Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias

"Criterios de diseño del Controlador Neuronal Autoajustable y su aplicación a un Sistema No Lineal"

TESIS

Que como parte de los requisitos para obtener el grado de Maestro en Ciencias
Línea Terminal: Instrumentación y Control Automático.

Presenta:
Fís. Alfonso Gómez Espinosa.

Dirigido por:
M.C. Alfonso Noriega Ponce.

SINODALES

M.C. Alfonso Noriega Ponce. Presidente	Firma
M.C. Victor M. Hernández Guzmán. Secretario	Firma
Dr. Ismael Espinosa Espinosa. Vocal	Firma
Dr. Arturo Zavala Rio. Suplente	Firma
Dr. Eduardo Castillo Castañeda. Suplente	Firma
M.I. José Jesús Hernández Espino. Nombre y Firma Director de la Facultad	Dra. Guadalupe Bernal Santos. Nombre y Firma Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
23 de Agosto de 1999.
México.

No Adq H61587
No. Título
Clas. 629.836
G633c
Ej 01

BIBLIOTECA CENTRAL UAQ
"ROBERTO RUIZ OBREGON"

A mi esposa,
Adriana Sierra Castillo.

A mis hijos,
Andrea y Arturo.

AGRADECIMIENTOS

Deseo expresar mi gratitud al M.C. Alfonso Noriega por su apoyo y confianza durante las distintas etapas de este trabajo.

Al Dr. Alberto Aguado por introducirme en el campo de las Redes Neuronales y por el genuino interés que mostró en el desarrollo de los experimentos.

Al Dr. Eduardo Castillo por facilitarme una amplificador con servomotor y esperar pacientemente a que se lo regresara al término de los experimentos.

Al M.C. Fernando Luengas por explicarme los conceptos de programación en "C" que utilicé para implementar los algoritmos de este trabajo.

Al Ing. Daniel Aguilera Longoria por la revisión preliminar de este documento.

Al técnico Arturo Martínez por el tiempo dedicado en la construcción del sistema mecánico que se utilizó para probar el Controlador Neuronal Autoajustable.

A CONACYT por el apoyo que me brindaron para realizar esta tesis.

A la Fundación TELMEX por distinguirme con su reconocimiento.

Al Centro de Investigación y Desarrollo Condumex por el apoyo recibido.

Finalmente, quiero agradecer a los maestros que intervinieron en mi formación académica, así como a los parientes y amigos que de alguna forma contribuyeron para la realización de este trabajo.

GRACIAS.

RESUMEN

El funcionamiento de un sistema de control de procesos industriales, equipado con un controlador convencional, puede verse degradado por retardos del sistema, zonas muertas, saturación de los mecanismos actuadores, incertidumbre en la estimación de los parámetros y ruido. Recientemente se demostró que las Redes Neuronales pueden ser utilizadas para controlar Sistemas No Lineales y se planteó la posibilidad de implementar un Controlador Neuronal Autoajustable (conectado en serie con un proceso real); en el que se sustituye la etapa de aprendizaje por una adaptación permanente, en tiempo real, de los coeficientes de peso. En el presente trabajo, se muestran los resultados obtenidos durante la implementación práctica del Controlador Neuronal Autoajustable en tres Sistemas No Lineales y se propone una metodología para ajustar sus parámetros. Como parte de la propuesta, se incluye un método para acelerar el aprendizaje de la red; el cual consiste en acotar el espacio de valores que pueden tomar las salidas de las neuronas y en utilizar un coeficiente de aprendizaje variable en el tiempo.

Palabras claves:

Redes Neuronales, Retropropagación, Control No Lineal, Control Autoajustable.

ABSTRACT

The performance of an industrial process control system equipped with a conventional controller may be severely degraded by a long system-time delay, dead zone and/or saturation of actuator mechanisms, model and/or parameter uncertainties, and process noises. Recently it has been demonstrated that Artificial Neural Networks can be used to control Non Linear Systems and it was also proposed that a Self-tuning Neural Network implementation is possible with a real time permanent adjustment of the weighting coefficients based on the control error. In the present thesis, the Self-tuning Neural Network practical implementation results in three Non Linear Systems are shown and a methodology to adjust the parameters is proposed. As part of the proposal we present a practical method to accelerate net learning by bounding neuron outputs and a time variable learning coefficient.

Keywords:

Neural Networks, Back-propagation, Non Linear Systems, Adaptive Control.

ÍNDICE

1) INTRODUCCIÓN Y OBJETIVOS

La sección mejor desarrollada de la Teoría Matemática de Sistemas, se ocupa del análisis y síntesis de los sistemas definidos por operadores lineales, usando técnicas bien establecidas que se basan en el Álgebra Lineal, la Teoría de Variable Compleja y la Teoría de Ecuaciones Diferenciales Lineales Ordinarias. Como las técnicas de diseño para los sistemas dinámicos están fuertemente relacionadas a sus propiedades de estabilidad y como las condiciones necesarias y suficientes para asegurar la estabilidad de los Sistemas Lineales Invariantes en el tiempo se encuentran desarrolladas, se han establecido métodos de diseño bien definidos para esos sistemas. En contraste, la estabilidad de los Sistemas No Lineales, en su mayor parte, sólo puede establecerse para cada sistema en particular y por lo tanto, no es sorprendente que los procedimientos de diseños que simultáneamente cumplen los requisitos de estabilidad, robustez y buena respuesta dinámica, actualmente no están disponibles para todas las clases de estos sistemas.

En el presente trabajo, se presentan los resultados obtenidos durante la implementación práctica del Controlador Neuronal Autoajustable en tres Sistemas No Lineales y se propone una metodología para ajustar sus parámetros. Como parte de la propuesta, se incluye un método para acelerar el aprendizaje de la red; el cual se basa en acotar los valores de los coeficientes de peso y utilizar un coeficiente de aprendizaje variable en el tiempo.

Esta tesis está organizada en la forma siguiente: En el capítulo 2) se presenta un panorama histórico con los trabajos que motivaron esta tesis. El capítulo 3) recopila los conceptos básicos sobre Redes Neuronales. En el capítulo 4) se muestra la estructura del Controlador Neuronal Autoajustable propuesto por Aguado et al, se introduce la propuesta de Cui y Shin para ajustar los coeficientes de peso de la red (de forma consistente con lo planteado para la estructura definida), se propone un método para acelerar la convergencia del algoritmo de Retropropagación; acotando el contradominio de la función de activación de las neuronas, se describen los sistemas No Lineales en los que se probó el controlador y se presenta los resultados obtenidos con distintos parámetros de entrenamiento. Por último, en el capítulo 5),

se hace un resumen de los resultados obtenidos, se propone una metodología para ajustar los parámetros del controlador y se dan algunos lineamientos para trabajos futuros.

2) ANTECEDENTES HISTÓRICOS

Los modelos de Redes Neuronales Artificiales (RNA) han sido estudiados por varios años con la esperanza de alcanzar resultados similares a los de los humanos en áreas como el reconocimiento de imágenes, voz, predicción, codificación, control y optimización; pero no es sino hasta fechas recientes que se tienen resultados prácticos considerables. Éstos modelos se componen por varios elementos que realizan operaciones matemáticas no lineales, operan en paralelo y están arreglados en estructuras que pretenden simular a las redes neuronales de los seres vivos. Los elementos que realizan las operaciones matemáticas (nodos) se conectan entre sí mediante coeficientes de peso que en general son ajustados durante su operación para mejorar su desempeño y de esta forma contar con capacidad de [13]: aprender de experiencias, generalizar de casos anteriores a nuevos casos, abstraer características esenciales a partir de entradas que representan información irrelevante, tolerancia a fallos, etc.

Warren McCulloch y Walter Pitts, en1943, lanzaron una teoría acerca de la forma en que trabajan las neuronas y modelaron una neurona simple mediante circuitos eléctricos. Posteriormente, en 1957, Frank Rosenblatt presentó el primer modelo del Perceptron; el cual, a pesar de tener una serie de limitaciones que no le permiten reconocer caracteres complejos, es útil para el reconocimiento automático de algunos patrones. En 1959, Bernard Widrow y Marcial Hoff desarrollaron el modelo ADALINE que fue usado en la construcción de filtros adaptivos para eliminar ecos en las líneas telefónicas. Finalmente, dentro de esta primer etapa, en 1967 Stephen Grossberg propuso la red "Avalancha" para enfrentar problemas tales como el reconocimiento continuo del habla y el aprendizaje del movimiento de un robot.

A pesar de los primeros logros de esta área de investigación, en el periodo comprendido entre 1969 y 1982, surgieron numerosas críticas (como la dificultad para reconocer patrones simples) que frenaron el crecimiento que estaba experimentando. [10] La aparición del libro

"Perceptrons", escrito por Marving Minsky y Seymour Papert, suele tomarse como causa del fallecimiento de esta tecnología; en esta obra, presentaron un análisis matemático detallado del Perceptron y consideraron (equivocadamente) que no era de utilidad extender su estructura a varios niveles (suponían que los resultados serían equivalentes a los de una sola capa). Durante el periodo de escepticismo, algunos investigadores continuaron desarrollando nuevos modelos como el "Asociador Lineal" de James Anderson, el "Neocognitrón" de Kunihiko Fukushima y la "Memoria de Hopfield" de John Hopfield. La aparición de este último trabajo, en 1982, coincide con el resurgimiento de la investigación sobre Redes Neuronales Artificiales; fruto de las teorías y resultados obtenidos de 1969 a 1982. Actualmente son numerosos los trabajos que se realizan y publican cada año en distintos países.

Es importante notar que los problemas a los que se hace referencia, en muchas recopilaciones sobre Redes Neuronales Artificiales, no están relacionados directamente con aplicaciones de control, sino con el procesamiento de señales y en particular con el reconocimiento de patrones [06] [07] [22]. Adicionalmente, se encontró que las recopilaciones en las que se aborda el problema del Control Automático, parten de suponer un entrenamiento previo de la red [19] [23]. Con lo anterior se pretende mostrar que, aún cuando la teoría de Redes Neuronales Artificiales se inició hace más de 50 años, no es sino hasta fechas recientes (1990, 1993 y 1997), que se abre la posibilidad de utilizar esta tecnología para implementar Controladores Directos con la capacidad de aprender a controlar un Sistema No Lineal sin contar con un entrenamiento previo.

Narendra y Parthasarathy [15], en 1990, demostraron que las Redes Neuronales pueden ser utilizadas efectivamente para la identificación y control de los Sistemas Dinámicos No Lineales. En este trabajo presentan estructuras de redes neuronales dentro de un esquema de control indirecto y comentan la falta de métodos para ajustar directamente los coeficientes de peso del controlador basándose en el error de salida (entre las salidas de la planta y del modelo de referencia). Basados en los resultados obtenidos en simulaciones, mencionan que el esquema sugerido se puede implementar en la práctica.

Posteriormente en 1993, Cui y Shin [24] proponen un algoritmo que permite ajustar directamente los parámetros del controlador neuronal y demuestran que es suficiente conocer el signo de la función de transferencia de la planta para entrenar la red con el error de regulación (en lugar del error de salida de la red). Con esto se abre la posibilidad de sintetizar controladores que no requieren la identificación previa de la planta. En ese trabajo no se discute la influencia de algunos parámetros de entrenamiento sobre la dinámica a lazo cerrado del sistema ni tampoco se menciona la posibilidad de sustituir el periodo de entrenamiento por una adaptación permanente, en tiempo real, de los coeficientes de peso de la red.

Recientemente, Aguado, Ordáz, Noriega y Rauch [03], proponen una estructura de la red neuronal en la que la etapa de aprendizaje es sustituida por una adaptación permanente, en tiempo real, de los coeficientes de peso. Después de realizar una prueba exhaustiva del algoritmo, mediante la simulación en tiempo real de múltiples Sistemas No Lineales, concluyen que en general no se requiere un entrenamiento previo de la red y que el comportamiento dinámico, a lazo cerrado, depende exclusivamente de la magnitud del coeficiente de aprendizaje. Finalmente comentan que, durante las simulaciones, resultó conveniente usar un coeficiente de aprendizaje variable ($\eta' = \eta + \alpha \text{ abs}[e_y]$), para acelerar la respuesta cuando el error es grande.

3) FUNDAMENTOS DE LAS REDES NEURONALES ARTIFICIALES

3.1) MODELO BIOLÓGICO

ESTRUCTURA DE UNA NEURONA

En una célula nerviosa típica perteneciente al sistema nervioso central, la membrana de la neurona separa el plasma intracelular del fluido intersticial que se encuentra fuera de la célula. La membrana es permeable para ciertas especies iónicas, y actúa de tal forma que se mantenga una diferencia de potencial entre el fluido intracelular y el fluido extracelular. Este efecto se consigue, principalmente, mediante la acción de una bomba de sodio – potasio. También están presentes otras especies iónicas como son los iones cloruro e iones orgánicos negativos.

Todas las especies iónicas se pueden difundir a través de la membrana, con la excepción de los iones orgánicos, que son demasiado grandes. Dado que los iones orgánicos no pueden salir de la célula por difusión, su carga negativa neta dificulta la entrada en la célula de iones cloro por difusión; por tanto, habrá una concentración más alta de iones cloro fuera de la célula. La bomba de sodio – potasio determina una concentración más alta de potasio dentro de la célula y una concentración más alta de sodio fuera de ella.

La membrana celular es selectivamente más permeable para los iones de potasio que para los iones de sodio. El gradiente químico de potasio tiende a hacer que los iones de potasio salgan de la célula por difusión, pero la fuerte atracción de los iones orgánicos negativos tiende a mantener dentro el potasio. El resultado de estas fuerzas opuestas es que se alcanza un equilibrio en el cual hay más iones de sodio y cloro fuera de la célula, y más iones orgánicos y de potasio dentro de ella. Además, el equilibrio resultante produce una diferencia de potencial a través de la membrana de la célula de unos 70 mV a 100 mV, siendo el más negativo el fluido intracelular. Este potencial, que se denomina potencial de reposo de la célula, se ha representado esquemáticamente en la Figura 3.1

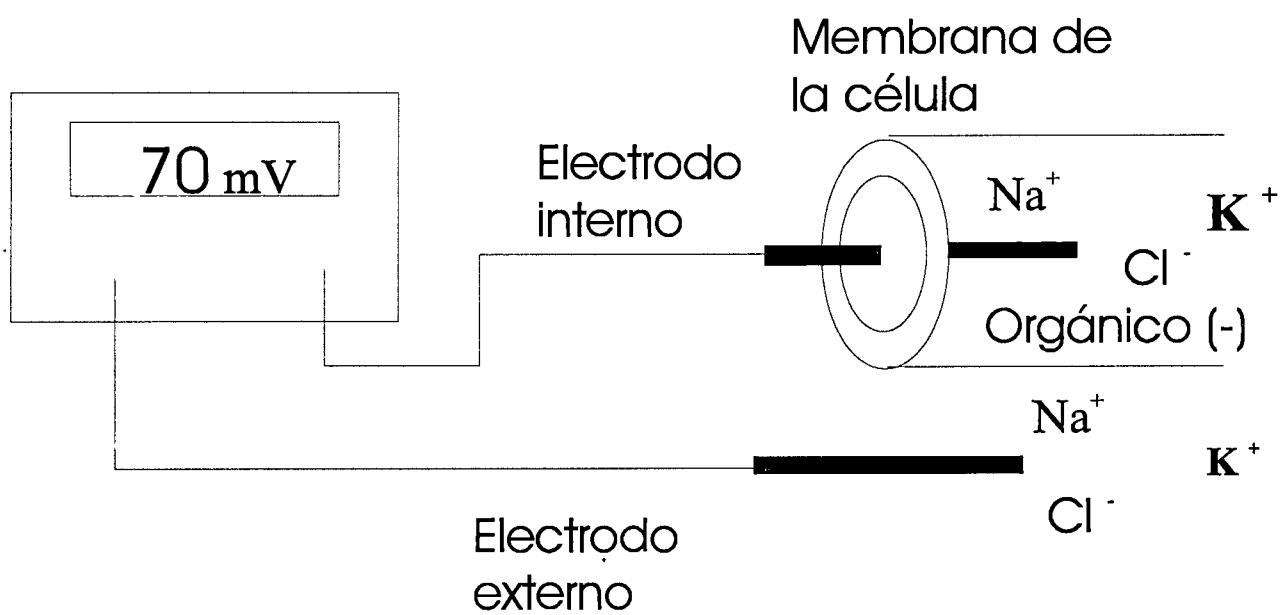


Figura 3.1 Potencial de reposo en ambos lados de la membrana de una neurona.

Las neuronas cuentan con múltiples conexiones de entrada en las que se reciben los potenciales provenientes de otras células. Otra característica importante es la de que el axón cuenta con una cubierta que se denomina vaina de mielina. Esta capa, de naturaleza aislante, es interrumpida en varios puntos por los nodos de Ranvier.

Las entradas excitatorias que llegan a la célula reducen la diferencia de potencial que existe entre los dos lados de la membrana celular. La despolarización resultante en el montículo del axón altera la permeabilidad de la membrana celular a efectos de los iones de sodio. Como resultado hay un fuerte flujo entrante de iones sodio positivos, que penetran en la célula, contribuyendo aún más a la despolarización. Este efecto autogenerado da lugar al potencial de acción.

Las fibras nerviosas en sí son malos conductores. La transmisión del potencial de acción a lo largo del axón es el resultado de una serie de despolarizaciones que tienen lugar en los nodos de Ranvier. Cuando uno de los nodos se despolariza, se desencadena la despolarización del siguiente nodo. El potencial de acción viaja a lo largo de la fibra en forma discontinua, de un nodo a otro. Una vez que un potencial de acción ha pasado por un cierto punto, ese punto no puede volver a ser excitado durante cosa de 1 milisegundo, que es el tiempo que tarda en volver a su potencial de reposo. Este período refractario limita la frecuencia de transmisión de los impulsos nerviosos a unos 1,000 por segundo.

Examinemos brevemente la actividad que se desarrolla en la unión existente entre dos neuronas, que se denomina unión sináptica o sinapsis. La comunicación entre las neuronas tiene lugar como resultado de la liberación de unas sustancias llamadas neurotransmisores por parte de la célula presináptica, al ser absorbidas estas sustancias por la célula postsináptica. Cuando el potencial de acción llega a la membrana presináptica, los cambios de permeabilidad de la membrana dan lugar a un flujo entrante de iones de calcio. Estos iones dan lugar a que las vesículas que contienen los neurotransmisores se fundan con la membrana sináptica, liberando así sus neurotransmisores en la separación sináptica.

Los neurotransmisores se difunden a través de la unión y se unen a la membrana postsináptica en ciertos lugares llamados receptores. La acción química que se produce en los receptores da lugar a cambios de permeabilidad de la membrana postsináptica para ciertas especies iónicas. Un flujo entrante de especies positivas hacia la célula tenderá a despolarizar el potencial de reposo; este efecto es excitatorio. Si entran iones negativos, se producirá un efecto hiperpolarizante; este efecto es inhibitorio. Estos dos efectos son locales, y actúan tan sólo a lo largo de una pequeña distancia hacia el interior de la célula; se suman en el montículo del axón. Si la suma es mayor que un cierto valor umbral se genera un potencial de acción

APRENDIZAJE DE HEBB.

Los sistemas neuronales biológicos no nacen preprogramados con todo el conocimiento y las capacidades que llegarán a tener eventualmente. Un proceso de aprendizaje que tiene lugar a lo largo de un período de tiempo modifica de alguna forma la red para incluir la nueva información.

Exploraremos una teoría relativamente sencilla del aprendizaje que sugiere una respuesta elegante para esta pregunta: ¿Cómo aprendemos?. La teoría básica procede de un libro de 1949 escrito por Hebb, *Organization of Behavior.* La idea principal se expresaba en forma de suposición, y la reproducimos aquí por su interés histórico:

"Cuando un axón de la célula A está suficientemente próximo para excitar a una célula B o toma parte en su disparo de forma persistente, tiene lugar algún proceso de crecimiento o algún cambio metabólico en una de las células, o en las dos, de tal modo que la eficiencia de A, como una de las células que desencadena el disparo de B, se ve incrementada".

Al igual que otros modelos del aprendizaje, el propuesto por Hebb, no cuenta toda la historia. Sin embargo, aparece de una forma o de otra en muchos de los modelos de redes neuronales que existen en la actualidad. Dado que la conexión entre neuronas se hace a través de las sinapsis, es razonable suponer que cualesquiera cambios que puedan tener durante el aprendizaje deberán producirse en ellas. Hebb sostenía la teoría de que aumentaba el área de la unión sináptica. Teorías más recientes afirman que el responsable es un incremento de la

velocidad con que se libera el neurotransmisor en la célula presináptica. En todo caso, ciertamente hay cambios que se producen en la sinapsis. Tanto la célula presináptica como la postsináptica resultarán alteradas en su totalidad, de forma que se podrían reforzar otras respuestas que no estuviesen relacionadas con el experimento condicionante.

3.2) ELEMENTOS DE LAS REDES NEURONALES ARTIFICIALES

NEURONA ARTIFICIAL

Las redes neuronales son modelos que intentan reproducir el comportamiento del cerebro. Como tal modelo, realiza una simplificación, averiguando cuáles son los elementos relevantes del sistema, bien porque la cantidad de información de que se dispone es excesiva o bien porque es redundante. Una elección adecuada de sus características, más una estructura conveniente, es el procedimiento convencional utilizado para construir redes capaces de realizar una determinada tarea.

Cualquier modelo de red neuronal consta de dispositivos elementales de proceso: las *neuronas*. A partir de ellas, se pueden generar representaciones específicas, de tal forma que un estado conjunto de ellas puede significar una letra, un número o cualquier otro objeto. En esta sección se realiza la idealización del funcionamiento neurobiológico descrito anteriormente, que sirve de base de las redes neuronales artificiales.

La neurona artificial pretende mimetizar las características más importantes de las neuronas biológicas. Cada i-ésima neurona está caracterizada en cualquier instante por un valor numérico denominado *valor o estado de activación* $a_i(t)$; asociado a cada unidad, existe una *señal de salida,* f_i , que transforma el estado actual de activación en una *señal de salida,* y_i . Dicha señal es enviada a través de los canales de comunicación unidireccionales a otras unidades de la red; en estos canales la señal se modifica de acuerdo con la sinapsis (el *peso,* w_{ji}) asociada a cada uno de ellos según una determinada regla. Las señales modulares que han llegado a la unidad j-ésima se combinan entre ellas generando así la *entrada total,* Net_j.

$$Net_j = \sum_i y_i w_{ij}$$

Una función de activación, F, determina el nuevo estado de activación $a_j(t+1)$ de la neurona, teniendo en cuenta la entrada total calculada y el anterior estado de activación $a_j(t)$.

La dinámica que rige la actualización de los estados de las unidades (evolución de la red neuronal) puede ser de dos tipos: modo asincrónico y modo sincrónico. En el primer caso, las neuronas evalúan su estado continuamente, según les va llegando información, y lo hacen de forma independiente. En el caso síncrono, la información también llega de forma continua, pero los cambios se realizan simultáneamente, como si existiera un reloj interno que decidiera cuándo deben cambiar su estado. Los sistemas biológicos quedan probablemente entre ambas posibilidades.

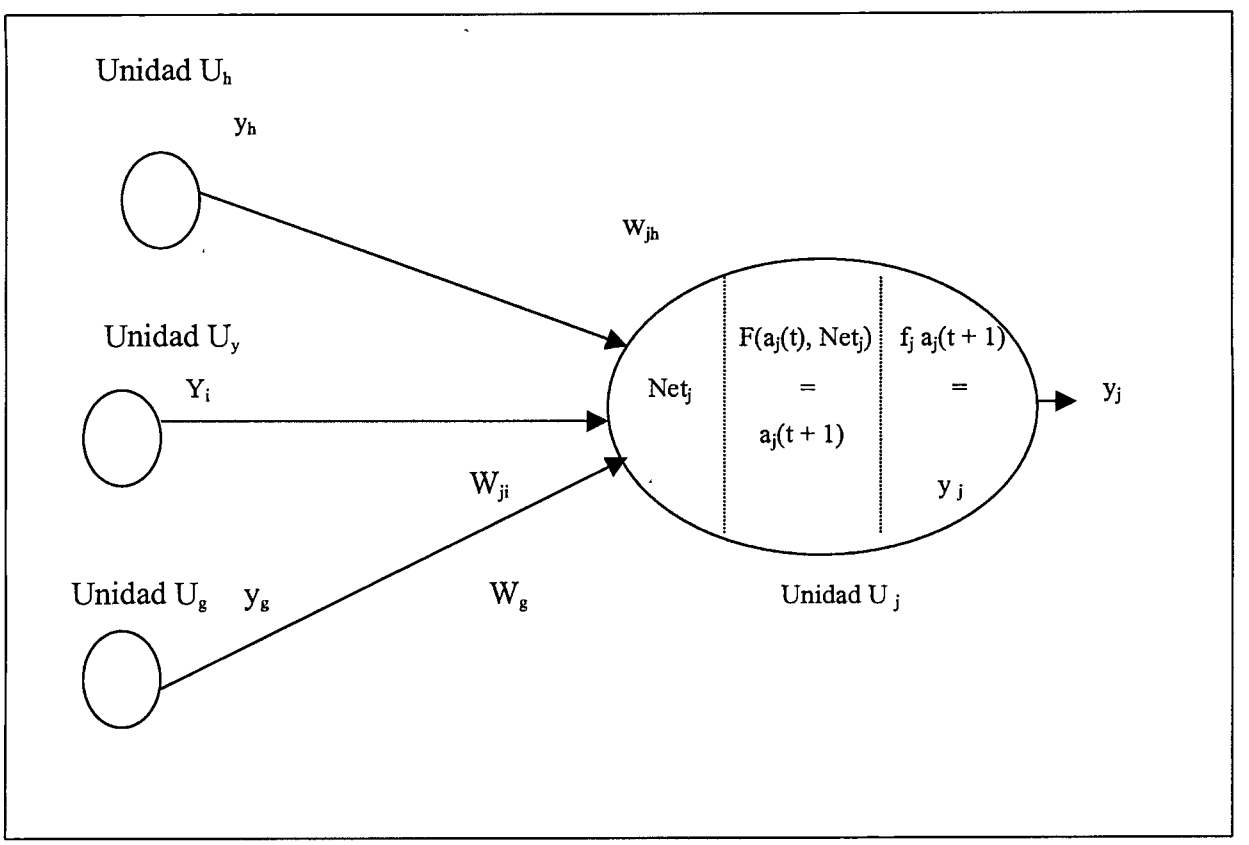


Figura 3.2 Entradas y salidas de una neurona U_j.

Si se tiene N unidades (neuronas), podemos ordenarlas arbitrariamente y designar la j-ésima unidad como U_j. Su trabajo es simple y único, y consiste en recibir las entradas de las células vecinas y calcular un valor de salida, el cual es enviado a las células restantes.

En cualquier sistema que se esté modelando, es útil caracterizar tres tipos de unidades: entrada, salida y ocultas. Las unidades de entrada reciben señales desde el entorno; estas entradas (que son a la vez entradas a la red) pueden ser señales provenientes de sensores o de otros sectores del sistema. Las unidades de salida envían la señal fuera del sistema (salida de la red); estas señales pueden controlar directamente actuadores u otros sistemas. Las unidades ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema, es decir, no tienen contacto con el exterior.

Se conoce como *capa* o *nivel* a un conjunto de neuronas cuyas entradas provienen de la misma fuente (que puede ser otra capa de neuronas) y cuyas salidas se dirigen al mismo destino (que puede ser otra capa de neuronas).

ESTADO DE ACTIVACIÓN

Adicionalmente al conjunto de unidades, la representación necesita los estados del sistema en un tiempo t. Esto se especifica por un vector de N números reales A(t), que representa el *estado de activación* del conjunto de unidades de procesamiento. Cada elemento del vector representa la activación de una unidad en el tiempo t. La activación de una unidad U_i en el tiempo t se designa por $a_i(t)$; es decir:

$$A(t) = (a_1(t), a_2(t), \ldots, a_i(t), \ldots, a_N(t))$$

El procesamiento que realiza la red se ve como la evolución de un patrón de activación en el conjunto de unidades que lo componen a través del tiempo.

Todas las neuronas que componen la red se hallan en cierto estado. En una visión simplificada, podemos decir que hay dos posibles estados, *reposo y excitado*, a los que denominamos globalmente *estados de activación,* y a cada uno de los cuales se le asigna un

BIBLIOTECA CENTRAL UAQ
"ROBERTO RUIZ OBREGON"

valor. Los valores de activación pueden ser continuos o discretos. Además, pueden ser limitados o ilimitados. Si son discretos, suelen tomar un conjunto pequeño de valores o bien valores binarios. En notación binaria, un estado activo se indicaría por un 1, y se caracteriza por la emisión de un impulso por parte de la neurona (potencial de acción), mientrás que un estado pasivo se indicaría por un 0, y significaría que la neurona está en reposo. En otros modelos se considera un conjunto continuo de estado de activación, en lugar de sólo dos estados, en cuyo caso se les asigna un valor entre (0,1) o en el intervalo (-1,1), generalmente siguiendo una función sigmoidal.

Finalmente, es necesario saber qué criterios o reglas siguen las neuronas para alcanzar tales estados de activación. En principio, esto va a depender de dos factores: a) Por un lado, puesto que las propiedades macroscópicas de las redes neuronales no son producto de actuación de elementos individuales, sino del conjunto como un todo, es necesario tener idea del mecanismo de interacción entre las neuronas. El estado de activación estará fuertemente influenciado por tales interacciones, ya que el efecto que producirá una neurona sobre otra será proporcional a la fuerza, peso o magnitud de la conexión entre ambas. B) Por otro lado, la señal que envía cada una de las neuronas a sus vecinas dependerá de su propio estado de activación.

FUNCIÓN DE SALIDA O DE TRANSFERENCIA

Entre las unidades o neuronas que forman una red neuronal artificial existe un conjunto de conexiones que unen unas a otras. Cada unidad transmite señales a aquellas que están conectadas con su salida. Asociada con cada unidad U_i hay una función de salida $f_i(a_i(t))$, que transforma el estado actual de activación $a_i(t)$ en una señal de salida $y_1(t)$: es decir:

$$y_1(t) = f_i(a_i(t))$$

El vector que contiene las salidas de todas las neuronas en un instante t es:

$$Y(t) = (f_1(a_1(t)), f_2(a_2(t)), \ldots, f_i(a_i(t)),\ldots, f_N(a_N(t)))$$

En algunos modelos, esta salida es igual al nivel de activación de la unidad, en cuyo caso la función, f_i es la función identidad, $f_i(a_i(t)) = a_i(t)$. A menudo f_i es de tipo sigmoidal, y suele ser la misma para todas las unidades.

Existen cuatro funciones de transferencia típicas que determinan distintos tipos de neuronas:
Función escalón
Función lineal y mixta
Sigmoidal
Función gaussiana

La función escalón o umbral únicamente se utiliza cuando las salidas de la red son binarias (dos posibles valores). La salida de una neurona se activa sólo cuando el estado de activación es mayor o igual que cierto valor umbral (la función puede estar desplazada sobre los ejes). La función lineal o identidad equivale a no aplicar función de salida. Se usa muy poco. Las funciones mixta y sigmoidal son las más apropiadas cuando queremos como salida función analógica. Veamos con más detalle las distintas funciones:

FUNCIÓN DE ACTIVACIÓN

Así como es necesario una regla que cambie las entradas a una neurona con los pesos de las conexiones, también se requiere una regla que combine las entradas con el estado actual de la neurona para producir un nuevo estado de activación. Esta función F produce un nuevo estado de activación en una neurona a partir del estado (a_i) que existía y la combinación de las entradas con los pesos de las conexiones (Net_j).

Dado el estado de activación $a_i(t)$ de la unidad U_i y la entrada total que llega a ella, Net_i, el estado de activación siguiente, $a_i(t + 1)$, se obtiene aplicando una función F, llamada *función de activación.*

$$a_i(t + 1) = F(a_i(t), Net_i)$$

En la mayoría de los casos, F es la *función identidad*, por lo que el estado de activación de una neurona en t + 1 coincidirá con el Net de la misma en t. En este caso, el parámetro que se le pasa a la función de salida, *f*, de la neurona será directamente el Net; Por tanto, y en lo sucesivo, consideraremos únicamente la función *f*, que denominaremos indistintamente de transferencia y de activación

REGLA DE APRENDIZAJE

Existen muchas definiciones del concepto general de *aprendizaje*, una de ellas podría ser: *La modificación del comportamiento inducido por la interacción con el entorno y como resultado de experiencias conducentes al establecimiento de nuevos modelos de respuesta a estímulos externos.* Esta definición fue enunciada muchos años antes de que surgieran las redes neuronales, sin embargo puede ser aplicada también a los procesos de aprendizaje de estos sistemas.

Biológicamente, se suele aceptar que la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con ellas mismas; es decir, el conocimiento se encuentra en las sinapsis. En el caso de las redes neuronales artificiales, se puede considerar que el conocimiento se encuentra representado en los *pesos* de las conexiones entre neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. En realidad, puede decirse que se *aprende* modificando los valores de los pesos de la red.

Al igual que el funcionamiento de una red depende del número de neuronas de las que disponga y de cómo estén conectadas entre sí, cada modelo dispone de su o sus propias técnicas de aprendizaje. Una de las reglas de aprendizaje mas potentes es la denominada Algoritmo de Retropropagación, esta se utilizó en este trabajo y se presenta en la sección 3.5.

3.3) ESTRUCTURA DE LA RED NEURONAL ARTIFICIAL

Algunos de los factores más importantes de la estructura de la red son los siguientes:

Número de niveles de capas.

Número de neuronas por nivel.

Tipo de conexiones.

Grado de conectividad.

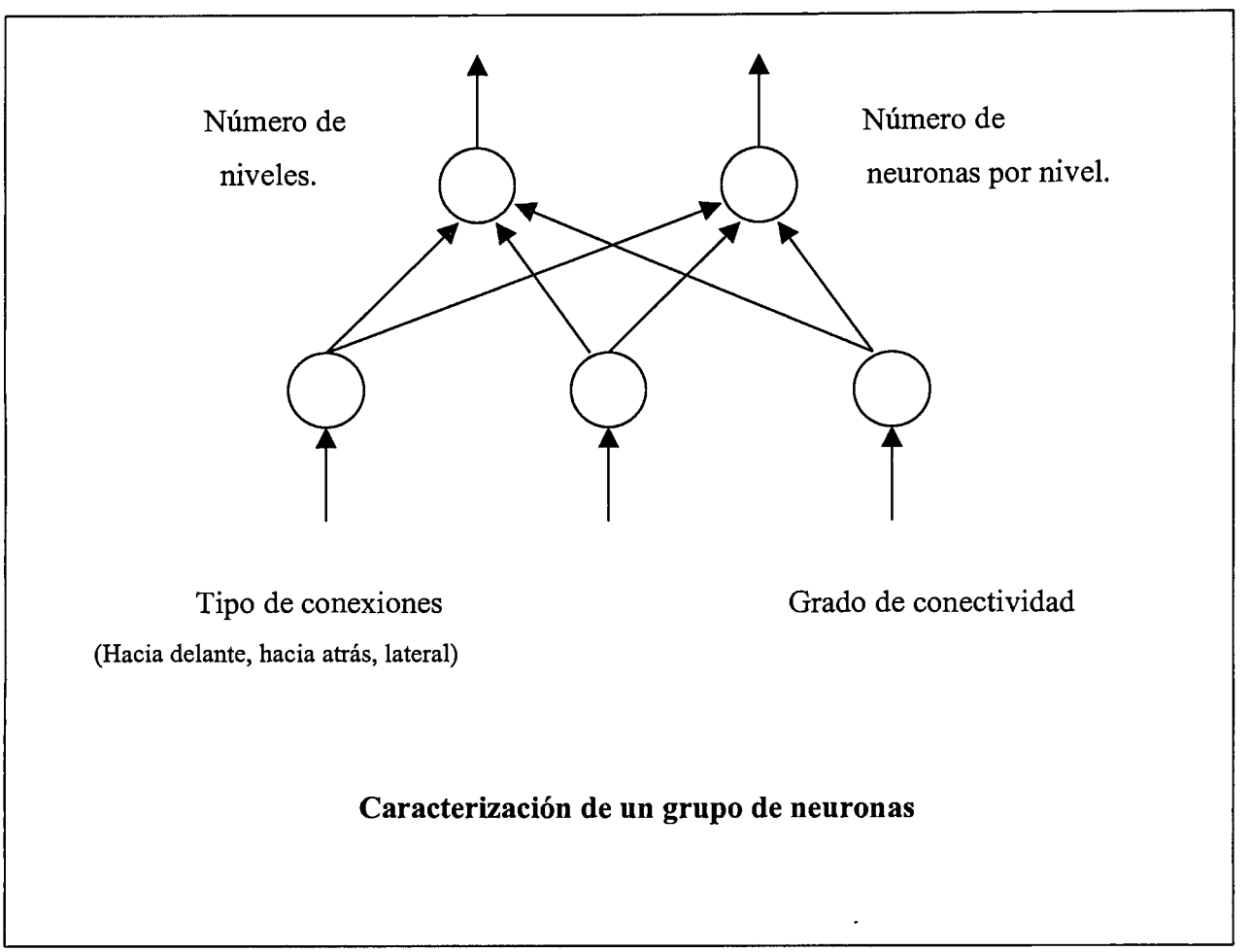


Figura 3.3 Factores modificables de una red neuronal.

NIVELES O CAPAS

La distribución de neuronas de la red se realiza formando niveles o capas de un número determinado de neuronas cada una. A partir de su ubicación dentro de la red, se pueden distinguir tres tipos de capas:

De entrada: Es la capa que recibe directamente la información proveniente de las funciones externas a la red.

Ocultas: Son internas a la red y no tienen contacto directo con el entorno exterior. El número de niveles puede estar entre cero y un número elevado. Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las distintas tipologías de redes neuronales.

De salida: Transfieren información de la red hacia el exterior.

En la Figura 3.4 se muestra el esquema de la estructura de una posible red multicapa en la que cada nodo o neurona únicamente está conectada con neuronas de un nivel superior, Nótese que hay muchas más conexiones que nodos. En este sentido, se dice que una red es *totalmente conectada* si todas las salidas desde un nivel llegan a todos los nodos del nivel siguiente.

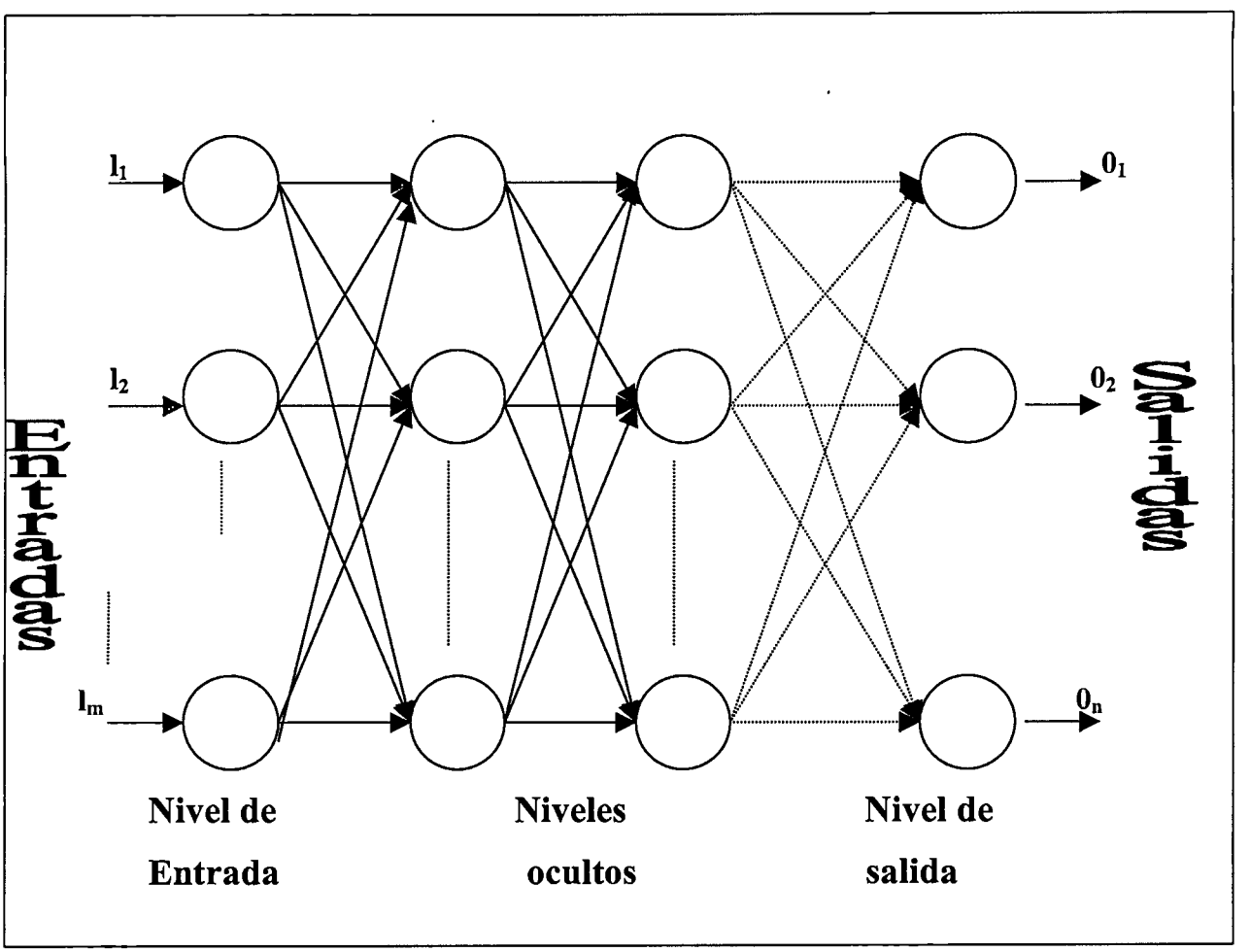
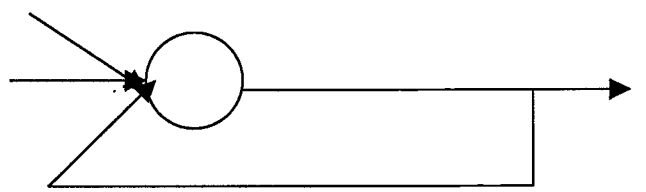


Figura 3.4 Estructura de una red multinivel con todas las conexiones hacia delante.

TIPOS DE CONEXIONES

La conectividad entre neuronas de una red neuronal está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso ser una entrada de sí mismo (conexión autorrecurrente).

Cuando ninguna salida de las neuronas es entrada de neuronas del mismo nivel o de niveles precedentes, la red se describe como de *propagación hacia delante* (Figura 3.4). Cuando las salidas pueden ser conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de *propagación hacia atrás* (Figura 3.5).



Nodo con propagación hacia atrás sobre sí mismo.

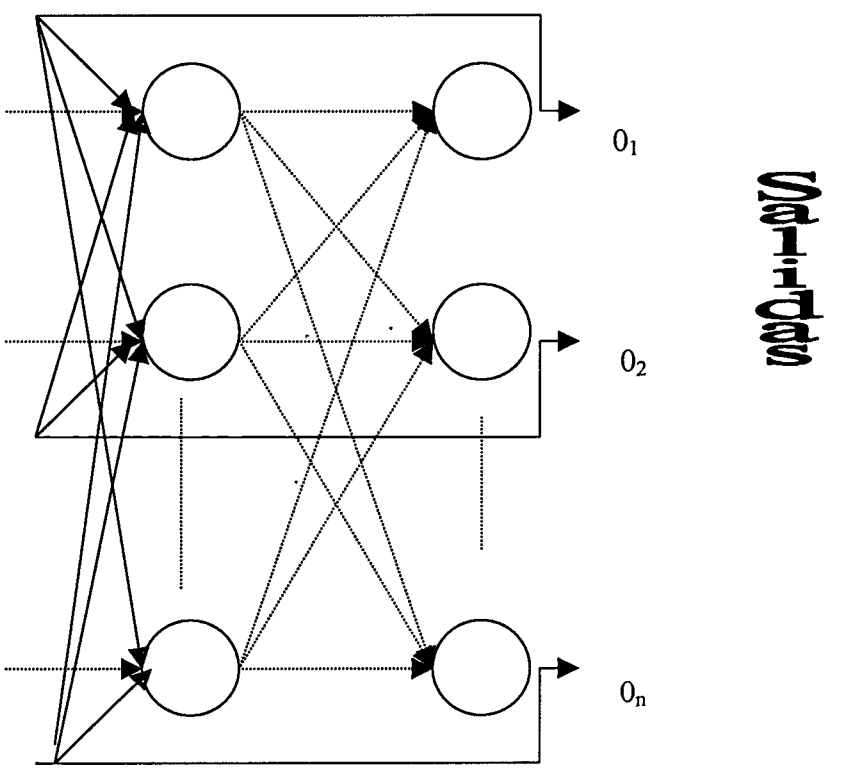


Figura 3.5 Ejemplos de conexiones con propagación hacia atrás.

3.4) PERCEPTRON

Este fue el primer modelo de red neuronal artificial desarrollado por Rosenblatt en 1958. Despertó un enorme interés en los años 60, debido a su capacidad para aprender a reconocer patrones sencillos: un Perceptron, formado por varias neuronas lineales para recibir las entradas a la red y una neurona de salida, es capaz de decidir cuándo una entrada, presentada a la red, pertenece a una de las dos clases que es capaz de reconocer.

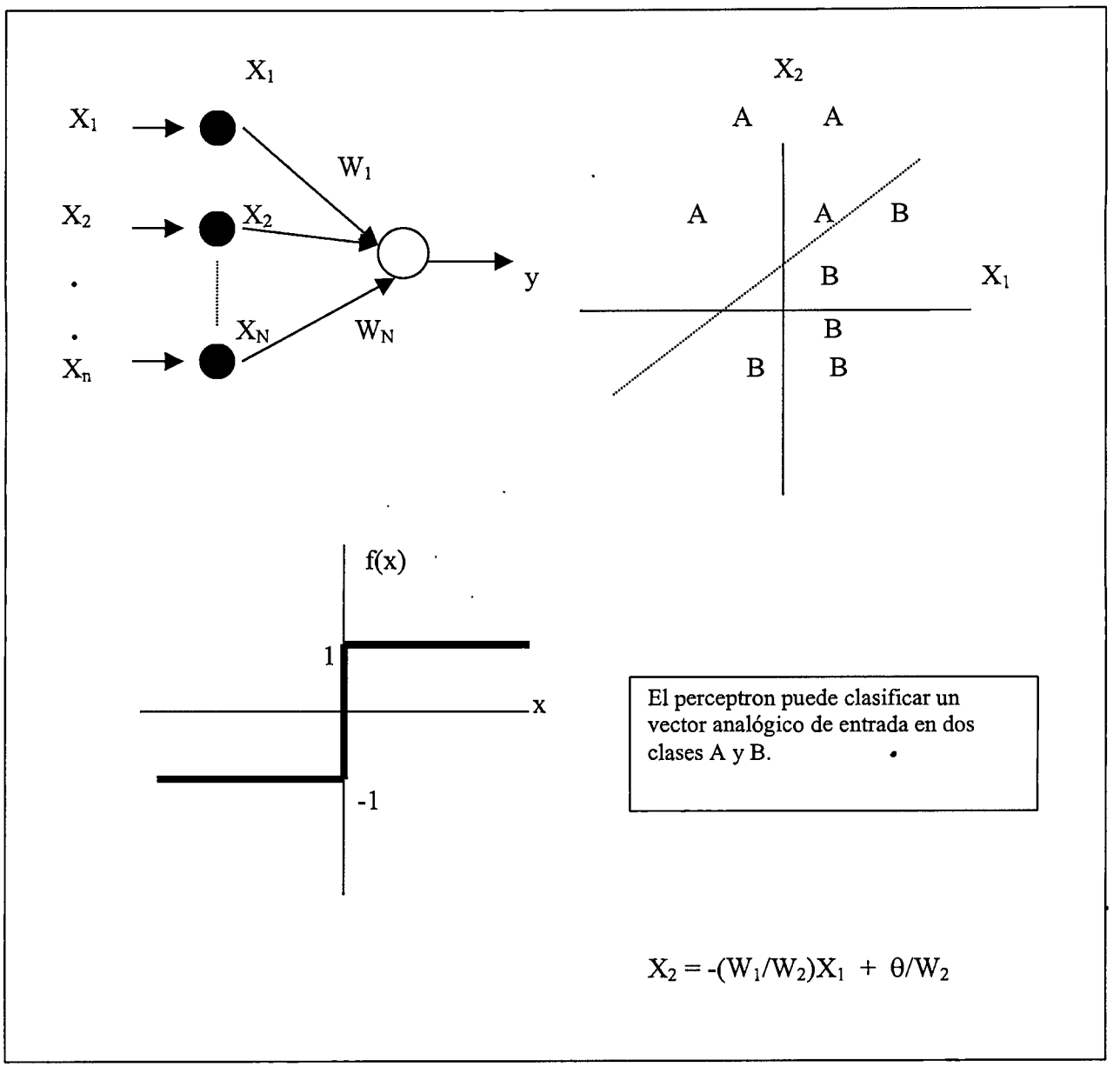


Figura 3.6 El Perceptron.

La única neurona de salida del Perceptron realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia de tipo escalón. La regla de decisión es responder +1 si el patrón presentado pertenece a la clase A, ó −1 si el patrón pertenece a la clase B (Figura 3.6). La salida dependerá de la entrada neta (suma de las entradas x_i ponderadas) y del valor umbral θ.

Una técnica utilizada para analizar el comportamiento de redes como el Perceptron es representar en un mapa las regiones de decisión creadas en el espacio multidimensional de entradas a la red. En estas regiones se visualiza qué patrones pertenecen a una clase y cuáles a otra. El Perceptron separa las regiones por un hiperplano cuya ecuación queda determinada por los pesos de las conexiones y el valor umbral de la función de activación de la neurona. En este caso, los valores de los pesos pueden fijarse o adaptarse utilizando diferentes algoritmos de entrenamiento de la red.

Sin embargo, el Perceptron, al constar sólo de una capa de entrada y otra de salida con una única neurona, tiene una capacidad de representación bastante limitada. Este modelo sólo es capaz de discriminar patrones muy sencillos, linealmente separables. El caso más conocido es la imposibilidad del Perceptron de representar la función OR-EXCLUSIVA (XOR); la cual activa su salida, únicamente, cuando sólo una de sus entradas se encuentra activa.

La separabilidad lineal limita, a las redes con sólo dos capas, a la resolución de problemas en los cuáles el conjunto de puntos (correspondientes a los valores de entrada) sean separables geométricamente. En el caso de dos entradas, la separación se lleva a cabo mediante una línea recta. Para tres entradas, la separación se realiza mediante un plano en el espacio tridimensional, y así sucesivamente hasta el caso de N entradas, en el cuál el espacio N-dimensional es dividido en un hiperplano.

REGLA DE APRENDIZAJE

El algoritmo de aprendizaje del Perceptron es de tipo *supervisado*, lo cual requiere que sus resultados sean evaluados y se realicen las oportunas modificaciones del sistema si fuera necesario. Los valores de los pesos pueden determinar, como se ha dicho, el funcionamiento

de la red; estos valores se pueden fijar o adaptar utilizando diferentes algoritmos de entrenamiento de la red. El algoritmo original de convergencia del Perceptron fue desarrollado por Rosenblatt. Se pueden usar Perceptrones como máquinas universales de aprendizaje. Desgraciadamente, no puede aprender a realizar todo tipo de clasificaciones: en realidad, sólo se pueden aprender clasificaciones *fáciles* (problemas de *orden* 1 en la terminología de Minsky y Papert. Esa limitación se debe a que un Perceptron usa un separador lineal como célula de decisión, con lo cual no es posible realizar sino una sola separación lineal (por medio de un hiperplano).

A continuación veremos el algoritmo de convergencia de ajuste de pesos para realizar el aprendizaje de un Perceptron (aprendizaje por corrección de error) con N elementos procesales y un único elemento procesal de salida:

1) Inicialización de los pesos y del umbral

Inicialmente se asignan valores aleatorios a cada uno de los pesos (w_i) de las conexiones y al umbral ($-w_0 = \theta$).

2) Presentación de un nuevo par (Entrada, Salida esperada)

Presentar un nuevo patrón de entrada $X_p = (x_1, x_2, \ldots, x_N)$ junto con la salida esperada d(t).

3) Cálculo de la salida actual

$$Y(t) = f\left[\sum_i w_i(t)\, x_i(t) - \theta\right]$$

Siendo $f(x)$ la función de transferencia escalón.

4) Adaptación de los pesos

$$w_i(t+1) = w_i(t) + \eta\,[\mathbf{d}(t) - y(t)]\, x_i(t) \qquad (0 \leq y \leq N)$$

donde d(t) representa la *salida deseada*, y será 1 si el patrón pertenece a la clase A, y –1 si es de la clase B. En estas ecuaciones, η es un factor de ganancia en el rango 0.0 a 0.1. Este factor debe ser ajustado de forma que satisfaga tanto los requerimientos de aprendizaje rápido como la estabilidad de las estimaciones de los pesos. Este proceso se repite hasta que el error que se produce para cada uno de los patrones (diferencia entre el valor de salida deseado y obtenido) es cero o bien menor que un valor preestablecido. Obsérvese que los pesos no se cambian si la red ha tomado la decisión correcta.

5) Volver al paso 2

Este algoritmo es extensible al caso de múltiples neuronas en la capa de salida. El Perceptron será capaz de aprender a clasificar todas sus entradas, en un número finito de pasos, siempre y cuando el conjunto de los patrones de entrada sea linealmente separable. En tal caso, puede demostrarse que el aprendizaje de la red se realiza en un número finito de pasos.

SOLUCIÓN AL PROBLEMA DE SEPARABILIDAD LINEAL

Para representar la función OR-EXCLUSIVA, tendríamos que descomponer el espacio en tres regiones: una región pertenecería a una de las clases de salida y las otras dos pertenecerían a la segunda clase. Si en lugar de utilizar únicamente una neurona de salida se utilizaran dos, se obtendrían dos rectas, por lo que podrían delimitarse tres zonas. Para poder elegir entre una zona u otra de las tres, es necesario utilizar otra capa con una neurona cuyas entradas serán las salidas de las neuronas anteriores. Las dos zonas o regiones que convierten los puntos (0,0) y (1,1) se asocian a una salida nula de la red, y la zona central se asocia a la salida con valor 1. De esta manera, es posible encontrar una solución para este caso particular.

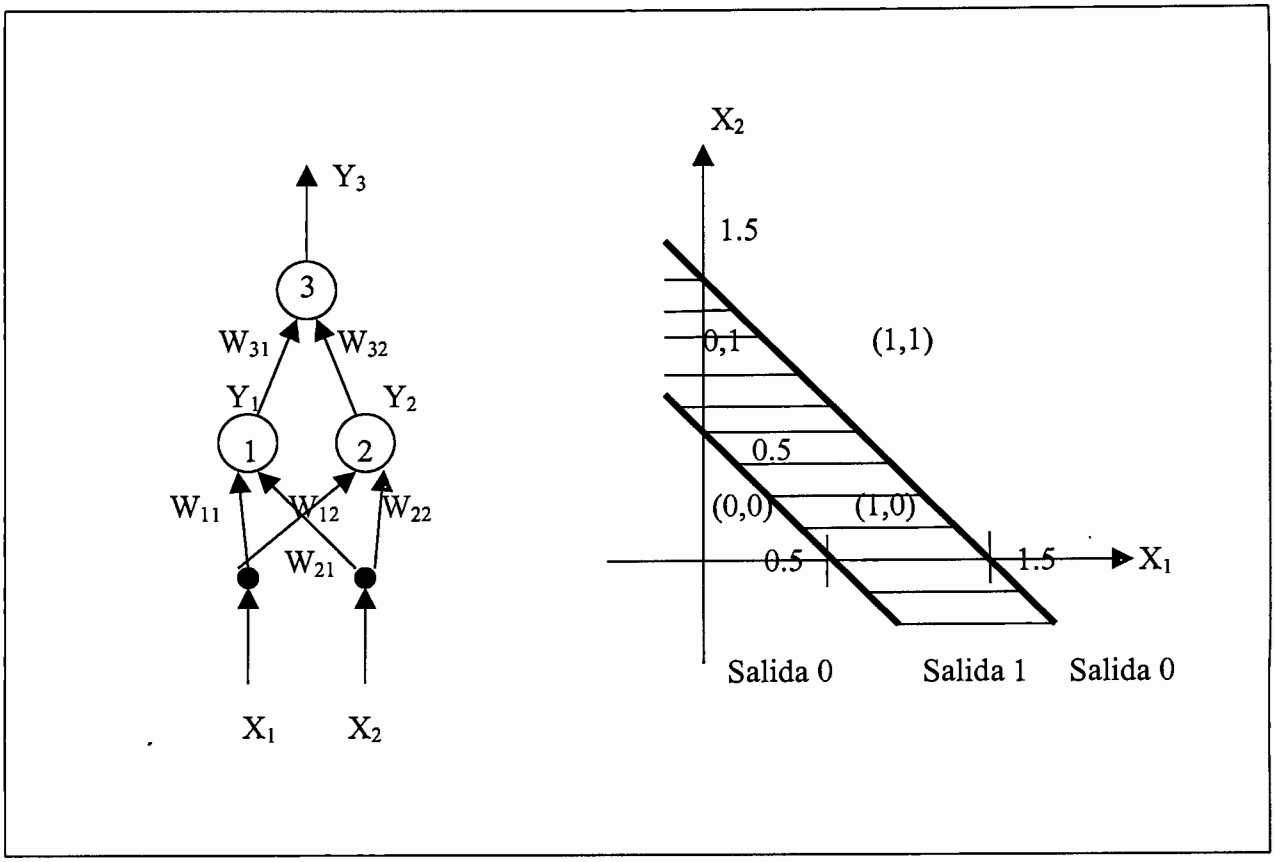


Figura 3.7 Solución del problema de la función XOR.

Hay que indicar que para el caso de la XOR se tienen que ajustar seis coeficientes de peso (sin incluir las conexiones que representan los umbrales). En el caso de los pesos de las conexiones de la capa de salida (W_{31} y W_{32}), el ajuste de los pesos se realiza de forma idéntica a la estudiada anteriormente, pues conocemos la salida deseada. Sin embargo, no se tiene porqué conocer cuál debe ser la salida deseada de las células de la capa oculta, por lo que el método indicado anteriormente, no es aplicable en la función XOR. Para ajustar los coeficientes de peso se requiere utilizar el algoritmo de Retropropagación; porque este permite calcular el error de las capas intermedias en función de la capa de salida.

EL PERCEPTRON MULTINIVEL

Un Perceptron multinivel o multicapa es una red de tipo "conexión hacia delante" *(feedforward),* compuesta de varias capas de neuronas entre la entrada y la salida de la misma.

Esta red permite establecer regiones de decisión mucho más complejas que las de dos semiplanos, como hacía el Perceptron de un sólo nivel.

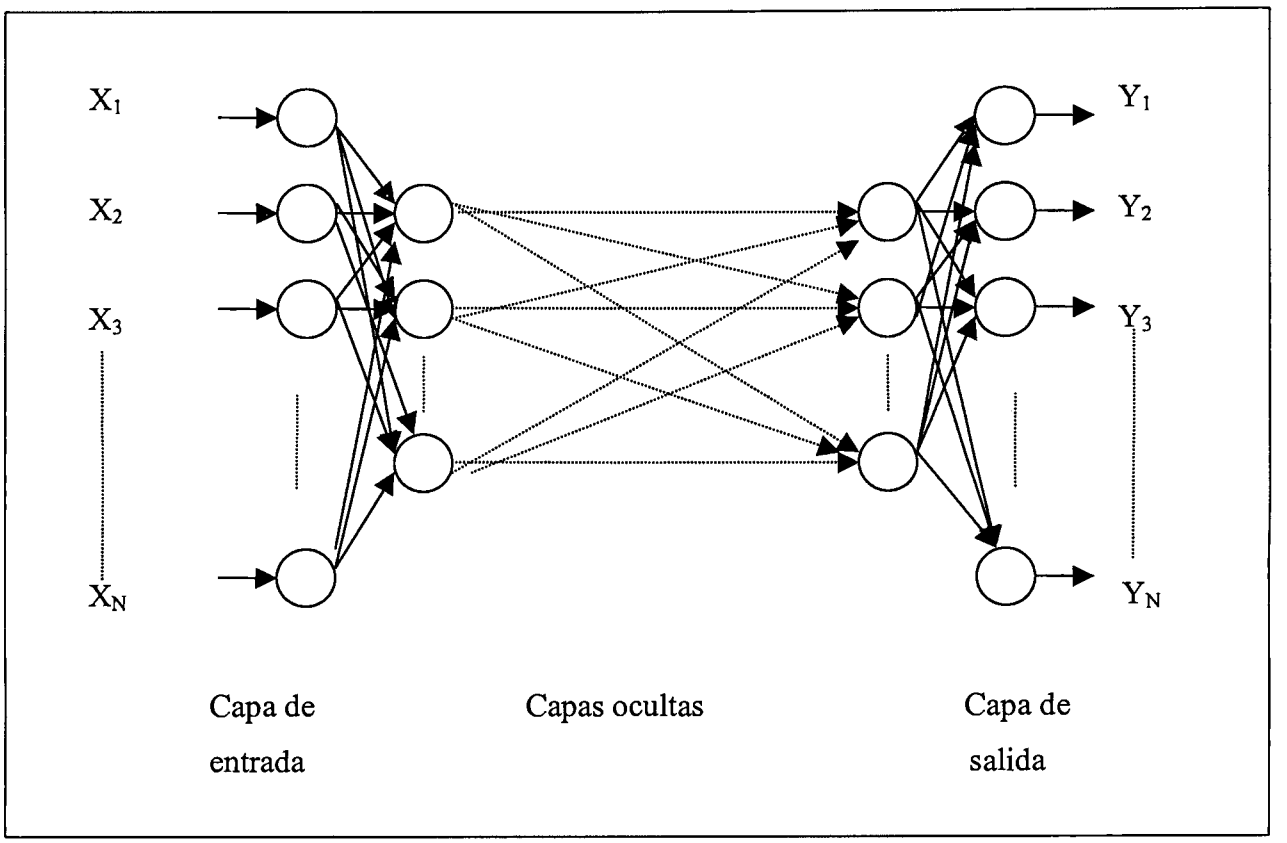


Figura 3.8 Perceptron multinivel (red feedforward multicapa).

Las capacidades del Perceptron con dos, tres y cuatro niveles o capas y con una única neurona en el nivel de salida, se muestra en la Figura 3.9. En la segunda columna se muestra el tipo de región de decisión que se puede formar con cada una de las configuraciones. En la siguiente columna se indica el tipo de región de decisión que se formaría para el problema de la XOR. En las dos últimas columnas se muestran las regiones formadas para resolver el problema de clases con regiones mezcladas y las formas de regiones más generales para cada uno de los casos.

El Perceptron básico de dos capas (la de entrada con neuronas lineales y la de salida con función de activación de tipo escalón) sólo puede establecer dos regiones separadas por una frontera lineal en el espacio de patrones de entrada.

Un Perceptron con tres niveles de neuronas puede formar cualquier región convexa en este espacio. Las regiones convexas se forman mediante la intersección entre las regiones formadas por cada neurona de la segunda capa.

Cada uno de estos elementos se comporta como un Perceptron simple, activándose su salida para los patrones de un lado del hiperplano. Si el valor de los pesos de las conexiones entre las N_2 neuronas de la segunda capa y una neurona del nivel de salida son todos 1 y el umbral de la de salida en $(N_2 - a)$, desde $0 < a < 1$, entonces la salida de la red se activará sólo si las salidas de todos los nodos de la segunda capa están activos. Esto equivale a ejecutar la operación lógica AND (sólo activa su salida con todas sus entradas activas), en el nodo de salida, resultando una región de decisión intersección de todos los semiplanos formados en el nivel anterior. La región de decisión resultante de la intersección serán regiones convexas con un número de lados a lo sumo igual al número de neuronas de la segunda capa.

Este análisis nos introduce en el problema de la selección del número de neuronas ocultas de un Perceptron de tres capas. En general, este número deberá ser lo suficientemente grande como para que se forme una región lo suficientemente compleja para la resolución del problema. Sin embargo, tampoco es conveniente que el número de nodos sea tan grande que la estimación de los pesos no sea fiable para el conjunto de patrones de entrada disponibles.

Un Perceptron con cuatro capas puede formar regiones de decisión arbitrariamente complejas. El proceso de separación en clases que se lleva a cabo consiste en la partición de la región deseada en pequeños hipercubos (cuadrados para dos entradas de la red). Cada hipercubo requiere 2N neuronas en la segunda capa (siendo N el número de entradas a la red), una por cada lado del hipercubo, y otra la 3ª capa, que lleva a cabo el AND lógico de las salidas de los nodos del nivel anterior. Las salidas de los nodos de este tercer nivel se activarán sólo para las entradas de cada hipercubo. Los hipercubos se asignan a la región de decisión adecuada mediante la conexión de la salida de cada nodo del tercer nivel sólo con la neurona de salida (cuarta capa) correspondiente a la región de decisión en la que está comprendido el hipercubo, llevándose a cabo una operación lógica OR en cada nodo de salida. La operación lógica OR se llevará a cabo sólo si el valor de los pesos de las conexiones de los

nodos de tercer nivel vale uno, y además el valor de los umbrales de los nodos de salida es 0.5. Este procedimiento se puede generalizar de manera que la forma de las regiones convexas sea arbitraria, en lugar de hipercubos.

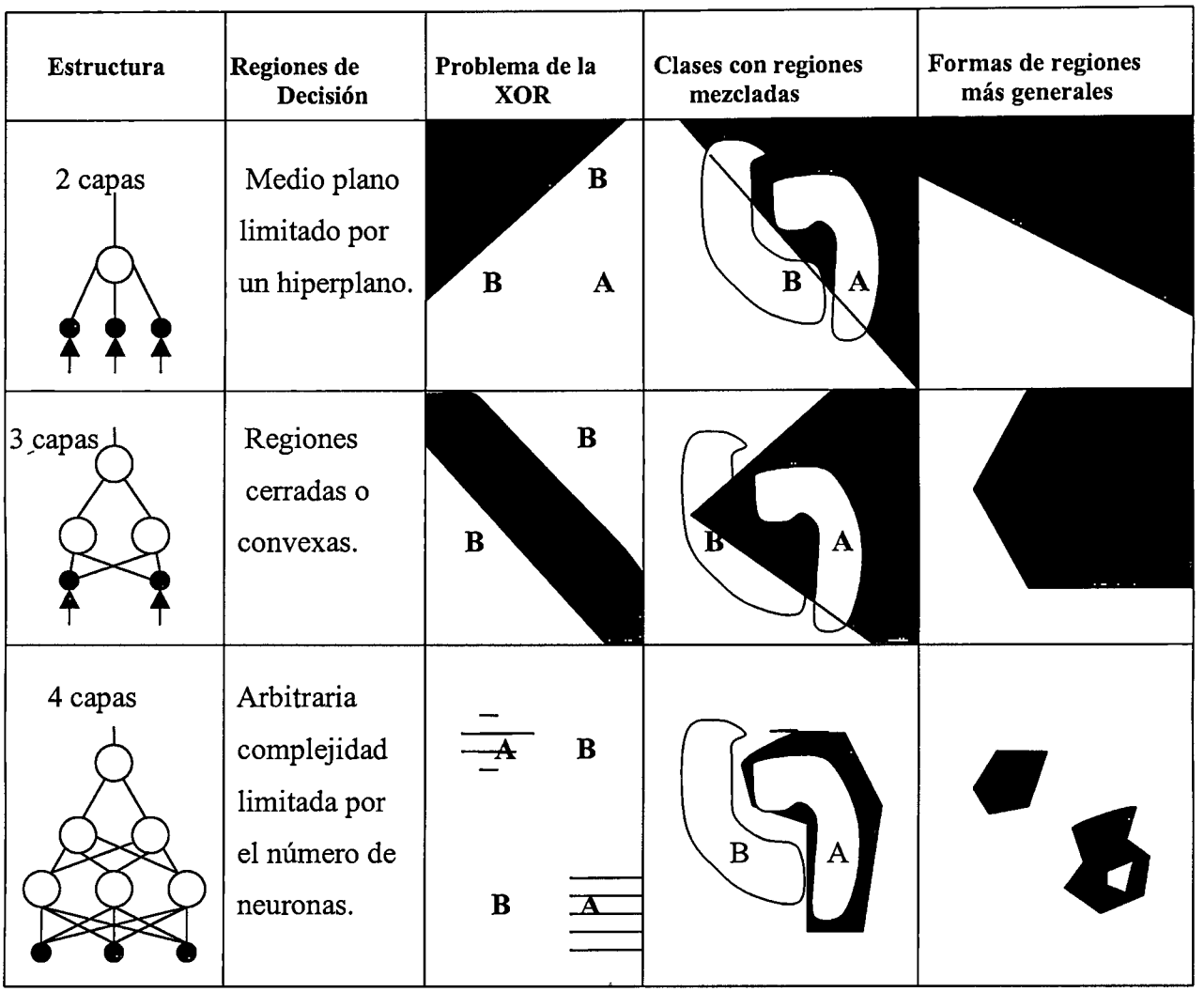


Figura 3.9 Formas de las regiones generadas por un perceptron multinivel. Lippmann [22]

El análisis anterior demuestra que no se requiere más de cuatro capas en una red de tipo Perceptron, pues, como se ha visto una red con cuatro niveles, puede generar regiones de decisión arbitrariamente complejas. Sólo en ciertos problemas se puede simplificar el aprendizaje mediante el aumento del número de neuronas ocultas. Sin embargo, la tendencia es el aumento de la extensión de la función de activación, en lugar del aumento de la

complejidad de la red. Esto de nuevo nos lleva al problema del número de neuronas que debemos seleccionar para un Perceptron con cuatro capas.

El número de nodos de la 3ª capa (N_3) debe ser mayor que uno cuando las regiones de decisión están desconectadas o endentadas y no se puede formar con una región convexa. Este número, en el peor de los casos, es igual al número de regiones desconectadas en las distribuciones de entrada. El número de neuronas en la 2ª capa (N_2) normalmente debe ser suficiente para proveer tres o más ángulos para cada área convexa generada por cada neurona de la 3ª capa. Así, deberá de haber más de tres veces el número de neuronas de la 3ª capa ($N_2 > 3N_3$). En la práctica, un número de neuronas excesivo en cualquier capa puede generar ruido. Por otro lado, si existe un número de neuronas redundantes se obtiene mayor tolerancia a fallos.

3.5) ALGORITMO DE RETROPROPAGACIÓN

En 1986, Rumelhart, Hinton y Williams, basándose en los trabajos de otros investigadores, formalizaron un método para que una red neuronal *aprendiera* la asociación que existe entre los patrones de entrada a la misma y las clases correspondientes, utilizando más niveles de neuronas que los que utilizó Rosenblatt para desarrollar el Perceptron. Este método, conocido en general como *Retropropagación* (propagación del error hacia atrás), está basado en la generalización de la regla delta y, a pesar de sus propias limitaciones, ha ampliado de forma considerable el rango de aplicaciones de las redes neuronales.

El algoritmo de propagación hacia atrás, o Retropropagación, es una regla de aprendizaje que se puede aplicar en modelos de redes con más de dos capas de células. Una característica importante de este algoritmo es la representación interna del conocimiento que es capaz de organizar en la capa intermedia de las células para conseguir cualquier correspondencia entre la entrada y salida de la red. Ya se ha mostrado en este capítulo que en muchos casos, como la resolución del problema de la OR exclusiva, es imposible encontrar los pesos adecuados para establecer la correspondencia entre la entrada y la salida mediante una red sin capas intermedias. Con una capa de neuronas ocultas, sí es posible establecer dicha correspondencia.

De forma simplificada, el funcionamiento de una red *Retropropagación* (BPN), consiste en un aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo, empleando un ciclo *propagación-adaptación* de dos fases: primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener y se calcula un valor del error para cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada; es decir, el error disminuya.

La importancia de la red *Retropropagación* consiste en su capacidad de autoadaptarse los pesos de las neuronas de las capas intermedias para *aprender* la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes. Para poder aplicar esa misma relación, después del entrenamiento, a nuevos vectores de entrada con ruido o incompletas, dando una salida activa si la nueva entrada es parecida a las presentadas durante el aprendizaje. Esta característica importante, que se exige a los sistemas de aprendizaje, es la capacidad de *generalización*, entendida como la facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento. La red debe encontrar una *representación interna* que le permita generar las salidas deseadas cuando se le dan las entradas de entrenamiento, y que pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje para clasificarlas según las características que compartan con los ejemplos de entrenamiento.

A continuación se muestra el algoritmo de Retropropagación, su deducción, para el caso del Controlador Neuronal Autoajustable, se presenta en el capítulo siguiente.

1. Se aplica el vector de entradas $X_p = (x_{p1}, x_{p2}, \ldots, x_{pN})^t$ a las unidades de entrada.

2. Se calculan los valores netos procedentes de las entradas para las unidades de la capa oculta: $$neta^h_{pj} = \sum_{i=1}^{N} w^h_{ji} x_{pi}$$

3. Se calculan las salidas de la capa oculta:

$$i_{pj} = f^h_j(neta^h_{pj})$$

4. Se pasa a la capa de salida. Se calculan los valores netos de las entradas para cada unidad:

$$neta^o_{pk} = \sum_{j=1}^{L} w^o_{kj} i_{pj}$$

5. Se calculan las salidas: $$o_{pk} = f^o_k(neta^o_{pk})$$

6. Se calculan los términos de error para las unidades de salida:

$$\delta^o_{pk} = (y_{pk} - o_{pk}) f^{o'}_k(neta^o_{pk})$$

7. Se calculan los términos de error para las unidades ocultas:

$$\delta^h_{pj} = f^{h'}_j(neta^h_{pj}) \sum_k \delta^o_{pj} w^o_{kj}$$

8. Se actualizan los pesos de la capa de salida:

$$w^o_{kj}(t+1) = w^o_{kj}(t) + \eta \delta^o_{pk} i_{pj}$$

9. Se actualizan los pesos de la capa oculta:

$$w^h_{ji}(t+1) = w^h_{ji}(t) + \eta \delta^h_{pj} x_{pi}$$

CONSIDERACIONES PRÁCTICAS

En general, se pueden utilizar todos los datos que estén disponibles para entrenar la red, aunque quizá no sea necesario utilizarlos todos. Con cierta frecuencia, lo único que se necesita para entrenar con éxito una red es un pequeño subconjunto de los datos de entrenamiento de los que se dispone. Los datos restantes pueden emplearse para probar la red,

con objeto de verificar que la red pueda llevar a cabo la asociación deseada al utilizar vectores de entrada que nunca haya encontrado durante el entrenamiento.

La BPN admite bien la generalización. Lo que quiere decir aquí *generalización* es que dados varios vectores de entrada distintos, todos los cuales pertenecen a una misma clase, una BPN aprenderá a adaptarse a las similitudes significativas de los vectores de entrada. Los datos irrelevantes serán ignorados.

En contraste con las generalizaciones, la BPN no extrapolará bien. Si una BPN se entrena de modo inadecuado o insuficiente empleando una clase concreta de vectores de entrada, la posterior identificación de miembros de esa clase puede ser imprecisa. Asegúrese de que los datos de entrenamiento cubran todo el espacio de entradas esperado. Durante el proceso de entrenamiento, seleccione aleatoriamente los pares de vectores de entrenamiento conjunto, si es que el problema se presta a ésta estrategia. En todo caso, no entrene por completo a la red con vectores de una clase, pasando después a otra clase; la red *se olvidará* del entrenamiento original.

Si la función de salida es una sigmoide, entonces será preciso aplicar una escala a los vectores de entrada. Como consecuencia de la forma de la función sigmoide, las salidas de la red nunca pueden alcanzar el cero ni el uno. Por tanto, hay que utilizar valores como 0.1 y 0.9 para representar los valores de entrada más pequeños y más grandes respectivamente. También se puede desplazar la sigmoide de tal manera que, por ejemplo, los valores limitantes pasen a ser ± 0.4. Además, se puede modificar la pendiente de la parte lineal de la curva sigmoide incluyendo una constante multiplicativa en la exponencial. Hay muchas posibilidades como éstas que dependen fuertemente del problema que se esté resolviendo.

Al igual que en el caso de las preguntas que se refieren a los datos correctos para el entrenamiento, no hay respuestas semejantes para determinar el número de neuronas de la capa intermedia. En general, tres capas son suficientes. Hay veces, sin embargo, en que parece que un problema es más fácil de resolver con más de una capa oculta. En este caso, *más fácil* quiere decir que la red aprende más rápido.

Determinar el número de unidades que hay que utilizar en la capa oculta no suele ser tan evidente como lo es para las capas de entrada y salida. La idea principal consiste en utilizar el menor número posible de unidades en la capa oculta, porque cada unidad supone una carga para la unidad central de proceso (UCP) durante las simulaciones. Por supuesto, en un sistema que esté construido el hardware en su totalidad (un procesador por cada elemento de proceso), la carga adicional para la UCP no es factor tan importante (sin embargo, la comunicación entre procesadores sí que puede ser problema). En una aplicación, si la red no llega a converger para llegar a una solución, es posible que se necesiten más nodos ocultos. Si converge, se puede probar con un número inferior de nodos ocultos y determinar un tamaño final basándose en el rendimiento global del sistema.

También es posible eliminar unidades ocultas que resulten superfluas. Si se examinan los valores de los pesos de los nodos ocultos periódicamente, a medida que se entrena la red, se verá que los pesos de ciertos nodos cambian muy poquito con respecto a sus valores iniciales. Estos nodos pueden no estar participando en el proceso de aprendizaje, y quizá baste con un número menor de unidades ocultas.

El algoritmo de *Retropropagación* encuentra un valor mínimo de error (local o global) mediante la aplicación del método del Gradiente Descendiente. Cada punto de la superficie de la función de error corresponde a un conjunto de valores de los pesos de la red. Con el gradiente descendente, siempre que se realiza un cambio en todos los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local de error.

Por tanto, uno de los problemas que presenta este algoritmo de entrenamiento de redes multicapa es que busca minimizar la función de error, pudiendo caer en un mínimo local o en algún estacionario, con lo cual no se llega a encontrar el mínimo global de la función del error. Sin embargo, ha de tenerse en cuenta que no tiene porqué alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo preestablecido.

La selección de un valor para el parámetro de velocidad de aprendizaje, η, tiene un efecto significativo en el rendimiento de la red. Normalmente, η debe ser un número pequeño (del orden de 0,05 a 0,25) para asegurar que la red llegue a asentarse en una solución. Un valor pequeño de η significaría que la red tendrá que hacer un gran número de interacciones, pero este es el costo en que se incurre.

En las técnicas de gradiente decreciente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos. Esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o lo cerca que se está del punto mínimo. Con incrementos grandes, se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarde más en llegar, se evita que ocurra esto.

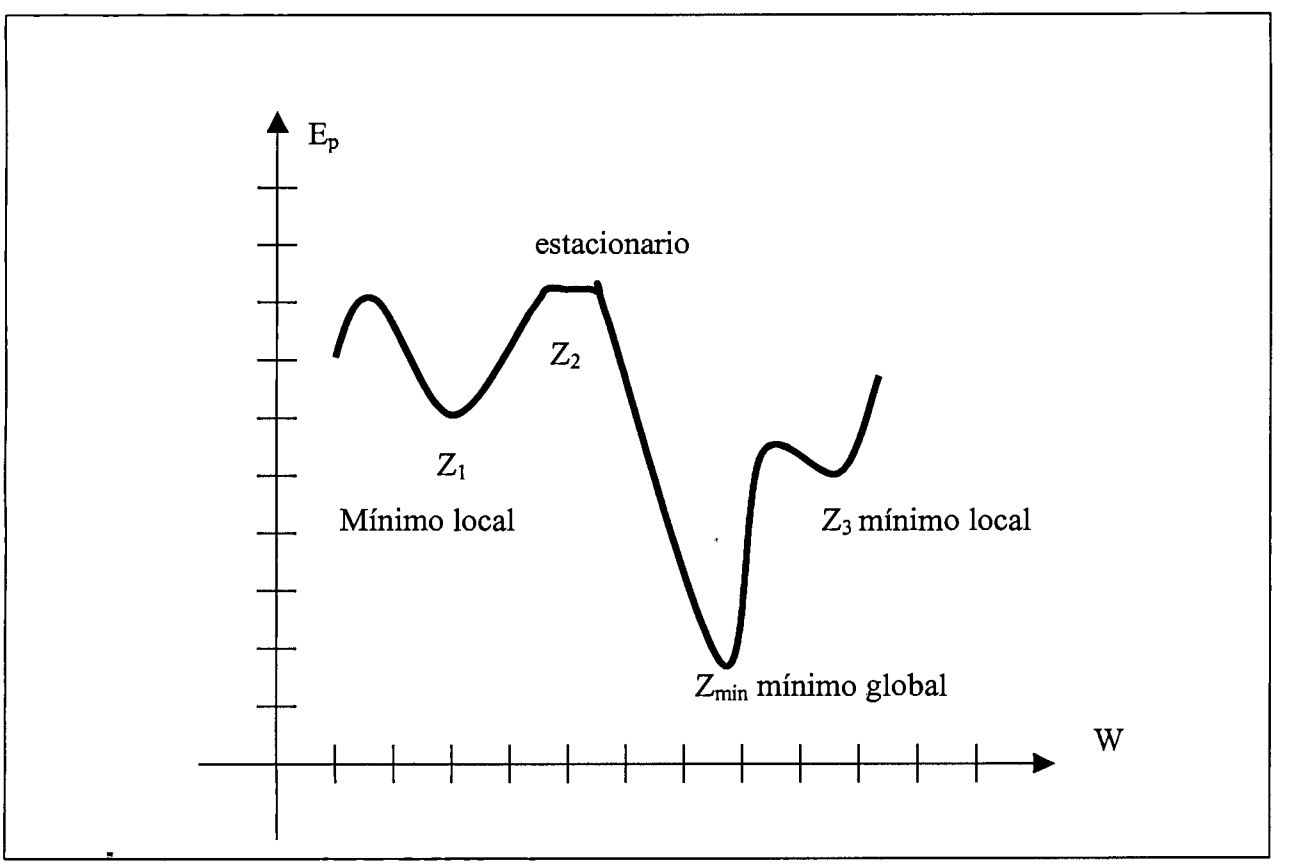


Figura 3.10 Ejemplo representativo de una forma de la superficie de error.

4) CONTROLADOR NEURONAL AUTOAJUSTABLE (C.N.A.)

4.1) ANTECEDENTES DIRECTOS

ESTRUCTURA DEL CONTROLADOR NEURONAL AUTOAJUSTABLE

En la Figura 4.1 se muestra el esquema del Controlador Neuronal Autoajustable propuesto por Aguado et al, el cual corresponde a una red neuronal tipo Perceptron de tres capas (con una capa oculta), cuyos coeficientes de peso son ajustados mediante un algoritmo de Retropropagación modificado. En este diagrama se modificó la flecha que indica la retroalimentación del error con el que se ajustan los coeficientes de peso, de forma que la simbología usada sea consistente con la de las referencias más recientes [01], [08], [12] y [17]. En este caso, para ajustar los coeficientes de peso, en lugar del error de salida de la red:

$$e_u(t) = u_d(t) - u(t) \tag{1}$$

se usa el error de salida del proceso:

$$e_y(t) = y_r(t) - y(t) \tag{2}$$

En la Figura 4.2 se representa la estructura de la red neuronal responsable del control. La capa de salida tiene sólo una neurona porque, por el momento, se restringe el análisis a procesos con una entrada y una salida. En la capa de entrada se toman en cuenta el valor actual y los dos inmediatos anteriores del error en regulación. Por último, en la capa intermedia se utiliza el mismo número de neuronas de la capa de entrada.

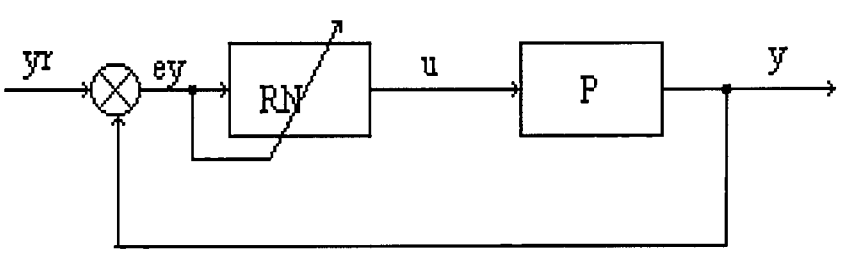


Figura 4.1 Esquema del Controlador Neuronal Autoajustable.

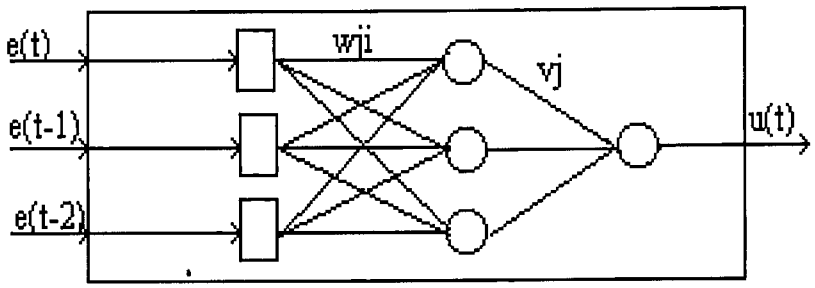


Figura 4.2 Estructura de la red neuronal responsable del control.

ALGORITMO DE ADAPTACIÓN DE LOS COEFICIENTES DE PESO

Siguiendo con la estructura de la Figura 4.2, los coeficientes de peso "w_{ji}" y "v_j" corresponden, respectivamente, a las conexiones de entradas de las capas intermedia y de salida. El índice "i" corre sobre el número de neuronas de entrada y el "j" sobre el número de neuronas de la capa intermedia. La función de activación propuesta es de tipo sigmoide, por lo que la salida de las neuronas de la capa intermedia "h_j" y la salida de la neurona de la capa de salida "u(t)", están dadas por las siguientes expresión:

Salida de la neurona j, de la capa escondida:

$$h_j = \frac{1}{1 + e^{-S_j}} \qquad \text{tal que: } S_j = \sum_{i=1}^{3} w_{ji} x_i \qquad \forall \text{ j=1,2,3.} \quad (3)$$

Salida de la neurona de la capa de salida:

$$u(t) = \frac{1}{1 + e^{-r}} \qquad \text{tal que: } r = \sum_{j=1}^{3} v_j h_j \quad (4)$$

Como criterio a minimizar, suponiendo que el tiempo ha sido discretizado en pequeños intervalos de tiempo, se escoge la siguiente función cuadrática del error:

$$E(t) = \frac{1}{2} \sum_{k=1}^{t} e_y(k)^2 \quad (5)$$

El procedimiento de minimización consiste en desplazarse en la dirección contraria a la del gradiente de la función "E(t)" con respecto a los coeficientes de peso "w_{ji}" y "v_j". El gradiente de la función "E(t)" está dado por la siguiente expresión:

$$\nabla E(t) = \begin{bmatrix} \dfrac{\partial E(t)}{\partial v_j} \\ \dfrac{\partial E(t)}{\partial w_{ji}} \end{bmatrix} \qquad (6)$$

Los componentes de la derivada parcial de la función "E(t)" con respecto a los coeficientes de peso de la neurona de salida, están dados por la relación siguiente:

$$\frac{\partial E(t)}{\partial v_j} = \frac{\partial E(t)}{\partial e_y} * \frac{\partial e_y}{\partial e_u} * \frac{\partial e_u}{\partial u(t)} * \frac{\partial u(t)}{\partial r} * \frac{\partial r}{\partial v_j}$$

$$= -e_y \frac{\partial e_y}{\partial e_u} u(t)(1-u(t))\, h_j \qquad (7)$$

En (7) se utilizó la suposición de que la función de activación de las neuronas es de tipo sigmoide y en ese caso la derivada parcial de "u(t)" con respecto a sus coeficientes de peso está dada por:

$$\frac{\partial u(t)}{\partial r} = \frac{\partial\left(\dfrac{1}{1+e^{-r}}\right)}{\partial r} = \frac{e^{-r}}{\left(1+e^{-r}\right)^2} = \frac{e^{-r}}{1+e^{-r}} \frac{1}{1+e^{-r}} = u(t)(1-u(t)) \qquad (8)$$

Para fines de cálculo, resulta conveniente obtener primero el valor del error en regulación multiplicado por (8), y después usarlo en el cálculo de cada uno de los términos de (7). Con esta finalidad, podemos reescribir (7) como se muestra a continuación:

$$\Rightarrow \quad \boxed{\frac{\partial E(t)}{\partial v_j} = -\delta^1 h_j \frac{\partial e_y}{\partial e_u}} \qquad \text{con: } \underline{\delta^1 = e_y\, u(t)(1-u(t))} \qquad (9)$$

En la expresión anterior, aparece el término $\frac{\partial e_y}{\partial e_u}$, el cual debe tener magnitud finita y se puede interpretar como un tipo de "ganancia equivalente" del proceso. Nótese que, en general, no se dispone de la información necesaria para evaluar este término, por esta razón, posteriormente, se tendrán que hacer algunas consideraciones que permitan evitar su cálculo.

Los componentes de la derivada parcial de la función "E(t)" con respecto a los coeficientes de peso de las neuronas de la capa intermedia, están dados por:

$$\frac{\partial E(t)}{\partial w_{ji}} = \frac{\partial E(t)}{\partial e_y} * \frac{\partial e_y}{\partial e_u} * \frac{\partial e_u}{\partial u(t)} * \frac{\partial u(t)}{\partial r} * \frac{\partial r}{\partial h_j} * \frac{\partial h_j}{\partial S_j} * \frac{\partial S_j}{\partial w_{ji}}$$

$$= -e_y \frac{\partial e_y}{\partial e_u} u(t)(1-u(t)) v_j h_j (1-h_j) x_i \qquad (10)$$

$$\Rightarrow \quad \boxed{\frac{\partial E(t)}{\partial w_{ji}} = -\delta^2{}_j \, x_i \frac{\partial e_y}{\partial e_u}} \qquad \text{con: } \underline{\delta^2{}_j = \delta^1 v_j h_j (1-h_j)} \qquad (11)$$

Usando las ecuaciones (9) y (11), el ajuste de los coeficientes de peso queda dado por :

$$v_j(t+1) = v_j(t) + (\eta \frac{\partial e_y}{\partial e_u}) \delta^1 h_j \qquad (12)$$

$$w_{ji}(t+1) = w_{ji}(t) + (\eta \frac{\partial e_y}{\partial e_u}) \delta^2{}_j \, x_i \qquad (13)$$

Una de las principales aportación del trabajo de Cui y Shin, consiste en que demostraron que sólo se requiere conocer el signo del término $\frac{\partial e_y}{\partial e_u}$ para asegurar la convergencia de los coeficientes de peso, debido a que la magnitud puede incorporarse como parte del coeficiente

de aprendizaje, este signo puede calcularse mediante un experimento en el que, por ejemplo, se someta la planta a una entrada en escalón. Con estas consideraciones, las ecuaciones de adaptación de los coeficientes de peso de la red toman la siguiente forma:

$$v_j(t+1) = v_j(t) + \eta' \; sign(\frac{\partial e_y}{\partial e_u}) \delta^1 h_j \qquad \text{con: } \eta' = \eta \cdot abs\left[\frac{\partial e_y}{\partial e_u}\right] \qquad (14)$$

$$w_{ji}(t+1) = w_{ji}(t) + \eta' \; sign(\frac{\partial e_y}{\partial e_u}) \delta^2{}_j \, x_i \qquad \text{con: } \eta' = \eta \cdot abs\left[\frac{\partial e_y}{\partial e_u}\right] \qquad (15)$$

Uno de los problemas que se presentan al utilizar (14) y (15) para ajustar los coeficientes de peso de la red, consiste en que durante el proceso de adaptación, la red, atraviesa por etapas en las que los coeficientes de peso cambian muy lentamente (incluso para valores grandes del error en regulación), en consecuencia la respuesta del sistema mejora lentamente e incluso, en una implementación práctica, puede quedarse estancado su progreso. Es indispensable reducir este efecto para poder implementar, en tiempo real, el Controlador Neuronal Autoajustable, por esta razón se desarrolla este tema en el inciso siguiente.

ACELERACIÓN DE LA CONVERGENCIA DEL ALGORITMO DE RETROPROPAGACIÓN

De las ecuaciones (9), (11), (14) y (15) se obtiene que, para cada periodo, los coeficientes de peso se modifican de acuerdo a las expresiones:

$$\Delta v_j = \eta' \; sign(\frac{\partial e_y}{\partial e_u}) \delta^1 h_j \qquad \text{con: } \delta^1 = e_y \, u(t)(1-u(t)) \qquad (16)$$

$$\Delta w_{ji} = \eta' \; sign(\frac{\partial e_y}{\partial e_u}) \delta^2{}_j \, x_i \qquad \text{con: } \delta^2{}_j = \delta^1 v_j \, h_j (1-h_j) \qquad (17)$$

En las Figuras 4.3a y 4.3b se muestran las gráficas de la función sigmoide [utilizada para calcular $u(t)$ y $h_j(t)$] y la de su derivada [correspondiente a los términos $u(t)(1-u(t))$ y $h_j(t)(1-h_j(t))$]. Cuando el valor de "$u(t)$"se aproxima a cualquiera de sus extremos, los valores de

"δ^1" y en consecuencia "δ^2" se hacen muy pequeños, ocasionando que el incremento de todos los coeficientes de peso también sea muy pequeño. Otro caso en el que los coeficientes de peso de las neuronas de la capa intermedia se hacen muy pequeños es cuando el valor de "h_j" tiende a cualquiera de sus extremos y los valores de "$\delta^2{}_j$" se hacen muy pequeños.

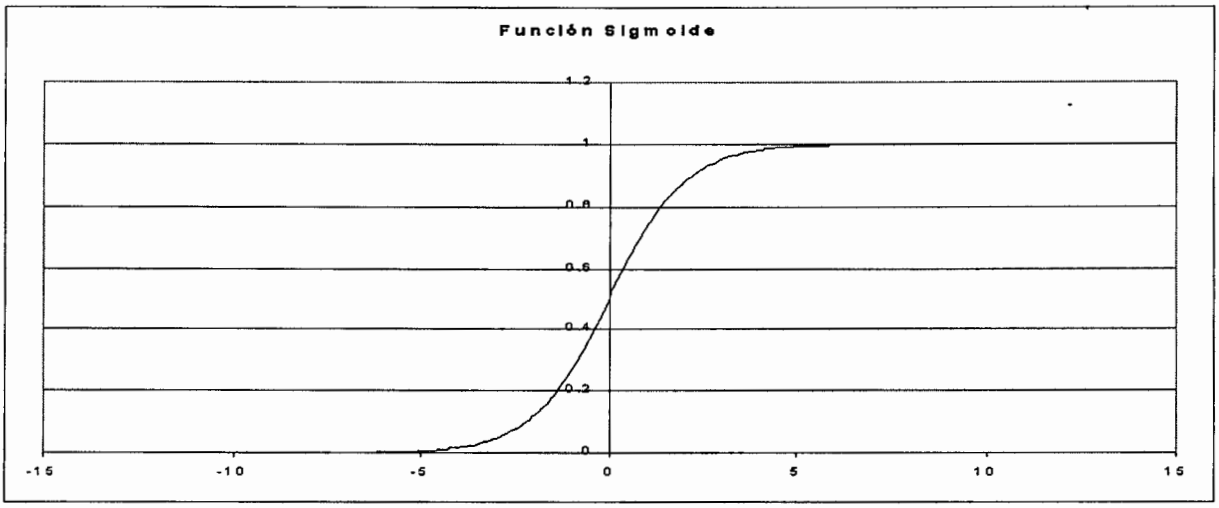


Figura 4.3a Gráfica de la función sigmoide.

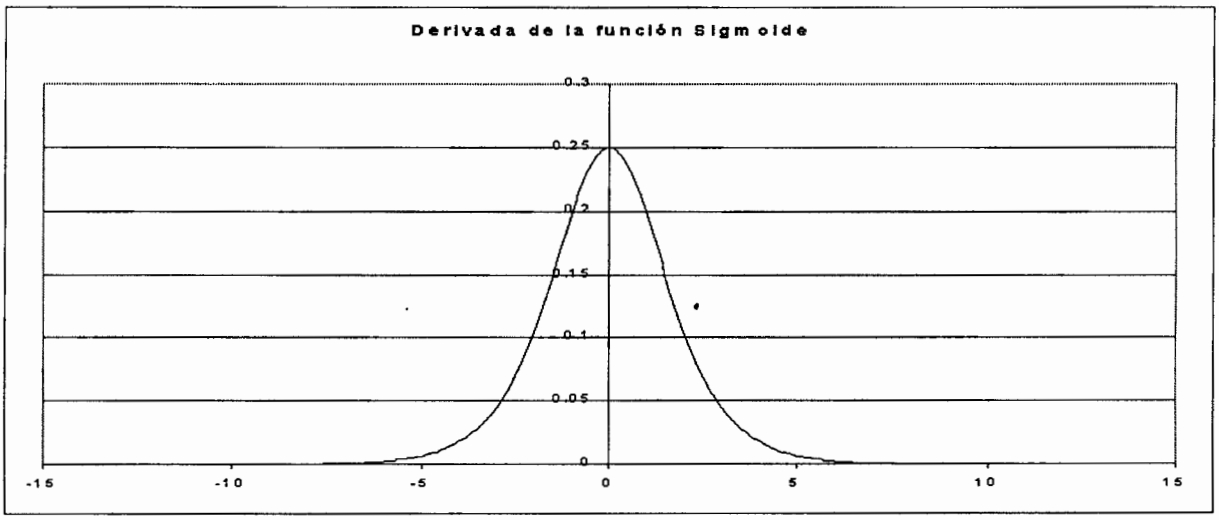


Figura 4.3b Gráfica de la derivada de la función sigmoide.

Desde un punto de vista teórico, esto ocasionaría que la red aprenda muy lentamente durante algunos periodos. En la práctica, debido a que la resolución de las computadoras y de las tarjetas de adquisición de datos es finita, pueden llegar a anularse los incrementos de los coeficientes de peso y detenerse el aprendizaje de la red (independientemente de que pueda existir un error en regulación grande).

En los experimentos de la sección V, se utilizó una tarjeta de adquisición de datos con resolución de 12 bits, esto ocasiona que el actuador reciba una acción de control del 100% para cualquier valor de "$u(t)$" mayor a 0.99976 y una acción de control de 0% para cualquier valor de "$u(t)$" menor a 0.00024. Esto quiere decir que cuando el argumento de la función de activación (sigmoide), se encuentra fuera del intervalo [-8.33, 8.33], el actuador recibirá una señal de control de 0% ó 100% sin importar que tan lejos se encuentre del intervalo.

Considerando que, en el caso que nos ocupa, el argumento de cada una de las funciones de activación está compuesto por tres sumandos, al acotar los coeficientes de peso en el intervalo de [-4,4] y normalizando los errores en regulación que se introducen a la red, se consigue que uno sólo de los sumandos tenga la capacidad para lograr el 98.2% de la respuesta de la neurona y que sea necesaria la intervención de los tres sumandos para lograr una salida, de la función de activación, que sature la señal de control.

Con este método, de normalizar el error en regulación y acotar los valores que pueden tomar los coeficientes de peso, se logra reducir el rango en el que las variaciones de "$u(t)$" no causan variaciones en la señal de control del actuador y se elimina la posibilidad de que los coeficientes de peso dejen de ajustarse cuando el error en regulación es distinto de cero. Por último, cabe aclarar que con las consideraciones anteriores, los incrementos "Δv_j" y "Δw_{ji}" quedan acotados en el rango: [-0.25η', 0.25η'], esto es conveniente para controlar la influencia del coeficiente de aprendizaje.

En [15] se explica que, una red neuronal de dos capas, con un número arbitrariamente grande de nodos en la capa oculta, puede aproximar cualquier función continua $\mathbf{f} \in C(\Re^n, \Re^m)$ sobre cualquier subconjunto compacto de $\Re^n$. Adicionalmente, en [16] se demuestra que es suficiente con pedir que la función sea continua, acotada y no constante para garantizar una aproximación arbitrariamente buena. Al tomar la decisión de acotar los coeficientes de peso en el intervalo [-4,4] y normalizar los errores en regulación, también estamos asegurando la convergencia de los coeficientes de peso para cualquier planta estable, con la condición de que se escoja un coeficiente de aprendizaje adecuado.

4.2) PROGRAMACIÓN DE UN C.N.A.

CONFIGURACIÓN DE LA TARJETA PCL-818HG.

La tarjeta de adquisición de datos, requiere que se ajusten algunos selectores; En nuestra aplicación, se utilizó la configuración siguiente:

SW1	Seleccionado para 16 canales sencillos.
SW2	Dirección base: 300(hex)
JP1	Canal para DMA = 3
JP2	Referencia interna de voltaje.
JP3	Referencia interna: -10 V.
JP4	Reloj de 1 MHz.
JP5	FIFO habilitado.
JP6	Señal de disparo externo = EXT. Señal del contador 0 = G0.
JP7	Interrupción del FIFO = IRQ2.

ALGORITMO DEL C.N.A.

Para implementar el Controlador Neuronal Autoajustable, se realizó un programa que contempla los siguientes puntos:

I) Ajuste de los rangos de voltaje para las entradas y selección de los canales de entrada.
II) Actualización del vector de entradas.
III) Cálculo de la salida de las neuronas.
IV) Escritura del canal D/A.
V) Lectura del canal A/D.
VI) Ajuste de los coeficientes de peso de la red.
VII) Regreso al paso II.

LISTADO DEL PROGRAMA DE UN C.N.A.

En el apéndice "A" se muestra el programa que desarrollamos para implementar el C.N.A.

4.3) IMPLEMENTACIÓN DEL C.N.A. EN SISTEMAS NO LINEALES

RESULTADOS EXPERIMENTALES

Los experimentos se realizaron utilizando una computadora con procesador Pentium a 166MHz, una tarjeta de adquisición de datos "PCL-818HG" con resolución de 12 bit (velocidad de muestreo de hasta 100kHz), y un amplificador para servomotores de corriente directa modelo "412" de 24-90 VCD, 10 A. En la Figura 4.4 se muestra una vista general de la implementación de los experimentos.

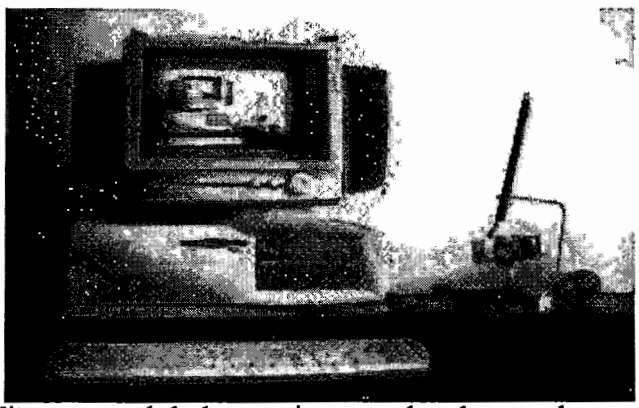


Figura 4.4 Vista general de los equipos usados durante los experimentos.

Se probó el Controlador Neuronal Autoajustable en tres sistemas mecánicos; el primero corresponde a un brazo con un grado de libertad; actuado por un motoreductor con relación de 1600:1, el segundo es similar al anterior; pero con un motoreductor con relación de 10:1 y el último corresponde a un péndulo invertido en el que la base es giratoria y está directamente acoplada a un servomotor. En lo que resta de esta sección se describen cada uno de los sistemas y se muestran los resultados más representativos de los experimentos realizados.

SISTEMA #1: BRAZO CON UN GRADO DE LIBERTAD Y REDUCTOR DE 1600:1.

En este sistema, la base del brazo móvil está conectada a un potenciómetro que descansa sobre un soporte fijo. Un motoreductor con relación de reducción de 1600:1, alineado con el eje de giro del potenciómetro, es responsable de generar el movimiento mediante un brazo en "L" conectado a la flecha del motor y terminado en un anillo (con un diámetro del doble que el del brazo móvil); por el que pasa el brazo móvil. La variación del punto en el que el brazo en "L" transmite el movimiento al brazo móvil, el juego que permite el anillo y el hecho de que el

motor no se montó rígidamente a su base, hacen que el sistema sea no lineal y ocasionan dificultades para controlarlo. En las Figuras 4.5a y 4.5b se muestra las imágenes de este sistema para dos posiciones distintas del brazo móvil.

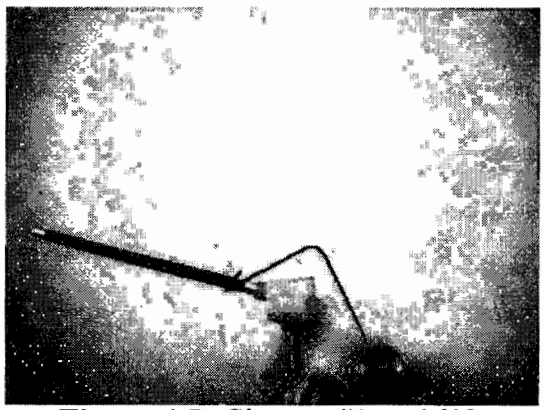


Figura 4.5a Sistema #1 en 160°.

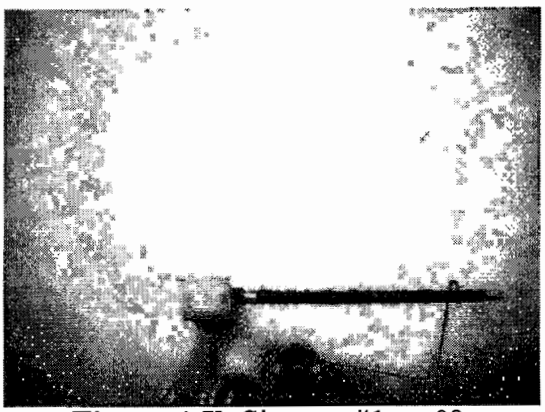


Figura 4.5b Sistema #1 en 0°.

Para tener un punto de comparación, se controló el sistema con un PID sintonizado para que responda rápidamente ante cambios en escalón de la referencia. Los parámetros de este controlador se ajustaron en K = 100, KTτd = 9.2 ms. y (K / Tτi) = (1 / 4.6 s). La Figura 4.6a muestra que, como resultado de estos ajustes, el tiempo requerido para pasar de 25° a 90° es de 6.2 s. y 8.3 s. para ir de 90° a 25°; Adicionalmente se encontró que no es posible reducir ese tiempo sin ocasionar oscilaciones cuando la referencia cambia de 90° a 155°.

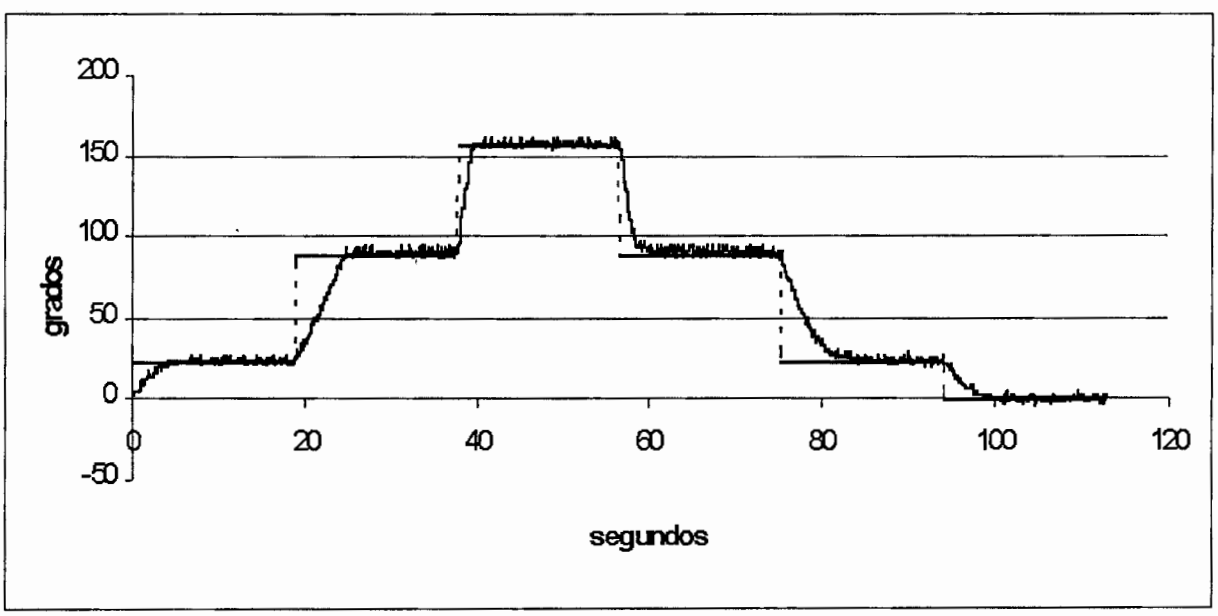


Figura 4.6a Sistema #1 controlado con un PID.

La señal de control correspondiente, a estos ajustes del controlador PID, se muestran en la Figura 4.6b. Como se puede apreciar, esta señal es considerablemente ruidosa (como consecuencia de que el potenciómetro utilizado para medir la posición del brazo introduce ruido al sistema y este es amplificado por efecto de la acción de control derivativa), esto produce calentamiento, vibración y ruido en el actuador y amplificador; Teniendo como resultado la reducción en la vida útil de estos componentes y en algunas ocasiones del sistema completo.

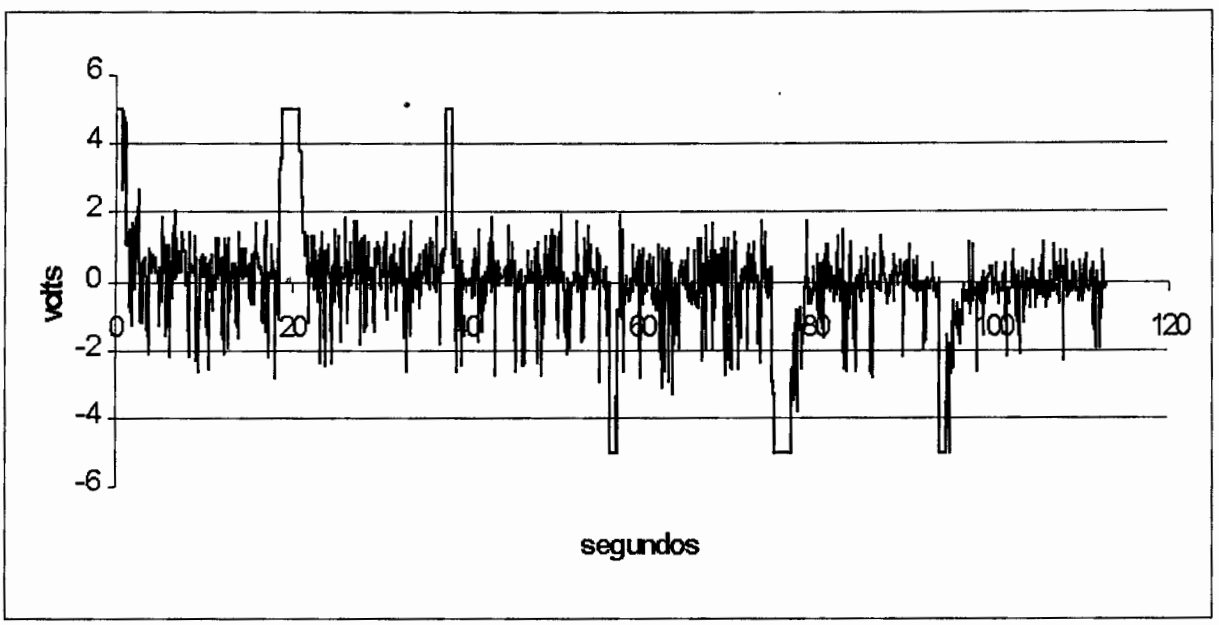


Figura 4.6b Señal de control del sistema #1 controlado con un PID.

En la Figura 4.7a se muestra que, al eliminar la acción diferencial y mantener sin cambio los demás parámetros del PID; se reduce en un segundo el tiempo necesario para pasar de 25° a 90° y en dos segundos el tiempo para ir de 90° a 25°. Pero en la Figura 4.7b se muestra un acercamiento de la Figura 4.7a, en el que se pueden apreciar las oscilaciones (con amplitud de 15°), que se generan al rededor de los 155°. Cabe hacer notar que, en un sistema mecánico industrial, se puede presentar el mismo efecto cuando se reduce la fricción en un amortiguador como consecuencia de una disminución en la viscosidad del fluido de este dispositivo.

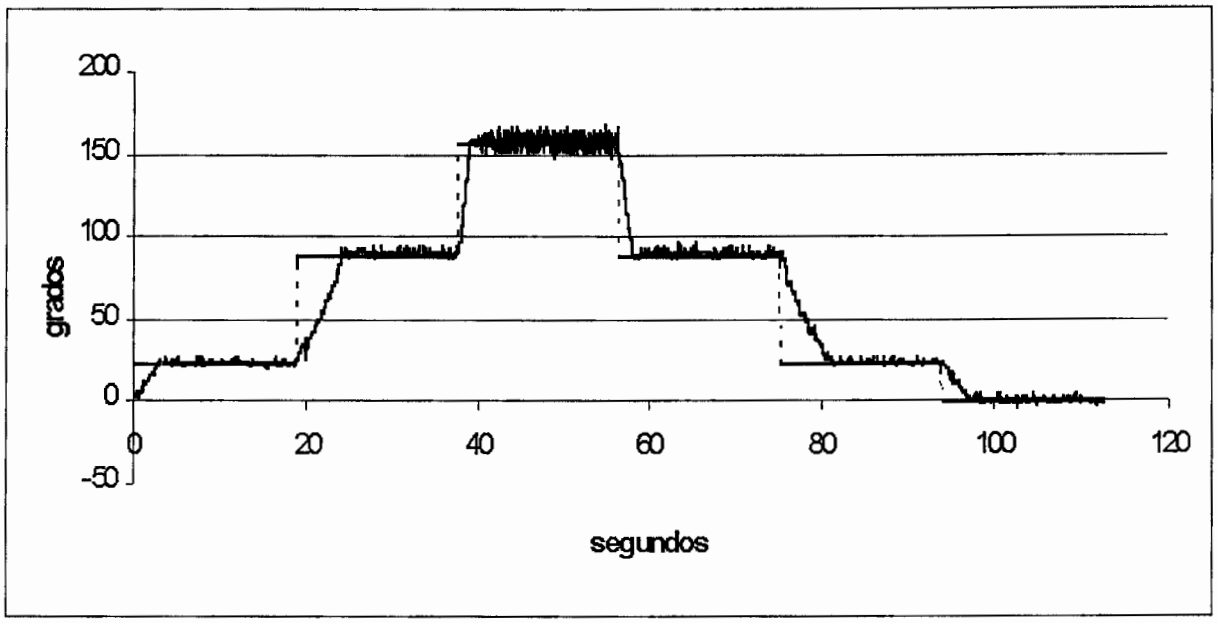


Figura 4.7a Sistema #1 controlado con un PI.

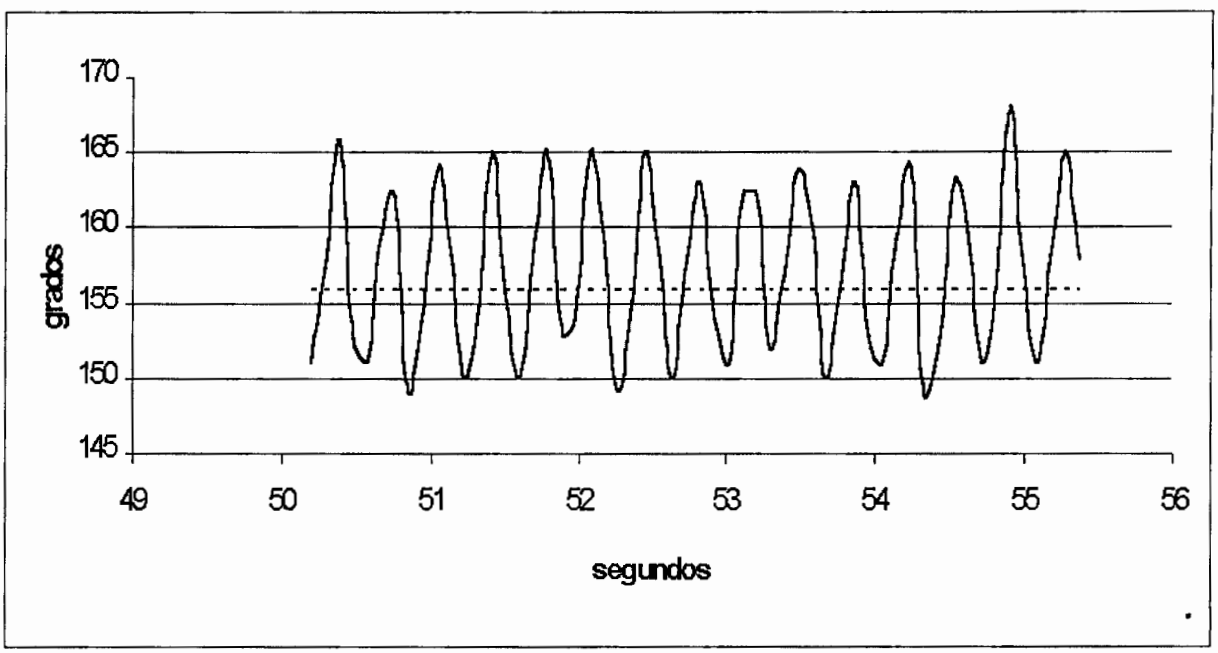


Figura 4.7b Detalle de oscilaciones en el sistema #1 controlado con un PI.

Teniendo como referencia los resultados de las dos figuras anteriores, pasaremos a revisar los resultados obtenidos, para este sistema, en los distintos experimentos realizados con el Controlador Neuronal Autoajustable (C.N.A.). En todos los casos se usó un periodo de muestreo de 0.47 ms.

En el experimento de la Figura 4.8 se usó un coeficiente de aprendizaje de $\eta = 0.01$. Podemos observar que el tiempo de adaptación de los coeficientes de peso de la red (el cual coincide con el necesario para alcanzar la primer referencia), es de 50 s. y que aparecen algunos sobrepasos de hasta 7°, que disminuyen gradualmente a medida que pasa el tiempo.

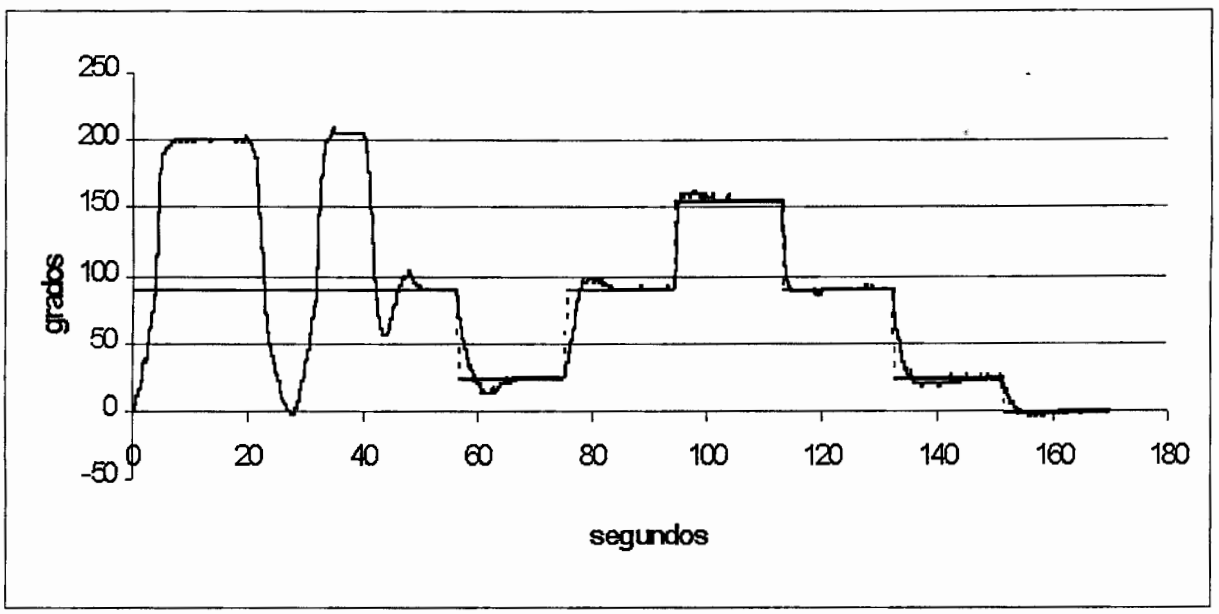


Figura 4.8 Sistema #1 controlado con un C.N.A. ($\eta = 0.01$).

Para alcanzar más rápido la primer referencia, podemos aumentar el valor de "η", pero esto tiene la desventaja de que también aumenta la amplitud de los sobrepasos. Una alternativa consiste en utilizar el coeficiente de aprendizaje propuesto en el trabajo de Aguado et al. ($\eta' = \eta + \alpha$ abs[e_y]); de esta forma aceleramos la adaptación de los coeficientes de peso cuando el error en regulación es grande. En la Figura 4.9 se muestra que, para un ajuste de $\eta = 0.01$ y $\alpha = 0.1$, se reduce a 26 s. el tiempo de adaptación de los coeficientes de peso de la red, pero tiene el inconveniente de que aumenta la magnitud de los sobrepasos; presentando para el primero un valor de 17° y valores decrecientes para los que se generan a medida que pasa el tiempo.

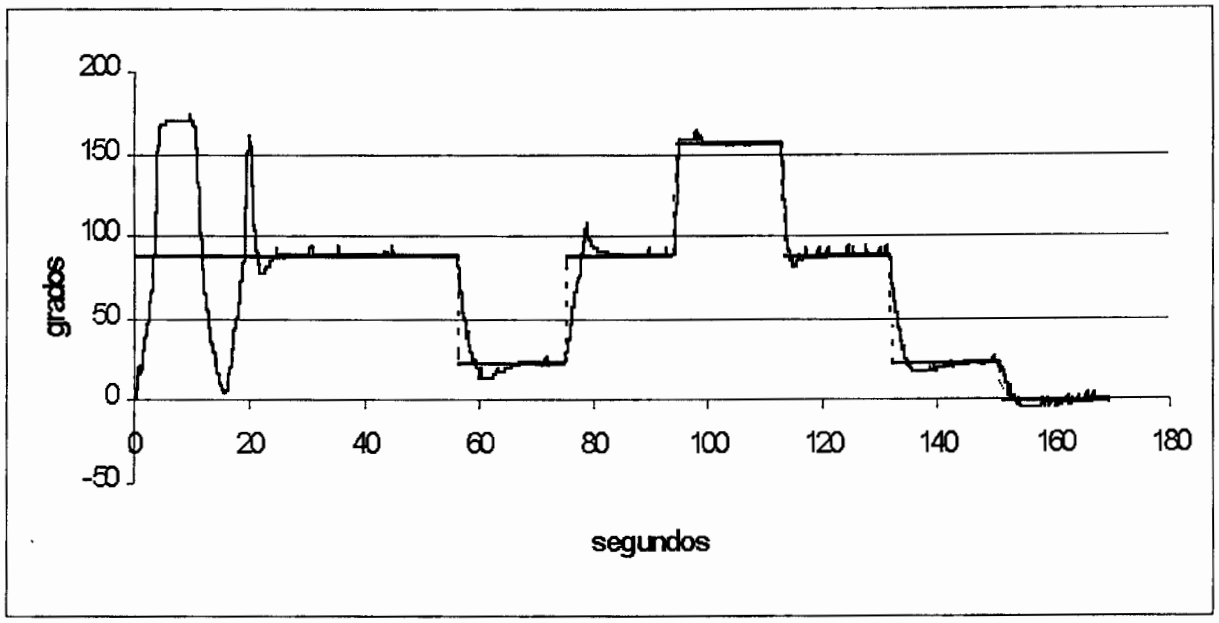


Figura 4.9 Sistema #1 controlado con un C.N.A. ($\eta = 0.01$ y $\alpha = 0.1$).

Durante los experimentos, nos dimos cuenta de la conveniencia de proponer un coeficiente de aprendizaje variable en el tiempo de la forma $\eta' = f(t)(\eta + \alpha \text{ abs}[e_y])$; de esta forma es posible lograr que el controlador se adapte rápidamente cuando no conoce el sistema y lentamente (para incorporar las variaciones qué pueda sufrir la planta), cuando los coeficientes de peso de la red estén suficientemente ajustados. Se experimentó haciendo que f(t) comenzara con un valor grande y que disminuyera en forma exponencial, lineal y en escalón; Encontrando que los mejores resultados corresponden a f(t) en escalón.

En la Figura 4.10a se muestran los resultados de ajustar ($\eta = 0.5$ y $\alpha = 5.0$) para la primer referencia, ($\eta = 0.05$ y $\alpha = 0.5$) para la segunda referencia y ($\eta = 0.001$ y $\alpha = 0.01$) de la tercer referencia en adelante. El tiempo de adaptación de los coeficientes de peso se redujo a 12 s. y el tiempo requerido para que el sistema pase de 25° a 90° es de 2.6 s. y de 3 s. para ir de 90° a 25° (42% y 36% de los tiempos requeridospor el controlador PID de la Figura 4.6a). La Figura 4.10b muestra que la señal de control es notoriamente más suave que la del PID y en las Figuras 4.10c a 4.10f se aprecia que la mayor parte de la convergencia de los coeficientes de peso se da durante el periodo inicial de adaptación; el cual corresponde con las primeras dos señales de referencia.

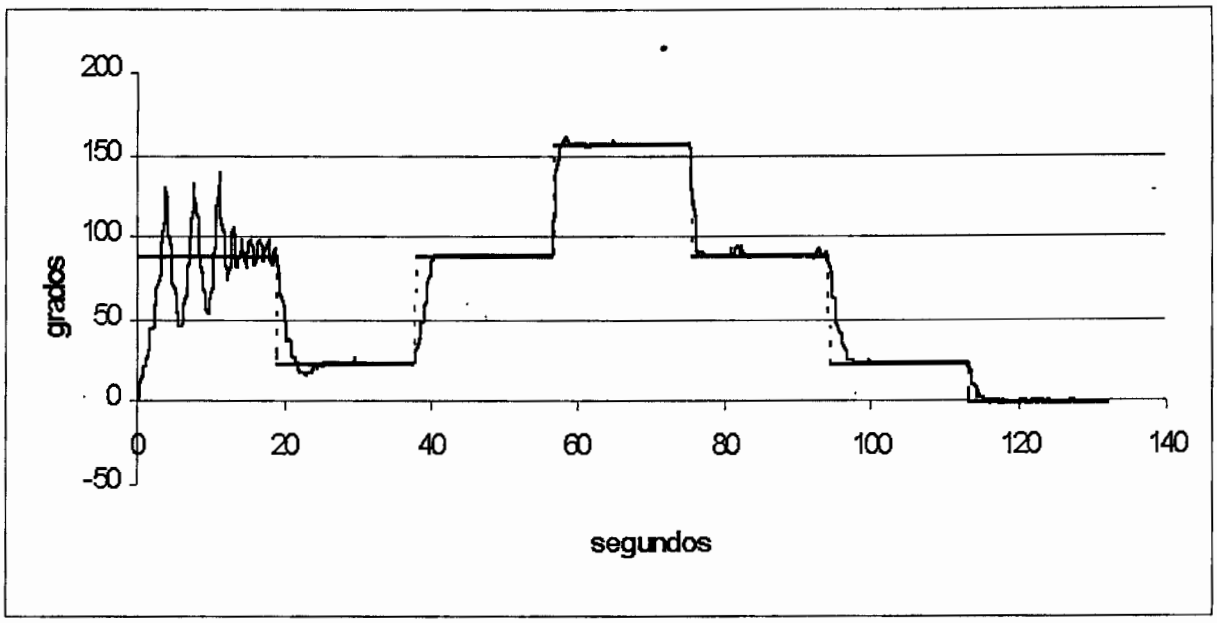


Figura 4.10a Sistema #1 controlado con un C.N.A. (η, α variando en forma escalonada).

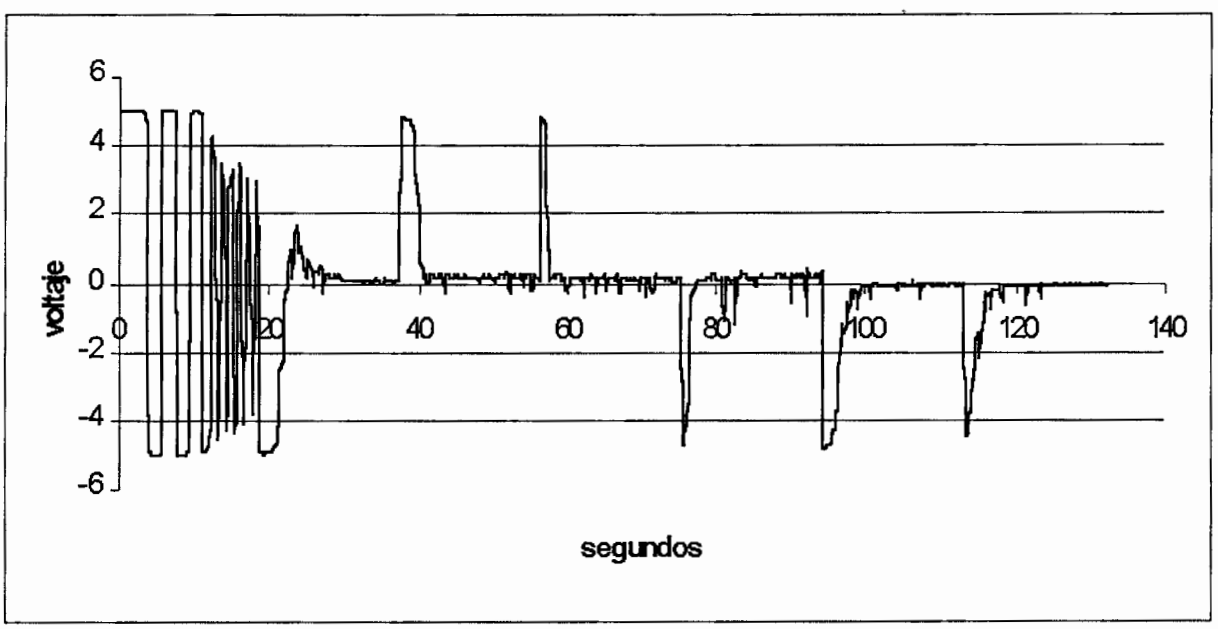


Figura 4.10b Señal de control del C.N.A. (η, α variando en forma escalonada).

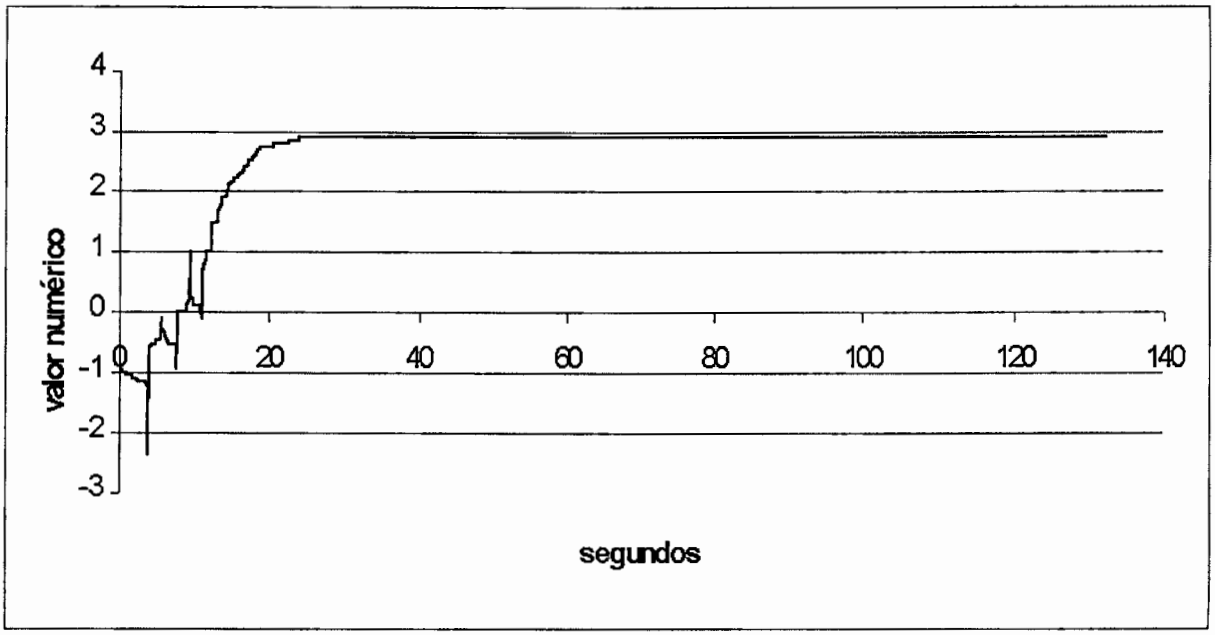


Figura 4.10c Coeficiente de peso de la capa intermedia: $w[1,2]$

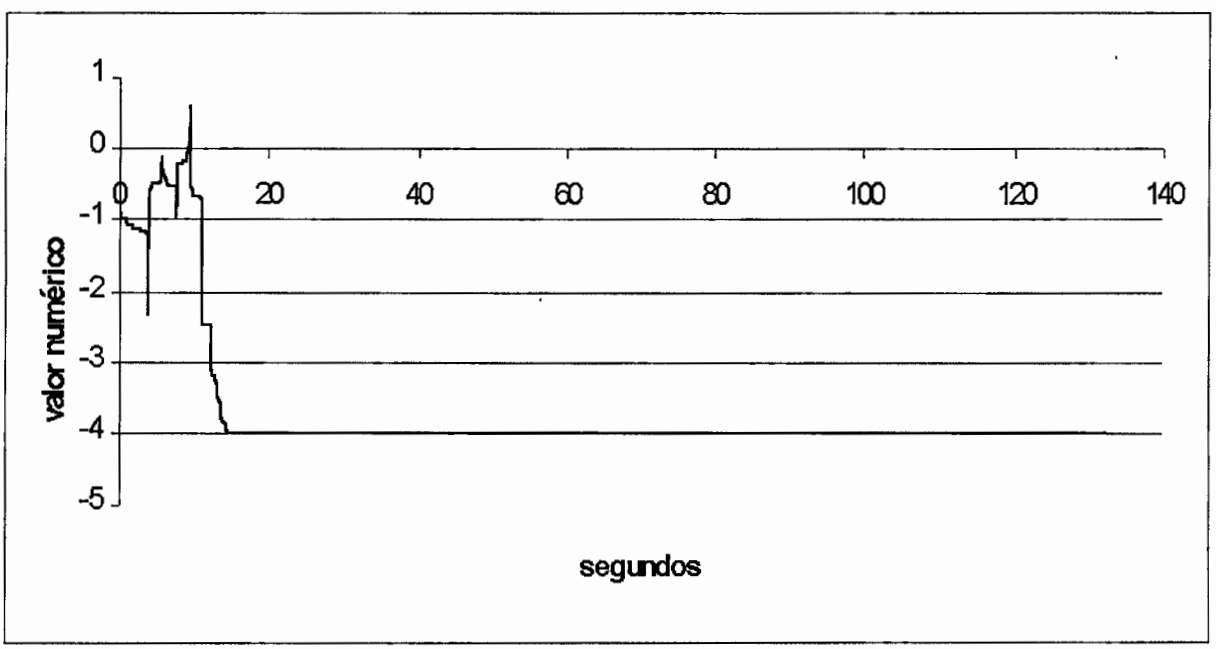


Figura 4.10d Coeficiente de peso de la capa intermedia: $w[2,2]$

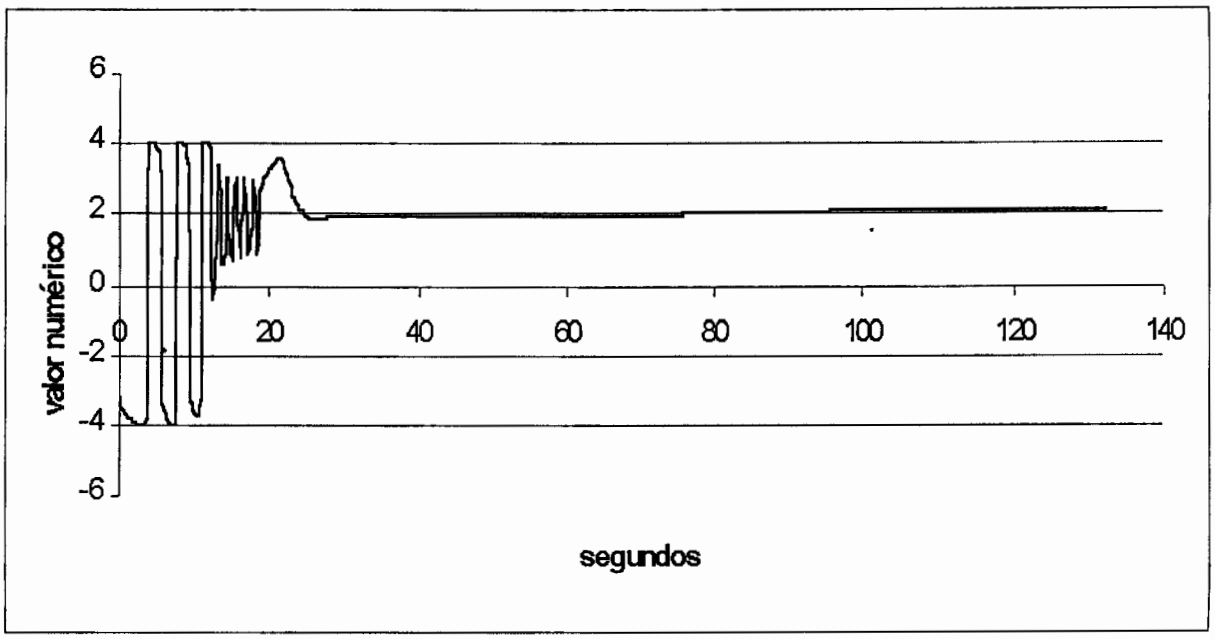


Figura 4.10e Coeficiente de peso de la capa de salida: $v[1]$

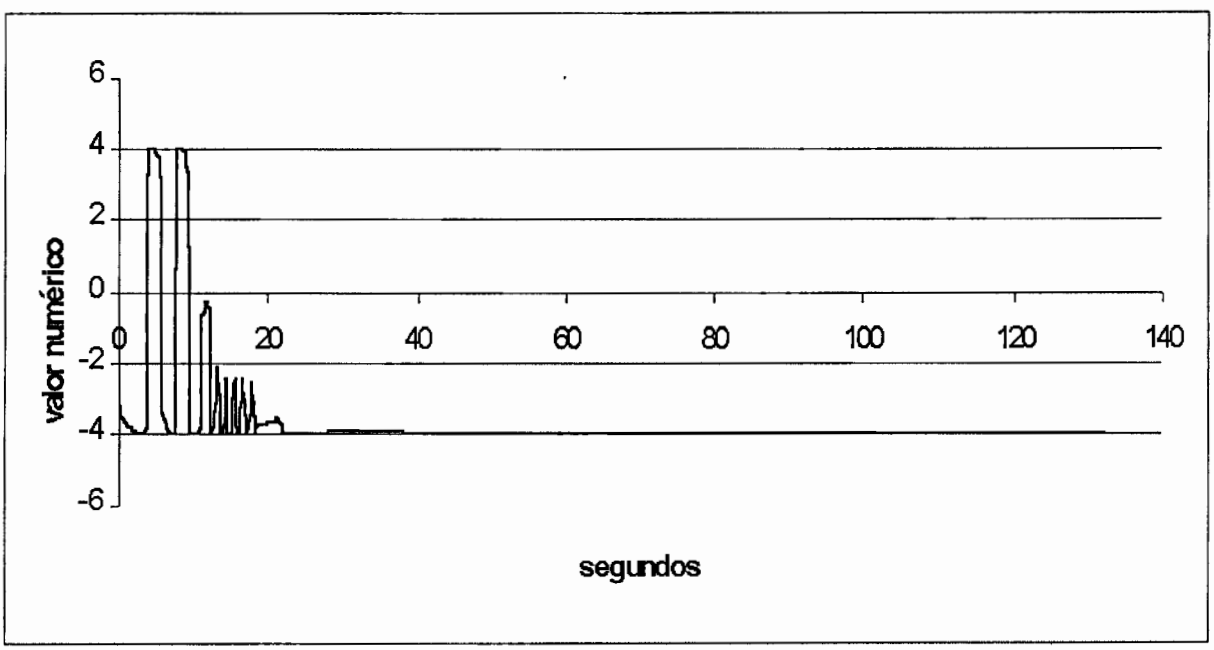


Figura 4.10f Coeficiente de peso de la capa de salida: $v[2]$

Con la idea de reducir el número de parámetros por ajustar en el C.N.A., se repitió el experimento ajustando ($\eta = 0.5$ y $\alpha = 0$) para la primer referencia, ($\eta = 0.05$ y $\alpha = 0$) para la segunda referencia y ($\eta = 0.001$ y $\alpha = 0$) de la tercer referencia en adelante. El resultado se

muestra en la Figura 4.11; en la que se puede apreciar que el resultado no es tan bueno como el de la Figura 4.10a, pero sigue siendo superior al del PID.

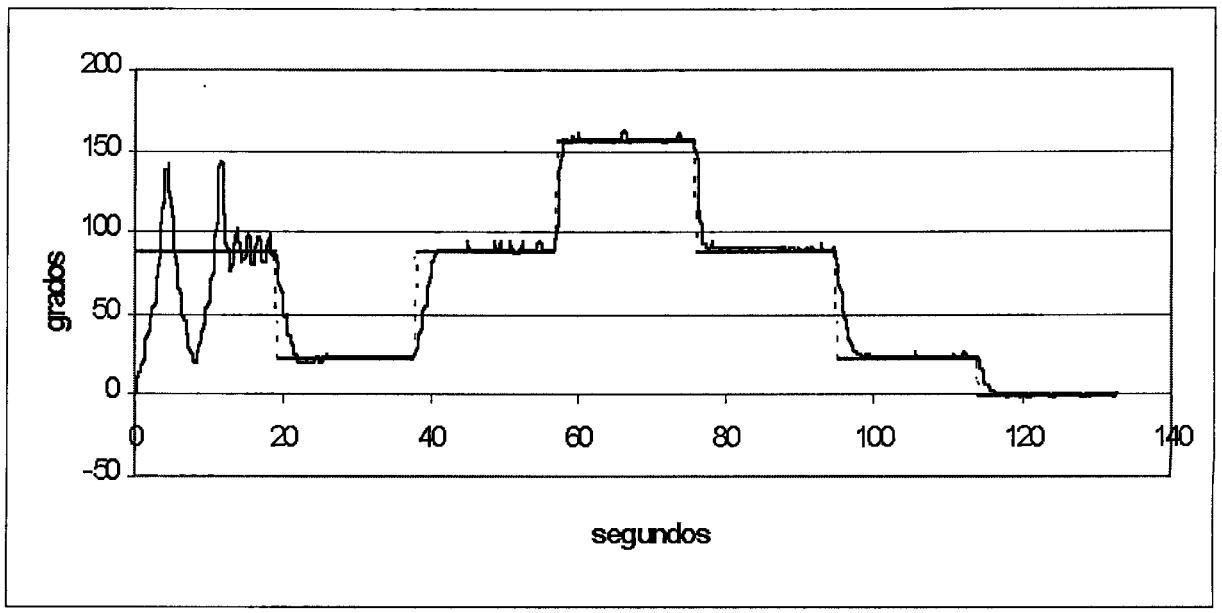


Figura 4.11 Sistema #1 controlado con un C.N.A. (η variando en forma escalonada).

SISTEMA #2: BRAZO CON UN GRADO DE LIBERTAD Y REDUCTOR DE 10:1.

Este sistema es similar al anterior pero en este caso se utilizó un motoreductor con relación de reducción de 10:1. Este motor alcanza velocidades mayores pero proporciona un par inferior al del motor del Sistema #1. En las Figuras 4.12a y 4.12b se muestra las imágenes de este sistema para dos posiciones distintas del brazo móvil.

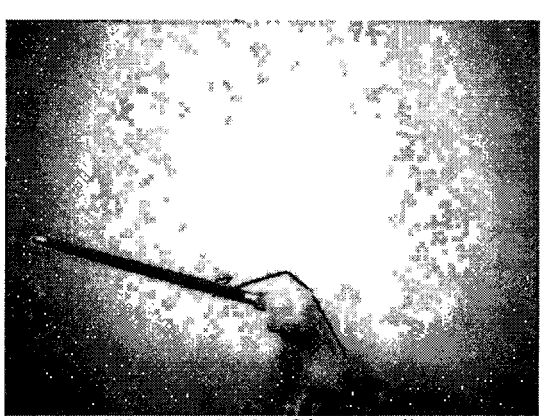


Figura 4.12a Sistema #2.

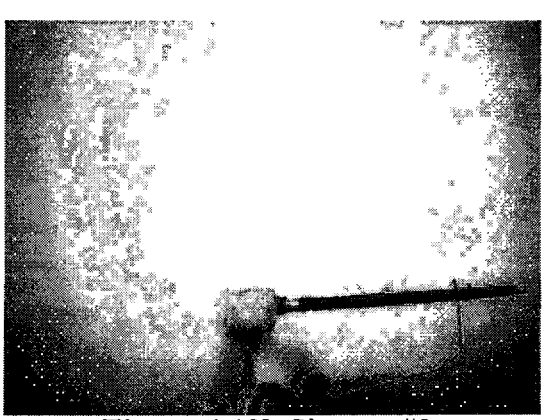


Figura 4.12b Sistema #2.

Se sintonizó un PID para que el sistema responda rápidamente ante cambios en escalón de la referencia. Los parámetros de este controlador se ajustaron en K = 3, KTτd = 0.46 ms. y (K / Tτi) = (1 / 0.23 s). La Figura 4.13 muestra que, como resultado de estos ajustes, se obtiene una respuesta oscilatoria del sistema. Al intentar reducir la amplitud del sobrepaso, aumentó considerablemente el tiempo necesario para alcanzar la referencia. Este sistema resultó ser más difícil de controlar debido a que el motor es de menor potencia que el del Sistema #1 y la relación del reductor es mucho menor; por lo que se presenta el doble efecto de reducir el par disponible en la flecha del motor y permitir el acoplamiento de las perturbaciones de la carga hacia el rotor del motor.

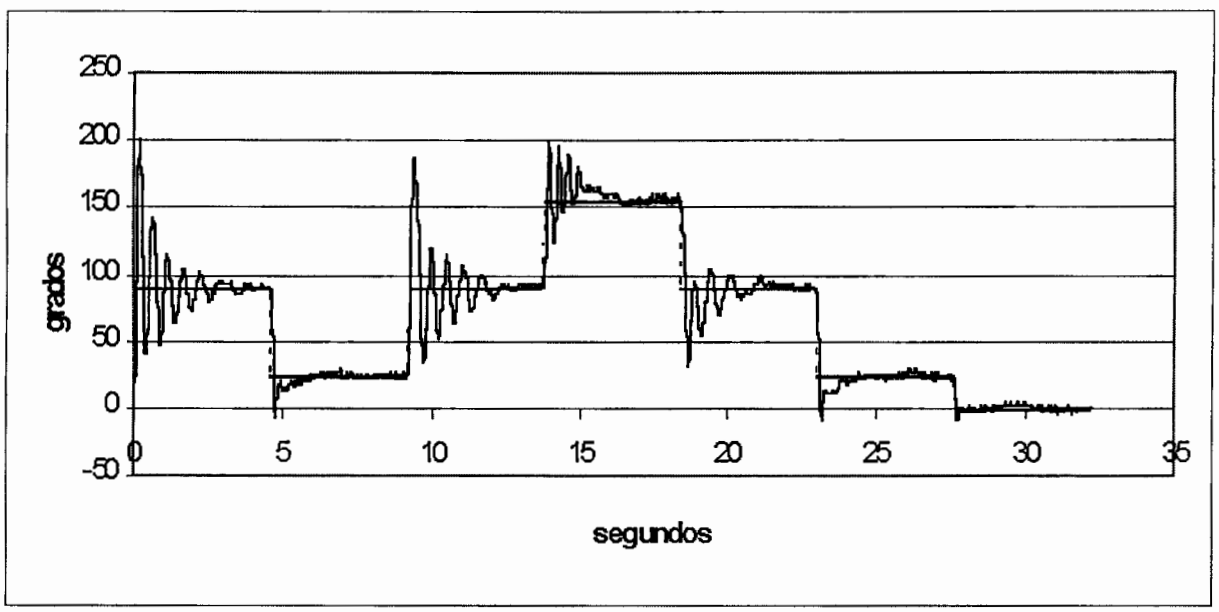


Figura 4.13 Sistema #2 controlado con un PID.

En la Figura 4.14 se usó un Controlador Neuronal Autoajustable con un coeficiente de aprendizaje variable en el tiempo; ajustando η = 0.5 para la primer referencia y η = 0.05 de la segunda referencia en adelante. Es interesante notar que en este caso, el tiempo de adaptación de los coeficientes de peso es del mismo orden de magnitud que el tiempo que requiere el PID para estabilizarse ante el primer cambio de referencia (6.2 s. del CNA contra 4.4 s. del PID). Adicionalmente, el CNA necesita la mitad el tiempo requerido por el PID para estabilizar el sistema cuando se lleva a 90°. En este experimento no se mejora la respuesta utilizando el parámetro "α" y por eso se ajustó con valor cero.

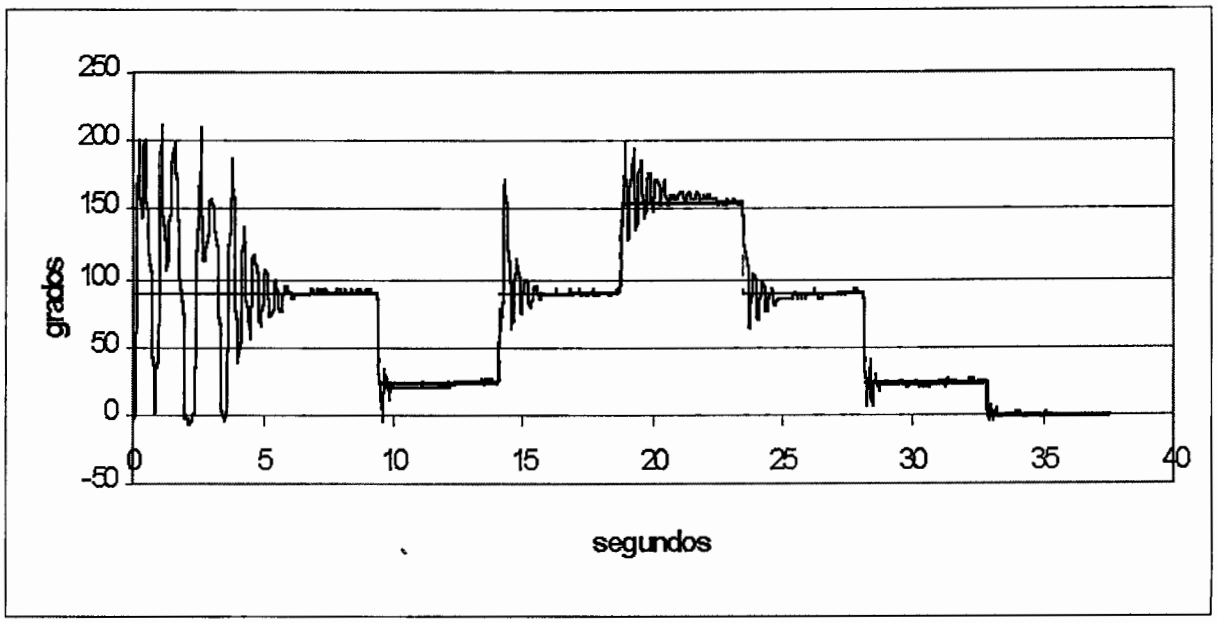


Figura 4.14 Sistema #2 controlado con un C.N.A.

SISTEMA #3: PÉNDULO INVERTIDO CON BASE GIRATORIA.

En la Figura 4.15 se muestra una imagen del péndulo invertido. En este sistema no fue posible ajustar el C.N.A.; únicamente se logro que oscilara el péndulo entre las placas que restringen su movimiento, sin que se lograra reducir la amplitud de las oscilaciones.

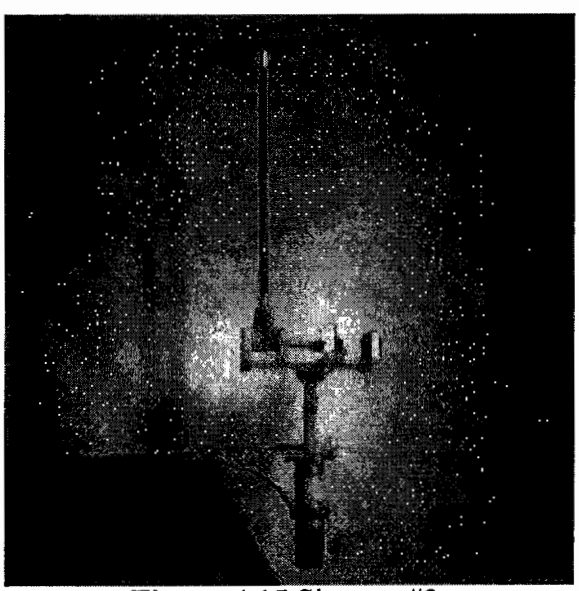


Figura 4.15 Sistema #3.

5) CONCLUSIONES

Los experimentos realizados, con los primeros dos Sistemas No Lineales, muestran que fue posible partir de una red sin entrenamiento previo y lograr una adaptación, en tiempo real, de los coeficientes de peso del Controlador Neuronal Autoajustable (C.N.A.); logrando reducir el tiempo de asentamiento (necesario para alcanzar la referencia de 90°), en más de 50% del tiempo requerido por un controlador PID. Para obtener estos resultados, fue decisivo el efecto de utilizar un coeficiente de aprendizaje variable (en función del error y de la evolución de los coeficientes de peso), y acelerar la convergencia del algoritmo de Retropropagación (mediante nuestra propuesta de acotar los coeficientes de peso dentro del rango [–4,4] y normalizar las lecturas del error en regulación). Adicionalmente se encontró que la señal de control del CNA es considerablemente menos ruidosa que la proporcionada por un PID; por lo que se puede esperar un incremento en la vida útil del sistema debido a la reducción de la vibración y calentamiento del amplificador y el actuador.

Durante las pruebas realizadas en el tercer Sistema No Lineal, se pudo controlar el sistema con un PID (en un rango pequeño de 85° a 95°), pero no se pudo controlar con el C.N.A. Por el momento no es posible concluir si esto se debe a las características propias del sistema o si es un efecto de las limitaciones, la tarjeta de adquisición de datos y de la computadora, para reducir los periodos de muestreo y de control; Sin embargo, es relevante el hecho de que el periodo de control del PID con el que se controló este sistema fue 20 veces menor que el del C.N.A. y por otro lado es significativo considerar que fueron necesarios 25,000 y 10,000 ciclos de adaptación de los coeficientes de peso para controlar los sistemas #1 y #2 respectivamente.

Los resultados, de este trabajo de tesis, fueron expuestos en el "II Simposio de Control Automático CIMAF'99" Habana, Cuba [05] y en el "Segundo Taller México - Hungría sobre Automatización Industrial y Ciencias de los Materiales" Qro. –S.J.R. México [04]. En ambos foros despertó interés la presentación de resultados experimentales y se puso de manifiesto la carencia de este tipo de trabajos; los cuales, aunque requieren más tiempo para implementar el sistema y no son tan fáciles de modificar como en la simulación, son determinantes para lograr que estas herramientas de control estén disponibles para ser implementadas en la industria.

Cabe aclarar que dentro de este enfoque práctico, las oscilaciones que presenta el sistema durante el periodo inicial de adaptación de los coeficientes de peso, no representan ningún problema para su implementación. Lo que se debe hacer es limitar la respuesta del sistema (mediante ajustes de hardware o consideraciones de software), dentro de los rangos permitidos de operación.

Con base en los experimentos realizados, se sugiere el siguiente procedimiento de ajuste del Controlador Neuronal Autoajustable:

1) Determinar (experimentalmente), el máximo valor de "η" que permite alcanzar la referencia con oscilaciones de amplitud menor al 10% del rango.

2) Determinar el mínimo valor de "η" que permite la adaptación del C.N.A. dentro del tiempo en el que podemos esperar variaciones de los parámetros de la planta.

3) Dividir el rango encontrado para "η" en unos cuantos escalones (sugerimos iniciar con tres). Estos valores se cambiarán, en forma decreciente durante los primeros cambios de la referencia; dejando, al final, el valor mínimo en forma permanente.

4) Variar la referencia y el coeficiente de aprendizaje en forma escalonada, tratando de cubrir todo el rango de la variable de control y esperando a que se alcance la referencia, en forma estable, antes de cambiarla nuevamente.

Nota: En el caso de usar el parámetro "α", se debe ajustar de forma similar a "η".

Para seguir desarrollando el Controlador Neuronal Autoajustable, como una herramienta de control industrial, es necesario realizar experimentos con otras familias de Sistemas No Lineales y extender la aplicación a sistemas con entradas y salidas múltiples (MIMO). De igual manera será importante implementar este controlador en dispositivos de hardware más rápidos (por ejemplo el procesador digital de señales (DSP)), así como ensayar otros algoritmos de entrenamiento [08] y estructuras de redes neuronales [18] que prometen ser de utilidad para reducir el periodo inicial de adaptación de los coeficientes de peso. Finalmente se abre la posibilidad de utilizar el C.N.A. como predictor (para aplicaciones en tiempo real), en un esquema de control óptimo que utilice Algoritmos Genéticos.

6) BIBLIOGRAFÍA

[01] A.M.S. Zalzala y A.S. Morris. "Neural Networks for Robotic Control". Ed. Ellis Horwood, 1996.

[02] A. Van Ooyen y B. Nienhuis. "Improving The Convergence of the Back-Propagation Algorithm". Neural Networks, pags. 465-471, vol. 5, Pergamon Press 1992.

[03] Alberto Aguado, Antonio Ordáz, Alfonso Noriega y Vladimir Rauch. "Self-Tuning Neural Controller". Documento preliminar, U.A.Q. 1997.

[04] Alfonso Gómez, Alfonso Noriega y Alberto Aguado. "Self-Tuning Neural Controller and its Application to a Non Linear System". II Taller México–Hungría sobre Automatización Industrial y Ciencias de los Materiales, Qro–S.J.R. México, 9-12 marzo 1999, pags. 163-174.

[05] Alfonso Gómez, Alfonso Noriega y Alberto Aguado. "Implementación Práctica de un Controlador Neuronal Autoajustable". II Simposio de Control Automático CIMAF'99, La Habana, Cuba, 22 -26 marzo 1999, pags. 265-277.

[06] Anil K. Jain, Jianchang Mao y K. M. Mohiuddin. "Artificial Neural Networks: A Tutorial". IEEE Computer, pags. 31-44, marzo 1996.

[07] Giovanni Leonetti del Negro. "Simulador de Redes Neuronales Artificiales". Tesis para obtener el título de Ingeniero en Computación, U.N.A.M. 1992.

[08] G.W.Ng "Application of Neural Networks to Adaptive Control of Nonlinear Systems". Ed. John Wiley & Sons Inc, 1997.

[09] Heng-Ming Tai, Junli Wang y Kaveh Ashenayi "A Neural Network-Based Tracking Control System". IEEE Transactions on Industrial Electronics, pags. 504-510, vol. 39, no.6, diciembre 1992.

[10] James A. Freeman y David M. Skapura. "Redes Neuronales Algoritmos, aplicaciones y técnicas de programación". Ed. Addison-Wesley, 1993.

[11] Jean-Jacqes E. Slotin y Weiping Li. "Applied Nonlinear Control". Ed. Prentice Hall, 1991.

[12] Jerzy Moscinski y Zbigniew Ogonowski. "Advanced Control with Matlab & Simulink". Ed. Ellis Horwood, 1995.

[13] José R. Hilera y Victor J. Martínez. "Redes Neuronales Artificiales Fundamentos, modelos y aplicaciones". Ed. Addison-Wesley, 1995.

[14] Julio Tanomaru y Sigeru Omatu. "Process Control by On-Line Trained Neural Controllers". IEEE Transactions on Industrial Electronics, pags. 511-521, vol. 39, no.6, diciembre 1992.

[15] Kumpati S. Narendra y Kannan Parthasarathy. "Identification and Control of Dynamical Systems Using Neural Networks". IEEE Transactions on Neural Networks, pags. 5-27, vol. l, no. 1, marzo 1990.

[16] Kurt Hornik. "Approximation Capabilities of Multilayer Feedforward Networks". Neural Networks, Vol. 4, pags. 251-257, 1991.

[17] M.M. Abdelhameed. "Adaptive Neural Network Based Controller for Robots". Mechatronics, Vol. 9, No. 2, pags. 147-162, marzo 1999.

[18] M.O. Efe, O. Kaynak. "A Comparative Study of Neural Network Structures in Identification of Nonlinear Systems". Mechatronics, Vol. 9, No. 3, pags. 287-300, abril 1999.

[19] Madan M. Grupta y Dandina H. Rao "Neuro-Control Systems: A Tutorial". A Selected Reprint Volume, IEEE Neural Networks Council, *Sponsor*, IEEE PRESS.

[20] Norio Baba. "A New Approach for Finding The Global Minimum of Error Function* of Neural Networks". Neural Networks, pags. 367-373, vol. 2, Pergamon Press 1989.

[21] Paul J. Werbos "Backpropagation Through Time: What It Does and How to Do It". Proceedings of the IEEE, pags. 1550-1560, vol. 78, no.10, octubre 1990.

[22] Richard P. Lippmann. "An Introduction to Computing With Neural Nets". IEEE ASSP Magazine, pags. 4-22, abril 1987.

[23] T. Fukuda y T.Shibat "Theory and Applications for Neural Networks for Industrial Control Systems". IEEE Transactions on Industrial Electronics, pags. 472-489, vol. 39, no.6, diciembre 1992.

[24] Xianzhong Cui y Kang G. Shin. "Direct Control and Coordination Using Neural Networks". IEEE Transactions on Systems , Man and Cybernetics, pags. 686-697, vol. 23, no.3, mayo 1993.

APÉNDICE

A

Listado del programa de un Controlador Neuronal Autoajustable

```
/*************************************************************************
 *                                UAQ / DEPFI                            *
 *      MAESTRIA EN CIENCIAS "INSTRUMENTACION Y CONTROL AUTOMATICO"      *
 *                                                                       *
 *  Tesis:  "Criterios de Diseño del Controlador Neuronal Autoajustable *
 *                  y su aplicación a un Sistema No Lineal".             *
 *  Programa: CONTROLADOR NEURONAL AUTOAJUSTABLE (TARJETA "PCL-818HG")   *
 *  Asesor:   M.C. Alfonso Noriega Ponce.                                *
 *  Alumno:   Fís. Alfonso Gómez Espinosa.                               *
 *  22/nov/98                                                            *
 *************************************************************************/

/***************************** DEFINICIONES ****************************/
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <graph.h>
#include <math.h>

#define BASE      0x300
#define AD_LOW    0x300
#define AD_HIGH   0x301
#define TRIGGER   0x300
#define RANGO     0x301
#define MUX_SCAN  0x302
#define DA_LOW    0x304
#define DA_HIGH   0x305
#define STATUS    0x308
#define CONTROL   0x309
#define FIFO_INT  0x306
#define COUNT_ENA 0x310

#define N 3

float W[N+1][N+1]= {{0,0,0,0},
                    {0,0.0001,-0.0001,0.0001},
                    {0,-0.0001,0.0001,-0.0001},
                    {0,0.0001,-0.0001,0.0001}};
float V[N+1]= {0,0.0001,-0.0001,0.0001};
float E[N+1]= {0,0,0,0};
float H[N+1]= {0,1,1,1};
float U[2];

float ETA_SIGN=0.0;
float ALFA=0.0;
float REF=2048;


/**********************PROTOTIPOS DE FUNCIONES*************************/
void  inicializa();
float lee();
void  escribe(float actuador);
void  controla();
void  actualiza();
void  calcula();
void  retro_pro();
void  esp();
```

```
/************************ PROGRAMA PRINCIPAL ***************************/
main()
{
_clearscreen(_GCLEARSCREEN);
 inicializa();

 ETA_SIGN=0.5;
 ALFA=5;
 REF=2048;
 controla();

 ETA_SIGN=0.05;
 ALFA=0.5;
 REF=1048;
 controla();

 ETA_SIGN=0.001;
 ALFA=0.01;
 REF=2048;
 controla();

 REF=3048;
 controla();

 REF=2048;
 controla();

 REF=1048;
 controla();

 REF=688;
 controla();

 escribe(2048);
}

/***************** FUNCION PARA INICIALIZAR LA TARJETA ******************/
void inicializa()
{
   int val=0x70;
   outp(CONTROL,val);
   outp(MUX_SCAN,0);
   outp(RANGO,4);
}

/****************** FUNCION PARA LEER EL CANAL A/D ********************/
float lee()
{
    unsigned char low,high,lo;
    float hi,res;
    outp(TRIGGER,0);
    do{}while((inp(STATUS)&0x10)==0);
    low =inp(AD_LOW);
    high=inp(AD_HIGH);
    lo =low>>4;
    hi =(float)high;
    outp(STATUS,0);
```

```
    res =(float)lo+hi*16;
    E[0]=(res-REF)/2048;
    return res;
}

/***************** FUNCION PARA ESCRIBIR EL CANAL D/A *****************/
void escribe(float actuador)
{
    unsigned char low,high;
    unsigned int temp;
    temp=(unsigned int)actuador;
    low= (unsigned char)(temp<<4);
    high=(unsigned char)(temp>>4);
    outp(DA_LOW,low);
    outp(DA_HIGH,high);
}

/********************* FUNCION GENERAL DE CONTROL **********************/
void controla()
{
 unsigned long    i=0;
 unsigned long    l=0;
 float sensor;
 for(i=0;i<=100;i++)
   {
    for(l=0;l<=400;l++)
      {
      actualiza();
      calcula();
      escribe(U[1]*4095);
      esp();
      sensor=lee();
      retro_pro();
      }
     printf("%2.2f\t%2.0f\t%2.0f\n",(float)(i*0.188),(sensor-
           688)*90/1360,(REF-688)*90/1360);
   }
}

/******************  ACTUALIZA EL VECTOR DE ERROR  ********************/
void  actualiza()
{
     int i;
     for(i=N;i>=1;i--)
       {
        E[i]=E[i-1];
       }
}

/****************  CALCULA LAS SALIDAS DE LAS NEURONAS *****************/
void  calcula()
{
     int i,j;
     float r,s;
     for(j=1;j<=N;j++)
     {
      s=0;
```

```
       for(i=1;i<=N;i++)
       {
           s=s+W[j][i]*E[i];
       }
       H[j]=1/(1+exp(-s));
      }
      r=0;
      for(j=1;j<=N;j++)
      {
       r=r+V[j]*H[j];
      }
      U[1]=1/(1+exp(-r));
}

/************************  RUTINA DE ESPERA  ************************/
void  esp()
{
 unsigned long    j=0;
 unsigned long    k=0;
 float ret=0;
      for(j=0;j<=0;j++)
       {
       for(k=0;k<=4000;k++)
         {
          ret=ret;
         }
       }
}

/******************  CALCULA LOS COEFICIENTES DE PESO ******************/
void  retro_pro()
{
     int i,j;
     float d1=0;
     float d2[N+1]= {0,1,1,1};

     d1=E[0]*U[1]*(1-U[1]);

     for(j=1;j<=N;j++)
     {
      d2[j]=d1*V[j]*H[j]*(1-H[j]);
     }

     for(j=1;j<=N;j++)
     {
      for(i=1;i<=N;i++)
      {
          W[j][i]=W[j][i]+(ETA_SIGN+ALFA*fabs(E[0]))*d2[j]*E[i];
          if(W[j][i]>4) W[j][i]=4;
          if(W[j][i]<-4) W[j][i]=-4;
      }
      V[j]=V[j]+(ETA_SIGN+ALFA*fabs(E[0]))*d1*H[j];
      if(V[j]>4) V[j]=4;
      if(V[j]<-4) V[j]=-4;
     }
}
```

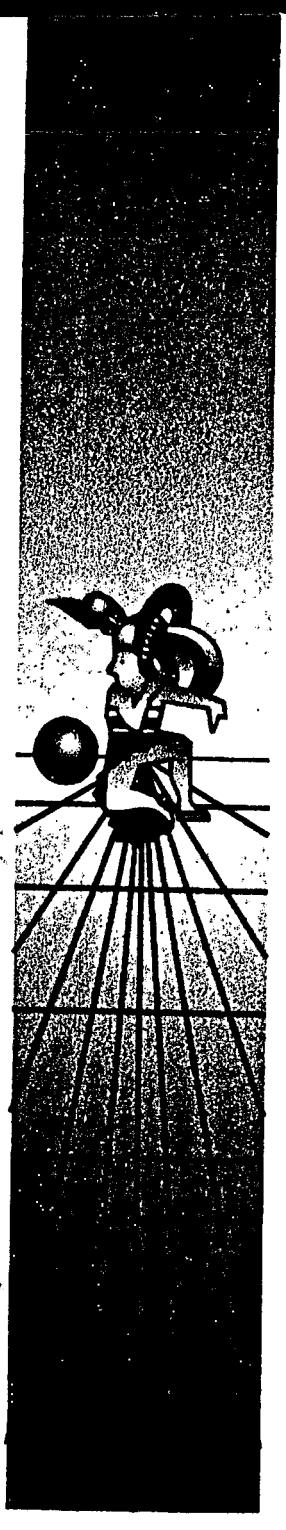
APÉNDICE

B

Constancia de participación:

"Self-Tuning Neural Controller and its application to a Non Linear System."

(Presentado en el: Segundo Taller México – Hungría sobre Automatización Industrial y Ciencias de los Materiales 9 – 11 marzo 1999, Qro. – S.J.R. México)



Universidad Autónoma de Querétaro a través de la
Facultad de Ingeniería otorgan la presente

Constancia

Al: **Ing. Alfonso Gómez Espinoza**

Por su participación como ponente en el

CONGRESO INTERNACIONAL EN AUTOMATIZACION INDUSTRIAL Y CIENCIA DE LOS MATERIALES MEXICO-HUNGRIA

Celebrado del 9 al 11 de Marzo de 1999.
Querétaro-San Juan del Río, Qro. México.

M. en I. José de Jesús Hernández Espino
Director de la Facultad de Ingeniería

Dr. Gilberto Herrera Ruiz
Coordinador de la Maestría en Ciencias
Instrumentación y Control Automático

Ing. Wenceslao Ortiz Vargas
Coordinador de la Carrera Electromecánica
Facultad de Ingeniería, San Juan del Río, Qro.

CIATEQ CONACYT CONCYTEQ amipa

APÉNDICE

C

Constancia de participación:

"Implementación Práctica de un Controlador Neuronal Autoajustable."

(Presentado en el: II Simposio de Control Automático
Conferencia Internacional CIMAF´ 99
22 – 26 marzo, 1999, Habana Cuba)

CONFERENCIA
INTERNACIONAL
CIENCIA Y TECNOLOGIA
PARA EL DESARROLLO

CIMAF '99

INSTITUTO DE CIBERNÉTICA, MATEMÁTICA Y FÍSICA

II SIMPOSIO DE CONTROL AUTOMÁTICO

Como reconocimiento al trabajo: *Implementación Práctica de un Controlador Neuronal Autoajustable.*

de los autores: *Msc. Alfonso Noriega Ponce, Dr. Alberto Aguado Behar, Msc. Alfonso Gómez Espín.*

expuesto en este evento.

La Habana, 22 - 26 marzo de 1999

Dr. Abelardo del Pozo Quintero.
Presidente del II Simposio de Control Automático.

Dr. Manuel Lazo Cortés.
Presidente del Comité Organizador CIMAF '99.