UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

CONTROL POR APRENDIZAJE NEURO-FUZZY DE UN SISTEMA SERVO-HIDRAULICO DE ALTA FRECUENCIA

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE
INGENIERO MECATRÓNICO

ELEAZAR CRISTIAN MEJIA SANCHEZ

PROMOCIÓN 2005-II

LIMA - PERÚ

2011

Dedicado a Mis padres:

Jesús y Agripina,

a mis hermanos:

Ronald y Liliana

y a todas las personas que han contribuido

en mi formación personal y profesional

CONTENIDO

PRÓLOGO

Los avances en el estudio de materiales sometidos a cargas cíclicas llevaron al desarrollo de modelos analíticos que describan su comportamiento. Por lo tanto, para validar estos modelos así como para determinar las propiedades de nuevos materiales, surgió la necesidad de desarrollar maquinas para ensayos mecánicos.

Uno de los ensayos de mayor importancia en el estudio de materiales son los relacionados con la fatiga, este tipo de ensayos requiere un elevado número de ciclos por lo cual surge la necesidad de desarrollar cada vez máquinas más rápidas con el propósito de disminuir el tiempo de los ensayos, por tal motivo es de vital importancia desarrollar controladores que puedan trabajar a altas frecuencias.

La presente Tesis muestra un modelo de control diseñado para este tipo de sistemas servo-hidráulicos, lo cual fue dividido en 5 capítulos como serán detallados a continuación.

Capítulo I: Se presentan los antecedentes, la justificación, el objetivo, la metodología de trabajo, los alcances y las limitaciones propias de la presente Tesis.

Capítulo II: Se describe en forma breve y simplificada los fundamentos de la inteligencia artificial, detallando los conceptos básicos de los sistemas fuzzy logic, redes neuronales y una descripción general de los sistemas híbridos principalmente de los sistemas híbridos neuro-fuzzy.

Capítulo III: Se presenta un resumen del modelamiento de un sistema servo-hidráulico de una máquina para ensayos de fatiga.

Capítulo IV: Este capítulo presenta el diseño de un control hibrido neuro-fuzzy para un sistema servo-hidráulico de alta frecuencia.

Capítulo IV: Se presentan el hardware para la implementación de un sistema experimental, así también los resultados de las simulaciones del controlador propuesto. Adicionalmente se detallan algunas características de la respuesta del sistema a las variaciones de algunos parámetros del sistema neuro-fuzzy.

Finalmente esta Tesis presenta una serie de conclusiones y recomendaciones obtenidas en base a los resultados de las simulaciones. Así mismo se presenta la bibliografía requerida para el desarrollo del presente trabajo.

CAPITULO I
INTRODUCCIÓN

1.1. Antecedentes.

Los avances tecnológicos y las innovaciones industriales hacen uso de los sistemas hidráulicos y neumáticos en diversas aplicaciones como por ejemplo, industrias automatizadas, explotación de minerales, maquinaria pesada, ensayo de materiales, industria aeroespacial, marítima, etc. Estos sistemas tienen muchas ventajas como su alta durabilidad y capacidad para generar fuerzas a altas velocidades. Infelizmente estos sistemas tienen características no lineales [1], las cuales surgen de la compresibilidad del fluido hidráulico, el rozamiento en el cilindro-pistón, etc. Las maquinas para ensayos de materiales son desarrollados utilizando sistemas servo-hidráulicos para poder generar fuerzas y torques relativamente altos, así como altas frecuencias de trabajo. El propósito de hacer estos ensayos es proporcionar al diseñador y personal de mantenimiento las características de los materiales para prevenir la vida útil de un equipamiento o estructura.

El estudio de la fatiga es importante porque la mayoría de las fallas de los elementos de maquinas en servicio se debe a la fatiga, debido a que la rotura por fatiga ocurre sin ningún aviso previo.

Una máquina para ensayos de fatiga es un equipamiento capaz de someter cuerpos de prueba a esfuerzos cíclico, con el propósito de poder prever el comportamiento de los mismo tanto en condiciones críticas como en condiciones normales de trabajo. En este contexto es necesario desarrollar un control eficiente para que el sistema pueda alcanzar frecuencias típicas de trabajo, que para el caso de los metales es de hasta cien veces por segundo [6]. Además el controlador tiene que ser capaz de superar las variaciones no lineales de la dinámica del sistema.

Debido a la no linealidad del sistema, el mejor desempeño no es alcanzado con controladores clásicos, debido a que toda la información del proceso no es conocida con anticipación. Una alternativa a este problema es el desarrollo de controladores con la capacidad de identificar y controlar el sistema dinámico no lineal.

En este trabajo fue desarrollado la optimización del control de una máquina servo-hidráulica para ensayos de fatiga utilizando técnicas de inteligencia artificial tales como: redes neuronales, lógica fuzzy y sistemas híbridos neuro-fuzzy; y se presenta las simulaciones computacionales del sistema para un ensayo de fatiga con cargas de amplitud constante y variable. Finalmente, es presentado el esquema general de una máquina servo-hidráulica, en la cual podría ser implementado el control por aprendizaje neuro-fuzzy propuesto.

1.2. Justificación

Los ensayos de materiales son realizados con el propósito de obtener información sobre los límites de esfuerzo y el tiempo de vida de una pieza o elemento de máquina, con la finalidad de determinar nuevos tipos de materiales, procesos de

fabricación o tratamientos térmicos de los materiales para así definir sus aplicaciones industriales.

Los ensayos de fatiga son un tipo de ensayo de gran importancia en las industrias para prever posibles fallas, hacer un programa de mantenimiento, y los fabricantes determinar los materiales adecuados que puedan soportar los esfuerzos repetitivos requeridos para la tarea del mecanismo. Este tipo de ensayo consiste en aplicar a una pieza o cuerpo de prueba apropiado esfuerzos cíclicos repetitivos, dependiendo del tipo de ensayo a ser realizados; por ejemplo, los ensayos de iniciación y de propagación de fisura.

Los equipos de ensayo de fatiga son constituidos por un sistema de aplicación de cargas, que permite alterar la intensidad y el sentido del esfuerzo, y por un contador del número de ciclos. La figura (1.1) presenta un ensayo de iniciación de fisura por fatiga, el cual proporciona datos cuantitativos de las características del material antes de la rotura.

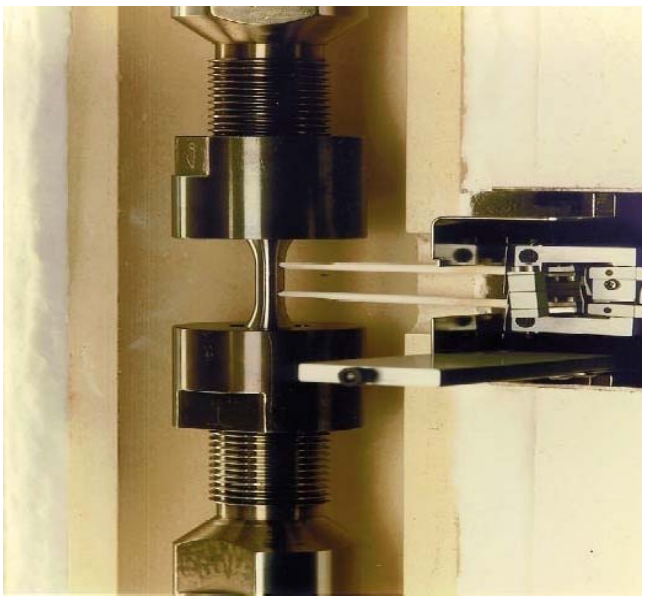

Figura 1.1. Ensayo de Fatiga.

Los ensayos de fatiga son casi independientes de la frecuencia de trabajo, excepto en polímeros, donde debido a las altas frecuencias el material es calentado, influenciando la resistencia útil del material; pero si el cuerpo de prueba pudiera ser refrigerado, en ese caso la frecuencia no afectaría en los resultados. Para un tipo de material dado y sometido a tensiones alternadas, la resistencia útil a la fatiga básicamente solo depende del número de ciclos aplicados al material de prueba, es por este motivo que los equipos para ensayos de materiales trabajan a altas frecuencias, procurando una reducción del tiempo y costos de los ensayos, sin que esto interfiera en los resultados.

Las técnicas de inteligencia artificial tales como: redes neuronales, lógica fuzzy y sistemas híbridos neuro-fuzzy tienen la capacidad de modelar una gran cantidad de plantas de dinámica no lineal con un arbitrario grado de precisión; esto permite diseñar un esquema de control óptimo aplicado a sistemas servo-hidráulicos, sin la necesidad de tener conocimiento de todo el modelo de la planta.

1.3. Planteamiento del Problema

En la presente Tesis, se desarrolla una técnica de control por aprendizaje neuro-fuzzy, el cual es aplicado en un sistema servo-hidráulico para ensayos de fatiga con el propósito de realizar ensayos de fatiga y así poder determinar el tiempo de vida útil de los materiales.

La implementación del control por aprendizaje neuro-fuzzy fue desarrollado con el objetivo de acelerar el proceso de aprendizaje. Esta metodología consiste en hacer un control de tipo bang-bang, restringiendo la servo-válvula a trabajar siempre en sus límites máximos de operación, tratando de mantener siempre

completamente abierto la servo-válvula en una dirección o otra. Para mantener la servo-válvula trabajando en sus límites de funcionamiento, los instantes óptimos reversiones son determinado por el sistema neuro-fuzzy, la cual tiene 2 variables de entrada, la variable *gama* que es el doble de la amplitud de la carga deseada y la variable *mínimo* que es el valor mínimo de carga deseada. Y el proceso de aprendizaje es realizado mediante las actualizaciones de los pesos del sistema neuro-fuzzy utilizando el error obtenido entre la carga deseada y la carga alcanzada a cada ciclo durante la ejecución de los ensayos de fatiga, mejorando la respuesta del sistema a cada evento. Estos puntos de reversión dependen de diversos factores, como la amplitud y carga media deseada, y también son influenciados por la dinámica del sistema. El desarrollo de un controlador por aprendizaje neuro-fuzzy fue motivado por la necesidad de tener un agente con la capacidad de aprendizaje y almacenamiento de los puntos óptimos de reversión de la servo-válvula.

1.4. Objetivo

El objetivo de esta tesis es desarrollar un controlador de fuerza por aprendizaje neuro-fuzzy basado en técnicas de inteligencia artificial de un sistema servo-hidráulicos de alta frecuencia. Este controlador es aplicado al modelo de una máquina para ensayo de fatiga y se evalúa su desempeño.

1.5. Alcances

Las maquinas para ensayos mecánicos de materiales tienen diferentes tipos de controladores, los cuales son mejorados con el propósito de alcanzar mayores

frecuencia de trabajo. En esta Tesis se plantea un modelo de control por aprendizaje neuro-fuzzy.

Algunos trabajos, para el control de sistemas servo-hidráulicos utilizaron un controlador no-lineal basado en Lyapunov [1], donde la ley adaptativa propuesto permite eliminar las incertidumbres en los parámetros del sistema hidráulicos. El desempeño del controlador no-lineal fue comparado con un controlador PD, del cual se puede concluir que el controlador propuesto superó ampliamente al controlador PD tanto en la simulación como experimentalmente.

En los últimos años, sistemas basados en redes neuronales y fuzzy fueron encontrando un camino en aplicaciones de control y muchas otras áreas de la ingeniería. Los resultados obtenidos capturaran la atención de ingenieros que trabajan con sistemas reales. Esto es debido principalmente a los resultados obtenidos y frecuentemente a la facilidad de implementación cuando se desarrollen sistemas de control basados en redes neuronales y fuzzy [2].

Otro trabajo [3] presentó el uso de un controlador por aprendizaje, utilizando técnicas neuro-fuzzy en el diseño de un control de trayectoria, modelando el actuador electro-hidráulico.

Otra aplicaciones utiliza un control híbrido adaptativo neuro-fuzzy por modelo de referencia ("*Adaptive Neuro-Fuzzy Model Reference Controller*", *ANFMRC*) [4], y fue desarrollado para mejorar el desempeño del control en un sistema neumático, este control híbrido es combinado con un control "*bang-bang*" aplicado cuando el error es grande, y un *ANFMRC* aplicado cuando el error es pequeño.

En la tesis de Alva [5], fue desarrollado un control por aprendizaje, restringiendo a la servo-válvula a trabajar siempre en sus límites extremos de operación "*bang-*

bang". En este caso, los puntos de reversión de la válvula siempre deben estar entre el valle y pico de una fuerza deseada (o entre pico y valle), y este parámetro depende de la amplitud y del menor valor de la carga solicitada. Para que la servo-válvula trabaje en sus límites de funcionamiento, una ley de aprendizaje obtiene los instantes óptimos para las reversiones. Y luego almacenado en tablas específicas para cada tipo de fuerza deseada, y actualizada continuamente.

Las principales máquinas servo-hidráulicas para ensayo de materiales que se encuentran en el mercado son de las marcas INSTRON y MTS. Las que son capaces de trabajar con células de carga desde 5 kN a 500 kN, a una frecuencia máxima teórica de 500Hz (para amplitudes muy pequeñas). Ellas pueden ejecutar ensayos de tracción, compresión, flexión y de fatiga. Tiene la habilidad de testar los más diversos materiales, incluyendo polímeros, metales y compuestos.

En la presente Tesis solo serán presentados las simulaciones del control por aprendizaje neuro-fuzzy basado en las fuerzas medidas sobre el cuerpo de prueba, aplicados a un sistema servo-hidráulico para cargas de amplitud contaste y variable, también se presenta un estudio del efecto del factor de aprendizaje "η" durante el proceso de aprendizaje.

1.6. Limitaciones

El modelo de la dinámica del sistema servo-hidráulico utilizado para las simulaciones del control por aprendizaje neuro-fuzzy fue el desarrollado por Alva [8], la cual representa una aproximación del comportamiento real de una máquina INSTRON modelo 8501 para ensayos mecánicos de fatiga.

Debido al elevado costo de los equipos para la implementación del prototipo de una máquina servo-hidráulica para ensayos mecánicos, en la sustentación de esta Tesis no se podrá realizar ninguna prueba ni presentar resultado experimental del modelo de control propuesto.

CAPITULO II

FUNDAMENTOS DE LA INTELIGENCIA ARTIFICIAL

2.1. Introducción

En este capítulo se presenta los conceptos fundamentales de la inteligencia artificial para la implementación de sistemas híbridos, tales como los sistemas híbridos neuro-fuzzy. En las últimas décadas, como una alternativa a los métodos convencionales de diseño de controladores, surgieron los modelos de control basado en lógica fuzzy (FL) y redes neuronales artificiales (ANN). Estas dos técnicas son aplicadas con éxito en diversas áreas en las cuales el control convencional presenta dificultades en el diseño del controlador.

Los sistemas basados en lógica fuzzy son apropiados para el diseño de controladores a partir del conocimiento explícito proporcionado por el especialista, donde desempeño del controlador depende básicamente de la experiencia del especialista. Entre tanto, los sistemas basados en ANN son adecuadas para el diseño de controladores basado en el conocimiento implícito embutido en un conjunto de datos, y donde su desempeño es principalmente afectado por el ajuste de sus parámetros (número de neuronas en cada capa, número de capas escondidas, etc.). Por consiguiente, muchos investigadores han

intentado integrar estas dos técnicas para generar un modelo híbrido que pueda aprovechar las ventajas de cada una de ellas y minimizar sus deficiencias. Así en los últimos años, diversos tipos de sistemas híbridos fueron desarrollados en la literatura, generalmente por la integración de las técnicas como Algoritmos Genéticos (GA), ANN y FL.

En la presente Tesis, se presentan diversas arquitecturas de sistemas híbridos basadas en la integración de sistemas fuzzy y redes neuronales artificiales.

Las redes neuronales artificiales, o comúnmente conocidas como "Neural Networks", fueron motivadas en principio por la extraordinaria capacidad del cerebro humano para ejecutar tareas de gran complejidad, no lineales y con procesamiento paralelo de la información, tales como reconocimiento de patrones, percepción, clasificación, generalización de conceptos y control motor. Estos hechos generaron el interés del estudio detallado de la constitución del cerebro y su mimetización en la concepción de sistemas con las capacidades referidas arriba fue designada por redes neuronales artificiales (ANN).

2.2. Lógica Fuzzy

La lógica fuzzy es una extensión de la lógica booleana, introducida por el Dr. Lofti Zadeh de la Universidad de Califórnica (*Berkeley*) en el año 1965. Fue desarrollado para expresar el concepto de verdad parcial, de manera que se puedan determinar valores entre el límite "completamente verdadero" y "completamente falso". Esto significa que un valor lógico difuso es un valor cualquiera en el intervalo de 0 a 1. La lógica fuzzy se torno importante en la medida en que el mundo en que vivimos no está constituido por formas absolutamente verdaderas o falsas.

Actualmente las técnicas de control pueden ser implementadas por modelos matemáticos deterministicos, las implementaciones basadas en la lógica fuzzy frecuentemente tienen un mejor desempeño, por los siguientes puntos [7]:

- Las estrategias de control fuzzy imitan un comportamiento basado en reglas obtenidas de la experiencia del especialista, en lugar de un control explícitamente restricto a modelos matemáticos como ecuaciones diferenciales. Por lo tanto, es robusto en sistemas no-lineales sin necesidad de un modelo matemático.
- El control fuzzy abarca un gran número de entradas, muchas de las cuales son apenas para condiciones especiales. Por lo tanto, algunas condiciones excepcionales (tales como alarmas) pueden ser implementadas con un menor esfuerzo computacional y flexible a modificaciones.
- La implementación de productos comerciales basados en estrategias de control fuzzy destinados al mercado tienen un menor costo, frecuentemente son más eficiente y fácilmente implementable en microprocesadores en comparación a estrategias de control convencionales, debido a una menor codificación y tiempo computacional de ejecución.

La lógica fuzzy tiene la capacidad de incorporar la forma humana de pensar en sistemas de control. De esta manera, el controlador fuzzy se comporta conforme el raciocinio que el especialista utiliza para inferir las reglas, basadas en la información que el ya conoce producto de la experiencia.

La lógica fuzzy es una variación de la lógica booleana, que solo presenta los valores de "0" y "1", sin ningún término medio. Mientras, que en la lógica fuzzy, una premisa puede asumir valores de pertenencia (grado de verdad) intermediarios. Permitiendo describir grandezas imprecisas como: altura (alto,

bajo), cantidad (mucho, razonable, poco), edad (joven, viejo), etc. En el sentido más amplio, la lógica fuzzy es casi sinónimo de la teoría de conjuntos fuzzy, una teoría en la cual los elementos tienen un grado parcial de pertenencia [8].

2.2.1 Conjuntos Fuzzy

En la teoría clásica de conjuntos, el concepto de pertenencia de un elemento a un conjunto es bien definido, de manera que para un conjunto "A" en un universo U, el elemento simplemente pertenece o no pertenece aquel conjunto, como se muestra en la siguiente función característica [9]:

$$f_A(x) = \begin{cases} 1 & \text{Si y solamente si } \; x \in A \\ 0 & \text{Si y solamente si } \; x \in A \end{cases} \tag{2.1}$$

En un sentido más amplio, Zadeh propone la generalización de la función característica, de modo que ella pueda asumir infinitos valores en el intervalo [0,1] y es representado por pares ordenados.

$$A = \{\mu_A(x) / x\} \quad x \in U \tag{2.2}$$

Donde:

$\mu_A(x)$: Representa el grado de pertenencia de x con el conjunto A

A : Conjunto fuzzy

x : La variable de interés

U : Universo de discurso

Adicionalmente, un elemento puede pertenecer a más de un conjunto fuzzy, con diferente grado de pertenencia.

En la figura 2.1 (a) se puede observar que si un elemento x fuera movido en dirección de los límites del conjunto A (color amarillo), en el punto de

cruce ocurrirá repentinamente un salto en el comportamiento de su grado de pertenencia, inicialmente de miembro para no miembro. También, el grado de pertenencia en los límites es indeterminado.

Por otro lado, la lógica fuzzy puede percibir las variaciones ocurridas en los puntos de transición de un color para otro. Los conjuntos (rango de colores) son fácilmente representables por medio de un lenguaje fuzzy (figura 2.1 b). Las funciones de pertenencia fuzzy pueden representar la variación gradual en las tonalidades. Por ejemplo, el mismo elemento x posee un grado de pertenencia 0,8 en la función de pertenencia de color amarillo, y grado de pertenencia 0,2 en la función de pertenencia de color rojo. En la figura 2.1, el eje "X" representa el universo de discurso del elemento x y el eje "Y" representa el grado de pertenencia definido entre 0 y 1.

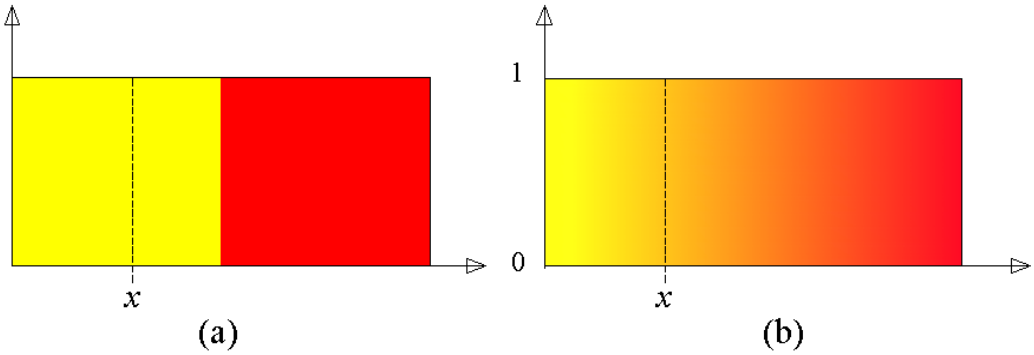

Figura 2.1. (a) Lógica booleana, (b) Lógica fuzzy.

En este momento, es importante definir el concepto de grado de pertenencia. El cual es definido por un número en el intervalo de [0,1] que determina "cuanto" una variable pertenece a un determinado conjunto. En la lógica booleana, solamente existen dos grados de pertenencia: 0% si no pertenece y 100% si pertenece al conjunto. Por otro lado, en la lógica fuzzy

existe un rango de valores entre 0% y 100%. El grado de pertenencia es descrito por la ecuación 2.2 [10].

2.2.2 Sistemas de Inferencia Fuzzy

Los sistemas fuzzy, es un modelo general que permite la identificación de los módulos que componen tal sistema, proporcionando así la idea del flujo de información dentro del mismo. Básicamente está constituido por 3 etapas, como es mostrado en la figura 2.2, donde están definidas las funciones de cada una de estas etapas.

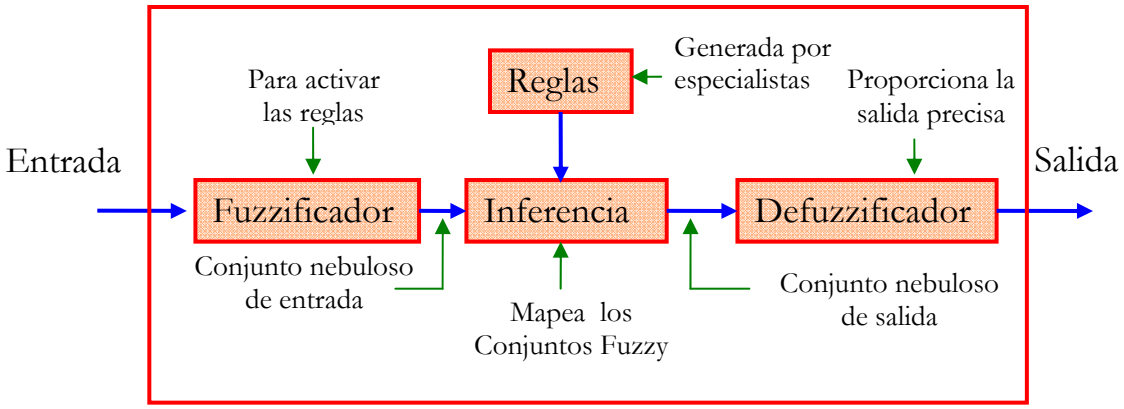

Figura 2.2. Sistema de Inferencia Fuzzy.

En la figura 2.2, se considera a la entrada como no fuzzy, debido a que en la mayoría de las aplicaciones prácticas es el resultado de mediciones. Por lo tanto, es necesario efectuarse la conversión de las entradas en una representación conocida como conjuntos fuzzy y a esto se le denomina la fuzzificación. Además de eso, en esta etapa se determinan los grados de pertenencia de las reglas para las diferentes situaciones.

En la segunda etapa, se establece el conjunto de reglas, como la relación de las variables de entrada y salida, las cuales son obtenidas por el

conocimiento y por la experiencia del especialista en la aplicación. Una vez obtenido el conjunto fuzzy de salida resultante del proceso de inferencia, es necesario efectuar la interpretación de esa información, pues en las aplicaciones prácticas son requeridas salidas precisas, el cual es realizado en la etapa de defuzzificación [9].

- **Fuzzificación**

  La fuzzificación es la conversión de las entradas exactas (número reales) para el dominio fuzzy. El fuzzificador atribuye valores lingüísticos (grados de pertenencia) empleando funciones de pertenencia a las variables de entrada. Esto se considera como una etapa de pre-procesamiento de las señales de entrada, reduciendo el número de valores a ser procesado lo que significa un menor esfuerzo computacional.

- **Reglas y Inferencia Fuzzy**

  Las regla fuzzy son implicaciones lógicas que relacionan los conjuntos fuzzy de entrada con los de salida. Generalmente son proporcionadas por un especialista, en forma de sentencias lingüísticas, constituyendo un aspecto fundamental en el desempeño de un sistema de inferencia fuzzy, como mostrado abajo.

  Si x es A y B, **entonces** z es C.

  Donde, A y B son los conjuntos fuzzy de entrada, relativos a la parte conocida como antecedentes o premisas, mientras C es el conjunto fuzzy de salida, relativo a la parte conocida como consecuente o conclusión [8].

  Estas reglas poden ser definidas previamente o alternativamente generadas automáticamente a partir de una base de datos. En la etapa de inferencia,

ocurren las operaciones de los conjuntos propiamente dichas, como la combinación de los antecedentes de las reglas del tipo *SI - ENTONCES*, generando el conjunto de salida fuzzy. En la figura 2.3, se presenta el proceso de inferencia fuzzy.

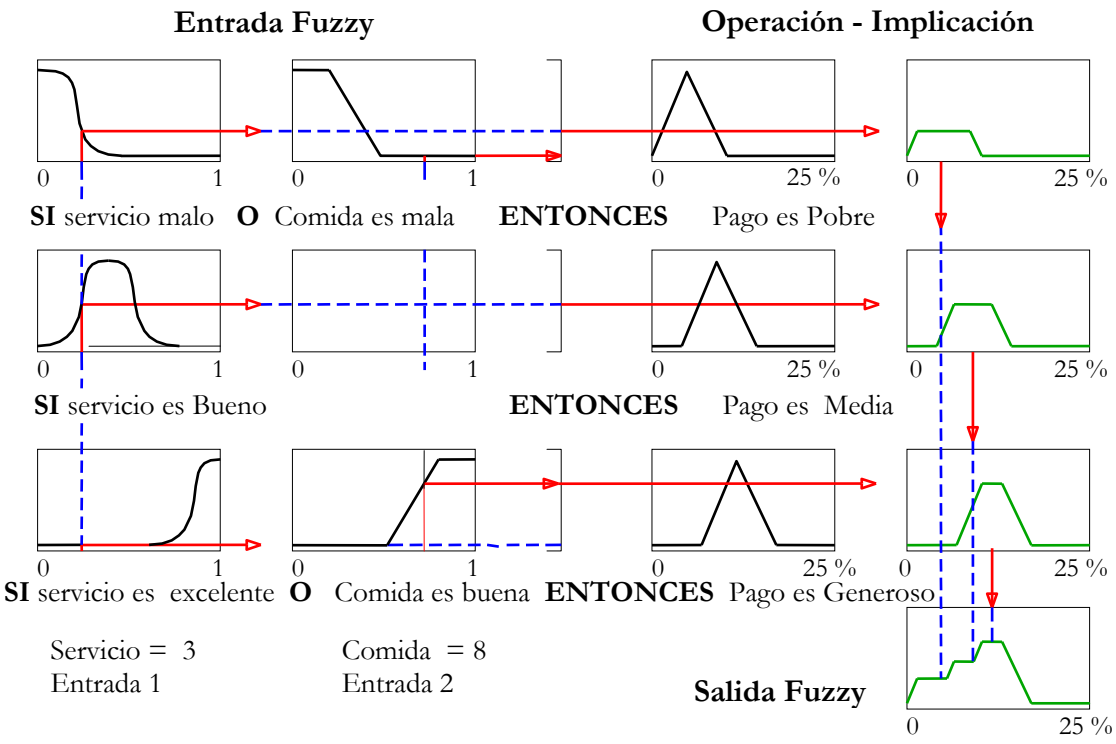

Figura 2.3. Regla y inferencia fuzzy [8].

En la figura 2.3, se muestra como ocurren las operaciones en la etapa de inferencia; la entrada exacta es transformada en entradas fuzzy, y con esos valores se calculan las operaciones - implicación y finalmente se obtiene la salida fuzzy.

- **Defuzzificación**

  En el proceso de defuzzificación, es efectuada la interpretación del conjunto fuzzy de salida inferida por las reglas, con el objetivo de obtener un valor numérico. Esto se hace necesario porque en aplicaciones prácticas son

requeridas salidas precisas con algún significado físico. Por ejemplo, en el control de una planta utilizando un controlador fuzzy, este debe proporcionar a la planta señales precisas, ya que presentar un conjunto fuzzy en la entrada de la planta no tendría significado algún.

En la práctica, el método de defuzzificación mas popular es el centro de área (C-o-A), el cual retorna el centro del área bajo la curva, como se muestra en la figura 2.4. Otros métodos utilizados son: centro del máximo (C-o-M), media del máximo (M-o-M) y el mayor del máximos [13].

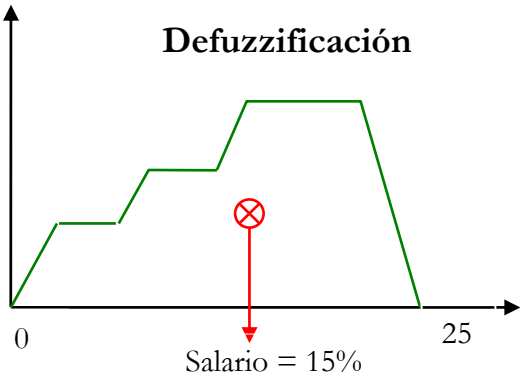

Figura 2.4. Proceso de defuzzificación.

2.2.3 Tipos de Sistemas Fuzzy

En la literatura existen varios modelos de sistemas fuzzy, en los cuales generalmente los antecedentes de las reglas están formados por conjuntos fuzzy (proposiciones lingüísticas), y la diferencia entre los modelos se da en el "consecuente" de las reglas. Alguno de los modelos más conocidos son el modelo de Mamdani y el modelo de Tagaki-Sugeno.

- **Modelo de Mamdani**

  El modelo de Mamdani utiliza conjuntos fuzzy en los "consecuentes" de las reglas fuzzy. En este modelo, la salida de la etapa de inferencia es

representada por un conjunto fuzzy, que es el resultado de la agregación de las salidas inferida por cada una de las reglas, la cual en la siguiente etapa genera una salida exacta utilizando uno de los métodos de defuzzificación ya mencionados.

La característica básica del modelo tipo Mamdani es el hecho que tanto los antecedentes como los consecuentes son mapeados con conjuntos fuzzy. Por ejemplo, una regla típica en un modelo Mamdani es de la siguiente forma:

Si Error es "*Grande*" y la Derivada de Error es "*Pequeña*" **entonces** Torque es "*Alto*".

En el caso en que cada una de las reglas de inferencia tenga más de una variable de entrada, es necesario aplicar una técnica de agregación de los conjuntos antecedentes, que en este caso generalmente es dada por la t-norma (min y producto). Además, en las aplicaciones prácticas se tiene N reglas en la etapa de inferencia, de las cuales son generados N conjuntos consecuentes, una por cada regla. Para obtener el conjunto final de salida de la etapa de inferencia, es hecha la composición de los N conjuntos consecuentes utilizando la s-norma (Max) [7- 10].

- **Modelo de Takagi-Sugeno**

  En el modelo de Takagi-Sugeno (TS), el consecuente de cada regla es representado en función de las variables de entrada; y la salida final de todas las reglas es determinada por la media ponderada de las salidas generadas por cada una de las reglas. En ese caso, los coeficientes de ponderación son definidos por los grados de activación de las respectivas reglas.

La fuzzificación de las entradas con la aplicación de los operadores fuzzy (operación de los antecedentes) es hecha de igual forma que en el modelo Mamdani, con la diferencia que la salida es una función lineal o constante. Una regla típica de un modelo Sugeno es de la siguiente forma:

Si $Error = x$ y la $Derivada\ de\ Error = y$ **entonces** Torque es $\tau_i = a.x + b.y + c$.

Donde, τ_i es el valor de salida de cada uno de las reglas. Además de eso, a partir del producto de la operación del antecedente de cada regla, se obtiene un peso ω_i, el cual determina el factor de influencia de cada regla en el resultado final. Por ejemplo, en la regla mostrada encima, con el tipo de operador AND, con $Error = x$ y $Derivada\ de\ Error = y$, el factor de influencia es:

$$\omega_i = AND(f(x), g(y)) \tag{2.3}$$

Donde, f y g son las funciones de pertenencia para las entradas error y derivada de error.

La salida final es determinada por la ecuación:

$$out = \frac{\sum_{j=1}^{n} w_i.\tau_i}{\sum_{j=1}^{n} w_i} \frac{n!}{r!(n-r)!} \tag{2.4}$$

2.3. Redes Neuronales

Las redes neuronales (NN) están constituidas por unidades básicas independientes designadas como neuronas (procesadores o nodos). Cada unidad posee ligaciones

para otras unidades, las cuales se comunican entre sí a través de sinapsis, formando una red de nodos, de ahí el nombre de redes neuronales.

Las primeras informaciones sobre redes neuronales surgieron en 1943, con el primer modelo lógico-matemático de una neurona biológica, que fue desarrollado por el neurofisiólogo Warren McCulloch, del Instituto Tecnológico de *Massachusetts*, y del matemático Walter Pitts, de la Universidad de Illinois. Desarrollaron un modelo matemático de neurona simulando el comportamiento de una célula nerviosa, la cual consistía en un modelo de resistores variables y amplificadores.

Las redes neuronales realizan el procesamiento de la información basado en la organización de los neuronas del cerebro, las cuales tiene la capacidad de aprender y tomar decisiones basadas en el aprendizaje. Las redes neuronales son interpretadas como una estructura neuronal de organismos inteligentes, capaz de almacenar conocimiento basado en el aprendizaje y la experiencia.

Una red neuronal es un procesador macizamente paralelo, distribuido, constituido por unidades de procesamiento simple, que tiene la capacidad de almacenamiento de conocimiento experimental y de tornar lo disponible para uso. Las redes neuronales son semejantes al cerebro por los siguientes aspectos [14]:

- El conocimiento es obtenido por la red neuronal a partir del ambiente, a través del proceso de aprendizaje.
- La fuerza de conexión entre cada neurona, conocida como peso sináptico, es usada para almacenar el conocimiento aprendido.

2.3.1 Estructura de la Neurona

La neurona es la unidad fundamental de procesamiento de información para a operación de una NN. Neuronas son elementos procesadores interconectados, trabajando en paralelo para desempeñar una determinada tarea. La figura 2.5 muestra la estructura de una neurona biológica, y en la figura 2.6 se presenta el modelo de una neurona artificial, en la cual se diseña la forma básica que constituye una NN. Los elementos básicos identificados en el modelo neuronal son: los pesos de conexión, el Net y la Función de activación [15].

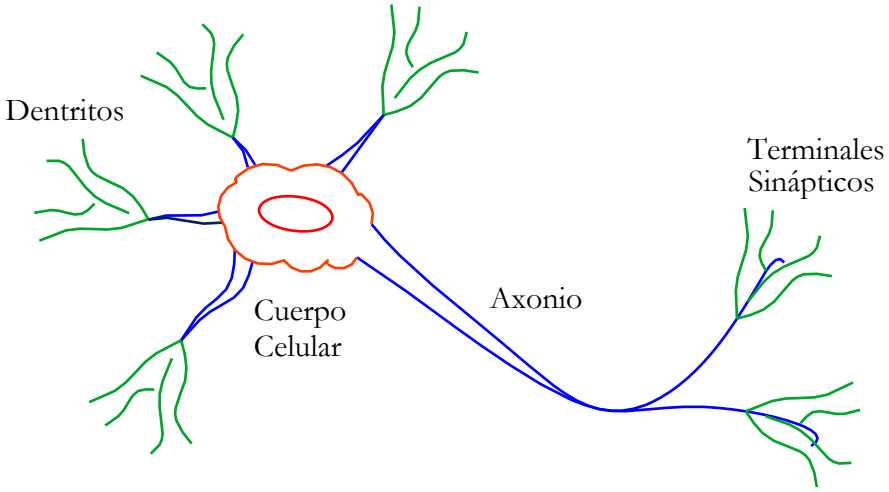

Figura 2.5. Esquema básico de una neurona biológica.

La neurona recibe múltiples señales de otras neuronas a través de sus dentritos, y cada uno de estas señales es multiplicado por el propio peso de la conexión. Estas señales son adicionados en el cuerpo celular o función sumatoria, y cuando esta señal compuesto alcanzar un valor umbral, una señal potencial es enviado por el axonio, el cual es la salida de la neurona [14].

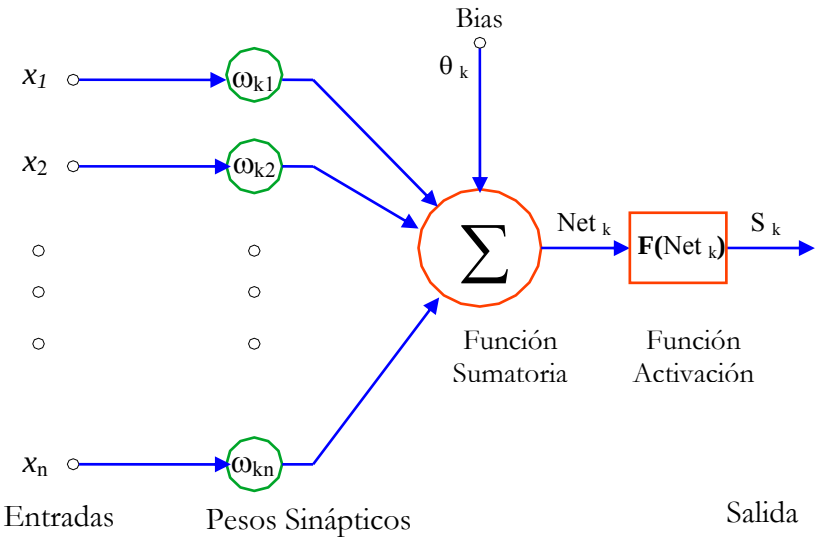

Figura 2.6. Modelo básico de una neurona artificial "perceptron".

Donde x son las entradas, ω_{kn} son los pesos o parámetros de la red, los cuales representan la memoria de la red, θ_k es conocido como el término polarizador (bias), Net_k representa la combinación lineal de los pesos, y S_k representa la salida de la neurona.

El modelo de la neurona en la figura 2.6, adicional a las entradas, tiene un término constante llamado "bias" θ_k, el cual tiene la función de aumentar o disminuir la entrada Net_k de la función de activación, modificando este positiva o negativamente.

2.3.2 Arquitectura de las Redes Neuronales

La arquitectura de las redes neurales artificiales puede ser formada por una o más capas de neuronas. Estas capas están formadas por neuronas en fila y ligados entre sí. La entrada de una capa intermediaria " p " es la salida de la capa anterior " $p-1$ " o la capa de entrada, y la salida de esta misma capa "

p " es una entrada de la capa siguiente " $p+1$ " o la capa de salida, como se presenta en la figura 2.7.

Las redes neuronales, basadas en su arquitectura, son clasificadas en dos grupos, las redes perceptron y las redes *multilayer perceptron* (MLP).

- **Perceptron.**

La red perceptron fue propuesto por Rosenblatt [1962], continuando el trabajo de McCulloch-Pitts. Esto se podría considerar como el primer modelo de redes neuronales, el cual es capaz de aprender solamente problemas linealmente separables. Su arquitectura consiste en una capa de entrada y una de salida, como se presenta en la figura 2.7.

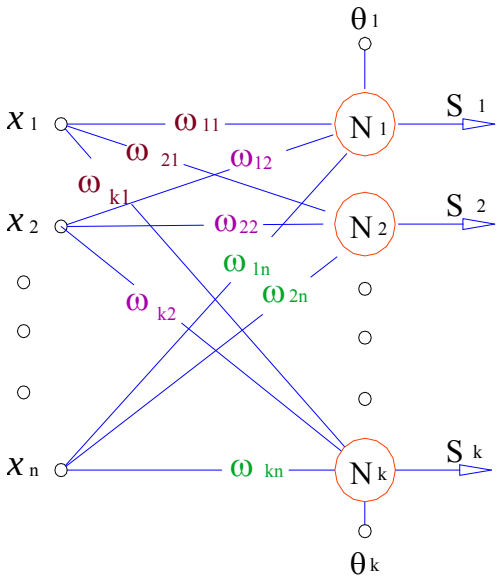

Figura 2.7. Red perceptron con k neuronas de salida.

Donde $X=[x_1\ x_2\ \ldots x_n]$ representa la capa de entrada, N_k es la $k-ésima$ neurona, ω_{jk} es el peso de la neurona N_k en relación a la entrada x_j, y S_k representa la $k-ésima$ salida.

Siguiendo la regla de propagación, el valor de Net de la $j-ésima$ neurona es definido por la ecuación

$$Net_j = \sum_{i=1}^{k} \omega_{ji}.x_i + \theta_j \tag{2.5}$$

Generalmente, en las redes perceptron la función de activación es la "función escalón", y su salida S_j es determinada por

$$S_j = \begin{cases} 1 & \text{Si} \quad Net_j > 0 \\ 0 & \text{Si} \quad Net_j \leq 0 \end{cases} \tag{2.6}$$

Durante el proceso de entrenamiento del perceptron, se busca encontrar un conjunto de pesos que determine una recta que separa las diferentes clases, de manera que la red pueda clasificar correctamente las entradas presentadas. El algoritmo de aprendizaje para obtener los pesos deseados es determinado de la manera descrita a seguir.

Para un padrón de entrada, se obtiene una salida de la red s_j, con su respectivo valor de salida deseado t_j. El error en la salida es definido por

$$e_j = t_j - s_j \tag{2.7}$$

Entonces, la variación del peso $\Delta\omega_{ji}$ es definida por

$$\Delta\omega_{ji} = \eta.x_i.e_j \tag{2.8}$$

Donde η es la tasa de aprendizaje.

Entre tanto, los pesos son actualizados después de cada entrada con la regla

$$\omega_{ji}^{t+1} = \omega_{ji}^{t} + \Delta\omega_{ji}^{t} = \omega_{ji}^{t} + \eta.x_i.e_j \tag{2.9}$$

El algoritmo de aprendizaje aplicado en una de las neuronas de la red perceptron es lo mismo para todos las demás neuronas.

El perceptron de una única neurona es limitado a trabajar en la clasificación de padrones con apenas dos clases. Mientras, el incremento de las neuronas en la capa de salida del perceptron permite clasificar padrones de más de dos clases, siempre que las clases fueran linealmente separables [17].

- **Red Multilayer Perceptron**

  La red multilayer perceptron (MLP) es considerada como una de las más importantes ANN. Este tipo de redes fueron aplicados con éxito en diversas áreas, en la solución de problemas complejos, desempeñando tareas de clasificación de padrones, reconocimiento de voz, reconocimiento de imágenes, y control. Una vez que las redes perceptron solo representan funciones linealmente separables, uno de los motivos del resurgimiento de la redes neuronales fue debido al desarrollo del algoritmo *Backpropagation*.

  La red MLP son una generalización de la red perceptron y están constituidas de tres partes: la primera parte es denominada capa de entrada (*input layer*), la cual está constituida de un conjunto de unidades sensoriales; la segunda parte está constituida de una o más capas escondidas (*hidden layers*); y la tercera parte está constituida de una capa denominada capa de salida (*ouput layer*). Con exepción de la capa de entrada, todas las otras capas están constituidas por neuronas, las cuales implican en un esfuerzo computacional. En la figura 2.8, se presenta la arquitectura de una red neuronal MLP con una capa de entrada constituida de n unidades sensoriales, 2 capas escondidas constituida de i y j neuronas respectivamente, y una capa de salida constituida de k neuronas, formando la arquitectura MLP (n-i-j-k).

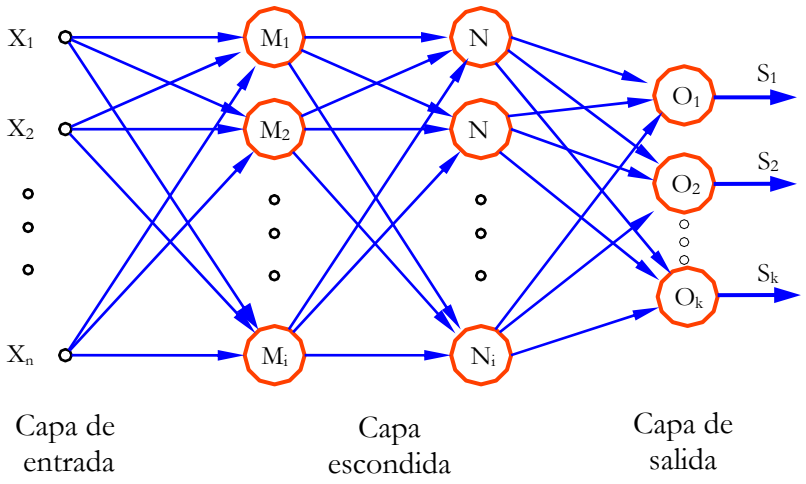

Figura 2.8. Arquitectura de una red neuronal MLP (n-i-j-k).

Algunas características básicas de las redes MLP son las siguientes:

La función de activación de cada una de las neuronas de la red es una función no lineal. Una característica importante de esta función es que precisa ser suave y diferenciable. Generalmente, la función de activación que satisface estos requerimientos es una función no lineal sigmoides, las más comunes utilizadas son las funciones *Logsig* y *Tansig*.

Las redes MLP son constituidas de una o más capas escondidas, que contienen neuronas que no forman parte de la capa de entrada o salida de la red. Estas neuronas escondidas permiten que la red pueda aprender tareas más complejas progresivamente, por el proceso de aprendizaje.

Una red MLP es una red *feedforward*, donde el flujo de datos es siempre en una única dirección. Así, la salidas de las neuronas de una capa cualquier se conectan únicamente a las entradas de las neuronas de la capa siguiente. Por tanto, la señal de entrada se propaga a través de la red en un solo sentido, esto es, no existe realimentación.

Las redes MLP presentan un alto grado de conectividad o pueden ser completamente conectados, el que es determinado por las sinapsis de la red, caso en que una neurona de una capa cualquier es conectada a todas las neuronas de la capa siguiente. Además de eso, una red MLP puede ser parcialmente conectada, en el caso en que alguna de las sinapsis este faltando. En la práctica, la falta de algunas sinapsis dentro de la red es representada haciendo el peso de las sinapsis igual a cero. Entre tanto, en este estudio se consideran redes MLP completamente conectadas.

La combinación de estas características, junto con la capacidad de aprendizaje, producto de la experiencia a través del proceso de entrenamiento de las redes MLP, determina el esfuerzo computacional.

El número de variables o nodos de la capa de entrada es determinado por la complejidad del problema y la dimensionalidad del espacio de observación, de donde son generadas las señales de entrada. El número de neuronas en la capa de salida es determinado por la dimensionalidad de la respuesta deseada. Además, la configuración y modelaje de una red MLP tiene como objetivo determinación de los siguientes aspectos:

a) La determinación del número de capas escondidas.

b) La determinación del número de neuronas en cada una de las capas escondida.

c) La determinación del valor de los pesos de conexión y las funciones de activación.

Existe una variedad de metodologías para escoger estos aspectos, pero normalmente estas son hechas de forma empírica. En la figura 2.9, se presentan estos aspectos.

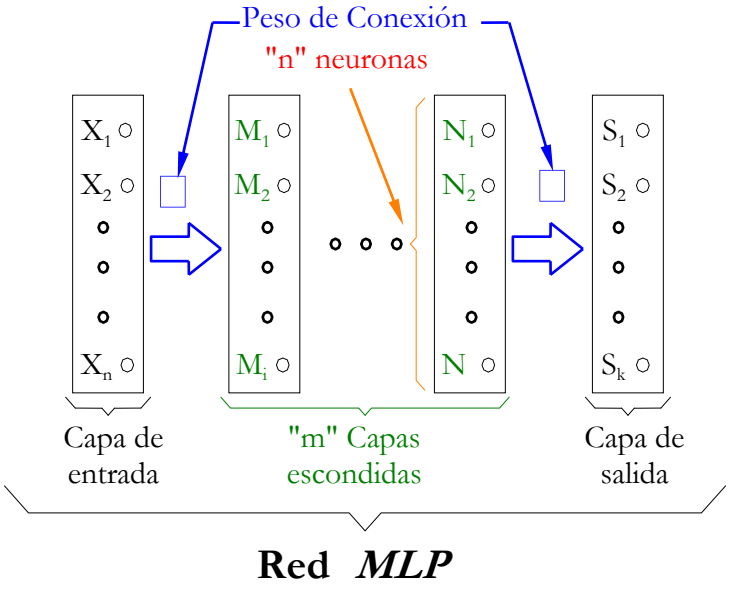

Figura 2.9. Configuración de la red MLP.

El tamaño del modelo de ANN es determinado por los aspectos 1 y 2 encima, mas no hay reglas para la determinación de tales especificaciones, siendo generalmente determinados basado en la experiencia del especialista. El número de capas escondidas en una ANN influye en la relación entrada-salida. Esto es, una ANN con un mayor número de capas escondidas tiene mejor capacidad de extraer las características de algún proceso aleatorio desconocido, sobre el cual la red esta tentando adquirir conocimiento.

El incremento excesivo del número de capas escondidas y neuronas dentro de la ANN producen una red muy grande que no responderá correctamente a padrones nunca vistos. Esto es, debido a la red extraer muchas características, ellas pueden intentar, de forma inapropiada, modelar

también el ruido. Y un número muy pequeño de capas escondidas y neuronas produce una ANN muy pequeña, que no es capaz de almacenar todos los padrones necesarios, al punto de no modelar fielmente los datos. Además, una red muy grande demanda un mayor esfuerzo computacional, entonces se debe tener un compromiso entre convergencia y generalización.

La Convergencia es la capacidad de la red de aprender todos los padrones del conjunto de entrenamiento, la cual está estrictamente relacionada con el tamaño de la red. Entre tanto, la Generalización es la capacidad de un modelo de aprendizaje responder correctamente a padrones nuevos que le son presentados. Un modelo con buena generalización es aquel que responde correctamente a padrones contenidos en la base de datos, y también a padrones nuevos contenidos en la base de teste. La capacidad de generalización es el principal objetivo en el proceso de aprendizaje.

Otro aspecto está relacionado con la determinación de la función de activación de cada una de las neuronas, y del algoritmo de entrenamiento utilizado para obtener los pesos de conexión deseados que resuelva el problema o tarea. Uno de los algoritmos de entrenamiento que ha sido aplicado en la solución de diversos problemas complejos es el algoritmo conocido en la literatura como *backpropagation*, el cual es basado en la retro-propagación del error.

La idea básica del algoritmo *backpropagation* fue descrito por Werbos en su tesis de doctorado en la *Universidad de Harvard* en 1974. No en tanto, o algoritmo de aprendizaje *backpropagation* surgió después de 1985, a través da

publicación del libro *Parallel Distributed Processing* de Rumelhart y McClelland en 1986.

Además del algoritmo *backpropagation*, existen varios algoritmos de aprendizaje, o también llamados reglas de aprendizaje, algunas de las cuales son el algoritmo de aprendizaje Levenberg Marquardt (utiliza una aproximación del método de Newton), el algoritmo del gradiente descendente (usado por el algoritmo *backpropagation*), algoritmos competitivos que se caracterizan por las conexiones laterales de las neuronas con sus vecinos, estableciendo una competencia entre las neuronas (usado por las redes Hopfiel y Kohonen).

La obtención de los pesos deseados de la ANN es obtenida mediante la actualización de pesos en el proceso de entrenamiento, el que pude ser realizado de dos maneras, Batch o Incremental.

En el proceso de entrenamiento Batch o por ciclos, la actualización de los pesos sucede solamente después de la presentación de todos los padrones del conjunto de entrenamiento. Así, todos los padrones son evaluados con la misma configuración de pesos. Entre tanto, en el entrenamiento Incremental (o por Padrón), la actualización de los pesos es realizado luego de la presentación de cada nuevo padrón. Esto lleva a que la red aprenda mejor el último padrón presentado, y por eso es recomendable presentar los datos de forma aleatoria.

La eficiencia de los dos métodos de entrenamiento depende del problema en estudio. El proceso de entrenamiento (aprendizaje) es mantenido hasta que los pesos se estabilicen y el error cuadrático medio sea suficientemente

pequeño, o sea, menor que un error admisible, de modo que el objetivo deseado sea alcanzado. Así, se debe encontrar un punto óptimo de parada con error mínimo y máxima capacidad de generalización.

Después del proceso de entrenamiento, algunas veces la ANN se puede especializar demasiadamente en relación a los padrones del conjunto de entrenamiento, generando un problema de aprendizaje conocido como súper-aprendizaje o *over-fitting*. Generalmente este problema puede ser evitado utilizando el método *Early Stopping*. Este método divide el conjunto de padrones en un nuevo conjunto de entrenamiento y validación, después cada barredura del conjunto de entrenamiento, la red es evaluada con el conjunto de validación. El algoritmo para, cuando el desempeño con el teste de validación deja de mejorar.

El procedimiento de entrenamiento utilizado es el entrenamiento supervisado, la red es entrenada con un conjunto de datos de entrenamiento que proporciona a la red los valores de entrada y sus respectivos valores de salida deseada, como se presenta en la figura 2.10.

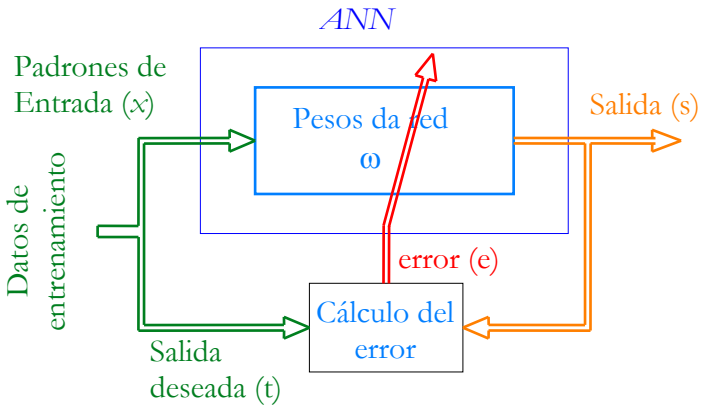

Figura 2.10. Entrenamiento Supervisado.

2.3.3 Algoritmo *Backpropagation*

Backpropagation Es el algoritmo de entrenamiento más utilizado por las *ANN*. Este algoritmo es de tipo supervisado, y consiste en el ajuste de los pesos de la red basado en la minimización del error medio cuadrático mediante el algoritmo de gradiente descendente. Por lo tanto, una parte esencial del algoritmo es el cálculo de las derivadas parciales del error en relación a los parámetros de la *ANN*. El desarrollo de este algoritmo permitió resolver el problema de la actualización de los pesos en las capas escondidas, y con este el resurgimiento de las redes neuronales.

Este tipo de algoritmo está compuesto de dos etapas *propagación* y *retro-propagación*, como son detallados a continuación.

- **Propagación (Forward)**

  En esta etapa, un conjunto de padrones son aplicados en la capa de entrada de la ANN, y la respuesta de esta capa es propagada como entrada en la siguiente capa. Este efecto es propagado a través de la red, capa a capa, hasta que finalmente es obtenido un conjunto de salida como la respuesta actual de la red.

  Durante esta etapa, los pesos sinápticos da red son fijos. Observe en la figura 2.11 (b) que la señal fluye a través de la red de izquierda hacia derecha. Estas señales que se propagan en esta dirección son denominadas Señales Funcionales [14].

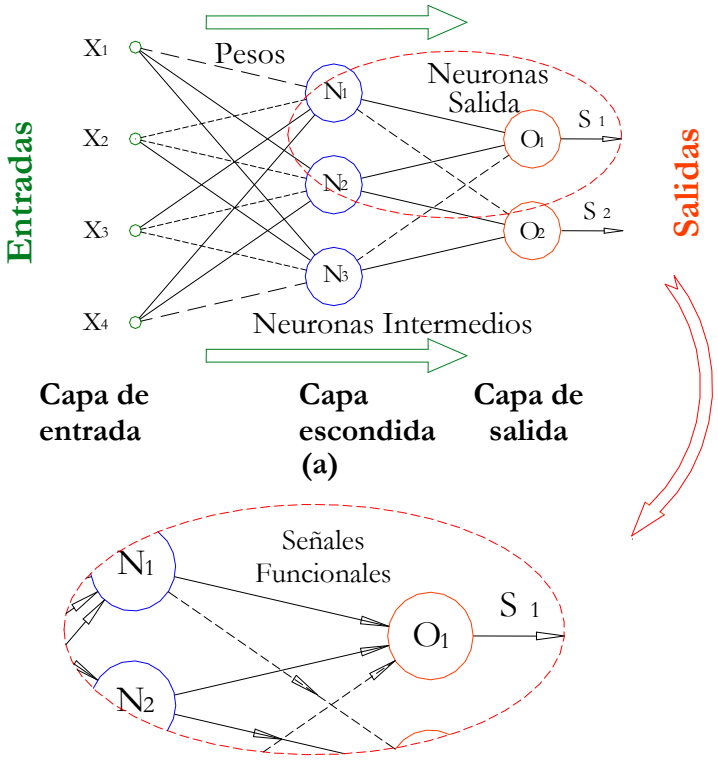

Figura 2.11. (a) Fase de propagación (b) Señales Funcionales.

- **Retro-Propagación (*Backward*)**

  En la etapa de Retro-propagación, desde la capa de salida hasta la capa de entrada son ajustados todos los pesos sinápticos en concordancia con una correlación del error. Específicamente, la respuesta actual de la red es substraída de una respuesta deseada (target), generando la señal de error. Esta señal de error nuevamente es retro-propagado a través de la red en sentido contrario a la señal funcional, de ahí el nombre de "error back-propagation".

  Los pesos de la ANN son ajustados para hacer que la respuesta actual de la red se aproxime a la respuesta deseada. El error originado en una neurona de la capa de salida de la red es propagado de vuelta a través de la red, capa

a capa, es denominado como señal de error. En la figura 2.12, se presenta la etapa de retro-propagación y las señales de error [14].

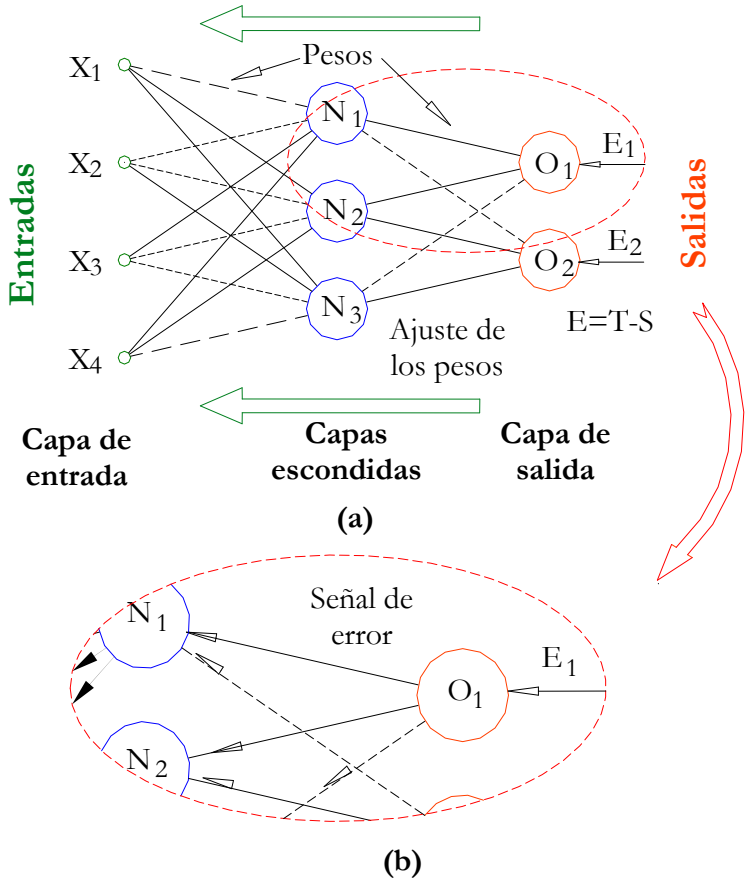

Figura 2.12. (a) Etapa de Retro-propagación (b) Señal de error.

- **Algoritmo de Aprendizaje**

  El algoritmo de aprendizaje *backpropagation* es basado en la propagación del error, que inicia en la capa de salida y va para las capas anteriores de la ANN, de acurdo con el grado de participación que cada neurona tuvo en el nivel posterior. Antes de pasar a la descripción del algoritmo, es conveniente hacer algunas consideraciones relacionado a las características básicas da ANN:

- Topología da red – Múltiples capas (*MLP*).

- Función de activación – Función no-linear, diferenciable en todo el dominio.
- Tipo de aprendizaje – supervisado.
- Regla de propagación – $Net_j = \sum_{i=1}^{k} \omega_{ji}.x_i + \theta_j$.
- Valores de entrada – salida – binarios o continuos.

El proceso de aprendizaje del algoritmo *backpropagation* tiene como objetivo al actualización de los pesos sinápticos de la red, el que es hecho por medio de la minimización del error cuadrático medio por el método del Gradiente Descendente. Basado en este método, el factor de actualización del peso ω_{ij} relativo a la entrada i de la neurona j es dado por

$$\Delta\omega_{ij} = -\eta.\frac{\partial E}{\partial \omega_{ij}} \tag{2.10}$$

Donde, $\Delta\omega_{ij}$ es la variación del peso de la neurona j de la conexión i, η es el factor de aprendizaje, y E es la sumatoria del error cuadrático medio total, el cual definido por la ecuación (2.11), como la suma del error cuadrático medio de todo os padrones [19].

$$E = \frac{1}{2}.\sum_{p}\sum_{i=1}^{k}(t_i^p - s_i^p)^2 \tag{2.11}$$

Donde, p es el numero de padrones, k es el numero de neuronas en la salida, t_i^p y s_i^p son el valor deseado y la salida generada por la red para la neurona i cuando es presentado el padrón p . Además, se puede asumir sin

pérdida de generalidad que la minimización de cada padrón va a llevar a la minimización del error total, de modo que el error pasa a ser definido por

$$E = \frac{1}{2}.\sum_{i=1}^{k}(t_i - s_i)^2 \tag{2.12}$$

- **Calculo de** $\Delta\omega_{ij}$

Utilizando la regla de la cadena en la ecuación (2.10), se puede expresar como

$$\Delta\omega_{ij} = -\eta.\frac{\partial E}{\partial \omega_{ij}} = -\eta.\frac{\partial E}{\partial net_j}.\frac{\partial net_j}{\partial \omega_{ij}} \tag{2.13}$$

donde, $net_j = \sum s_i.\omega_{ij} + \theta_j$ y así

$$\frac{\partial net_j}{\partial \omega_{ij}} = S_i \tag{2.14}$$

Donde, s_i es el valor de entrada recibido por la conexión i de la neurona j

Entre tanto, el otro término de la ecuación (2.13) es el valor calculado del error de la neurona j. Este término depende de la capa en la cual se encuentra la neurona, y es representado por

$$e_j = -\frac{\partial E}{\partial net_j} \tag{2.15}$$

Así, de las ecuaciones de encima, se puede establecer que

$$\Delta\omega_{ij} = \eta.s_i.e_j \tag{2.16}$$

Entre tanto, el valor del peso actualizado es definido por

$$\omega_{ij}^{t+1} = \omega_{ij}^{t} + \Delta\omega_{ij}^{t+1} = \omega_{ij}^{t} + \eta.s_i.e_j \tag{2.17}$$

Donde, ω_{ij}^{t+1} es el valor de peso actualizado, ω_{ij}^{t} es el valor de peso en la iteración anterior en el proceso de aprendizaje, y $\Delta\omega_{ij}^{t}$ es la variación del peso.

- **Calculo del Error (e_j)**

Utilizando la regla de la cadena en la ecuación (2.15), se puede establecer que:

$$e_j = -\frac{\partial E}{\partial net_j} = -\frac{\partial E}{\partial s_j}.\frac{\partial s_j}{\partial net_j} \tag{2.18}$$

Donde, $s_j = \varphi(net_j)$ entonces:

$$\varphi'(net_j) = \frac{\partial s_j}{\partial net_j} \tag{2.19}$$

Entre tanto, el valor del término $\frac{\partial E}{\partial s_j}$ depende de la capa a la cual pertenece la neurona j.

- **En la Capa de Salida**

La neurona j pertenece a la capa de salida, como es presentado en la figura 2.13. El valor de E es calculado por la ecuación (2.12), de modo que el termino $\frac{\partial E}{\partial s_j}$ es calculado por

$$\frac{\partial E}{\partial s_j} = -(t_j - s_j) \tag{2.20}$$

Así, el valor do de e_j para la neurona j en la capa de salida es

$$e_j = (t_j - s_j).\varphi'(net_j) \tag{2.21}$$

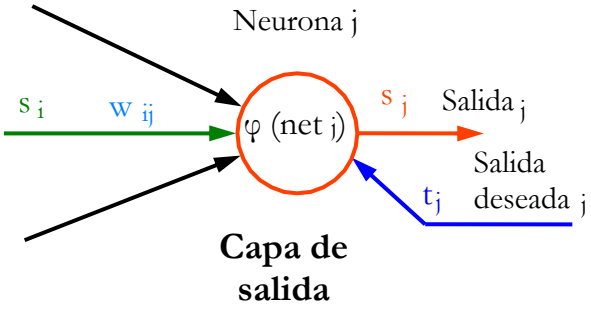

Figura 2.13. Cálculo del error e_j en la capa de salida.

- **En la Capa Escondida**

El caso en que la neurona j pertenece a la capa escondida es presentado en la figura 2.14.

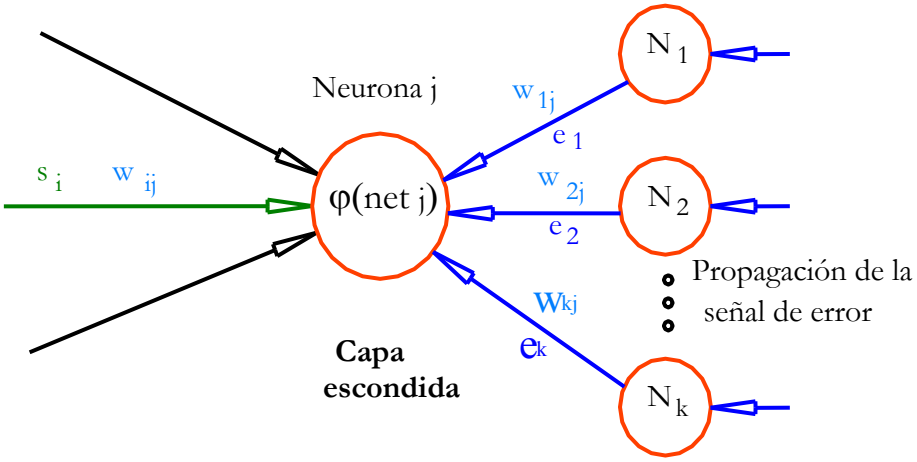

Figura 2.14. Calculo del error e_j en la capa escondida.

El valor de E para esta neurona j es calculado por la ecuación (2.12), tomando en consideración la propagación del error de la capa siguiente. El termino $\frac{\partial E}{\partial s_j}$ es calculado por $E = \frac{1}{2}.(t_k - s_k)^2$.

$$\frac{\partial E}{\partial s_j} = -\sum_k (t_k - s_k).\frac{\partial s_k}{\partial s_j} \qquad (2.22)$$

Usando la regla de la cadena en la ecuación (2.22), se puede establecer que:

$$\frac{\partial E}{\partial s_j} = -\sum_k (t_k - s_k).\frac{\partial s_k}{\partial net_k}.\frac{\partial net_k}{\partial s_j} = -\sum_k (t_k - s_k).\varphi'(net_k).\omega_{jk}$$

$$\frac{\partial E}{\partial s_j} = -\sum_k (e_k.\omega_{jk}) \tag{2.23}$$

El valor de e_j para la neurona j en la capa escondida es dado por

$$e_j = \sum_k (e_k.\omega_{jk}).\varphi'(net_j) \tag{2.24}$$

2.3.4 Parámetros de Aprendizaje

El algoritmo backpropagation realiza el aprendizaje de la red mediante un proceso iterativo de ajuste de los pesos, siguiendo una trayectoria de movimiento sobre la superficie de error en el espacio de pesos sinápticos, la cual, a cada instante, sigue la dirección de minimización del error, pudiendo llegar al mínimo global. Algunos parámetros que influencian el desempeño del proceso de aprendizaje son Factor de Aprendizaje y Término de Momentum.

- **Factor de Aprendizaje (η)**

  El factor de aprendizaje (η) es un parámetro constante en el intervalo [0,1] que interfiere en la convergencia del proceso de aprendizaje. La influencia de este parámetro está relacionada al cambio en los pesos sinápticos. Así, un factor de aprendizaje muy pequeña implica en una trayectoria suave y pequeños cambios de los pesos a cada iteración. Por otro lado, se requiere un tiempo de entrenamiento muy largo, y dependiendo de la inicialización de los pesos es posible caer en el problema de mínimo local, pues la *ANN* en ese caso no consigue calcular un cambio en los pesos que haga a la red

salir del mínimo local. En la figura 2.15 se presenta el efecto producido con una η pequeña.

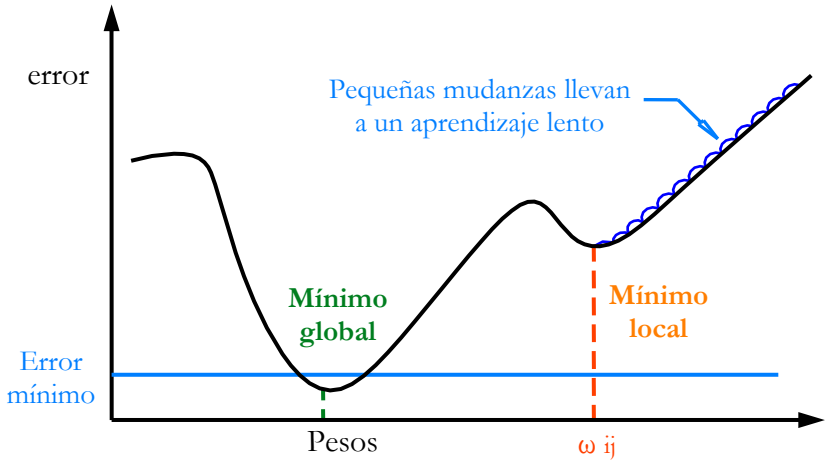

Figura 2.15. Factor de aprendizaje pequeño, con problema de mínimo local.

Entre tanto, si el factor de aprendizaje fuera muy grande (cerca de 1), ocurre un mayor cambio en los pesos, aumentando la velocidad del aprendizaje, lo que podría llevar a oscilaciones en torno del mínimo global. Así, que el factor de aprendizaje no debe ser muy pequeña, ni muy grande. En la figura 2.16, se muestra el efecto causado por un factor de aprendizaje muy grande [19].

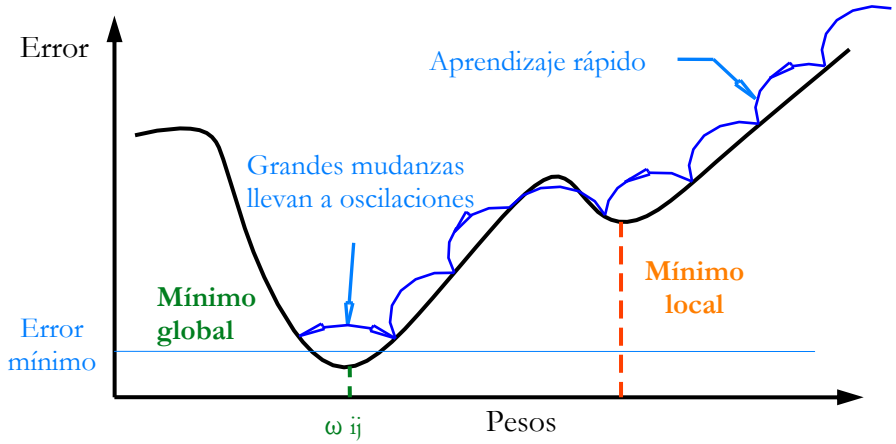

Figura 2.16. Factor de aprendizaje grande, con problema de oscilaciones.

Lo ideal sería utilizar el mayor factor de aprendizaje posible que no llevase a una oscilación, obteniendo un aprendizaje más rápido. De acuerdo con BEALE [20], el factor de aprendizaje (η) debería ser disminuido progresivamente, de modo que el gradiente descendente por el cual son actualizados los pesos pueda encontrar una mejor solución. De esta forma, se utiliza un factor de aprendizaje (η) adaptativo, el cual inicia con un valor grande, y va decrece exponencialmente a medida que el entrenamiento converge.

- **Termino de Momentum** (α)

  Una manera de aumentar el factor de aprendizaje sin llevar a oscilaciones durante la ejecución del algoritmo *backpropagation* está en la modificación de la ecuación (2.16) para incluir el termino momentum, el cual lleva en consideración el efecto de los cambios anteriores de los pesos en la dirección del movimiento actual de los pesos. El cambio de los pesos tomando en consideración el efecto del término de momentum es determinada por

$$\Delta\omega_{ij}^{t+1} = \eta.s_i.e_j + \alpha.\Delta\omega_{ij}^{t} \tag{2.25}$$

  Donde, $\Delta\omega_{ij}^{t+1}$ y $\Delta\omega_{ij}^{t}$ son la variación del peso de la neurona j en relación a la conexión i en el instante $t+1$ y t respectivamente, η es el factor de aprendizaje, y α es el termino de momentum. En la figura 2.17, se presenta el efecto del término de momentum en el aprendizaje de la red [14], [19].

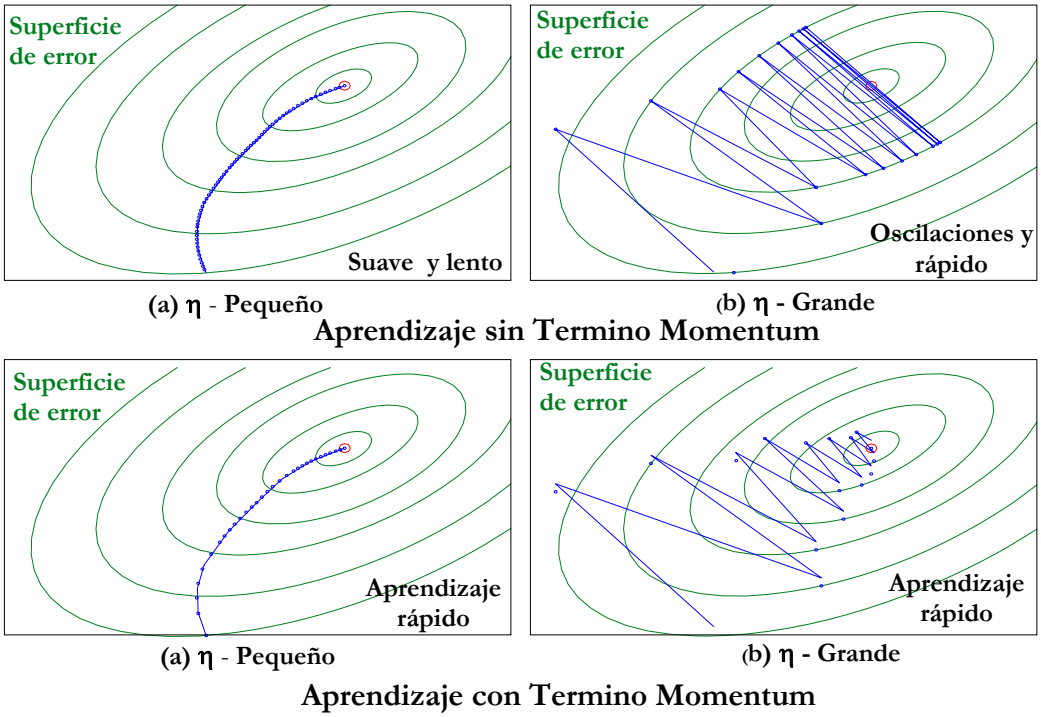

Figura 2.17. Aprendizaje sin y con término de Momentum.

2.4. Sistemas Híbridos

2.4.1 Sistemas Híbridos

Los sistemas híbridos son la sinergia obtenida por la combinación de dos o más técnicas de inteligencia. El foco de estos sistemas está en obtener un sistema más poderoso y con menos deficiencias. Dependiendo de la forma básica de construcción, una técnica puede ser aplicada para mejorar las deficiencias del otro en un mayor o menor grado. Algunas de estas formas son detalladas a continuación.

- **Sistema Híbrido Secuencial**

  En un sistema híbrido secuencial, la salida de un subsistema con "*Técnica 1*" actúa como entrada de otro subsistema con "*Técnica 2*", como es mostrado en la figura 2.18.

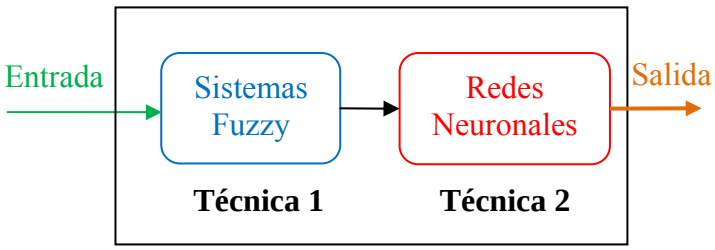

Figura 2.18. Sistema Híbrido Secuencial.

En la figura anterior, se muestra un sistema híbrido secuencial constituido de dos técnicas o subsistemas: la primera técnica es un sistema fuzzy que trabaja como un pre-procesador fuzzy de los dados, y la segunda técnica es una *ANN*. En la literatura, frecuentemente este tipo no es considerado como sistema hibrido, por ser la forma más simple de hibridización.

- **Sistema Híbrido Auxiliar**

  Un sistema híbrido auxiliar, es constituido de dos subsistemas un principal y otro auxiliar. El subsistema principal con "*Técnica 1*" llama al subsistema auxiliar con "*Técnica 2*" para realizar una determinada tarea.

  En el ejemplo de la figura 2.19, el subsistema principal basado en una *ANN*, invoca un subsistema auxiliar basado en un algoritmo genético para la optimización de sus pesos. Este tipo de sistema tienen un mayor grado de hibridización que el sistema hibrido secuencial.

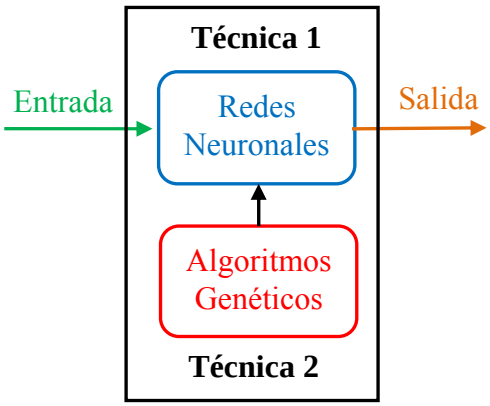

Figura 2.19. Sistema Hibrido Auxiliar.

- **Sistema Híbrido Incorporado**

En la práctica, los sistemas híbridos incorporados son los más utilizados, pues presentan el mayor grado de hibridización, al punto que no es posible la separación entre los dos subsistemas. En estos sistemas, el grado de hibridización es tan elevado que se puede decir que el primero subsistema contiene el segundo, o viceversa.

En la figura 2.20, se presenta un ejemplo de un sistema híbrido incorporado, el cual es constituido de un *Sistema Fuzzy* y una *ANN*. Estos tipos de sistemas híbridos son conocidos como Sistemas neuro-fuzzy, en los cuales un sistema de inferencia fuzzy es implementado conforme la estructura paralela de una *ANN*.

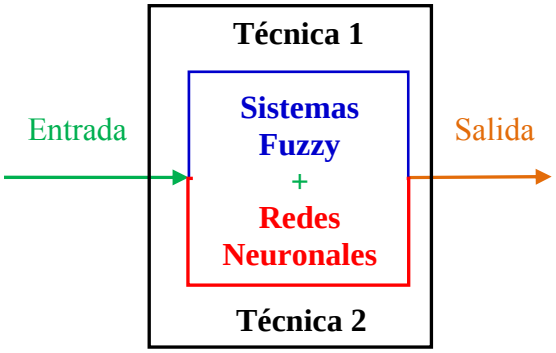

Figura 2.20. Sistema Hibrido Incorporado.

En esta Tesis, se utiliza el sistema híbrido incorporado neuro-fuzzy, pues este sistema ha recibido una gran atención de los investigadores en aplicaciones en la área de control, por la necesidad de controladores cada vez más eficientes.

2.4.2 Sistemas Neuro-fuzzy

Un *sistema neuro-fuzzy* (*SNF*), es un tipo de sistema híbrido incorporado constituido por la combinación de dos técnicas de modelamiento muy conocidas como las *ANN* y la *FL*. En la actualidad, los *SNF* se están tornando de gran interés, pues traen los beneficios tanto de *ANN* cuanto de sistemas de *FL*, eliminando así las desventajas individuales al combinar las características comunes. Además de eso, diferentes arquitecturas de *SNF* vienen siendo investigados en diversas áreas de aplicación, especialmente en el control de procesos [23].

La idea básica de un *SNF* es la construcción de un *sistema de inferencia fuzzy* (*FIS*), en una estructura paralela distribuida de tal forma que los algoritmos de aprendizaje de las redes neuronales puedan ser aprovechados en estos sistemas híbridos para ajustar los parámetros del *FIS*. La figura 2.21 presenta la estructura de un *SNF* que es dividido en 5 capas.

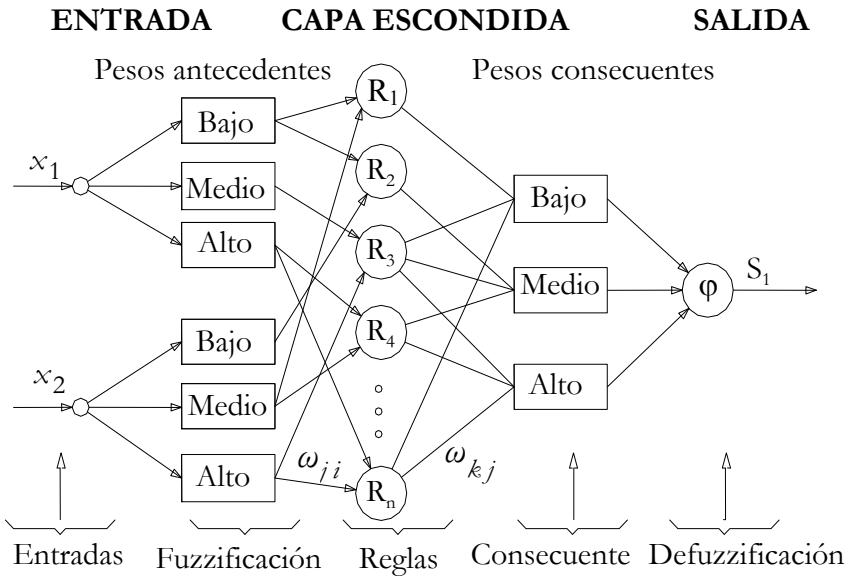

Figura 2.21. Arquitectura básica de un sistema neuro-fuzzy.

La capa de entrada representa las variables de entrada, las cuales son normalizadas y escalonadas dentro del intervalo numérico [0,1] o [-1,1]. La segunda capa es la etapa de fuzzificación. En esta etapa, os intervalos de cada variable de entrada son divididos en diversos niveles (Bajo, Medio y Alto), los cuales indican los pesos de la red para cada entrada. la tercera capa es definida por las reglas del *FIS*, la capa 4 es determinada por los consecuentes de las reglas, y la capa 5 o capa de salida es la etapa de defuzzificación, donde se calcula el valor numérico de salida.

- **Características de Sistema Neuro-fuzzy**

  Los *SNF* permiten la extracción de conocimiento basado en la forma de reglas de inferencia fuzzy, mediante la integración del conocimiento explícito generado por la experiencia del especialista y del conocimiento implícito obtenido a partir de un conjunto de datos. Así, estos sistemas asocian la capacidad de aprendizaje y de tolerancia a fallas de las *ANN*, con la interpretabilidad de los *FIS*'s.

  Debido a su naturaleza dual, estos sistemas heredan las características de sus "genitores". En este sentido, se dividirán las características en dos categorías principales, como se muestra en la figura 2.22.

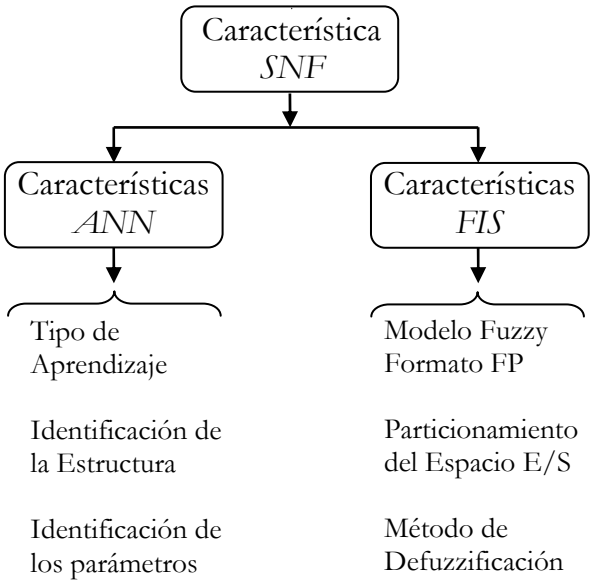

Figura 2.22. Características del sistema neuro-fuzzy.

Características Fuzzy del *SNF*

Las características fuzzy del *SNF* son agrupadas en 4 sub-clases, clasificadas en relación a los parámetros de la estructura fuzzy, de modo que la combinación de estas características determina el modelo del *SNF*. En la figura 2.23, se presenta esta clasificación.

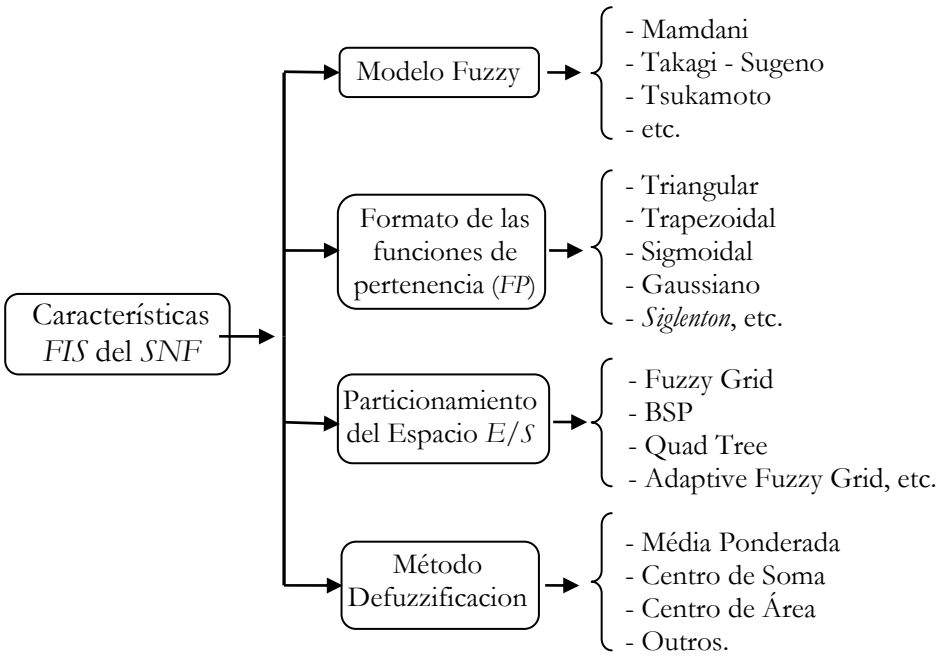

Figura 2.23. Características fuzzy del sistema neuro-fuzzy.

El modelo fuzzy implementado en el *SNF* determina el formato de las reglas fuzzy (modelo de inferencia), el cual forma la parte fundamental de la estructura de conocimiento en un sistema de inferencia fuzzy. Entre tanto, los formatos de las funciones de pertenencia influencian en los grados de pertenencia asociados a cada variable, siendo el formato triangular y o trapezoidal os más usados por la ventaja de ser computacionalmente simple. Así, las funciones de pertenencia "*singlenton*" generalmente son utilizadas en los consecuentes de los sistemas híbridos, esto debido a la ventaja de simplificar el proceso de defuzzificación del sistema fuzzy.

El particionamiento del espacio de las variables de entrada y salida (*E*/*S*) define internamente regiones fuzzy de espacio, los cuales son relacionados a través de las reglas fuzzy. Así, el particionamiento del espacio de salida generalmente es más simple pues está asociado a los consecuentes de las reglas.

Después de hacer las evaluaciones del conjunto de reglas fuzzy, el valor real de la salida del *SNF* es determinado por el método de defuzzificación escogido.

Características Neuronales del SNF

Las características neuronales del *SNF* están relacionadas con la capacidad de aprendizaje do sistema híbrido, estando divididas en 3 sub-clases como representados en la figura 2.24.

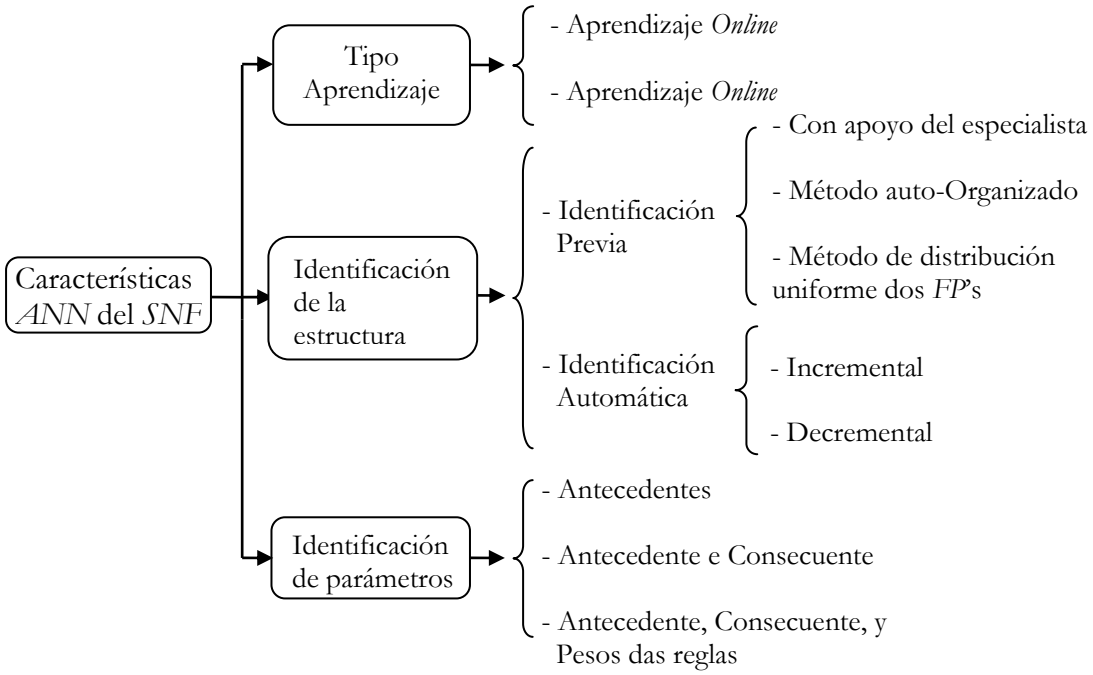

Figura 2.24. Características RNA del sistema neuro-fuzzy.

El tipo de aprendizaje, en relación a la forma de presentación de los padrones, se divide en aprendizaje *Off-line* y aprendizaje *On-line*. En el aprendizaje *On-line*, la actualización de los parámetros de la estructura es realizada para cada padrón presentado. Por otro lado, en el aprendizaje *Off-line* la actualización de los parámetros es realizado después de la presentación de todos los datos del conjunto de entrenamiento.

La identificación de la estructura de un *SNF* está relacionada con la determinación del número adecuado de reglas fuzzy, y de un particionamiento satisfactorio de las E/S. Esta característica se divide en dos categorías: *Identificación Previa* y *Identificación Automática*. En la *Identificación Previa*, el conocimiento de la estructura del sistema es realizado *a priori* con apoyo de algún especialista o método auto-organizado, antes del inicio de la actualización de los parámetros. En la *Identificación Automática*, el

conocimiento es obtenido de forma incremental o decremental, de modo que el sistema no precisa de conocimiento previo.

Las características del *SNF* en relación a la identificación de los parámetros son basadas en el método utilizado para el ajuste de los pesos fuzzy que definen los perfiles de las *FP*'s de los antecedentes y consecuentes de las reglas fuzzy. Generalmente, la mayoría de los SNF presenta identificación de los antecedentes y consecuentes, con peso fijo de cada regla y de valor unitario. Un ejemplo de este tipo de SNF son os modelos ANFIS (*Adaptative Network Based Fuzzy Inference System*). Entre tanto, algunos *SNF* solo presentan ajuste en los antecedentes de las reglas fuzzy; un ejemplo típico es el modelo NEFCLASS (*Neuro Fuzzy Classification*). Además, existen sistemas que presentan ajuste en el antecedente, consecuente, y en los pesos de cada una de las reglas fuzzy.

2.4.3 Modelos de Sistemas Neuro-fuzzy

Básicamente, los modelos de *SNF* son determinados por el diseño las características de sus "genitores". En esta sección se describen brevemente algunos de los SNF más conocidos: *ANFIS*, *FSOM*, *NEFCLASS*, *NEFCON* y *NFHQ*. Con esto se espera mejorar la comprensión de un *SNF* cualquiera.

- ***Adaptive Network Based Fuzzy Inference System*** **(ANFIS)**

  El sistema neuro-fuzzy ANFIS fue creado por Roger Jang. Esta arquitectura fue usada satisfactoriamente en aplicaciones de previsión y aproximación de funciones. Además el autor propone algunas variaciones

de este modelo dependiendo de las aplicaciones, así aumentando su popularidad al punto de ser incorporado dentro del *software* MATLAB©. En la figura 2.25 se muestra la arquitectura *ANFIS*.

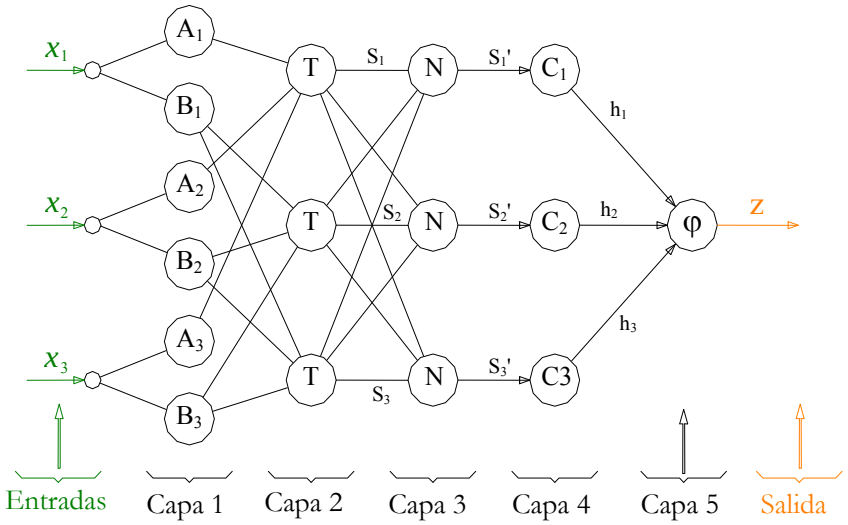

Figura 2.25. Arquitectura ANFIS

En la figura de encima, los nodos que componen una misma capa tienen funciones semejantes, las cuales son detalladas a continuación.

- *Capa 1*: La salida de esta capa son los grados de pertenencia de las entradas, basado en la premisa de cada regla. En este caso, cada entrada apenas tiene dos funciones de pertenencia (*Ai* = *Alto* y *Bi* = *Bajo*), pudiendo ser este número mayor.
- *Capa 2*: En esta capa se calcula el grado de pertenencia al cual es sometido el consecuente de cada regla. Cada nodo o neurona de esta capa realiza la operación *t_norm* y corresponde a una regla:

$$\begin{aligned} S_1 &= A_1(x_1) * A_2(x_2) * A_3(x_3) \\ S_2 &= B_1(x_1) * B_2(x_2) * A_3(x_3) \\ S_3 &= B_1(x_1) * B_2(x_2) * B_3(x_3) \end{aligned} \tag{2.26}$$

donde " * " representa el operador *t -norm*.

- *Capa 3:* Esta capa realiza la normalización de los grados de activación de las reglas. Cada nodo de esta capa realiza la función normalización, la cual es utilizada como un pre-procesamiento para la defuzzificación.

$$\begin{aligned} S_1' &= S_1 / (S_1 + S_2 + S_3) \\ S_2' &= S_2 / (S_1 + S_2 + S_3) \\ S_3' &= S_3 / (S_1 + S_2 + S_3) \end{aligned} \tag{2.27}$$

- *Capa 4*: En esta capa, la salida de cada neurona es calculada por el producto de la salida normalizada de la capa anterior y el grado de activación del consecuente. Este valor de salida es dado por:

$$\begin{aligned} h_1 &= S_1'.C_1 \\ h_2 &= S_2'.C_2 \\ h_3 &= S_3'.C_3 \end{aligned} \tag{2.28}$$

donde C_i's corresponden a los valores de los consecuentes, los cuales, por ejemplo, pueden ser funciones de pertenencia "*singlenton's*".

- *Capa 5*: La salida de esta capa proporciona la salida precisa del sistema ANFIS. Esta etapa de defuzzificación se facilita por el cálculo de las capas 3 y 4, dado por:

$$\begin{aligned} Z &= \frac{\sum S_i . f_i}{\sum S_i} = \sum S_i' . f_i \\ Z &= h_1 + h_2 + h_3 \end{aligned} \tag{2.29}$$

El espacio de particionamiento de las *E/S* utilizadas es del tipo *Fuzzy Grid Adaptativo*. Su proceso de aprendizaje del sistema para la identificación de la estructura y parámetros es realizado en dos etapas, repetidas hasta alcanzar el criterio de parada:

Etapa 1: Los parámetros de los consecuentes son ajustados por el método de *MQO (Mínimos cuadrados Ordinarios)*, mientras que los antecedentes permanecen fijos.

Etapa 2: Los parámetros de los antecedentes son ajustados por el método de *GD (Gradient Decrescent)*, mientras que los consecuentes se mantienen fijos.

La idea principal del sistema ANFIS es implementar un sistema fuzzy en una red neuronal, donde generalmente las funciones de pertenencia utilizadas son del tipo sigmoidales.

- ***Neuro Fuzzy Classification (NEFCLASS)***

  Este modelo fue propuesto por *Nauck* y *Kruse*, y es aplicado con éxito básicamente en problemas de clasificación. Además, *Nauck* desarrollo otras dos arquitecturas, el modelo *Neuro-Fuzzy Control (NEFCON)* para aplicaciones en control y el modelo *Neuro-Fuzzy Function Approximation (NEFPROX)* para aplicaciones en previsión y aproximación de funciones. Todos estos modelos utilizan un modelo genérico denominado *Fuzzy Perceptron*. La figura 2.26 ilustra una arquitectura *NEFCLASS*.

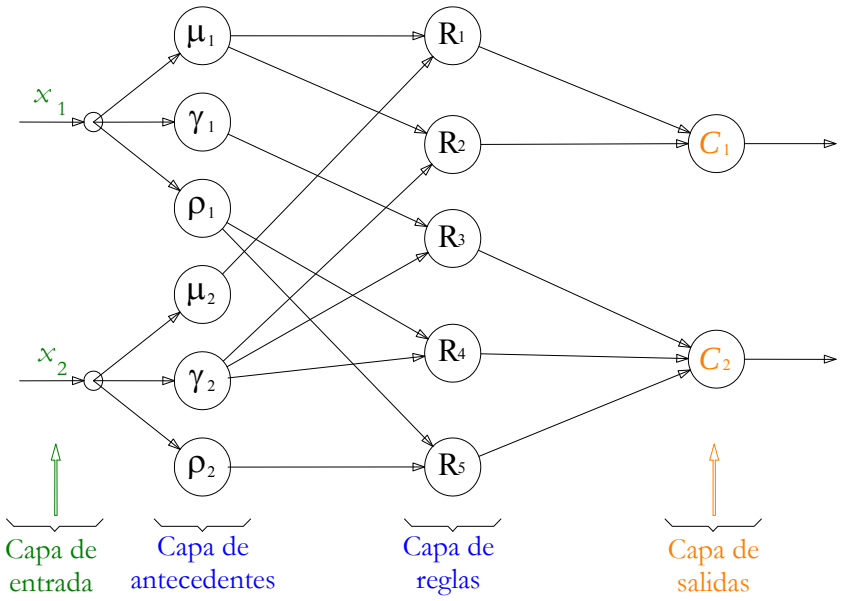

Figura 2.26. Arquitectura NEFCLASS.

Estos tipos de arquitecturas están constituidas de 4 capas y son descritas conforme se sigue:

- *Capa de Entrada*: Esta capa solo tiene la funciono de direccionar los valores de las entradas (x_1 y x_2) hacia las entradas de las funciones de pertenencia de los antecedentes de las reglas. Estas reglas fuzzy tienen la forma:

  Si $x_1 \in \mu_1$ **y** $x_2 \in \mu_2$ **Entonces** padron (x_1, x_2) pertence a la clase i

- *Capa de Antecedentes*: Esta capa proporciona el grado de pertenencia de los antecedentes de cada regla, cada variable de entrada fue dividida en tres funciones de pertenencia (*Bajo, Medio, Alto*). El particionamiento del espacio de entrada utilizado en esta capa es "*Adaptative Fuzzy-Grid*".

- *Capa de Reglas*: En esta capa, se realizan la operación t-norm entre los grados de pertenencia de los antecedentes de cada regla, generando así su grado de activación.

- *Capa de Salida*: Esta capa proporciona la salida del sistema, obtenida realizando la operación *t-conorm* con los grados de activación de la capa de reglas. Los pesos de inter-ligación entre la capa de reglas y la capa de salida son unitarios.

  El aprendizaje de este modelo es realizado en dos etapas:

  Etapa 1: En la primera etapa, se utiliza un algoritmo de aprendizaje de la base de reglas, que puede ser construida de dos maneras: inicializada a partir de un conocimiento previo sobre el conjunto de padrones y después refinada por aprendizaje aumentando o disminuyendo reglas o inicializado con un conjunto vacio de reglas y aumentando por aprendizaje incremental.

  Etapa 2: Después de la creación de la base de reglas, en esta fase se ejecuta un algoritmo de aprendizaje basado en el *GD(Gradient Descent)* para ajustar los parámetros de las funciones de pertenencias de los antecedentes de las reglas.

El sistema híbrido *NEFCLASS* es una red *Fuzzy-Perceptron* que presenta la ventaja de interpretación de la estructura en forma de reglas, y hereda todas las características de las *ANN* del tipo *Perceptron Multicapas(MLP).*

- ***Fuzzy Self-Organized Map (FSOM)***

Este modelo fue desarrollado por *Vuorimaa*, posee una estructura muy semejante al modelo *ANFIS*, como se ilustra en la Fig. 2.27. Este modelo utiliza particionamiento *Fuzzy-Box* en el espacio de entrada.

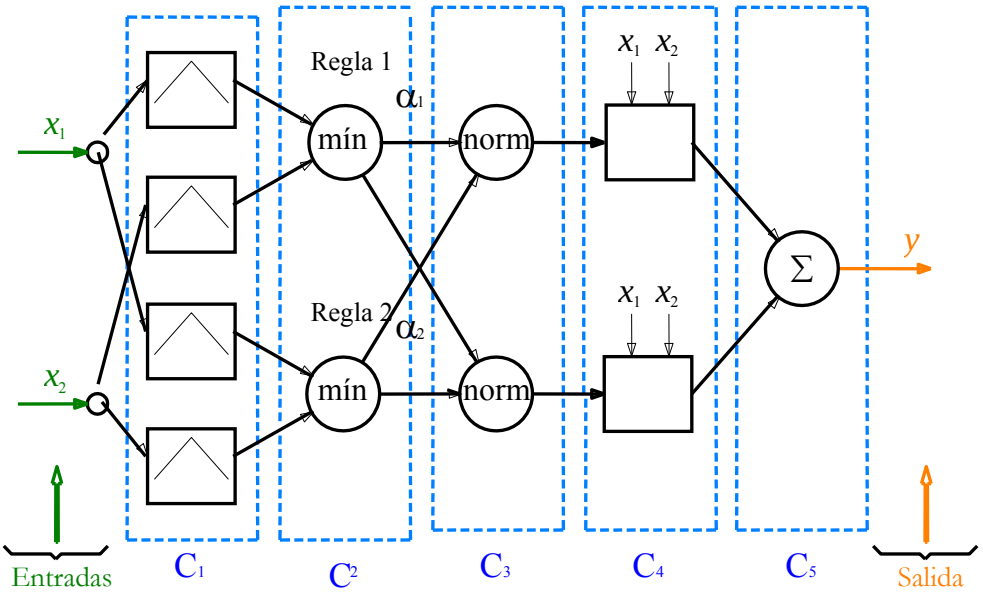

Figura 2.27. Arquitectura FSOM.

La descripción de las capas es semejante al modelo ANFIS, a excepción de algunos detalles descritos abajo.

En la *Capa 1*, las funciones de pertenencia de los antecedentes son de formato triangular, y el formato de la reglas fuzzy son

$$\textbf{Si } x_1 \in U_{i,1} \textbf{ y } x_2 \in U_{i,2} \textbf{ Entonces } y = S_i$$

Donde, $x_1 \in U_{i,1}$ es el grado de pertenencia de la entrada $x_j \in U_{i,j}$ en el conjunto fuzzy $U_{i,j}$.

En la *capa 2*, la operación *t-norm* para el cálculo del grado de pertenencia de las reglas es

$$\alpha_i = min\left\{\mu_{U_{i,1}}(x_1) + \mu_{U_{i,2}}(x_2)\right\} \tag{2.30}$$

Donde, α es el grado de pertenencia calculado para cada regla.

En la capa 5, el cálculo de la defuzzificación es realizado utilizando el método de la media ponderada, y es dado por:

$$y = \frac{\sum \alpha_i . S_i}{\sum \alpha_i} = \sum \alpha_i' . S_i \qquad (2.31)$$

Donde, S_i es el grado de activación de las funciones de pertenencia del consecuente de cada regla.

El proceso de aprendizaje de este modelo es realizado en tres fases.

- *Etapa 1*: Los parámetros de los centros C_i de las funciones de pertenencia triangulares son auto-organizados utilizando el algoritmo *SOM* de Kohonen.
- *Etapa 2*: Los parámetros de los conjuntos fuzzy son formados alrededor de los centros obtenidos en la etapa 1, se utiliza un ancho ω_0.
- *Etapa 3*: En esta etapa, son ajustados los parámetros de los antecedentes por un algoritmo supervisado semejante al LVQ, el cual también es generado por Kohonen.

Esta arquitectura pode ser usada en la implementación de sistemas para aproximación de funciones y control.

CAPITULO III

MODELAMIENTO DEL SISTEMA SERVO-HIDRAULICO

3.1. Introducción

En este capítulo es presentada una breve descripción del modelamiento dinámico del sistema servo-hidráulico. Que está constituido de un conjunto de componentes tales como: fuente de potencia hidráulica, servo-válvula, pistón hidráulico, mangueras, y sensores.

En la figura 3.1, se muestra el sistema hidráulico de una máquina utilizada para ensayos de fatiga.

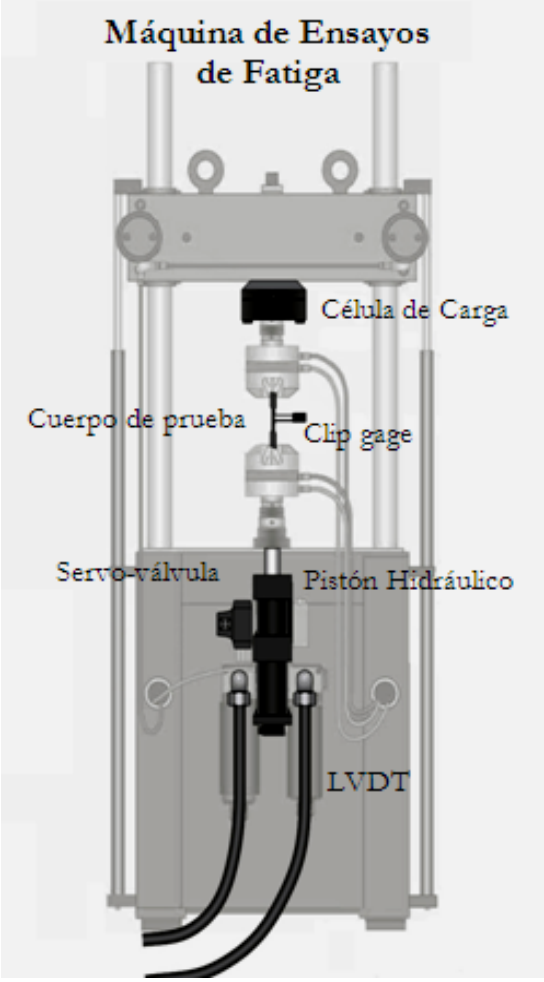

Figura 3.1. Sistema Servo-Hidráulico

En la presente tesis, es utilizado el modelamiento del sistema hidráulico de una máquina para ensayos de fatiga desarrollada por Alva, este modelo es utilizado para evaluar el desempeño de la simulación del controlador propuesto.

El modelamiento del sistema servo-hidráulico es determinado por la dinámica de sus componentes, los cuales son detallados a continuación.

3.2. Modelamiento de la Servo-válvula.

El modelamiento del comportamiento dinámico de la servo-válvula envuelve una variedad de parámetros, como es ilustrado en la figura 3.2. Muchos de estos

parámetros son determinados de diferentes fuentes de información, lo cual hace que el modelo no refleje exactamente su comportamiento real.

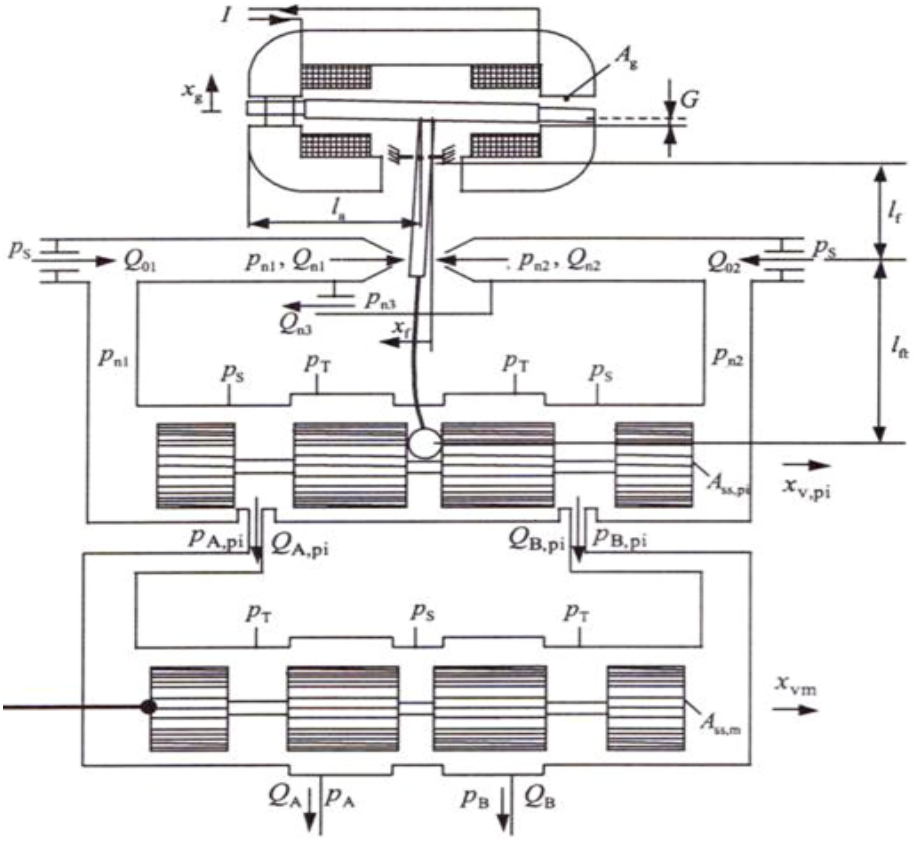

Figura 3.2. Representación esquemática de la servo-válvula de 3 estados.

3.2.1. <u>Modelamiento de la Válvula Piloto</u>

La válvula piloto consiste de un sistema flapper-bocal accionado por el torque de un motor y un carretel (spool) de válvula.

- **Dinámica del Torque del Motor**

De la figura 3.2, el torque electromagnético del motor que direcciona el flapper, es controlado por una corriente eléctrica I. El torque generado en la armadura teóricamente es dado por:

$$T_a = \frac{\mu_0 A_g l_a}{4}\left[\left(\frac{M_0 + IN}{G - x_g}\right)^2 - \left(\frac{M_0 - IN}{G + x_g}\right)^2\right]\frac{1}{2} \tag{3.1}$$

Donde, μ_0 es la permeabilidad de una región particular del circuito magnético, A_g es el área transversal de la abertura, l_a es la longitud de la armadura, M_0 es la fuerza magneto-motriz de los imanes permanentes, N es el número de bobinas, y G es la longitud de la abertura en relación a la posición neutra de la armadura.

Frecuentemente se asume que el torque de la armadura es linear en relación a la corriente de entrada para pequeñas rotaciones de la armadura.

$$T_a = \beta I \tag{3.2}$$

Donde β es conocido como la ganancia de torque del motor.

- **Dinámica del Flapper – Bocal**

El desplazamiento de la punta de la armadura x_g debido a las rotaciones de la armadura está relacionado a la deflexión x_f del flapper entre los bocales y el desplazamiento del flapper l_f .

$$x_g = \frac{l_a}{l_f} x_f \tag{3.3}$$

Como el flapper solo gira sobre pequeños ángulos (≈ 0.01 rad), la ecuación del movimiento puede ser expresado en términos de la deflexión del flapper:

$$\frac{J_a}{l_f}\ddot{x}_f + \sigma_f \dot{x}_f + K_a x_f = T_a + T_{fl} - T_{fb} \tag{3.4}$$

Donde, J_a es la inercia de la armadura del flapper, σ_f es el coeficiente de rozamiento viscoso del flapper, K_a es la rigidez del tubo flexible que

conecta el flapper con la carcasa, y T_{fl} es el torque debido a las fuerzas del flujo. T_{fb}, representa el torque de realimentación del resorte.

- **Continuidad en los Bocales**

Aplicando la ecuación de continuidad en las cámaras de la válvula, y considerando el volumen entre los bocales y el orificio de salida, tenemos

$$\dot{p}_{n1} = \frac{E'}{V_{n1}}\left(Q_{01} - Q_{n1} + A_{ss,pi}\dot{x}_{v,pi}\right) \tag{3.5}$$

$$\dot{p}_{n2} = \frac{E'}{V_{n2}}\left(Q_{02} - Q_{n2} + A_{ss,pi}\dot{x}_{v,pi}\right) \tag{3.6}$$

$$\dot{p}_{n3} = \frac{E'}{V_{n3}}\left(Q_{n1} - Q_{n2} - Q_{n3}\right) \tag{3.7}$$

Donde, $V_{ni,}i = 1,2,3$ son los volúmenes de las cámaras de la válvula, $A_{ss,pi}$ y $\dot{x}_{v,pi}$ son el área lateral del spool y la velocidad del spool respectivamente. Los flujos a través de la entrada son expresados como:

$$Q_{01} = A_0\alpha_{dn}\sqrt{\frac{2}{\rho}\left(p_s - p_{n1}\right)} \tag{3.8}$$

$$Q_{02} = A_0\alpha_{dn}\sqrt{\frac{2}{\rho}\left(p_s - p_{n2}\right)} \tag{3.9}$$

Donde, A_0 es la área del orificio de la entrada. Los flujos de los bocales Q_{n1} y Q_{n2} es dado por:

$$Q_{n1} = \alpha_{dn}\pi d_n(x_{f0} + x_f)\sqrt{\left(p_{n1} - p_{n3}\right)} \tag{3.10}$$

$$Q_{n2} = \alpha_{dn}\pi d_n(x_{f0} - x_f)\sqrt{\left(p_{n2} - p_{n3}\right)} \tag{3.11}$$

Donde, $p_{ni,} i = 1,2,3$ son las presiones en los bocales, x_f es el desplazamiento del flapper, x_{f0} es la distancia flapper – bocal en posición neutra, d_n es el diámetro del bocal, y α_{dn} es el coeficiente de descarga del bocal. El flujo del bocal Q_{n3} a través del orificio de salida es calculado por:

$$Q_{n3} = A_{n3}\alpha_{dn}\sqrt{\frac{2}{\rho}(p_{n3} - p_T)} \tag{3.12}$$

Donde, A_{n3} es el área del orificio de salida.

- **Dinámica del Carretel Piloto**

Aplicando la segunda ley de Newton a las fuerzas sobre el carretel,

$$m_{s,pi}\ddot{x}_{v,pi} + F_f(\dot{x}_{v,pi}) = A_{ss,pi}(p_{n1} - p_{n2}) - \frac{T_{fb}}{l_f + l_{fb}} - F_{ax} \tag{3.13}$$

Donde, $m_{s,pi}$ es la masa do carretel piloto, $F_f(\dot{x}_{v,pi})$ es la fuerza de rozamiento dependiente de la velocidad (por ejemplo, $\alpha_s\dot{x}_{v,pi}$), l_{fb} es la longitud del resorte de realimentación, y F_{ax} es la fuerza del flujo axial sobre el carretel.

3.2.2. <u>Modelamiento de la Etapa Principal</u>

Aplicando la segunda ley de Newton, las fuerzas sobre el carretel principal de la servo-válvula se puede obtener las siguientes ecuaciones:

$$m_{sm}.\ddot{x}_{vm} + F_f(\dot{x}_{vm}) = A_{ss,m}.(P_{A,pi} - P_{B,pi}) - F_{ax,m} \tag{3.14}$$

$$P_{A,pi} = \frac{E'}{V_{A,pi}}.(Q_{A,pi} + A_{ss,m}.\dot{x}_{vm}) \tag{3.15}$$

$$P_{B,pi} = \frac{E'}{V_{B,pi}}.(Q_{B,pi} + A_{ss,m}.\dot{x}_{vm}) \tag{3.16}$$

Donde, $m_{sm}, A_{ss,m}, \dot{x}_{vm}$ es la masa, área y velocidad del carretel principal respectivamente, $F_f(\dot{x}_{vm})$ es la fuerza de rozamiento, $F_{ax,m}$ es la fuerza de flujo axial sobre el carretel, y $P_{A,pi}, Q_{A,pi}, P_{B,pi}, Q_{B,pi}$ son las presiones y flujos en los puntos A y B del carretel piloto.

Como las fuerzas de rozamiento y aceleración son mucho menores que la fuerza impulsora, y la relación entre el área lateral y los volúmenes de las cámaras en el carretel principal son relativamente grandes, la presión dinámica en el carretel principal puede ser despreciado. Así, simplificando las ecuaciones (3.1) y (3.3), se obtiene dos relaciones:

$$Q_{B,pi} = A_{ss,m}\dot{x}_{vm} = -Q_{A,pi} \tag{3.17}$$

$$A_{ss,m}(P_{A,pi} - P_{B,pi}) = 0 \tag{3.18}$$

Finalmente, las ecuaciones que describen la dinámica de los flujos del actuador son:

$$\begin{aligned} Q_A = {} & C_{v1}.sg(x_{vm}).sign(P_s - P_A).\sqrt{|P_s - P_A|} \\ & -C_{v2}.sg(-x_{vm}).sign(P_A - P_T).\sqrt{|P_A - P_T|} \end{aligned} \tag{3.19}$$

$$\begin{aligned} Q_B = {} & C_{v3}.sg(-x_{vm}).sign(P_s - P_B).\sqrt{|P_s - P_B|} \\ & -C_{v4}.sg(x_{vm}).sign(P_B - P_T).\sqrt{|P_B - P_T|} \end{aligned} \tag{3.20}$$

Donde, Q_A, P_A, Q_B, P_B son las presiones y flujos del carretel primario en los puntos A y B, x_{vm} es la posición del carretel principal.

3.3. Modelamiento de la Fuente Hidráulica

En la práctica, asumir una fuente de presión constante es usualmente justificable, siempre que algunas condiciones estrictas están presentes (Viersma, 1980), como:

- El acumulador es colocado muy cerca de la bomba (distancia de hasta 0.3 m).
- La línea de conexión entre la línea principal y el acumulador es la más corta posible (preferiblemente menor que 0,05 m).
- Un acumulador suficientemente grande de gas (1 dm3) es colocado muy cerca de la servo-válvula, solo si las pérdidas del estado de equilibrio entre la bomba y la servo-válvula no sean demasiado bajas. Este acumulador sirve de amortiguador.

Así, Viersma [6] propone un modelo muy simple para las bombas controladas:

$$G_{pu}(s) = \frac{P_s(s)}{P_{s,ref}(s)} = \frac{K_{pu}}{1 + T_{pu}s} \tag{3.21}$$

Pero, la variación de la presión de la fuente P_s, dependiendo del flujo debe ser tomada en consideración en el modelo de la simulación. Es importante para sistemas de sensoriamento de carga o para sistemas de alto desempeño en relación a la velocidad del pistón, donde los límites de operación pueden ser alcanzados, teniendo por resultado una caída de la presión.

3.4. Modelamiento del Pistón Hidráulico.

Las ecuaciones de la dinámica del pistón fue obtenida aplicando la segunda ley de Newton sobre las fuerzas ejercidos en el pistón hidráulico, así tenemos

$$m_t.\ddot{x}_p + F_f(\dot{x}_p) = A_p.(P_A - \alpha.P_B) - F_{ext} \tag{3.22}$$

Donde, m_t es la masa total, constituida por la masa del pistón (m_p), y por las masas de las cámaras del cilindro y sus tabulaciones, dadas por $m_{A,fl}$ y $m_{B,fl}$ respectivamente.

$$m_t = m_p + m_{A,fl} + m_{B,fl} \tag{3.23}$$

La masa del fluido es determinada por

$$m_{A,fl} = \rho.[V_{PL,A} + (x_{p0} + x_p).A_p] \tag{3.24}$$

$$m_{B,fl} = \rho.[V_{PL,B} + (x_{p0} + x_p).\alpha.A_p] \tag{3.25}$$

La masa del fluido puede ser despreciado en relación a la masa del pistón. Aplicando la ecuación de continuidad para cada una de las cámaras del cilindro, se obtiene.

$$Q_A - Q_{Li} = \dot{V}_A + \frac{V_A}{\dot{E}.(P_A)}.\dot{P}_A \tag{3.26}$$

$$Q_B + Q_{Li} - Q_{Le} = \dot{V} + \frac{\dot{V}_A}{\dot{E}.(P_A)}.\dot{P}_A \tag{3.27}$$

Donde, V_A es el volumen de la cámara del pistón, V_B es el volumen de la cámara del anillo y líneas de conexión, Q_{Li} y Q_{Le} son los flujos de escape interno y externo, respectivamente.

3.5. Modelamiento de las Conexiones

Los componentes de los sistemas hidráulicos son usualmente conectados por mangueras. la longitud de estas mangueras no debe exceder cierto límite, como:

$$l < \frac{c}{10 f_{\max}}$$

Donde, c es la velocidad del sonido (o velocidad de la onda) en el aceite, $f_{\max}$ es el mayor valor de la frecuencia. Caso contrario, el comportamiento dinámico de las mangueras hidráulicas poseen parámetros distribuidos que deben ser considerados. La velocidad de onda en líneas rígidas (c_s) y en mangueras (c_w) son:

$$c_s = \sqrt{\frac{E}{\rho}} \qquad\qquad c_w = c_s\left(1+\frac{E}{E_h}\frac{2r_h}{s_h}\right)^{-1/2}\frac{1}{2}$$

Donde, r_h es el radio interno de la manguera y s_h es el espesor de la pared de la manguera. Generalmente, la velocidad de la onda es determinado usando el modulo efectivo de compresibilidad, que introduce los efectos de la entrada del aire y conformidad mecánica:

$$c = \sqrt{\frac{E'}{\rho}}$$

En algunas aplicaciones industriales, los volúmenes ineficientes parecen ser significativamente grandes. Esto es debido a la instalación de diversas características de seguridad hidráulica entre la válvula y el actuador.

3.6. Diseño de la Función de Transferencia

Combinando los modelos dinámicos de la servo-válvula con el modelo del pistón hidráulico, se pude obtener las funciones de transferencia que modela el sistema servo hidráulico, como es detallado a continuación.

3.6.1. Función de Transferencia de Posición

La función de transferencia para el control de posición es determinada por la ecuación

$$G_x(S) = \frac{X_p(S)}{I(S)} = \left[\frac{K_v.K_Q}{\frac{1}{\omega_v^2}.S^2 + \frac{2.D_v}{\omega_v}.S + 1} \right] \left[\frac{\frac{A_p}{m_p}}{S^2 + 2.D_h.\omega_h.S + \omega_h^2} \right] \left[\frac{1}{S} \right] \quad (3.28)$$

Donde ω_h es la frecuencia natural del sistema servo-hidráulico, la cual es determinado por

$$\omega_h = \sqrt{\frac{\sigma}{m_p.T_h} + \frac{A_p}{m_p}.K_d} \quad (3.29)$$

La tasa de amortiguamiento D_h es dada por

$$D_h = \frac{\frac{1}{T_m} + \frac{1}{T_h}}{2.\omega_h} \quad (3.30)$$

Los parámetros T_m y K_d son determinados por

$$\begin{aligned} T_m &= \frac{m_p}{\sigma} \\ K_d &= \frac{A_p}{C_h} \end{aligned} \quad (3.31)$$

Donde m_p e A_p son la masa y área del pistón, y C_h es la capacitancia hidráulica.

3.6.2. Función de Transferencia de Fuerza

La función de transferencia para el control de fuerza es obtenida a partir de la función de transferencia de posición, siendo conocida la siguiente ecuación:

$$P_L = \left(\frac{m_p}{A_p} + \frac{\sigma}{A_p}.\frac{1}{S} \right) \quad (3.32)$$

Utilizando la relación $F_L = A_P.P_L$, se obtiene:

$$G_F(S) = \frac{F_L(S)}{I(S)} = \left[\frac{K_v.K_Q}{\frac{1}{\omega_v^2}.S^2 + 2.D_v.\omega_v.S + 1} \right] \left[\frac{A_p.\left(S + \frac{\sigma}{m_p} \right)}{S^2 + 2.D_h.\omega_h.S + \omega_h^2} \right] \quad (3.33)$$

Donde, ω_v es la frecuencia natural y D_v es la tasa de amortiguamiento, y los demás parámetros fueron definidos en la ecuación (3.28).

3.6.3. Función de Transferencia de Deformación

Para el control de la deformación del cuerpo de prueba durante el ensayos, la rigidez de la máquina debe ser mucho mayor que la de los cuerpos de prueba testados. Asumiendo que esto sea verdad, la función de transferencia para el control de deformación es dada por:

$$G_F(S) = \frac{F_L(S)}{I(S)} = \left[\frac{K_v.K_Q}{\frac{1}{\omega_v^2}.S^2 + 2.D_v.\omega_v.S + 1} \right] \left[\frac{A_p.\left(S + \frac{\sigma}{m_p} \right)}{S^2 + 2.D_h.\omega_h.S + \omega_h^2} \right] \left[\frac{1}{m_p.S^2 + b.S + k} \right] \quad (3.34)$$

CAPITULO IV

CONTROL POR APRENDIZAJE NEURO-FUZZY

4.1 Introducción

En este capítulo se presenta el control por aprendizaje neuro-fuzzy, este modelo de control utiliza un sistema híbrido neuro-fuzzy para calcular y actualizar los puntos de reversión de la servo-válvula del sistema servo-hidráulico. También se presenta una comparación con otros modelos de controle por aprendizaje y todos los programas de simulación fueron implementados en el software de MATLAB©.

4.2 Esquema de Control por Aprendizaje Neuro-fuzzy

En este modelo de control por aprendizaje, la información desconocida es estimada o aprendida por el sistema neuro-fuzzy y proporcionada al controlador "*bang-bang*". Así, a medida que el sistema neuro-fuzzy almacena más información, el controlador es actualizado mejorando el desempeño del sistema.

En la figura 4.1 se presenta el diagrama de bloques que ilustra el controle por aprendizaje neuro-fuzzy. En este modelo de control, la información es representada por la variable U_{IJ} que representa la salida del sistema neuro-fuzzy. Esta información que genera el valor de U_{IJ} es almacenada en los pesos de la estructura del sistema neuro-fuzzy. La variable U_{IJ}, utilizada para cambiar la acción de control sobre la servo-válvula, es actualizada después de cada ciclo de operación a través del ajuste de los pesos de la estructura neuro-fuzzy, utilizando un algoritmo de aprendizaje basado en los errores medidos.

El objetivo del sistema neuro-fuzzy es proporcionar el valor de U_{IJ} al controlador, determinando así el punto de reversión en el cual la servo-válvula va revertir su sentido, de manera que la máquina (el sistema servo-hidráulico) se mantenga trabajando en sus límites de operación.

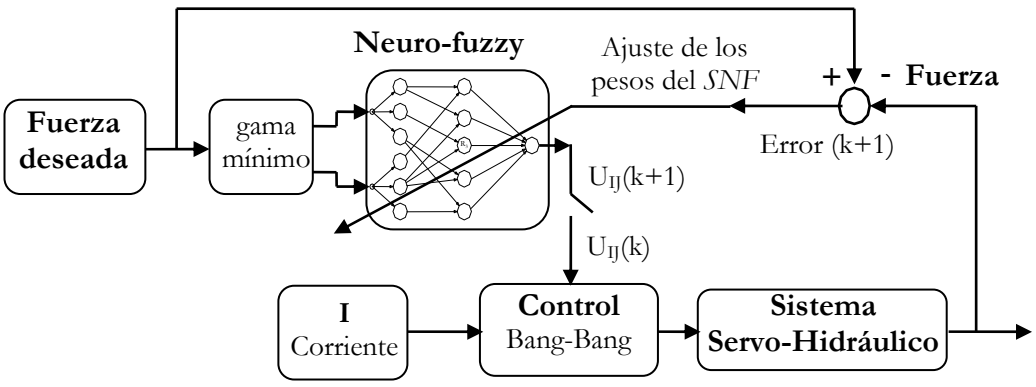

Figura 4.1. Diagrama de bloques del control por aprendizaje neuro-fuzzy.

En la figura 4.1, el valor de $U_{IJ}(k)$ es actualizado por el *switch* con el valor de $U_{IJ}(k+1)$, y el error normalizado $error(k+1)$ es obtenido en función de la fuerza deseada y de la fuerza real medida.

Las entradas del sistema neuro-fuzzy son *gama* (doble de la amplitud) y *mínimo* de la grandeza controlada, la salida del sistema es la variable U_{IJ}. El sistema neuro-fuzzy es constituido por dos capas escondidas: la capa Fuzzy y la capa de reglas. Los pesos del sistema neuro-fuzzy ω_j entre la capa de reglas y la capa de salida son actualizados utilizando el algoritmo de aprendizaje *Backpropagation*, basado en él $error(k+1)$ a cada iteración. La figura 4.2 representa la estructura del sistema neuro-fuzzy propuesto.

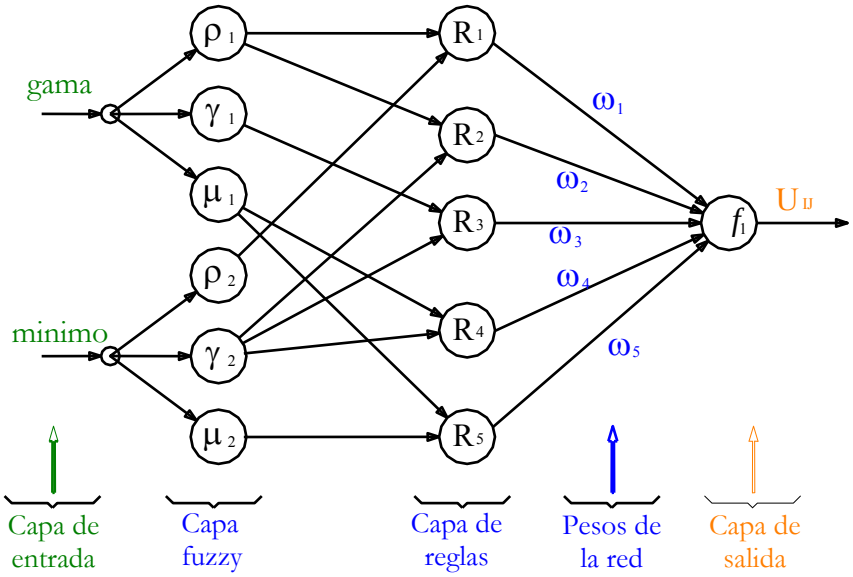

Figura 4.2. Estructura del sistema neuro-fuzzy.

4.3 Modelamiento del Control por Aprendizaje Neuro-fuzzy

El controle por aprendizaje basado en sistemas híbridos neuro-fuzzy (SNF) generalmente están constituido por la combinación de sistemas fuzzy y redes neuronales (ANN). Estos sistemas combinan las ventajas de las ANN tales como la habilidad de aprendizaje, optimización, y conexión en estructura, con las ventajas de los sistemas Fuzzy, el cual utiliza un raciocinio semejante al humano, con facilidad de incorporar informaciones de especialistas. El modelamiento de

este sistema es determinado por las configuraciones de las características de sus "genitores", la parte Fuzzy y Neuronal.

4.3.1 Modelamiento Fuzzy del Sistema NF

El modelamiento de las características de la parte Fuzzy es determinado por la configuración de los parámetros de las siguientes 4 categorías:

- **Modelo Fuzzy**

  El tipo de inferencia Fuzzy (formato de las reglas) implementado en este *SNF* fue el modelo Takagi-Sugeno, con conjuntos Fuzzy de salida del tipo "*singlenton*" para cada una de las reglas. Estas reglas fuzzy son de la siguiente forma:

$$\text{Regla j : Si "}gama\text{" es } \rho_1 \text{ y "}minimo\text{" es } \mu_2 \text{ Entonces } U_{IJ} = \omega_j \tag{4.1}$$

- **Formato de las Funciones de Pertenencia**

  Las funciones de pertenencia en la capa Fuzzy del SNF son generalmente funciones simétricas, tales como funciones triangulares, trapezoidales, y sigmóidales. En esta tesis, se escogió 8 funciones de pertenencia del tipo triangular para cada una de las variables de entrada, por la simplicidad para su implementación experimental. En la figura 4.3, se muestra la función de pertenencia triangular, cuya ecuación es expresada por:

$$\mu(x_i) = 1 - \frac{2.|x_i - c_i|}{b_i} \tag{4.2}$$

  Donde, x_i es el valor de la variable de entrada, c_i es el centro del triangulo de la función de pertenencia, y b_i es el ancho de la base del triangulo.

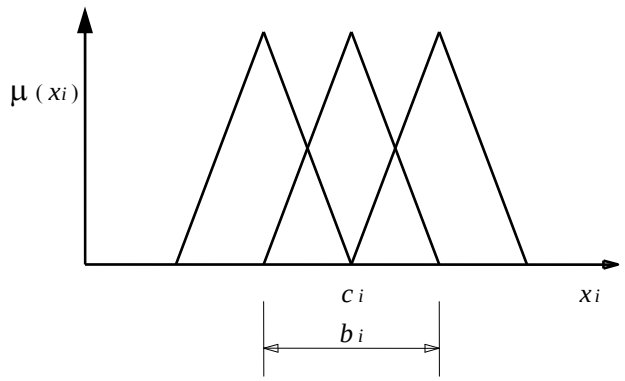

Figura 4.3. Función de pertenencia triangular del *SNF*.

- **Particionamiento del Espacio de E/S**

  El particionamiento del espacio de las variables de entrada y salida es del tipo *fuzzy grid*, el cual mapea internamente regiones fuzzy relacionados a través de sus reglas. En la figura 6.4 se presenta el particionamiento fuzzy grid.

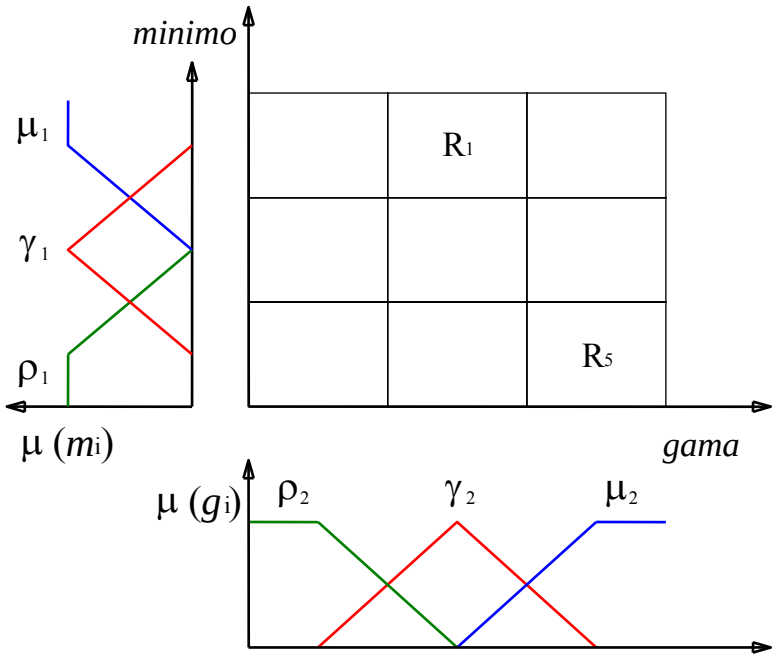

Figura 4.4. Particionamiento *Fuzzy Grid*, donde ρ_1, γ_1, μ_1 y ρ_2, γ_2, μ_2, son los grados de pertenencia de los conjuntos Fuzzy de las variables "*minimo*" y "*gama*" respectivamente.

- **Método de Defuzzificación**

  El consecuente de las reglas es una función del tipo "*singlenton*", y la salida del sistema U_{IJ}, es obtenida de la media ponderada de los grados de activación de cada regla. La salida U_{IJ} es calculada por:

$$U_{IJ} = f\left(P(\mu_i, \mu_j), \omega\right) \tag{4.3}$$

  Donde, $P(\mu_i, \mu_j)$ es el grado de activación correspondiente a cada regla, y ω es el peso de la estructura (salida "*singlenton*").

4.3.2 Modelamiento de la Parte Neuronal

El modelamiento de las características de la parte neuronal está relacionada con la capacidad de aprendizaje del SNF, y determinada por la configuración de los parámetros de las siguientes sub-clases:

- **Tipo de Aprendizaje**

  El aprendizaje utilizado es del tipo online, esto debido a que durante la operación de la máquina el sistema tiene que tener la capacidad de cambiar los puntos de reversión, de manera que los pesos del SNF tienen que ser actualizados a cada iteración.

- **Identificación de la Estructura**

  El número de reglas es determinado por la combinación de los conjuntos fuzzy de las variables de entrada.

- **Identificación de los Parámetros**

  En este tipo de SNF solo presentan aprendizaje en los parámetros del consecuente de cada regla, basado en la medida del error normalizado.

4.4 Algoritmo del Control por Aprendizaje Neuro-fuzzy

4.4.1 Calculo del Valor de U_{IJ}

El valor de la variable a dimensional U_{IJ} es obtenido como resultado de evaluar el SNF para cada combinación de mínimo y gama. Así, para un cargamento con valor mínimo igual a min_i, y gama igual a $gama_j$, se tiene U_{IJ} como salida del SNF.

La fuerza generada por el sistema servo-hidráulico está en el rango de $[-100\ ,\ 100]$ kN. Así, los valores de la variable de entrada, mínimo y gama esta en el rango de $[-100\ ,100]$ y $[-200\ ,\ 200]$ respectivamente. Los valores de gama son positivos cuando el sistema va de un valle para un pico y negativo cuando va de un pico para un valle. Estas variables de entradas del SNF (mínimo y gama) son normalizadas en el rango de [-1,1] utilizando la ecuación 4.4.

$$x_n = \frac{2.(x - \min)}{Max - \min} - 1 \tag{4.4}$$

Donde, x_n es el valor normalizado de la variable x, $\min$ y Max son los valores de mínimo y máximo de la variable x.

La normalización de las variables de entrada, mínimo y gama se obtiene substituyendo en la ecuación 4.4 como se muestra a continuación:

$$min_n = \frac{min}{100} \tag{4.5}$$

$$gama_n = \frac{gama}{200} \tag{4.6}$$

Después de la normalización de las variables de entrada en la capa de entrada, ver la figura 4.5, en la capa fuzzy se calcula el grado de pertenencia con que las entradas satisfacen a los conjuntos fuzzy asociado a cada entrada.

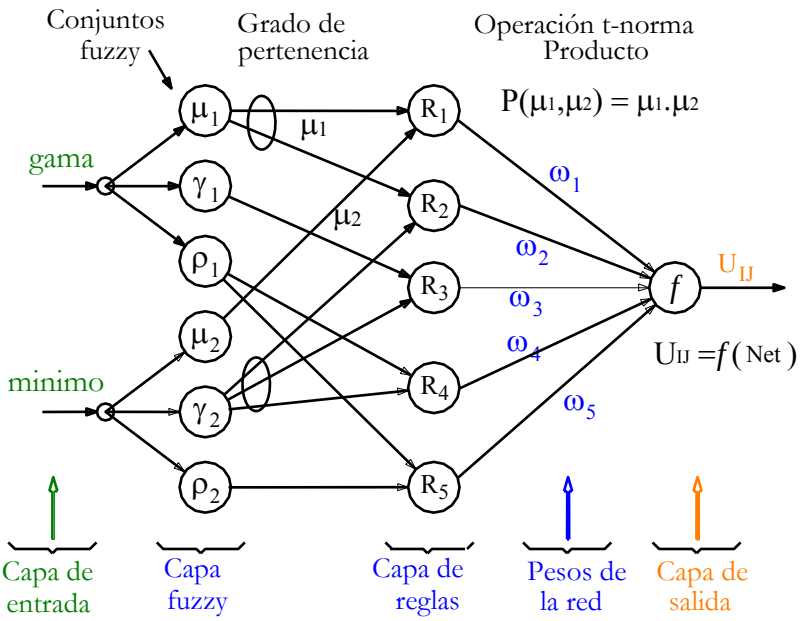

Figura 4.5. Calculo de U_{IJ} y descripción de las capas del SNF

En la capa de reglas, se calcula el grado de activación correspondiente a cada una de las reglas, ejecutando la operación t-norm (producto).

$$P_k(\mu_i, \mu_j) = \mu_i . \mu_j \tag{4.7}$$

En la capa de salida se calcula el valor de U_{IJ} en función del $P_k(\mu_i, \mu_j)$ y ω_k.

$$U_{IJ} = f(Net) \tag{4.8}$$

$$Net = \frac{\sum_{k=1}^{M} P_k(\mu_i, \mu_j).\omega_k}{\sum_{k=1}^{M} P_k(\mu_i, \mu_j)} \tag{4.9}$$

En esta tesis, se considero f como una función de activación lineal por la facilidad en la implementación experimental, pero se puede aplicar otras funciones del tipo sigmoidales (logsig o tansig).

$$U_{IJ} = \frac{\sum_{k=1}^{M} P_k(\mu_i, \mu_j).\omega_k}{\sum_{k=1}^{M} P_k(\mu_i, \mu_j)} \tag{4.10}$$

Donde, $P_k(\mu_i, \mu_j)$ es el resultado de la operación t-norma en la capa de reglas y ω_k es el peso de conexión de la regla k y el neurona de salida.

Las ecuaciones presentadas arriba determinan el valor de U_{IJ} para cualquier tipo de cargamento. Una vez calculado el valor de U_{IJ}, los puntos de reversión de la servo-válvula son calculados, por la siguiente ecuación:

$$Punto\_Reversion = \begin{cases} min + U_{IJ}.gama & (en\ la\ subida) \\ (min+gama) - U_{IJ}.gama & (en\ la\ bajada) \end{cases} \tag{4.11}$$

4.4.2 Ley de Aprendizaje del SNF

El aprendizaje del SNF es realizado por las actualizaciones de los pesos ω_k a cada instante con el valor actual calculado. Todos los valores de ω_k son inicializados con 0,5 y después actualizados en función del error normalizado, el factor de aprendizaje y el grado de activación de cada una de las reglas. El error normalizado es el error obtenido entre el pico (o valle) deseado x_d y el pico (o valle) alcanzado x, dividiendo entre el gama obtenido entre el pico (o valle) deseado y el valor de pico (o valle) alcanzado en la reversión anterior x', o sea:

$$error = \frac{x_d - x}{x_d - x'} \tag{4.12}$$

Si el sistema fuera de un valle para un pico (subida), los valores de x y x_d están cerca del pico y x' habría sido un valle, por lo tanto $x_d - x'$ es positivo. Así, en el caso que $x < x_d$ y el $error > 0$, se tiene *undershoot*. Y en el caso que él $error < 0$, se tiene *overshoot*.

En el caso en que el sistema fuera de un pico para un valle (bajada), los valores de x y x_d están cerca del valle y x' habría sido un pico, por lo tanto $x_d - x'$ es negativo. En este caso, como la carga está disminuyendo, si $x > x_d$ el error es positivo ($erro > 0$), un caso de *undershoot*, mientras que si se da que $x < x_d$ el error es negativo ($erro < 0$), un caso de *overshoot*. De esto se concluye que errores positivos están asociados a *undershoot*, y negativos a *overshoot*, tanto en la subida (valle – pico) como en la bajada (pico-valle). Por lo general, el valor del error esta en el rango de [-1,1] y el algoritmo de actualización de los pesos ω_k del SNF es dado por la siguiente ley de aprendizaje.

$$\omega_k(t+1) = \omega_k(t) + \Delta\omega_k(t) \tag{4.13}$$

$$\Delta\omega_k(t) = \eta.error.\frac{P_k(\mu_i, \mu_j)}{max[P_k(\mu_i, \mu_j)]} \tag{4.14}$$

Donde, $\omega_k(t)$ peso de conexión correspondiente a la regla k, η es el factor de aprendizaje, *error* es el error normalizado y $P_k(\mu_i, \mu_j)$ es el grado de activacion correspondiente a la regla k.

4.4.3 Algoritmo del Control por Aprendizaje NF

El diagrama de flujo del algoritmo de controle por aprendizaje neuro-fuzzy presenta los pasos seguidos para realizar las simulaciones del controlador neuro-fuzzy aplicado a un sistema servo-hidráulico. El controlador determina los puntos de reversión de la servo-válvula, la cual depende del valor de U_{IJ} obtenido de evaluar el SNF para un cargamento deseado.

Por otro lado, la ley de aprendizaje actualiza los pesos ω_k del SNF, el cual utiliza un factor de aprendizaje η que permite acelerar el proceso de aprendizaje.

En la figura 4.6 es presentado el diagrama de flujo del algoritmo de control por aprendizaje neuro-fuzzy.

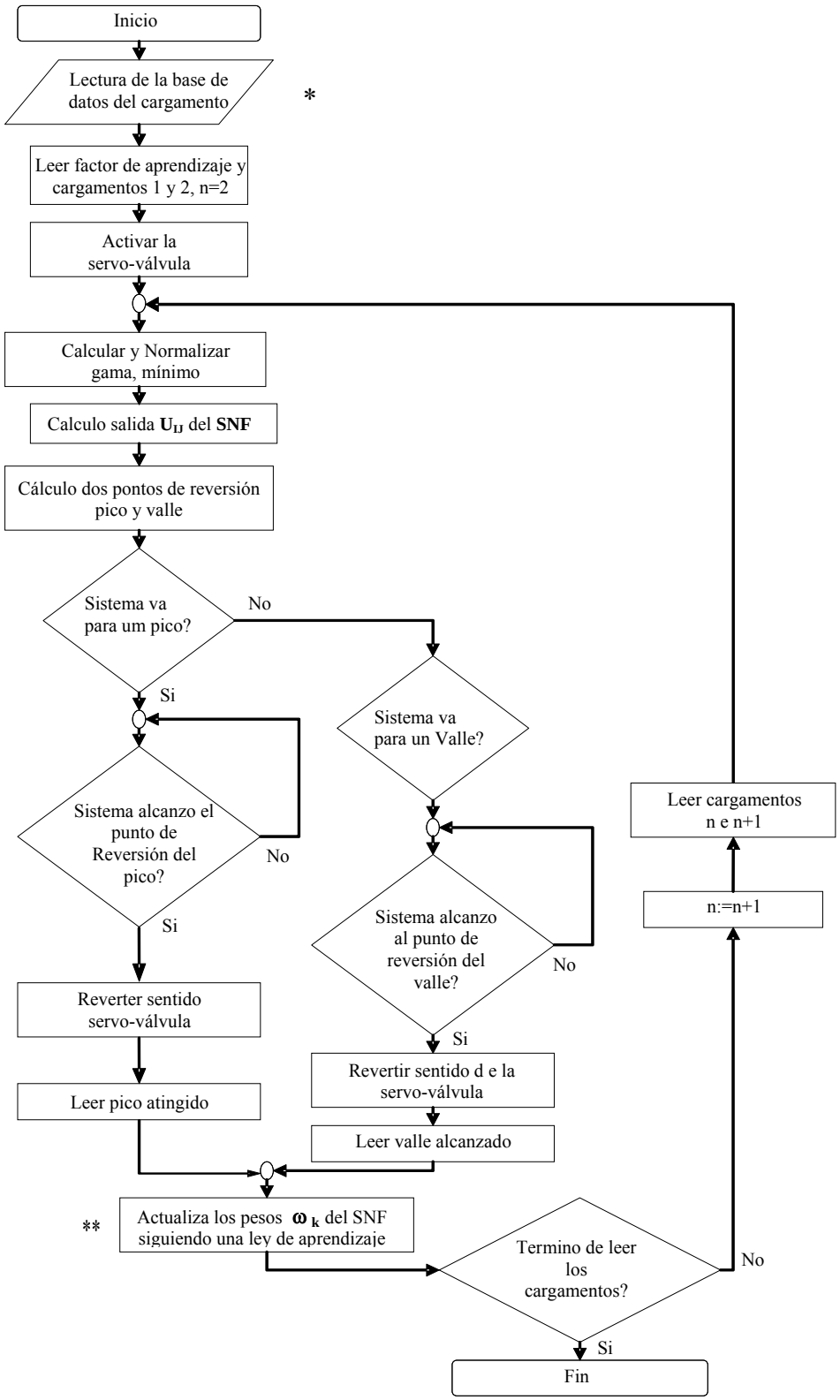

Figura 4.6. Algoritmo de control por aprendizaje neuro-fuzzy

* Los cargamentos son picos e valles requeridos para el ensayo de fatiga

** Para actualizar ω_k, es calculado el error del pico y/o valle y usado una ley de aprendizaje.

CAPITULO V

RESULTADO DE LAS SIMULACIONES

Y HARDWARE

5.1 Introducción

En este capítulo serán presentados los resultados de la simulación del control por aprendizaje neuro-fuzzy propuesto, las simulaciones fueron realizados para cargamentos de amplitud constante y variable, así como un breve análisis de la influencia de los parámetros de aprendizaje. Además se presenta el sistema experimental en la cual fue implementado el control propuesto.

5.2 Resultado de las Simulaciones

Las simulaciones del control por aprendizaje neuro-fuzzy propuesto fueron implementados en el software de Matlab©. Y para las simulaciones se escogieron cargamentos de amplitud constante y variable.

5.2.1 Cargamento Constante

En las figuras 5.1 y 5.2, se muestran las simulaciones para cargamentos de amplitud constante de ± 10 kN y ± 75 kN, respectivamente, con un factor

de aprendizaje $\eta = 0.95$. Se puede observar que los puntos de reversión de pico/valle son modificados a medida que son presentados nuevos cargamentos, hasta convergir en un valor optimo de reversión. Así, en el futuro para cargamentos de la misma amplitud, el controlador puede responder satisfactoriamente sin la necesidad de re-aprendizaje.

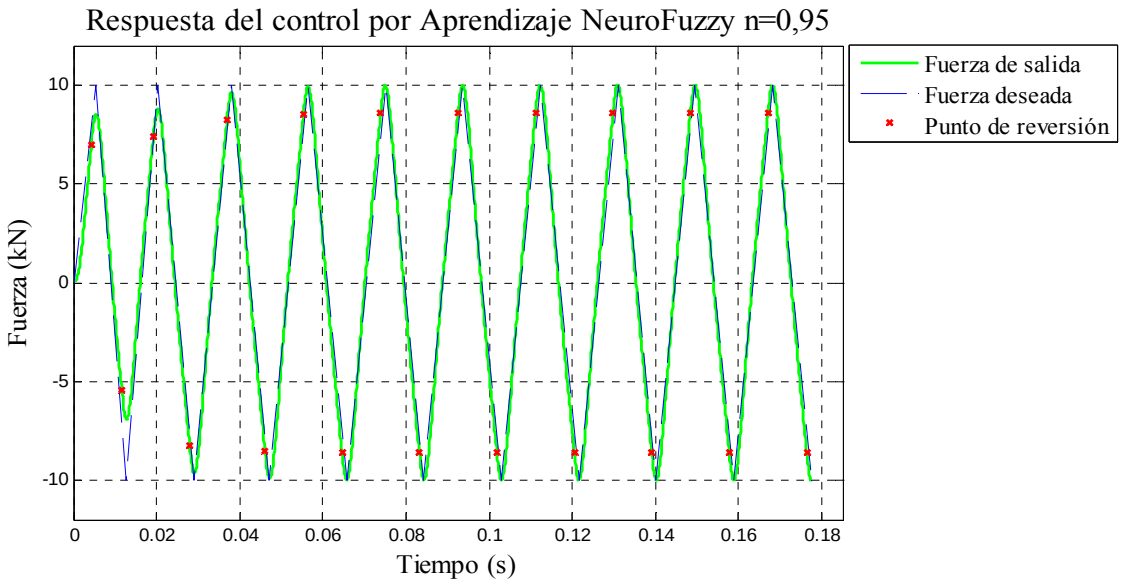

Figura 5.1. Respuesta del control por aprendizaje neuro-fuzzy para un cargamento de amplitud constante de $\pm$ 10 kN.

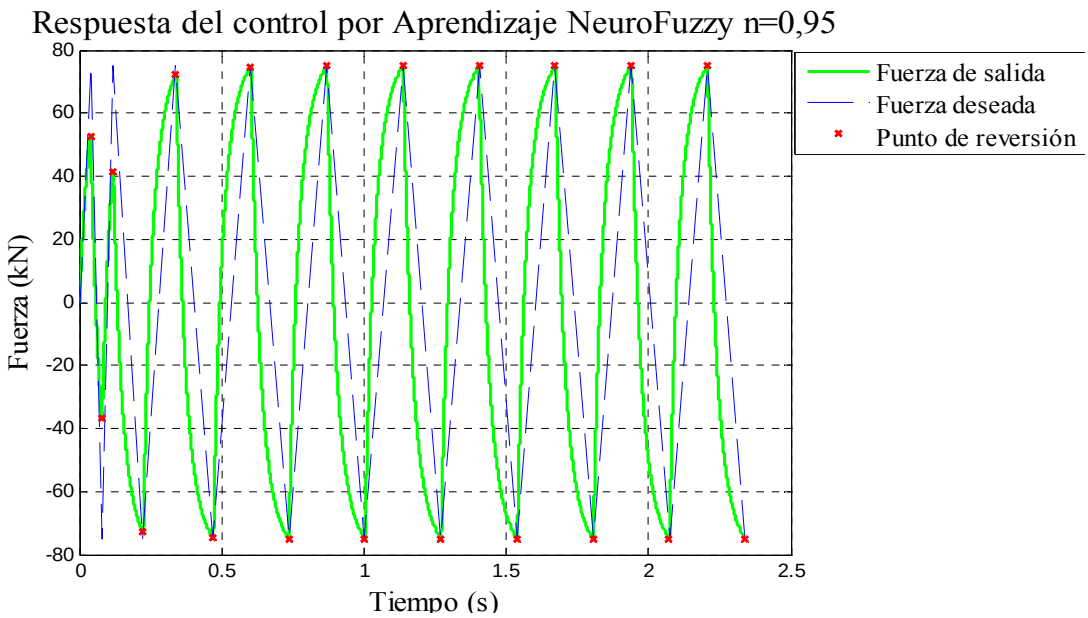

Figura 5.2. Respuesta del control por aprendizaje neuro-fuzzy para un cargamento de amplitud constante de $\pm$75 kN.

En la figura 5.3, las curvas representan la convergencia del error normalizado a lo largo del proceso de aprendizaje. La línea "roja" representa la convergencia del error para el control por aprendizaje desarrollado por Alva, y la línea "verde" del aprendizaje neuro-fuzzy. Se puede apreciar que el aprendizaje neuro-fuzzy tiene una rápida convergencia, con una mayor velocidad de aprendizaje en relación al otro modelo de control.

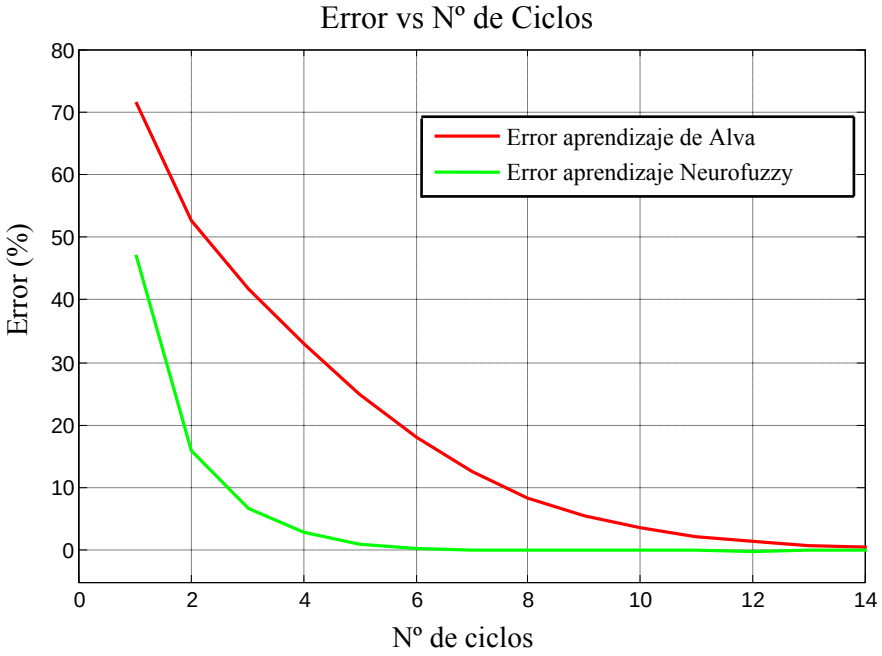

Figura 5.3. Desempeño de los modelos de control por aprendizaje para cargamento constante de $\pm$ 20 kN.

5.2.2 Efecto del Factor de Aprendizaje "η"

Todas las simulaciones presentadas en las figuras anteriores fueron simulados para un factor de aprendizaje $n = 0{,}95$. Pero este parámetro influye en el proceso de aprendizaje, haciendo que el aprendizaje sea más rápido o lento. En la figura 5.4 se presenta la convergencia del erro al

alcanzar los picos o valles para los dos modelos de control por aprendizaje para una $n = 0{,}1$.

Se puede observar que el error del aprendizaje neuro-fuzzy (línea verde) tiene una convergencia más rápida en relación a otros tipos de control. Pero aun así, el valor de $n = 0{,}1$ hace que tenga un aprendizaje lento llegando a convergir después de 40 ciclos.

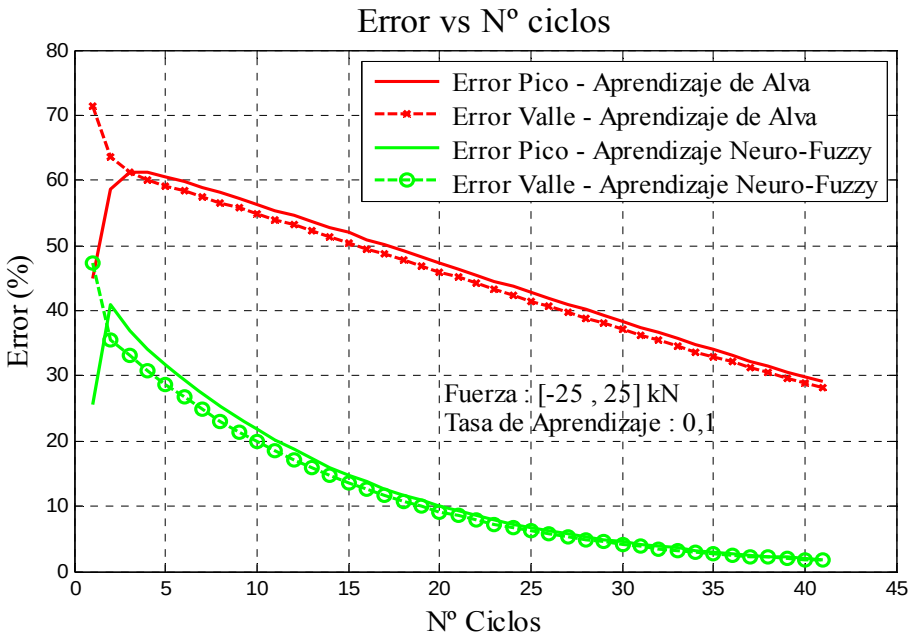

Figura 5.4. Desempeño de los modelos de control por aprendizaje para $n = 0{,}1$ y cargamento de amplitud constante ± 25 kN.

A medida que el factor de aprendizaje aumenta, la velocidad de convergencia del error aumenta, luego la velocidad de aprendizaje del control es mayor.

En la figura 5.5 es presentado la convergencia del error pico/valle de los dos modelos de control por aprendizaje, para un factor de aprendizaje $n = 0{,}95$. Se puede observar que el error del control por aprendizaje neuro-

fuzzy (línea verde) converge para cero después de 8 ciclos y el control por aprendizaje de Alva luego de 30 ciclos.

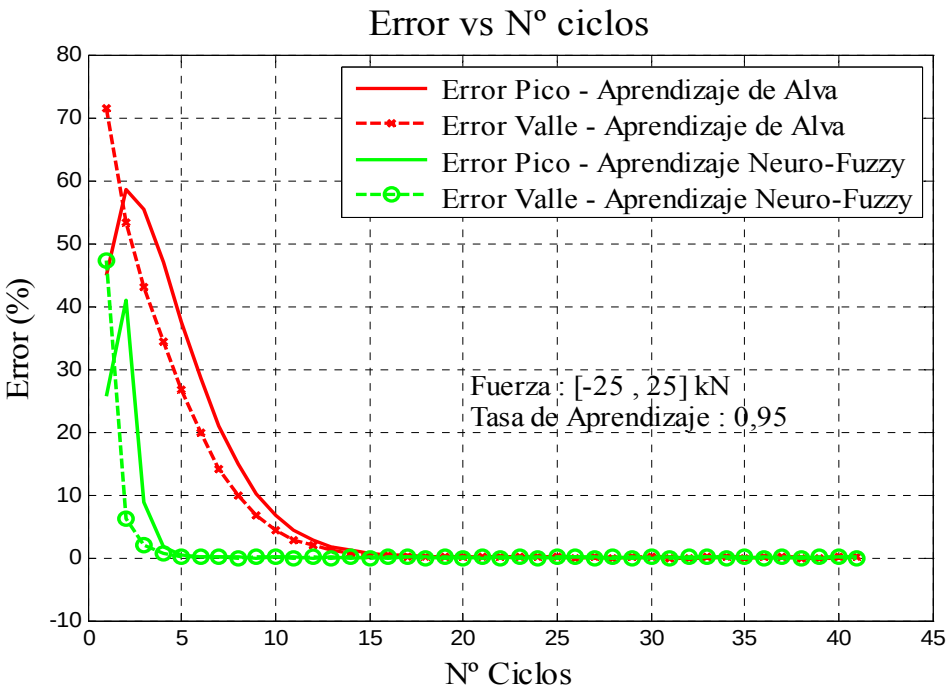

Figura 5.5. Desempeño del control por aprendizaje para $n = 0{,}95$ y cargamento de amplitud constante de ± 25 kN.

El aumento del factor de aprendizaje permite que la velocidad de aprendizaje sea mayor. Por otro lado, una tasa de aprendizaje muy alto presenta oscilaciones en el control y el problema de *overshoot*, que es prejudicial en ensayos de fatiga por introducir efectos de sobrecarga.

La figura 5.6 presenta el comportamiento del error durante el proceso de aprendizaje para $n = 1{,}5$. El control por aprendizaje neuro-fuzzy (línea verde) presenta una rápida convergencia, pero presenta el problema de oscilaciones, indeseado en este tipo de aplicaciones, además de un mayor número de ciclos para la convergencia que en el caso de $n = 0{,}95$.

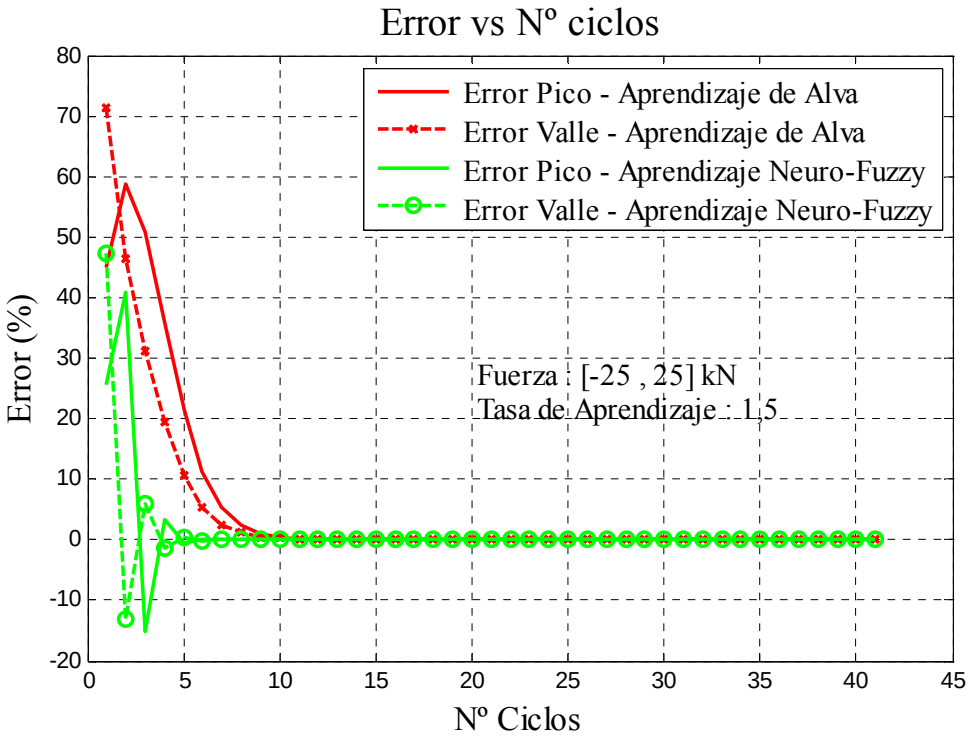

Figura 5.6. Desempeño del control por aprendizaje para $n = 1{,}5$ y cargamento constante de ± 25 kN.

La figura 5.7 presenta el número de ciclos hasta que el control consiga alcanzar un error menor que un valor admisible, en este caso definido en 2% (98% de exactitud), en función del factor de aprendizaje, para un cargamento de ± 25 kN. El control por aprendizaje neuro-fuzzy (línea verde) tiene una aprendizaje mucho más rápido en relación al control por aprendizaje desarrollado por Alva. También se muestra que para tasas de aprendizaje en el rango de [0.1-1] el número de ciclos hasta la convergencia disminuye a medida que aumenta el factor de aprendizaje. Y para valores $\eta > 1$, el número de ciclos para convergencia aumenta con el factor de aprendizaje.

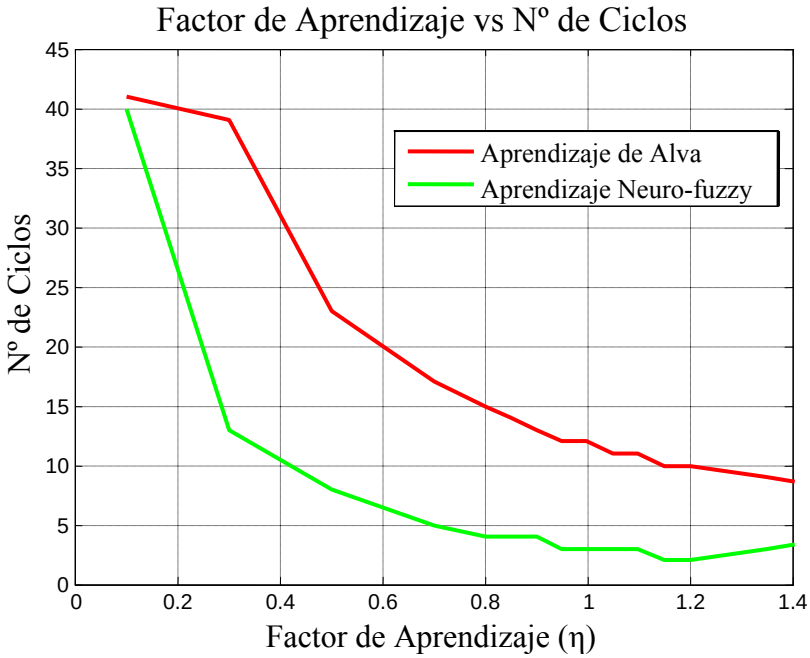

Figura 5.7. Número de ciclos de convergencia en función del factor de aprendizaje, cargamento de ± 25 kN.

La figura 5.8 presenta el desempeño del error en cada pico o valle en función del número de ciclos para un cargamento de ± 80 kN y un factor de aprendizaje $\eta = 0{,}1$. El comportamiento es semejante al obtenido para un cargamento de ± 25 kN.

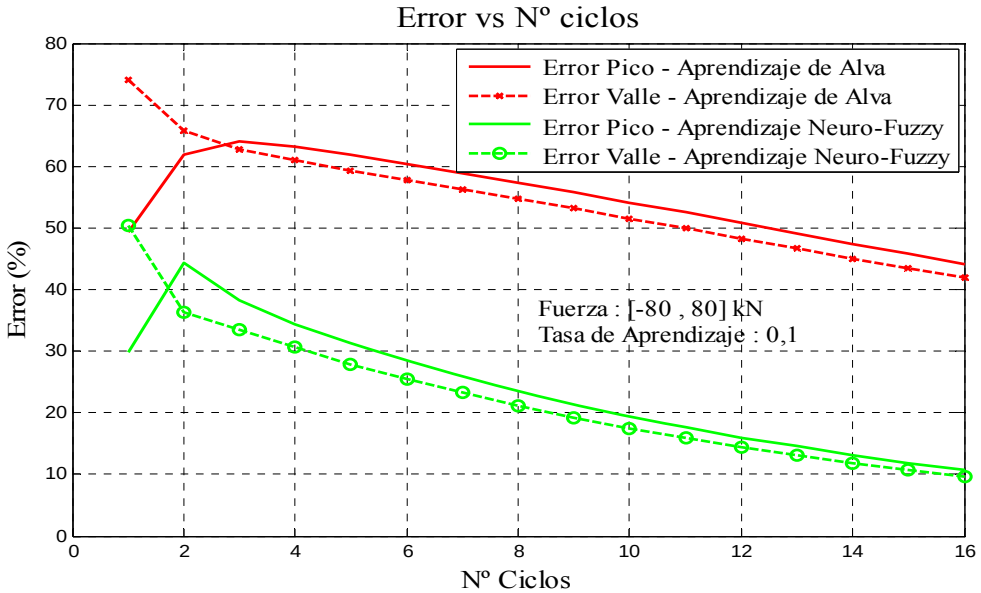

Figura 5.8. Desempeño del control por aprendizaje para $\eta = 0{,}1$ y cargamento de amplitud constante de ± 80 kN.

En la figura 5.9 es presentado el desempeño del error para los mismos niveles de cargamento y con un factor de aprendizaje $\eta = 0{,}85$. En este caso, para un cargamento mayor, y para un factor de aprendizaje menor en relación al caso da figura 5.8, el control neuro-fuzzy tiene un mejor desempeño, convergiendo luego de 6 ciclos.

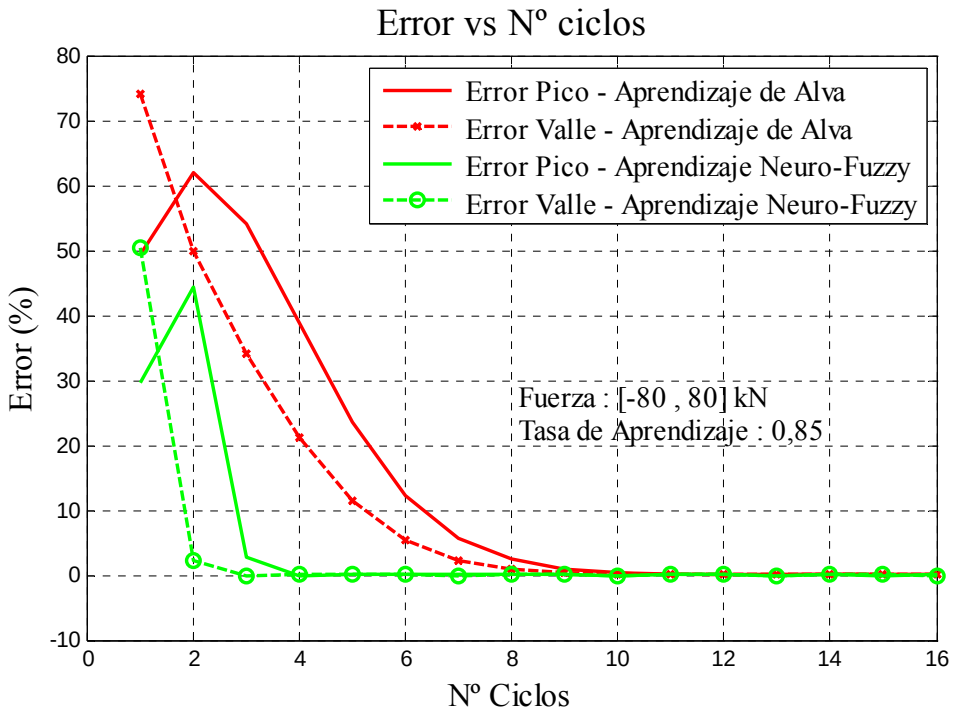

Figura 5.9. Desempeño del control por aprendizaje para $\eta = 0{,}85$ y cargamento de amplitud constante de ± 80 kN.

La figura 5.10 muestra que el controle presenta oscilaciones para un factor de aprendizaje de $\eta = 0{,}95$ para a amplitud de ± 80 kN, el que muestra que el desempeño del aprendizaje también depende de los niveles de cargamento deseado.

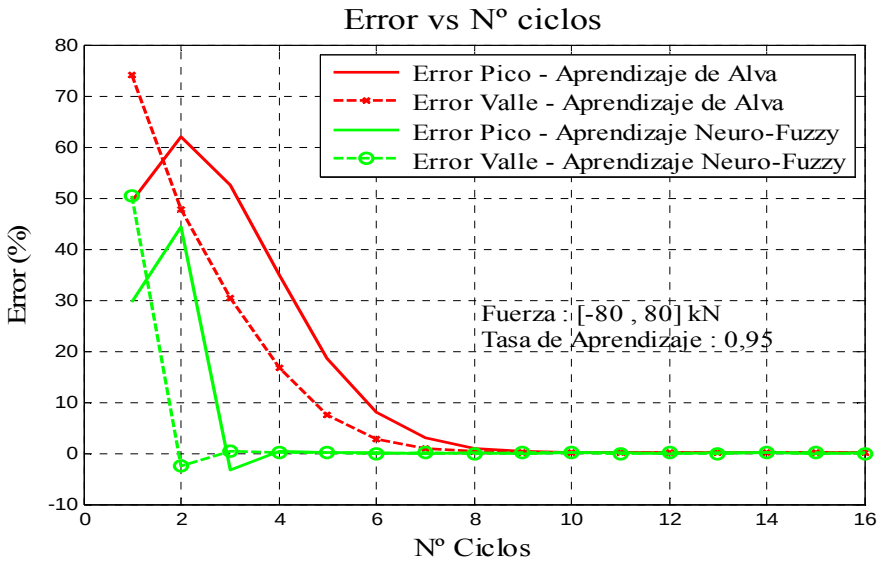

Figura 5.10. Desempeño del control por aprendizaje para $\eta = 0{,}95$ y cargamento de amplitud constante de ± 80 kN.

La figura 5.11 presenta el número de ciclos hasta que el control consiga alcanzar el error de 2% (98% de exactitud), en función del factor de aprendizaje, para un cargamento de ± 80 kN.

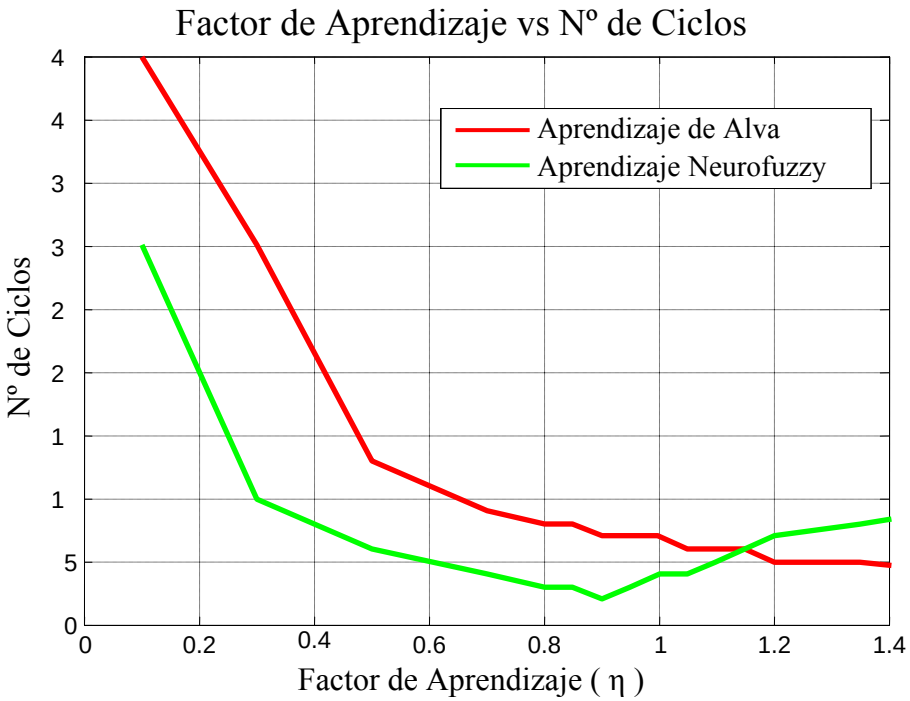

Figura 5.11. Número de ciclos de convergencia en función del factor de aprendizaje, cargamento de ± 80 kN.

La tendencia del desempeño es similar al mostrado en la figura 5.7, con la diferencia que en el control por aprendizaje neuro-fuzzy (línea verde), para valores $\eta > 0{,}9$, el número de ciclos aumenta con el factor de aprendizaje.

Así, el valor óptimo del factor de aprendizaje con el cual el error de control convergió con el menor número de ciclos depende del nivel de cargamento deseado.

Esto es presentado en la figura 5.12, donde se puede observar que el valor óptimo del factor de aprendizaje η para el sistema neuro-fuzzy para un cargamento de ± 25 kN (línea rojo) es cerca de 1.2, para el cargamento de ± 55 kN(línea azul) es 1.1, y para el cargamento ± 80 kN (línea verde) es 0.9.

Como la selección de un valor de factor de aprendizaje mayor que el óptimo presenta oscilaciones y un mayor número de ciclos para la convergencia, es conveniente escoger el menor η optimo, que probablemente estará asociado a la mayor amplitud de la historia de cargamentos. Esto debido a que las oscilaciones presentan casos de overshoot que con indeseados en los ensayos de fatiga.

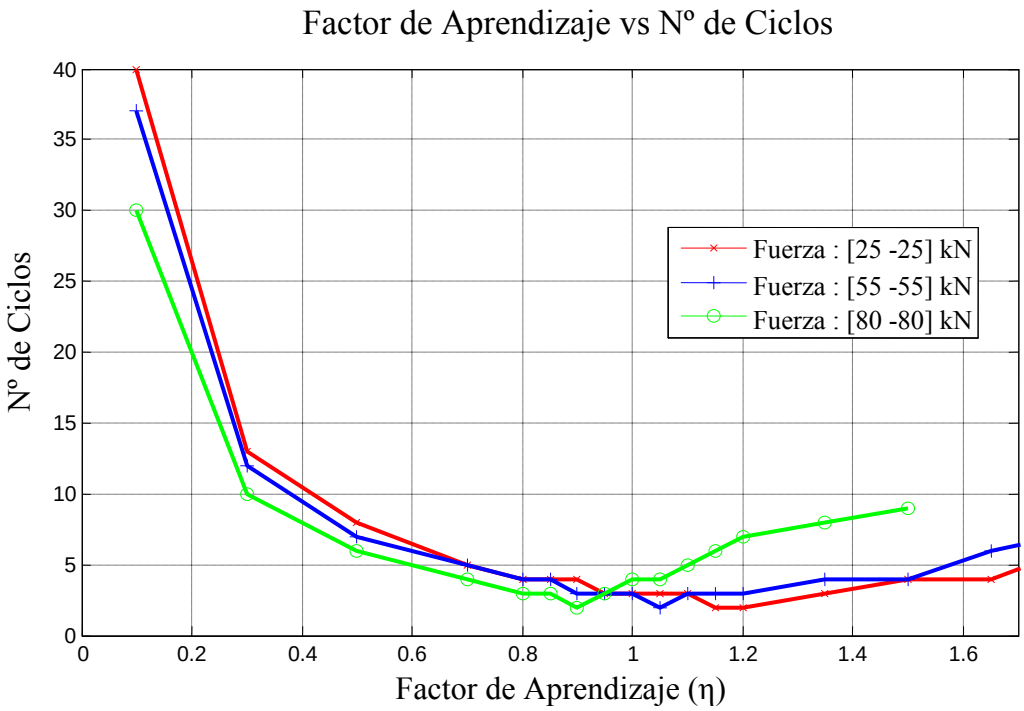

Figura 5.12. Número de ciclos de convergencia en función del factor de aprendizaje, para diferentes cargamentos.

5.2.3 Cargamento Variable

En la figura 5.13 es presentado el desempeño del controle por aprendizaje neuro-fuzzy para cargamentos de amplitud variable. El algoritmo de aprendizaje actualiza los pesos de la estructura neuro-fuzzy para cada combinación de mínimo y gama.

Así, los valores de los puntos de reversión son actualizados después de cada ciclo hasta alcanzar puntos de reversión óptimos, como se muestra en la figura.

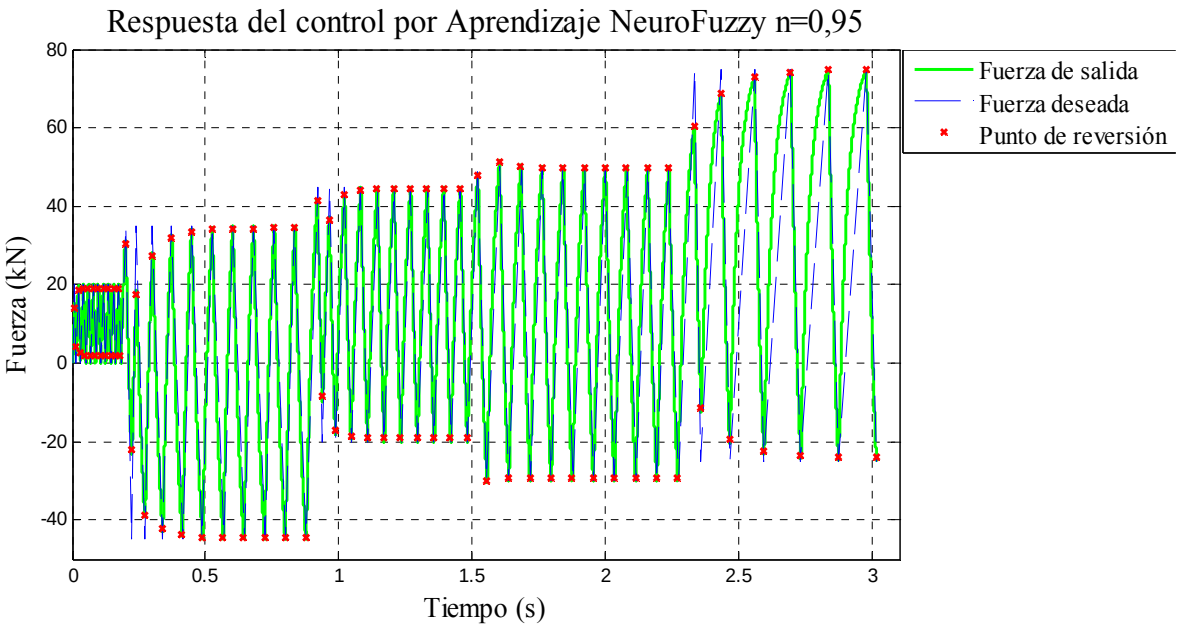

Figura 5.13. Respuesta del control por aprendizaje neuro-fuzzy para diferentes amplitudes de cargamento.

Toda la información obtenida a lo largo de la operación del controlador es almacenada en los pesos de la estructura del sistema neuro-fuzzy. Por lo tanto, en el futuro el sistema no tiene la necesidad de re-aprendizaje para cargamentos que ya fueron presentados anteriormente.

En la figura 5.1-5.2 y 5.13 se muestra que la fuerza de salida no supera a la fuerza deseada, esto debido a que en los ensayos de fatiga el *overshoot* (superar la fuerza deseada) es indeseado. Entre tanto, el *undershoot* (no superar la fuerza deseada) es tolerable.

La figura 5.14 presenta la respuesta del control por aprendizaje neuro-fuzzy para un cargamento que ya fueron presentados 5 veces anteriormente. Se puede apreciar que el sistema presenta un óptimo desempeño desde el primer ciclo.

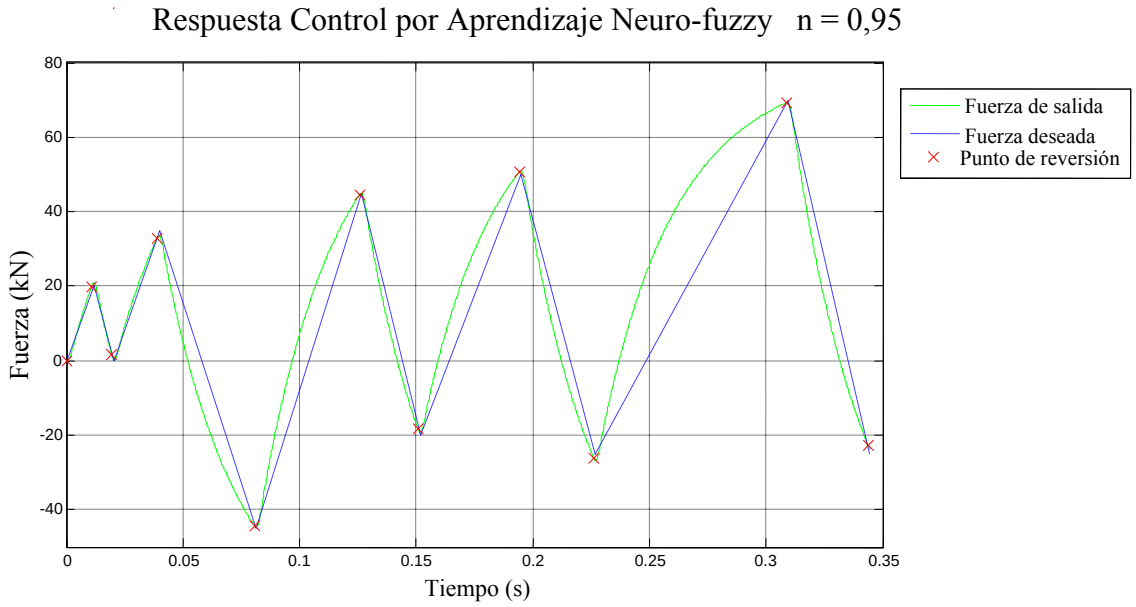

Figura 5.14. Respuesta del control por aprendizaje neuro-fuzzy para cargamentos ya presentados anteriormente.

5.3 Sistema Experimental

5.3.1 Máquina Servo-Hidráulica

Los modelos de control propuestos en esta teses podría ser implementado en una máquina INSTRON modelo 8501 utilizada para ensayos de fatiga uni-axial, la cual está constituida de una bomba hidráulica que proporciona una presión de 190 bar, un actuador hidráulico cilindro-pistón con capacidad de 100 kN, y es comandada por una servo-válvula MOOG modelo D562, que tiene como entrada de controle una señal de corriente de -40mA a 40mA.

Además, la máquina dispone de tres sensores: un LVDT, que mide el desplazamiento del actuador en el rango de [-60 mm, +60 mm], un clip gages que mide las deformaciones del cuerpo de prueba, y una célula de carga de capacidad de 100 kN. El controlador que posee estas maquinas

llegan a alcanzar frecuencias de la orden de 50 Hz para un cuerpo de prueba de acero sometido a un cargamento de 25 kN de amplitud [4].

Figura 5.15. Máquina de Ensayos INSTRON 8501.

Para la implementación del control por aprendizaje en el sistema experimental, es necesario utilizar un módulo computacional capaz de realizar un control en tiempo real. Esto es posible utilizando, por ejemplo, el modulo CompactRIO de la *National Instrument*, que juntamente con sus módulos de entrada/salida analógica, y sus módulos de excitación para extensómetros, es posible alcanzar frecuencias de control del orden de los kHz. En la figura 5.16 es presentado el esquema de conexiones del sistema experimental para el control por aprendizaje propuesto.

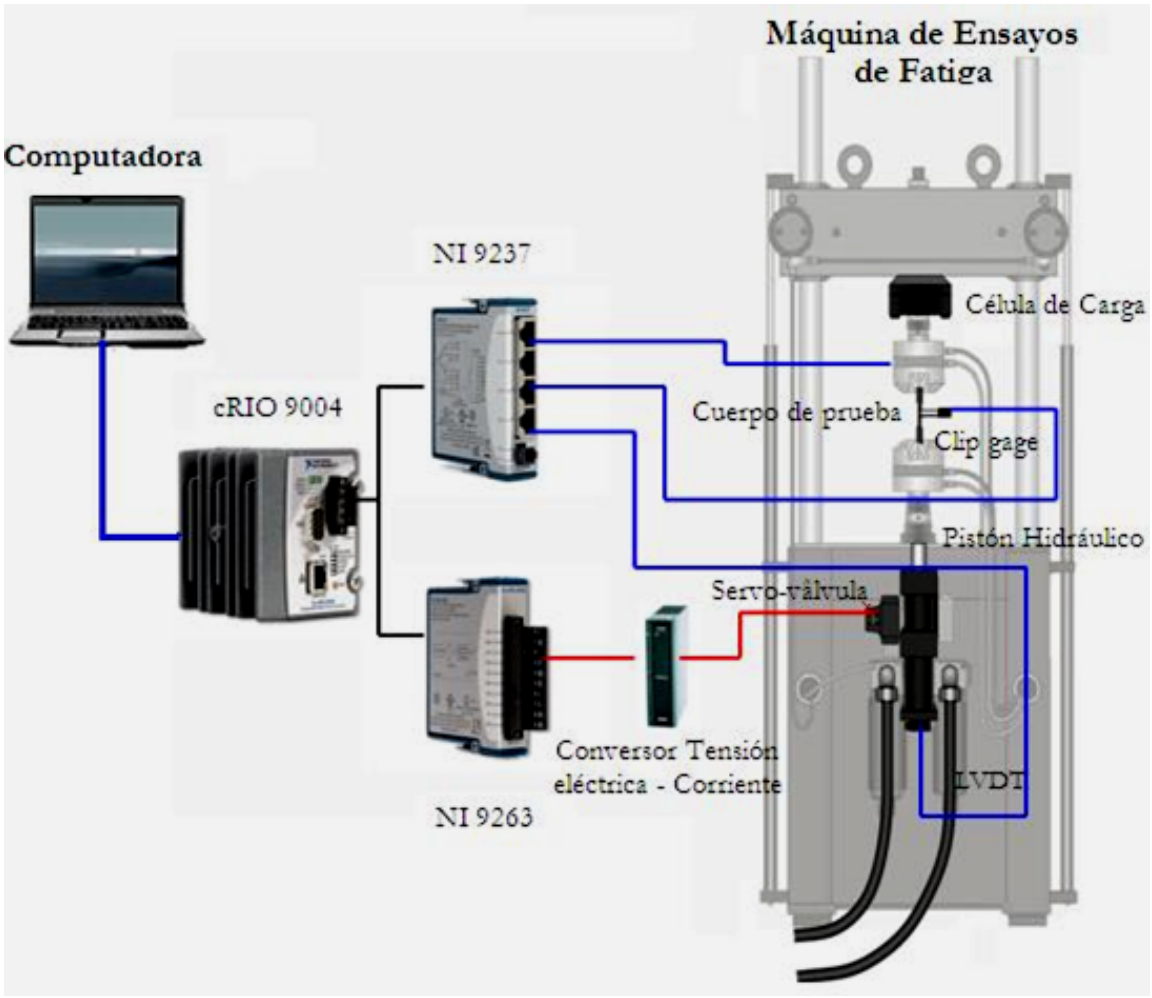

Figura 5.16. Conexiones para el sistema de control por aprendizaje.

5.3.2 Modulo de Control Compact-Rio

El *CompactRIO* de la *National Instruments* es un controlador programable de automatización, que tiene un sistema de control reconfigurable y adquisición de datos diseñado para aplicaciones que requieren alto desempeño y respuesta en tiempo real con alta confiabilidad.

El combina un procesador en tiempo real integrado a un chip FPGA de alto desempeño y robustez, con módulos de entrada/salida intercambiables. El FPGA es conectado al procesador en tiempo real vía un bus PCI de alta velocidad, y cada módulo de entrada/salida es conectado directamente al

FPGA. El CompactRIO usado en este sistema de control es el cRIO 9004, presentado en la figura 5.8.

Figura 5.17. Controlador c*RIO*-9004.

El cRIO-9004 tiene incorporado un procesador industrial clase Pentium de 195 MHz para ejecutar aplicaciones deterministicas en tiempo real desarrolladas en el software LabVIEW Real Time.

El cRIO tiene una memoria de 64 MB en DRAM y 512 MB de almacenamiento Compact Flash no volátil, además de una puerto Ethernet para la programación por red. El LabVIEW Real Time tiene funciones internas para transferir datos entre el FPGA y el procesador en tiempo real dentro del sistema del CompactRIO.

- **Modulo LabVIEW Real Time**

  El Módulo LabVIEW Real-Time amplía la inteligencia distribuida de LabVIEW a hardware de *National Instrument* que ejecuta sistemas operativos en tiempo real. LabVIEW Real-Time puede establecer una interfaz directamente con hardware de E/S reconfigurable (cRIO) para análisis de punto flotante y integración de E/S determinísticas.

Al instalarse el módulo LabVIEW Real-Time de *National Instruments*, este software compila el código gráfico de *National instrument* LabVIEW y lo optimiza para el objetivo en tiempo real seleccionado. Con el Módulo LabVIEW Real-Time se puede desarrollar y desplegar aplicaciones a todos los objetivos hardware de NI en tiempo real, incluyendo PXI, Compact FieldPoint, FieldPoint, CompactRIO y PC de escritorio estándar. El RTOS embebido para estos objetivos es un solo kernel dedicado que proporciona máxima fiabilidad para código embebido.

El módulo LabVIEW Real-Time de *National Instruments*, es utilizado en algunas aplicaciones como: Registro y monitoreo de vibraciones de maquinas rotativas, Simulador para pruebas de un sistema de control de una turbina de viento, Sistemas de control de un brazo robot industrial, etc.

- **Modulo LabVIEW FPGA**

  El FPGA (*Field Programmable Gate Arrays*) es un chip de silicio reprogramable, que utiliza bloques de lógica pre-construidos con entradas lógicas que no están conectadas inicialmente, y que después son configuradas y re-configuradas entre sí para las diferentes aplicaciones que están siendo implementadas (ver figura 5.18). Aplicaciones con algoritmos donde se requiere respuesta en tiempo real, sincronización, precisión, y ejecución de tareas simultáneas de forma paralela, son desarrolladas en el FPGA.

  El paralelismo es conseguido debido al hecho de que el módulo LabVIEW FPGA ejecuta su lógica en el hardware, teniendo el programa la ventaja de

procesar las tareas tales como aplicaciones de control, lectura y grabación de salidas analógicas y/o digitales, en tiempo real y de forma deterministica.

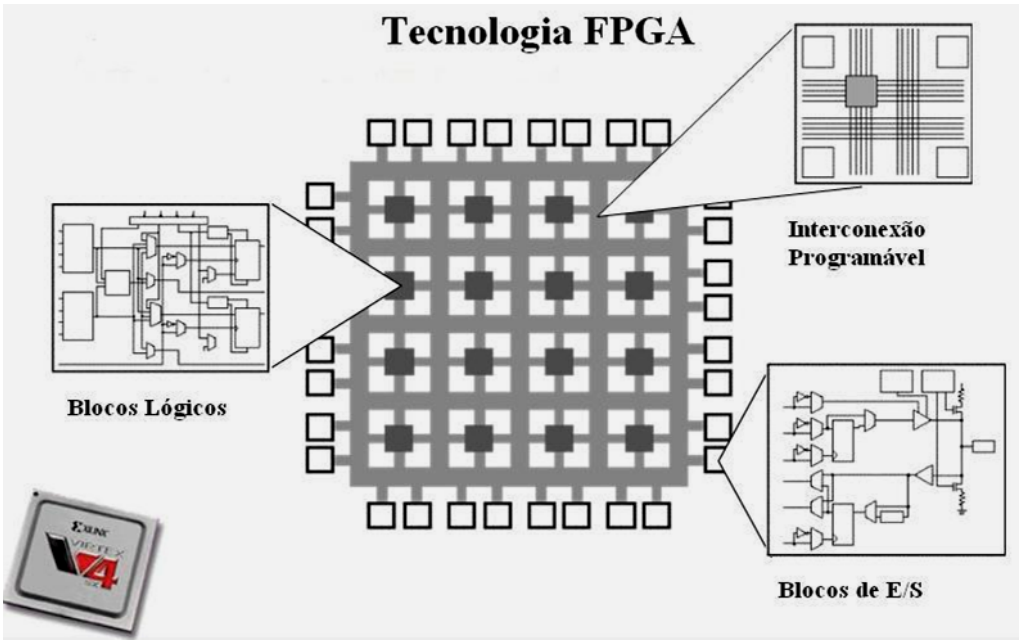

Figura 5.18. Componentes del Chip FPGA.

Para el control de la máquina servo-hidráulica, se trabajo con módulos de entradas y salidas analógicas y con un módulo de excitación de extensómetros.

El módulo NI cRIO 9263 ilustrado en la figura 5.19 (a) es el modulo de salidas analógicas usado para generar tensiones eléctricas entre -10 V y +10 V. Esas salidas analógicas son convertidas en salidas de corriente de -40 mA a +40 mA a través de un conversor de tensión eléctrica para corriente, el cual fue desarrollado por Alva [5].

Él modulo NI cRIO 9237 presentado en la figura 5.19 (b) es el módulo excitador de extensómetros utilizado para excitar y medir el valor de la fuerza aplicada al cuerpo de prueba a través de la célula de carga, y también

puede medir las deformaciones del cuerpo de prueba a través de un clip gage.

(a) (b)

Figura 5.19. (a) NI cRIO 9263 y (b) NI cRIO 9237.

5.3.3 Software Desarrollado en Labview

Los modelos de control por aprendizaje de la máquina servo-hidráulica para ensayos de fatiga fueron desarrollados utilizando el software LabView. Este software utiliza tres ambientes de programación: un computador conectado al cRio, el Real Time del cRio y el FPGA del cRio. La parte del control bang-bang, incluyendo las lecturas de los datos de las entradas analógicas, y del excitador de strain gage, fue implementado en el FPGA, para tener la seguridad de una respuesta en tiempo real, teniendo como referencia experimental que el FPGA puede trabajar con frecuencias de hasta 55 kHz para datos analógicos, y para datos digitales frecuencias de hasta algunos MHz.

La estructura del sistema neuro-fuzzy, las tablas de aprendizaje (aprendizaje de Alva) y los algoritmos de aprendizaje, son implementados en el Real

Time. Finalmente, las configuraciones y presentación de resultados son realizadas por medio del computador.

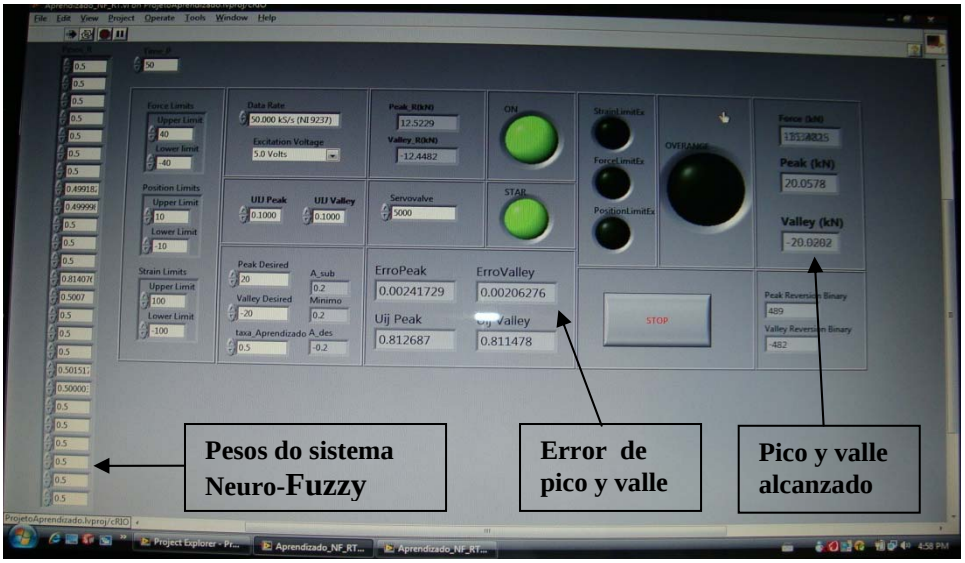

Figura 5.20. Pantalla de control por aprendizaje neuro-fuzzy.

Para determinar la cantidad de dispositivos a usar (sensores, transmisores, controladores, etc.), inicialmente se identificaron todos los lazos de control a implementar así como todas las variables que necesitamos monitorear.

CONCLUSIONES Y RECOMENDACIONES

- **Conclusiones**

  En este trabajo fue desarrollado un sistema de control por aprendizaje neuro-fuzzy para accionar un sistema servo-hidráulicos, este modelo de control propuestos fue simulado y aplicados en el modelo de una máquina para ensayos de fatiga uniaxial.

  El modelo de control por aprendizaje neuro-fuzzy en comparación del aprendizaje desarrollado por Alva no necesita el uso de tablas debido a que todas las actualizaciones y la información del proceso de aprendizaje son almacenadas en los pesos del sistema neuro-fuzzy.

  Este modelo propuesto en comparación al aprendizaje desarrollado por Alva, presenta mejores resultados en las simulaciones tanto para cargas de amplitud constante como para amplitud variable, confirmando así una buena aplicabilidad para el control de sistemas servo-hidráulicos.

  A partir de los resultados obtenidos con el control propuesto se espera generar experimentalmente frecuencias más altas que las del controlador original de la máquina de ensayos INSTRON 8501.

- **Recomendaciones**

  A partir del estudio del efecto del factor de aprendizaje en el aprendizaje del sistema en este tipo de aplicaciones, se recomienda utilizar un valor en el intervalo de [0,5 - 1] debido que una $\eta < 0{,}5$ muy pequeño produce un

aprendizaje muy lento y un $\eta > 1$ produce *overshoots* el cual es indeseado en estas aplicaciones.

El valor optimo del factor de aprendizaje durante la implementación experimental podría ser diferente al obtenido en las simulaciones, debido que estos fueron realizados en un modelo aproximado de un sistema servo-hidráulico (INSTRON 8501), ya que la $\eta = 0{,}95$ fue obtenido con el propósito de mostrar la importancia del factor de aprendizaje durante el proceso de aprendizaje.

- **Trabajos Futuros**

  El sistema de control por aprendizaje neuro-fuzzy propuesto podría ser implementando experimentalmente en una máquina servo-hidráulica INSTRON modelo 805 utilizando el módulo de control CompactRIO de la *National Instruments*.

  Trabajos futuros incluyen el estudio de otros modelos de servo-válvula y módulos de control más rápidos, el cual permita aumentar aún más la frecuencia de ensayo por encima de los 500 Hz.

  El presente trabajo solo contempla el estudio de control basado en la fuerza aplicado sobre el cuerpo de prueba, el cual es útil para los ensayos SN, trabajos futuros contemplarían un modelo de control por aprendizaje neuro-fuzzy basado en las deformaciones medidas sobre el cuerpo de prueba para ensayos tipo εN.

BIBLIOGRAFÍA

[1] Jelali, M, Kroll, A. "Hydraulic Servo - System: Modeling, Identification and Control". New York: Springer, 2003.

[2] F. L. Lewis, J. Campos, R. Selmic. "Neuro-Fuzzy Control of industrial System with Actuator Nonlinearities". Philadelphia, 2002.

[3] P. J. Costa Branco, J. A. Dente. "Design of an Electro-Hydraulic System Using Neuro-Fuzzy Techniques". Mechatronics Laboratory. Department of Electrical and Computer Engineering. Instituto superior técnico, Lisbon. Portugal – 1998.pp 190-206.

[4] Somyot Kaitwanidvilai, M. Parnishkun. "Force Control in a Pneumatic System using Hibrid Adaptive Neuro-Fuzzy Model Reference Control". School of Advanced Technologies, Asian Institute of Technology. Thailand – 2004.pp 23-41.

[5] Juan. G. C. Alva, "Control por aprendizaje de Sistemas Servo-hidráulicos de alta frecuencia". Dissertação de mestrado, Departamento de Engenharia Mecânica, PUC - Rio. Rio de Janeiro, 2008, pp.64 – 89.

[6] Manual INSTRON. Modelo 8500 Plus. 1995. 200 pp.

[7] Shaw, I.S; Godoy M. "Controle e Modelagem Fuzzy". 1ª. edição. São Paulo: Edgard Blucher: FAPESP, 1998. 165p.

[8] Manual Fuzzy Logic Toolbox do MATLAB, versão 7.0 . May, 2004.

[9] Tanscheit, R. "Fundamentos de Lógica Fuzzy e Controle Fuzzy". http//www.ica.ele.puc-rio.br.

[10] Fernandes, R. T. "Supervisão de um Sistema Hibrido Eólico/Diesel usando Lógica Fuzzy". Dissertação de mestrado, Universidade Federal de Mato Grosso do Sul. Campo Grande, 2005, 118p.

[11] Lewis, H. W. "The Foundations of Fuzzy Control". New York: Plenum Press, 1997. 299p.

[12] Delgado. M. R. "Projeto Automatico de Sistemas Nebulosos: uma Abordagem CoEvolutiva". UNICAMP. Tese de Doutorado. Campinas, SP,2002. 186p.

[13] Pedrycz, W., Gomide, F. "Na Introduction to Fuzzy Sets: Analysis and Design". MIT Press. Cambridge, 1998. 456p.

[14] Simon Haykin "Neural Networks - A comprehensive Foundation". Canada. Pearson Prentice Hall, 2001. pp 23.

[15] Braga. A. P., Carvalho. A. P. de L. F., Ludernir. T. B. "Fundamentos de Redes Neurais Artificiais". Editora LTC, 2000 1º Edição.

[16] Z. L. Kovács, "Redes Neurais Artificiais". Editora Acadêmica São Paulo, São Paulo, 1996.

[17] M. H. Hassoun, "Fundamentals of Artificial Neural Networks". MIT Press, Massachusetts, 1995.

[18] Fayal Assis M. A, "Previsão de Vazão por Redes Neurais Artificiais e Transformada Wavelets". Dissertação de mestrado, Departamento de Engenharia Mecânica, PUC - Rio. Rio de Janeiro, 2008.

[19] Vellasco. M. M. B. R. "Inteligência Computacional", ICA: Núcleo de Pesquisa e IA. http//www.ica.ele.puc-rio.br.

[20] Beale R., Jackson T. "Neural Computing: an Introduction". Bristol: Adam Hilger, 1990.

[21] Baron R. "Knowledge Extraction from Neural Network". A Survey. (NeuroCOLT Technical Report), 1995.

[22] Mendes E., Carvalho A. "Target Recognition using Evolutionary Neural Networks". Proceeding of V Brazilian Symposium on Neural Networks, 1998.

[23] Souza F. J. "Sistemas Neuro-Fuzzy". ICA: Nucleo de Pesquiza em IA, http://www.ele.puc-rio.br/labs/ica/icahome.html.

ANEXO

CODIGO MATLAB© DE LAS SIMULACIONES DEL CONTROL POR APRENDIZAJE NEURO-FUZZY DE UN SISTEMA SERVO-HIDRAULICO

```
%-------------------------------------------------------------------------
%CONTROL POR APRENDIZAJE NEURO-FUZZY DE UN SISTEMA SERVO-HIDRAULICO
%-------------------------------------------------------------------------
clear all, close all, clc;
format short;
%-------------------------------------------------------------------------
%---------Vector de Entrada de las Fuerza Deseada ------------------
%-------------------------------------------------------------------------
Fd=[];
 for k=0:9
     i=   10;
     j=  -10;
     Fd = [Fd i j];
 end
%-------------------------------------------------------------------------
%----- Asegura la Dirección inicial de la servovalvula -------------
%-------------------------------------------------------------------------
if Fd(1)>0          % La valvula trabaja totalmente abierto
    servovalve = 1; % 1 En la dirección subida, hacia un Pico
    k=0;
else
    servovalve = -1;% -1 En la dirección bajada, hacia un Valle
    k=1;
end
%-------------------------------------------------------------------------
%--------------Declaración de variables ---------------------------
%-------------------------------------------------------------------------
Nc = size(Fd,2);    % numero de carrgamentos
tfinal=8;           % Tiempo de simulación
dt = 0.000005;       % delta de tiempo
%-------------------------------------------------------------------------
%---Funcion de Transferencia( planta del sistema hydraulico)--------
%-------------------------------------------------------------------------
[numd,dend] =Build_Sist_Hydr(dt);
%-------------------------------------------------------------------------
%------    Creacion del Controlador Neuro-Fuzzy --------------------
%-------------------------------------------------------------------------
N_in=8;
controle=crea_neurofuzzy(N_in);
%-------------------------------------------------------------------------
%--------- Inicializando Variables ------------------------------
%-------------------------------------------------------------------------
SERV=[];
II=[];         % inicializamdo variables
UIJV=[];
%-------------------------------------------------------------------------
%x Posicion del piston
%F Fuerza aplicada por el piston (equivalent to the current i)
FF=[];
F4=0;   %F(i+4)
F3=0;   %F(i+3)
```

```
F2=0;   %F(i+2)
F1=0;   %F(i+1)
F0=0;   %F(i)

I3=0;   %I(i+3)
I2=0;   %I(i+2)
I1=0;   %I(i+1)
I0=0;   %I(i)

Imax=0.04;  % Entrada maxima de alimentacion a la servo-valvula
F=[F4];     % Vector de salida ( Fuerza)
t=[0];      % Vector tiempo
taux=[0];   % Historia con los picos y valles
tswitch=[]; % Historia del tiempo en el momento q la servo-valvula
revierte
xswitch=[]; % Historia de la fuerza en el momento q la servo-valvula
revierte
i=0;        % Indice de tiempo
xlastvalley=0; % Instante del valle anterior
xlastpeak=0;   % Instante del pico anterior
Ft = 0;        % Fuerza en el t=0
j=1;           % Indica el numero de fuerza j
UIJ=0.5;       % Factor de reversion
ERRO=[];       % Historia de Error de pico y valle
Am=0;Minimo=0; % Amplitud y minimo zero
n_taxa=0.95;   % Factor de aprendizaje

while j<=Nc
    i=i+1;
    ti=i*dt;
    % Valores de Amplitud y Minimo Normalizado
    A_forca = (Fd(j)-Ft)/160;
    M_forca =min(Fd(j),Ft)/80;

    % calculo aproximado de UIJ
    if ((Am~=A_forca)||(Minimo~=M_forca))
        [UIJ,Fuzz] = eval_Neurofuzzy([A_forca, M_forca],controle);
        UIJV =[UIJV UIJ];   % Almacenamiento de los valores de UIJ.
    else
        UIJ=UIJ;
    end
    % Corriente maxima dependiendo del sentido de la servo-valvula
    I3=servovalve*Imax;
    II=[II I3];
    % Presicion es muy importante si dt es pequeño, no para acumular
error:
    F4 = (numd(2)*I3 + numd(3)*I2 + numd(4)*I1 + numd(5)*I0)...
        -(dend(2)*F3 + dend(3)*F2 + dend(4)*F1 + dend(5)*F0);
    FF=[FF F4];
    oldservovalve=servovalve;% Determina sentido de la servo-valvula
    if (F4>F3)&(F3>F2)       % El sistema va hacia un pico
        if ( (F4-xlastvalley)>=UIJ*(Fd(j)-xlastvalley)) % Tiempo
para ir al valle
            servovalve=-1;
        else
            servovalve=1;
        end
```

```
    elseif (F4<F3)&(F3<F2) % El sistema va hacia un valle
        if ( (xlastpeak-F4)>=UIJ*(xlastpeak-Fd(j))) % Tiempo para ir
al pico
            servovalve=1;
        else
            servovalve=-1;
        end
    elseif (F4<=F3)&(F3>=F2)&(F3~=0) % El sistema es un Pico
        xlastpeak=F3;
        if mod(j+k,2)==1 %siempre que el sistema va hacia un pico
            if (Fd(j)-xlastvalley)~=0
                peakerror=(Fd(j)-F3)/(Fd(j)-xlastvalley);
              % Aprendizaje del controle Neuro-fuzzy
              controle=nef_learning(controle,Fuzz,peakerror,n_taxa);
                Ft =Fd(j);
                erro=(Fd(j)-F3);
                err=(erro*100)/abs(Fd(j));
                ERRO=[ERRO ; err];
            end
            taux=[taux (ti-dt)];
            j=j+1; % proxima fuerza deseado
        end

    elseif (F4>=F3)&(F3<=F2)&(F3~=0)% El sistema es un valle
        xlastvalley=F3;
        if mod(j+k,2)==0 % siempre q el sistema va hacia un valle
            if (xlastpeak-Fd(j))~=0
                valleyerror=(F3-Fd(j))/(xlastpeak-Fd(j));
            % Aprendizaje del controle Neuro-fuzzy
            controle=nef_learning(controle,Fuzz,valleyerror,n_taxa);
                Ft = Fd(j);
                erro=(F3-Fd(j));
                err=(erro*100)/abs(Fd(j));
                ERRO=[ERRO ; err];

            end
            taux=[taux (ti-dt)];
            j=j+1; % Proxima fuerza deseado
        end
    end

    if (oldservovalve*servovalve==-1)% La servovalvula fue revertido
        tswitch=[tswitch ti];
        xswitch=[xswitch F4];
    end

    plotstep=20;
    if mod(i,plotstep)==1  % grafica cada 20 puntos
        F=[F F4];
        t=[t ti];
    end
    Am=A_forca;
    Minimo=M_forca;
    SERV=[SERV servovalve];
```

```
    % Actualización de variables
    F0=F1; F1=F2; F2=F3; F3=F4; I0=I1; I1=I2; I2=I3;

end

title('Respuesta del control por Aprendizaje NeuroFuzzy
n=0,95','FontSize',16,'FontName','Times New Roman')
ner=size(ERRO,1);
ERRO=[ERRO(1:2:ner,:),ERRO(2:2:ner,:)];
% Guarda la estructura del controlador
save 'controle.mat' controle

% Grafica de la Simulación
plot(t,F,'g','LineWidth',2)
hold on
plot(taux,[0 Fd],'b--','LineWidth',1)
plot(tswitch,xswitch,'rx','LineWidth',2)
xlabel('Tiempo (s)','FontSize',14,'FontName','Times New Roman')
ylabel('Fuerza (kN)','FontSize',14,'FontName','Times New Roman')
legend({'Fuerza de salida','Fuerza deseada','Punto de
reversion'},'FontSize',12,'FontName','Times New Roman')
xlim([0 0.2])
ylim([-12 12])
grid on
```

Función de Transferencia de la Máquina Servo-hidráulica Instron

```
function [Num,Den ] = Build_Sist_Hydr(dt)
%   Construye la Funcion de Transferencia del sistema hidraulico
%   Num : Retorna el numerador de la funcion de transferencia Z
%   Den : Retorna el denominador de la funcion de transferencia Z
%   dt  :   paso
Wv = 1160;      % rad/seg
Dv = 0.7;       % (no dimensional)
Kv = 4;         %
num1 = 4*8000;                          % Numerador de la FT 1
den1 = [(1/1160)^2 2*(0.7)/1160 1];     % Denominador de la FT 1
TF1 = tf(num1,den1);                    % Funcion de Transferencia 1
num2 = 40*[0.047 1];                    % Numerador de la FT 2
den2 = [1 2*(1.09)*(24.9) (24.9)^2];    % Denominador de la FT 2
TF2 = tf(num2,den2);                    % Funcion de Transferencia 2
tfc = series(TF1,TF2);                  % Funcion de Transferencia
total 1+2
tfd = c2d(tfc,dt,'zoh');                % 'foh', 'imp', 'tustin'
[Num,Den] = tfdata(tfd,'v');            % Funcion de Transferencia Z
```

Función que Crea la Estructura del Controlador Neuro-fuzzy

```
function out=crea_neurofuzzy(N_in)
% Crea la estructura del controlador neuro-fuzzy
% N_in : numero de conjuntos fuzzy de amplitud y minimo
nefmat=newfis('Control_neurofuzzy');
```

```
% creando las entradas.
% Amplitude
nefmat = crea_mf(nefmat, 'input', 'Amplitude', N_in, 'trimf', [-1.2
1.2]);
% Minimo
nefmat = crea_mf(nefmat, 'input', 'Minimo', N_in, 'trimf', [-1.2
1.2]);
% Punto de Reversion
nefmat=addvar(nefmat,'output','Reversao',[0.4 1.0]);
nefmat=addmf(nefmat,'output',1,'Pr1','trapmf',[0.3 0.35 0.4 0.65]);
nefmat=addmf(nefmat,'output',1,'Pr2','gaussmf',[0.06 0.6]);
nefmat=addmf(nefmat,'output',1,'Pr3','gaussmf',[0.03 0.75]);
nefmat=addmf(nefmat,'output',1,'Pr4','gaussmf',[0.03 0.85]);
nefmat=addmf(nefmat,'output',1,'Pr5','gaussmf',[0.03 0.9]);
nefmat=addmf(nefmat,'output',1,'Pr6','gaussmf',[0.03 0.94]);
nefmat=addmf(nefmat,'output',1,'Pr7','trapmf',[0.95 0.96 1.2 1.5]);

% Reglas del control neuro-fuzzy
ruleList=[];
for i=1:N_in
    for j=1:N_in;
        ruleList=[ruleList;[j i 4 1 1]];
    end
end

nefmat=addrule(nefmat,ruleList);
N_rule=size(nefmat.rule,2);
nefmat.peso=ones(1,N_rule)*0.5;
nefmat.d_peso=zeros(1,N_rule);
out=nefmat;
```

Función que Evalúa el controlador Neuro-fuzzy

```
function [out,Fuzz] = eval_Neurofuzzy(x,nef_mat)
%%  Evalua el controlador Nuero-Fuzzy para la variable de entrada x
%   ------------------- Entrada ----------------------------------
%   x       ==> Es el vector de entrada
%   nef_mat ==> Estructura del sistema Neuro-Fuzzy
%   ---------------- Salida ------------------------------------
%   out     ==> Salida al evaluar al sistema Neuro-Fuzzy
%   ------------------------------------------------------------
%   Por ejemplo:
%           N_in=8;
%         nef_fis = crea_neurofuzzy(N_in);
%         x1=[2 1];x2=[4 9];
%         out = eval_Neurofuzzy(x1,nef_fis)
%   El Resultado es:
%         out:
%              7.0147
%% ------------------------------------------------------------
if nargin < 1
    error('No hay suficientes argumentos de Entrada.')
end
if nargin < 2
    error('Especificar a estrutura NEFCON.')
```

```
end
m = size(x, 2);
if (m <= 1)
    error('Muy pocos parametros de entrada de x!');
end
m = size(x, 1);
if (m ~= 1)
    error('Soporta solo 2 vector fila de entrada!');
end
%% salida de cada rega luego de la fuzzificacion
Fuzz=nef_fuzzif(x,nef_mat);
% numero de reglas
Nr=size(nef_mat.rule,2);
bias=0.2;
net=0;
for Ind_rule=1:Nr
    W_rule=nef_mat.peso(Ind_rule);
    net_rule=Fuzz(Ind_rule)*W_rule;
    net=net+net_rule;
end
%% calculo de la sumatoria de las salidas de toda las reglas
S_fuzz=sum(Fuzz);
if(S_fuzz ~= 0)
    net=net/S_fuzz;
else
    net=net;
end
net=net+bias;
out=fun_valid(net);


function [Fuzz,Gper]=nef_fuzzif(x,nef_mat)
% fus=nef_fuzzif(x,N)
% ---------------- Entradas ----------------------------------------
% x         ==> Valores de entrada para los cuales se calcula la
respuesta fuzzificada.
% nef_mat   ==> Estructura del sistema NEURO-FUZZY
%-----------------  Salida   ----------------------------------------
% Fuzz      ==> Salida fuzzificado, para cada inferencia de las
reglas.
% Gper       ==> Grado de perteneNcia para cada uno de las MF de las
reglas.
%-------------------------------------------------------------------
if nargin < 1
    error('No hay suficientes argumentos de Entrada.')
end
if nargin < 2
    error('Especificar a estrutura NEURO-FUZZY.')
end
%---------------------------------------
fus=[];
Nrul=size(nef_mat.rule,2);  % numero de regLas
N_x=size(x,2);              % numero de entradas
if (Nrul~=0)
    for i=1:Nrul
        Na=size(nef_mat.rule(i).antecedent,2);% Numero de
antecedentes de la regra(i)
```

```
        N_c=nef_mat.rule(i).connection;       % Tipo de conexion de
la regla(i)
        for j=1:Na
            N_mf=nef_mat.rule(i).antecedent(j);% Tipo de funcion de
pertenencia de la regla(i)
            if (N_mf~=0)
                tipo=nef_mat.input(j).mf(N_mf).type;
                parametro=nef_mat.input(j).mf(N_mf).params;
                out=evalmf(x(1,j),parametro,tipo);
                Sal(i,j)=out;
            else
                if (N_c==1)
                    Sal(i,j)=1;
                elseif (N_c==2)
                    Sal(i,j)=0;
                end
            end
         end
       % calculo da inferencia dos antecedentes de cada regra.
       if (N_c==1)
          % Produto  ==> P(u1,u2) = u(1)*u(2);
          Fuzz(i,1)= Sal(i,1)*Sal(i,2);
       elseif (N_c==2)
          % Soma Probabilistica ==> P*(u1,u2)= u(1)+u(2)-u(1)*u(2);
          Fuzz(i,1)= Sal(i,1)+Sal(i,2)-Sal(i,1)*Sal(i,2);
       end
   end
   Gper=Sal;
else
    error('A Estrutura Neurofuzzy nao tem Regras')
end
```

Función de Aprendizaje del Controlador Neuro-fuzzy

```
function [out,d_peso]=nef_learning(nef_mat,fuzz,erro,n)
%  Funcion que hace el aprendizaje del controlador neuro-fuzzy
%-----------------------------------------------------
% Atualizacion de los pesos de la red.
if nargin < 4
    n=0.9;
end
% Halla el peso de la regla de mayor funcio de ativacion
max_fuzz=max(fuzz);
I_peso=find(fuzz);
n_fa=length(I_peso);
for i=1:n_fa

    d_peso=n*erro*fuzz(I_peso(i))/max_fuzz;
    nef_mat.peso(I_peso(i))=nef_mat.peso(I_peso(i))+ d_peso;
    nef_mat.d_peso(I_peso(i))=d_peso;
    clear d_peso;
end

out=nef_mat;
```