is an oscillatory system, described by four second order differential equations, representing a system with four degrees of freedom but only two inputs which makes the system sub-actuated.

E. Simulation Results

This section described the simulation results, obtained using the given up three schemes of adaptive neural control. As a measure of comparison we use the final MSE of the output variables for the two schemes of adaptive neural control.

Direct Feedforward Adaptive Neural Control with I-Term. The simulation was executed with the following RCVNN configuration: $m = 2, n = 6, p = 2$. The dimension of the hidden RCVNN layer was chosen by trial and error, based on the step response of the system. The same was done for the gain of the I-Term. The graphical results of the first neural control scheme outputs are given on Fig.8 for the first link and in Fig.9 for the second link. The output of the plant is compared with a constant stepwise reference. The reference signal is different for each of the links. It can see that the step response presented oscillations at the beginning, and then converge to the reference.

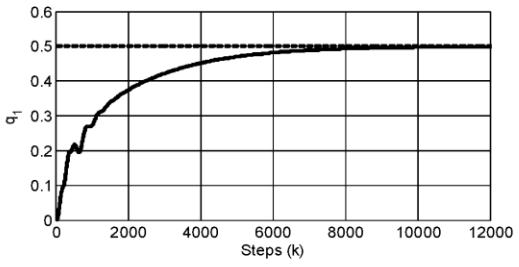


Fig. 8 Graphical simulation results for ***the plant outut*** q_1, controlled by the first control scheme; plant output (continued line); system reference (dashed line)

ISBN: 978-1-61804-244-6

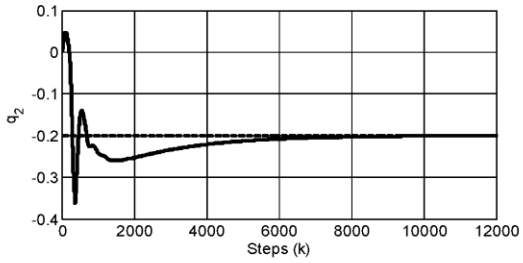


Fig. 9 Simulation results of q_2, controlled by the first control scheme; plant output (continue line); system reference (dashed line)

Direct Feedforward Adaptive Neural Control with State Feedback. This scheme does not implement I-Term. The simulation was executed with the following NN configurations: RCVNN-1 has dimensions: $m = 2, n = 6, p = 2$; the dimension of the hidden layer of RCVNN-1 was chosen by trial and error, using the fact that the whole system has order eight; RCVNN-2 has dimensions: $m = 2, n = 4, p = 2$; the dimension of the hidden layer of RCVNN-2 was chosen by trial and error; RCVNN-3 has dimensions: $m = 2, n = 6, p = 2$; the dimension of the hidden layer of RCVNN-3 was chosen by trial and error.

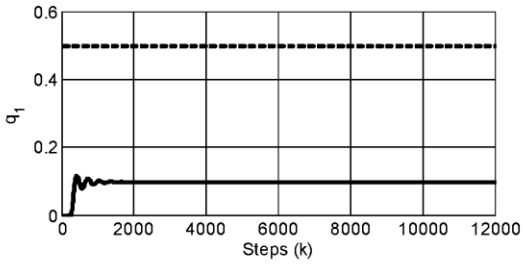


Fig. 10 Simulation results of q_1, controlled by the second control scheme; plant output (continue line); system reference (dashed line)

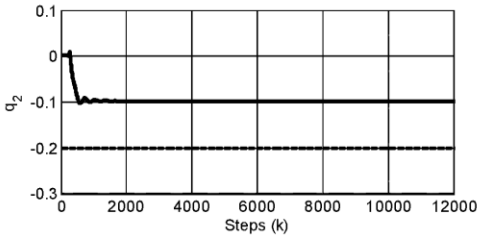


Fig. 11 Simulation results of q_2,controlled by the second control scheme; plant output (continue line); system reference (dashed line)

The graphical results of the second neural control scheme plant outputs are given on Fig.10 for the first link and in Fig.11 for the second link. Each output of the plant is compared with a constant reference. The reference is different for each of the links. It could be seen that the responses of the plant outputs presented oscillations at the beginning and generate static errors; which is due to the control signal behavior and the lack of I-term.

Direct Feedforward Adaptive Neural Control with I-Term and State Feedback. The simulation was executed with the following RCVNN configurations: RCVNN-1 has dimensions $m = 2, n = 6, p = 2$. The number of the hidden neurons was chosen by trial and error, based on the step response of the system and using the fact that the whole system has order eight; RCVNN-2 has dimensions $m = 2, n = 4, p = 2$. The number of the hidden neurons was chosen by trial and error based on the step response of the system; RCVNN-3 has dimensions $m = 2, n = 6, p = 2$. The number of the hidden neurons was chosen by trial and error, based on the step response of the system, also the gains of the I-term. The graphical results of the third neural control scheme are given on Fig.12 for the first link output and in Fig.13 for the second link output. The respective output of the plant is compared with a constant reference. The reference is different for each one of the links. It could be seen that each response does not present oscillation at the beginning which is achieved applying a state feedback control.

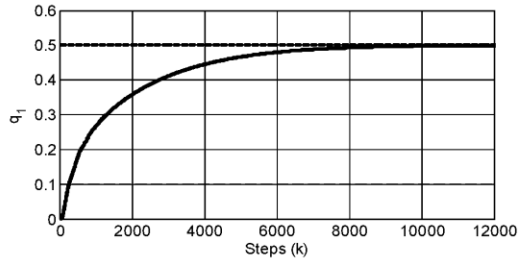


Fig. 12 Simulation results ofq_1,controlled by the third control scheme; plant output (continue line); system reference (dashed line)

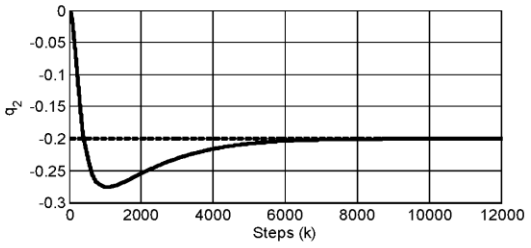


Fig.13 Simulation results ofq_2, controlled by the third control scheme, plant output (continue line); system reference (dashed line)

Comparative final MSE simulation results, obtained by the three adaptive control schemes. The MSE values obtained at

ISBN: 978-1-61804-244-6

the final of the simulation (12000 iteration steps) with the three adaptive neural control schemes are shown on Table I. It could be seen that the use of the state feedback affects the step response of the system, decreasing the oscillations of the plant output step responses. Also the MSE decreases more quickly and converges to a small final value.

Table I. Final MSE of adaptive neural control schemes

DOF	Direct Feed-forward Adaptive Neural Control with I-Term	Direct Feedforward Adaptive Neural Control with State Feedback.	Direct Feed-forward Adaptive Neural Control with I-Term and State Feedback.
q_1	0.0088	0.1629	0.0090
q_2	0.0010	0.0109	0.0007

In Table I, we can see that the first control scheme and the last control scheme have smaller MSE with respect to the second control scheme, but the suppression of the oscillations is greater using the state feedback. So we could conclude that the third control scheme containing state feedback and I-term have better behavior. Also the I-Term reduces the error in steady state.

IV. Conclusions

In the present paper we purposed three schemes for direct adaptive control using RCVNNs. We applied diagrammatic rules for the complex value case in order to derive the back-propagation training algorithm of the RCVNN topology. Comparative simulation results of flexible-joint robot model using the three schemes of direct complex value control, are obtained. These results lead us to the conclusion that the presence of the state feedback decreases the oscillations in the output step response, and the I-Term reduces the error in steady state. As we can see in the Table I, the MSE is almost equal between the scheme one and the scheme three, but the suppression of oscillations for the third control scheme is considerable. The obtained results confirmed the good quality of the proposed control methodology.

References

[1] A. Hirose, *Complex-Valued Neural Networks*, 2nd ed., S. i. C. Intelligence, Springer Verlag, 2012, vol. 400.

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

[3] L. Ferariu, "Nonlinear System Identification Based on Evolutionary Dynamic Neural," in *Proc. of European Control Conference*, Cambridge, UK, 2003.

[4] K. Kawashima and T. Ogawa, "Complex-Valued Neural Network for Group-Movement Control of Mobile Robots," in *Proc. SICE Annual Conference 2012*, Japan, 2012.

[5] A. Hirose, "Motion Controls Using Complex-Valued Neural Networks with Feedback Loops," in *Proc. IEEE International Conference on Neural Networks*, vol. 1, San Francisco, CA, 1993, pp. 156-161.

[6] H. Leung and S. Haykin, "The Complex Backpropagation Algorithm," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2101-2104, 1991.

[7] T. Nitta, *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*. IGI Global, 2009.

[8] C. Woo and D. S. Hong, "Adaptive Equalization Using the Complex Backpropagation Algorithm," in *Proc. of IEEE International Conference on Neural Networks*, vol. 4, Washington, DC, 1996, pp. 2136-2141.

[9] N. Miklos and B. Salik, "Neural Networks with Complex Activations and Connection Weights," *Complex Systems*, vol. 8, pp. 115-126, 1994.

[10] G. Georgiou and C. Koutsougeras, "Complex Domain Backpropagation," *IEEE Transactions on Circuits and Systems-11: Analog and Digital Signal Processing*, vol. 39, no. 5, pp. 330-334, 1992.

[11] V. M. Arellano-Quintana and I. S. Baruch, "Identification of Dynamical Systems Using Recurrent Complex-Valued Neural Networks," in *18th International Conference on Circuits, Systems, Communications and Computers* , Santorini, Greece, 2014 (accepted for presentation).

[12] S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators*, C. J. Harris, Ed. UK: World Scientific Publishing Co. Pte. Ltd., 1998, vol. 19.

ISBN: 978-1-61804-244-6