Capítulo 2

Control clásico/moderno y control inteligente.

2.1 Introducción

En este capítulo se presentan brevemente los conceptos principales en que se basan el control clásico y el control moderno, así como algunos de los esquemas utilizados en el control inteligente como son el control difuso, el control neuronal[1] y el control neurodifuso. Se mencionan algunos de los aspectos importantes de cada uno de los esquemas, así como su utilización para el control de procesos. El objetivo primordial de este capítulo es presentar los elementos más importantes en que se basan los controladores implementados en los capítulos 4 y 5.

2.2 Esquemas de control clásico y moderno

La teoría desarrollada para el control de procesos, desde el punto de vista clásico y moderno, tiene su base esencial en el conocimiento de la dinámica del proceso que se desea controlar. Esta dinámica normalmente se expresa haciendo uso de ecuaciones diferenciales ordinarias, y en el caso de sistemas lineales, se hace uso de la transformada de Laplace para obtener una representación matemática que relaciona la señal que se quiere controlar y la señal de entrada al sistema. Esta relación matemática se conoce como función de transferencia.

Desde la teoría clásica de control, considerando el caso más sencillo de un sistema lineal de una entrada y una salida, la dinámica se puede representar como en la figura 2.1. En esta figura se representa el bloque etiquetado como "Proceso" o "Planta", que es el sistema que se desea controlar. A este sistema le llegan dos señales, una etiquetada como "Entrada de control" que será la señal que genera el controlador que se ha de diseñar y la señal etiquetada como "Entrada incierta" que puede representar cualquier señal indeseable externa al sistema y que se conoce también como "perturbación" o "ruido". Finalmente la señal de "Salida" que será la señal que se desea que se comporte de una forma determinada. La señal de salida también se conoce como señal controlada.

[1] El término de "**control neuronal**" también es usado como "**control neural**"; y se refiere a las técnicas de control de sistemas dinámicos asociadas al uso de redes neuronales artificiales.

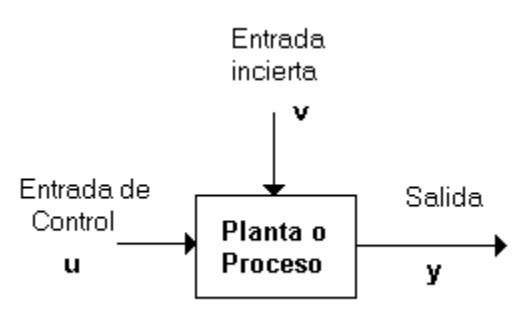

Figura 2.1 Representación a bloques de un sistema.

La función de transferencia, expresada como una relación de dos polinomios puede ser representada en forma general como se muestra en la ecuación (2-1).

$$\frac{Y(s)}{U(s)} = \frac{Ko(s+z_1)(s+z_2)\cdots(s+z_m)}{(s+p_1)(s+p_2)\cdots(s+p_n)} \qquad \textbf{(2-1)}$$

La representación anterior puede ser un poco más general si se hace uso de la teoría de control moderna, en donde la representación matemática utiliza el concepto denominado estado del sistema. Su representación más general se muestra en la figura 2.2. De la planta ahora se observa una señal adicional denominada "Estado[2]" **x** del sistema, que es la señal que nos proporcionará información más completa de la planta.

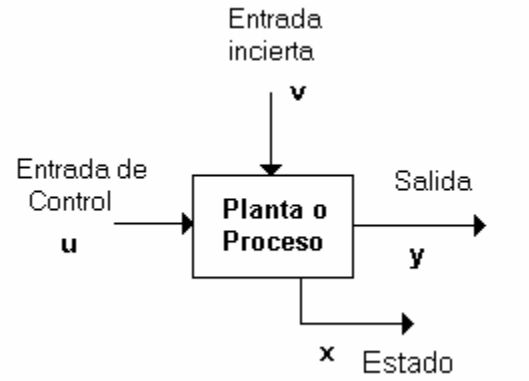

Figura 2.2 Diagrama de un sistema dinámico.

La ecuación matemática desde el punto de vista de la teoría moderna del control se puede expresar mediante la relación de la ecuación (2-2)

$$\frac{dx}{dt} = f(x,u,v) \qquad \text{(2-2)}$$

[2] El estado de un sistema dinámico es un conjunto mínimo de parámetros (variables de estado) que permiten representar de manera única al sistema.

Partiendo de cualquiera de estas representaciones matemáticas, se utiliza el concepto del control retroalimentado que en forma de diagrama de bloques tiene la estructura mostrada en la figura 2.3. Este es el esquema más común para el control automático.

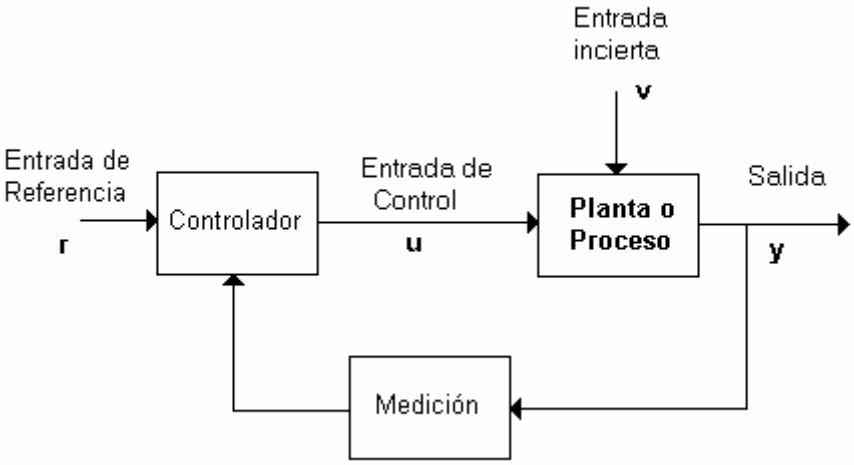

Figura 2.3 Esquema de un sistema retroalimentado de control.

El problema de control se restringe, una vez que ha sido seleccionado el mejor sistema de medición y que también es representado mediante ecuaciones, al diseño del controlador que busca determinar la relación funcional más adecuada para generar la entrada **u** (Entrada de Control) de manera que el modelo del sistema, sujeto a entradas de comando (Entrada de referencia) y posiblemente entradas inciertas, genere una respuesta del estado **x**(t) o una respuesta en la señal de salida y(t) con propiedades específicas o comportamiento aceptable[3].

En algunas aplicaciones, el controlador puede generar la misma señal de comando (Entrada de Referencia según se ilustra en la figura 2.3), **u** = **r**(t). La señal de comando es una señal externa al controlador y es, como su nombre lo expresa, la señal que comandará al sistema de control; esta función de comando **r**(t) debe ser conocida para propósitos de diseño. En otras aplicaciones el controlador podrá ser una función de la señal de comando y también del tiempo, **u** = **u**[**r**(t),t]. Estos dos tipos de relaciones funcionales para el controlador corresponden a los sistemas denominados de lazo abierto que no se esquematizan aquí. La relación funcional para el controlador puede ser una función de la entrada de comando y de la salida del sistema **u** = **u**[**r**(t),**y**]; a este tipo de sistema se le conoce como control en lazo cerrado o control retroalimentado. En este esquema, la entrada incierta **v** generalmente es desconocida e independiente del controlador. Este tipo de sistema de control es el que se usará en este trabajo.

Los esquemas de control retroalimentado se pueden clasificar en dos. Aquellos esquemas que retroalimentan propiamente la señal de salida, como el mostrado en la figura 2.3 y que se denomina como retroalimentación de la señal de salida y es el esquema utilizado en la teoría de control clásico. La otra forma es retroalimentar el estado, conocido el esquema como retroalimentación de las variables de estado y cuya función se puede expresar como **u** = **u**[**r**(t),**x**]. Al usar estos esquemas de retroalimentación de las variables de estado, se pueden

[3] Este comportamiento "aceptable" tiene que ver con las especificaciones de diseño que se puedan establecer para el sistema de control.

obtener también dos formas diferentes de hacerlo, una denominada directa en donde las variables de estado se retroalimentan directamente al controlador y cuyo diagrama a bloques se muestra en la figura 2.4. Es indispensable en este esquema que se retroalimenten todas las variables de estado.

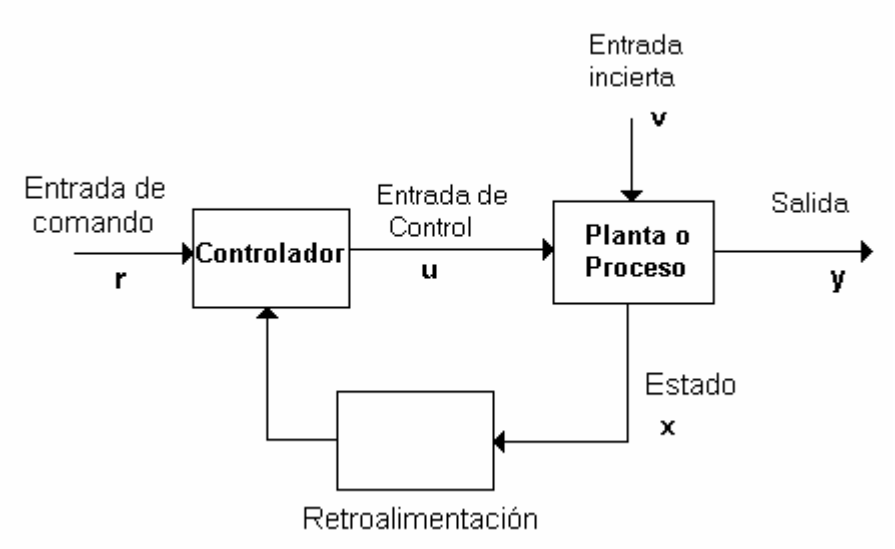

Figura 2.4 Retroalimentación directa de estado.

Para el caso en que no se pueda tener acceso a los estados del sistema y sólo se tenga un conjunto de variables de salida, entonces se usa el esquema de la figura 2.5

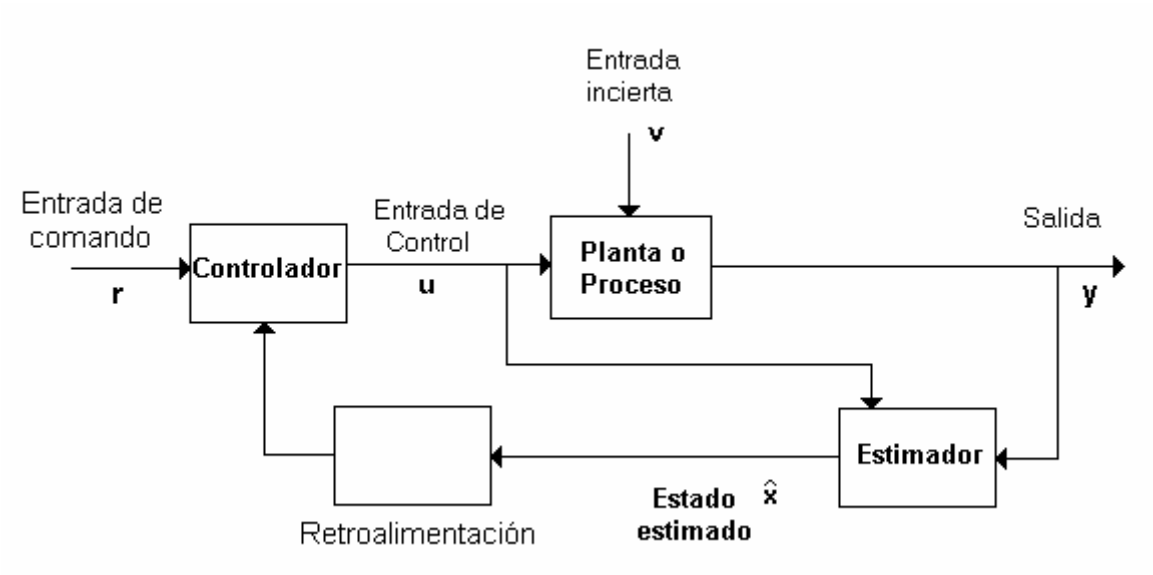

Figura 2.5 Retroalimentación de estado usando un estimador.

Para poder aplicar este esquema es necesario diseñar un elemento denominado estimador[4] [1] que, alimentado por la información de la salida y la entrada disponibles del sistema, pueda

[4] Un estimador o un observador es un sistema alimentado por las salidas y entradas disponibles de un sistema original con objeto de reconstruir el vector de estado de ese sistema.

reconstruir el estado **x** del sistema. En general un estimador requerirá como entradas tanto la salida del sistema como la entrada de control **u**, como se aprecia en el esquema de la figura 2.5

El problema fundamental de control está asociado con la transferencia del estado del sistema **x**(t) a un conjunto destino determinado en el espacio de estados o cerca de ese destino. (Cambiar o moverse de un estado a otro). Si el conjunto destino es un conjunto constante (usualmente un punto fijo) en el espacio de estados, el problema de control se conoce como problema de control de regulación. Si el conjunto destino es variante con el tiempo, especificado mediante una entrada de comando que depende del tiempo, entonces el problema se conoce como problema de control de seguimiento. El problema que se trata en este trabajo es un problema de control de regulación.

Asociadas a estas características, aparecen otros dos conceptos fundamentales en la teoría del control y que son abordados normalmente por la teoría de control moderna, estos conceptos son la controlabilidad y la observabilidad. El tema de controlabilidad está relacionado con una entrada de control que se supone como existente y que llevará al estado del sistema a un destino determinado. Un requerimiento primordial en el diseño de un sistema de control automático es contar con sistemas controlables. El objetivo es que la salida **y**(t) se aproxime o siga a la entrada de comando **r**(t). La tarea del controlador es llevar al sistema a la condición de operación deseada. Si el sistema es controlable, el objetivo de diseño corresponde a hacer que el sistema global sea estable alrededor del punto de operación. El tema de observabilidad se enfoca en el problema de determinar el estado **x**(t) a partir de las mediciones **y**(t). Frecuentemente las mediciones contienen solamente algunos de los estados. Se dice que un sistema es observable si es posible inferir el estado inicial x(0), a partir de un conjunto de mediciones de la salida y(t) sobre un intervalo finito de tiempo [0,T].

Todos estos conceptos se tratan en la teoría clásica y moderna del control y normalmente se realizan con formalismo matemático riguroso. Mientras más complejo es el sistema, el procedimiento se vuelve también muy complejo. También se han establecido procedimientos empíricos para determinar las características o parámetros del controlador sin pasar por toda la herramienta matemática, sin embargo los resultados no siempre son los mejores.

Cuando se desea obtener comportamientos realmente satisfactorios y apegados lo más fielmente posible a las especificaciones de diseño, entonces las técnicas de control óptimo resultan ser las más convenientes, estas técnicas buscan mediante procedimientos matemáticos de optimización, generar los mejores parámetros de control, considerando algunos criterios de optimización. Aparece luego el concepto de control adaptivo, que de alguna manera será brevemente mencionado más adelante y que se puede considerar como inicio de las técnicas de control inteligente. En el caso en donde el sistema contenga múltiples entrada y salidas, aparece la teoría de control multivariable. Y para sistemas en donde aparecen fenómenos aleatorios se utiliza la teoría de control estocástico.

2.3 Esquemas de control inteligente. [2]

El incremento de las demandas tecnológicas en nuestros tiempos, ha generado sistemas muy complejos que requieren controladores altamente sofisticados para asegurar alto desempeño dentro de condiciones adversas. Estas y otras condiciones de control no se pueden cumplir con controladores convencionales, debido principalmente a la falta de conocimiento preciso acerca del proceso que se desea controlar. La adquisición de conocimiento adecuado del sistema en ocasiones es problemática o impráctica debido a la complejidad del sistema y al hecho de que la estructura y los parámetros en muchos sistemas cambian de manera significativa e impredecible con el tiempo. Es bajo estas condiciones en donde se utilizan las técnicas del control inteligente.

El control inteligente es una generalización del concepto de control y se puede ver como un campo dentro de la disciplina del control. El control inteligente [3] es la disciplina donde los métodos de control se desarrollan para emular algunas características importantes del ser humano. Estas características incluyen adaptación y aprendizaje, planeación bajo gran incertidumbre y el trabajo con gran cantidad de datos.

Las metodologías de control inteligente están siendo aplicadas a la robótica, las comunicaciones, la manufactura, el control de tráfico, por mencionar algunas pocas. Las áreas donde se está realizando trabajo alrededor del control inteligente son: redes neuronales, control difuso, algoritmos genéticos, sistemas de planeación, sistemas expertos y sistemas híbridos (combinación de más de una de las técnicas anteriores).

Un sistema de control inteligente debe ser autónomo; esto significa que tiene el poder de autogobernarse. Existen varios grados de autonomía: un controlador totalmente autónomo debería tener la habilidad de aún reparar su propio hardware si uno de sus componentes falla. Un control fijo convencional como los mostrados en el apartado 2.2 se consideran con un bajo grado de autonomía; un control adaptivo[5] convencional tiene un alto grado de autonomía. La autonomía es el objetivo en los sistemas de control complejos y los controladores inteligentes son una manera de lograrlo.

Los sistemas de control convencionales se diseñan usando los modelos matemáticos de sistemas físicos. Se selecciona un modelo matemático que captura el comportamiento de la dinámica de interés y entonces se aplican las técnicas de diseño, tal vez ayudados por sistemas CAD, para diseñar el modelo matemático apropiado del controlador. Luego se realiza el controlador ya sea en hardware o en software en el sistema físico. Este procedimiento puede llevar varias iteraciones hasta lograr el mejor comportamiento. El modelo matemático de la planta deberá ser "bastante simple" para que pueda ser analizado con técnicas matemáticas disponibles, y "bastante exacto" tal que describa los aspectos importantes y relevantes del comportamiento de la planta. Por lo general el comportamiento de la planta se aproxima en las vecindades de un punto de operación para hacer más sencillo el diseño del controlador.

Esto significa que los controladores pueden diseñarse para cumplir las especificaciones alrededor de un punto de operación, donde el modelo lineal es válido. En sistemas de control con alto grado de autonomía necesitamos incrementar significativamente el rango de operación.

[5] El Control adaptivo lo constituye una variedad especial de controladores cuyos parámetros varían con el tiempo. Poseen una rutina para adaptarse a las modificaciones del proceso para seguir controlándolo adecuadamente.

La complejidad del modelo de un sistema dinámico y la demanda creciente de funcionamiento, hacen necesario el uso de controladores más complejos y sofisticados. La forma en cómo se incrementa la complejidad de un controlador se puede describir de la forma siguiente.

En el nivel más bajo. El control retroalimentado determinístico basado en la teoría de control convencional se utiliza para las plantas que pueden ser representadas con modelos lineales más simples que son esencialmente, buenas aproximaciones al comportamiento real. Si se incrementa la complejidad de la planta, los controladores necesitarán estimadores de estado. Si se incrementa la señal de ruido, se necesitan filtros Kalman[6] u otro tipo de filtros. Si se requiere completar una tarea de control en un tiempo mínimo (o energía mínima), se utilizan técnicas de control óptimo. Cuando hay características cuantificables estocásticas en la planta, se usa la teoría de control estocástico. Si hay variaciones significativas en los parámetros de la planta, tal que la teoría del control robusto sea inapropiada, se emplean técnicas de control adaptivo.

Para plantas aún más complejas es necesario usar control de aprendizaje o auto – organizado[7][4]. En el nivel más alto de jerarquía, la complejidad de la planta es tal y las especificaciones son tan demandantes que se usan técnicas de control inteligente. Se cambia a controladores más sofisticados solamente si los más simples no pueden lograr los objetivos buscados. La necesidad de usar control autónomo inteligente se origina de la necesidad por incrementar la habilidad de tomar decisiones autónomas para ejecutar tareas complejas de control.

Aunque el proceso en donde se aplicarán estos controladores, en este trabajo de tesis, no es un proceso muy complejo, tiene algunos elementos no lineales que pueden ser un buen ejercicio para aplicar los esquemas de control inteligente. El objetivo de este trabajo radica esencialmente en comparar el funcionamiento de los tres esquemas y sobre todo, aplicar el esquema de control neurodifuso NEFCON, del cual no se han reportado muchas aplicaciones en procesos reales y sobre todo contribuir al conocimiento de estas técnicas de control para el ITESO.

2.3.1 Control difuso[8]. ("*Fuzzy Control*")[5]

Para el diseño de un controlador difuso, como ya se mencionó anteriormente, no se requiere el modelo analítico completo del sistema dinámico. El resultado de este diseño es un controlador heurístico basado en conocimiento, utilizado para controlar un sistema complejo e indefinido. En este apartado se explicará de manera breve la base necesaria para el diseño de un controlador difuso, así como su estructura fundamental, algunas características especiales a tomar en cuenta y algunas definiciones que serán necesarias para entender los conceptos aquí mencionados. Un controlador difuso es esencialmente un controlador no lineal.

Tomando como referencia el esquema de un control retroalimentado mostrado al principio de este capítulo (ver figura 2.3), el esquema de control difuso tiene ahora la estructura que se muestra en la figura 2.6

[6] Un filtro Kalman es una estructura matemática empleada para estimar los estados de un sistema en presencia de ruido Gaussiano aleatorio.
[7] El aprendizaje auto-organizado o también llamado "aprendizaje no supervisado" es otro tipo de estrategia de aprendizaje. La clave aquí es el aprendizaje sin un "maestro".
[8] El término difuso equivale en español al término "*Fuzzy*" en inglés.

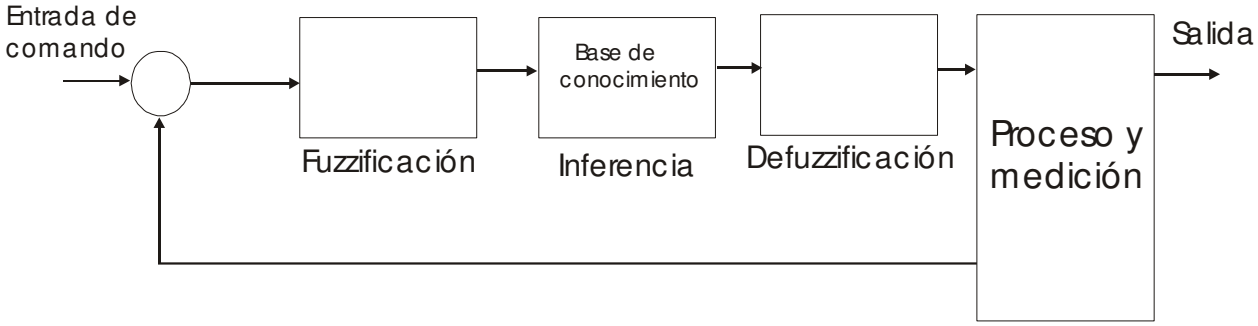

Figura 2.6 Esquema de un controlador difuso insertado en el esquema general de un control retroalimentado.

En la figura 2.6 se aprecian tres nuevos bloques que no aparecen en la figura 2.3, uno etiquetado como Fuzzificación, otro como Defuzzificación y otro como Inferencia. La función involucrada en el primer bloque, la fuzzificación, tiene el objetivo de tomar el valor de la señal leída del proceso (en este esquema es la señal de error) y transformarla en una señal entendible por el sistema difuso. La función Defuzzificación[9] tiene el objetivo de convertir el resultado del bloque etiquetado como inferencia, que es un resultado difuso, a un resultado entendible para el proceso. Finalmente el bloque etiquetado como Inferencia se encarga de realizar todo el razonamiento del sistema de control.

Cuando se diseña un controlador difuso, es necesario determinar las variables de entrada y de salida. Las variables de salida representan el resultado de la operación que realizará el controlador y serán determinadas por el objetivo de control. La decisión de las variables de entrada depende de la situación en particular.

Generalmente, en sistemas de múltiples entradas y salidas (MIMO[10]), es difícil generar reglas de control. Existen aplicaciones exitosas del control difuso en sistemas de una entrada y una salida (SISO[11]). También se han aplicado en sistemas MIMO con buenos resultados [6]. El ejemplo del sistema que se diseña en el capítulo 4 es un sistema con dos variables de entrada y una variable de salida, un sistema MISO.

El diseño de un controlador difuso involucra la construcción de reglas de control. En muchos casos, se pueden obtener estas reglas describiendo las acciones de los operadores del sistema de control en el formato IF-THEN. Sin embargo no es un método genérico para construir reglas de control, se pueden construir reglas IF-THEN no sólo de las acciones de los operadores sino de la respuesta característica del sistema a controlar. Una vez establecidas las reglas IF-THEN se puede realizar la estrategia de control usando razonamiento difuso. La estructura del controlador difuso es en sí misma la estructura del razonamiento difuso.

Se ha mencionado que es necesario determinar las reglas de control para el diseño de controladores difusos, sin embargo, la determinación de estas reglas no es un proceso directo,

[9] En este capítulo no se explicarán los procedimientos utilizados para ello; se da por supuesto que el lector tiene un conocimiento de estos procesos.
[10] MIMO (Multiple Input Multiple Output)
[11] SISO (Single Input, Single Output)

debido a que el procedimiento incluye la identificación de parámetros difusos para los conjuntos difusos. Se puede asumir que los parámetros de los conjuntos difusos representan los parámetros para el controlador. Por lo tanto la sintonización significa ajustar los parámetros para los conjuntos difusos, incluidas las reglas de control. Es posible mejorar el funcionamiento del controlador con la sintonización de tales parámetros. Sin embargo, la sintonización sola de parámetros en ocasiones no puede mejorar el funcionamiento del controlador. Esto es porque las reglas de control fueron construidas inadecuadamente. En este caso, se puede mejorar el funcionamiento revisando las reglas de control y sintonizando los parámetros para las nuevas reglas.

Algunas definiciones importantes para clarificar los conceptos utilizados en la definición de un controlador difuso son las siguientes.

Sea X un conjunto determinado de valores que denominaremos "el universo de discurso"[12]*, formado por todos los números reales.*

Definición de conjunto difuso: *Un conjunto difuso A en X es un subconjunto de X cuya función de membresía $\mu_A(x)$ tiene valores en el intervalo real [0,1].*

Función de membresía: *Cualquier función de la forma A: X $\rightarrow$ [0,1] describe a una función de membresía asociada con un conjunto difuso A.*

Etiqueta: *A estos conjuntos difusos se les asocia un nombre o "etiqueta" que representa un término lingüístico conocido como por ejemplo: el conjunto difuso A puede ser etiquetado con el nombre de "Negativo".*

Partición difusa: *N conjuntos difusos A_1, ... , A_n, que cubren el espectro completo X, es decir, el universo de discurso.*

Las reglas difusas tienen la siguiente estructura:

IF (antecedente(1)) and (antecedente(2)) and …. and (antecedente(n))
THEN (consecuente(1)) … (consecuente(n))

En el control difuso se distinguen también dos tipos de controladores, caracterizados esencialmente por la forma en que se definen los consecuentes, estos tipos de controladores denominados Mamdami y Takagi Sugeno, se definen de la forma siguiente [7]:

Control difuso tipo Mamdami. *Un control difuso tipo Mamdami es aquel control que tiene reglas difusas cuya forma general es la siguiente:*

r_k : **if** x_1 is A_1 **and** … **and** x_n is A_n **then** y_1 is B_1,…,y_m is B_m

Un ejemplo particular de este tipo de control se puede escribir de la forma siguiente:

[12] Lara, Fernando, Curso tutorial de Lógica Difusa, ITESO, 1995 (¿es correcto el año?)

if el error es grande **and** el cambio del errror es pequeño **then** la señal de control es grande

Control difuso tipo Takagi-Sugeno. *Un control difuso de este tipo contiene reglas cuya forma general es la siguiente:*

r_k : **if** x_1 is A_1 **and** … **and** x_n is A_n **then** $y_1 = f_{1,k}(x_1,\ldots,x_N),\ldots,y_m=f_{m,k}(x_1,\ldots,x_N)$

En esta forma, se aprecia que el consecuente de la regla son funciones de las entradas al controlador.

Un controlador difuso tipo Takagi-Sugeno tiene como consecuente una o varias funciones que dependen directamente de las señales que se están leyendo del sistema o de las variables de estado. Un controlador tipo Mamdami tiene como consecuente un valor determinado con anterioridad y no una función. Esta tesis trata de la implementación de controladores difusos tipo Mamdami.

Las funciones de membresía tienen además trayectorias ya definidas. Pueden ser del tipo triangular, trapezoidal, singleton, gausianas. Las usadas en esta tesis serán funciones triangulares y trapezoidales. ¿tengo que fundamentar porqué?

A continuación se describen los pasos para la definición de este controlador.

Procedimiento para diseño.

- Definición de las señales de entrada y salida.

Considérese que al controlador difuso le llegan dos entradas del sistema de control: la señal de error *e(n)* y la señal cambio del error *de(n)*. El universo de discurso para cada una de estas señales está dado por los siguientes intervalos: $0 \leq e(n) \leq 10$ y $0 \leq de(n) \leq 10$. La señal de salida del controlador difuso se llama acción de control *m(n)* y su universo de discurso está definido en el siguiente intervalo $0 \leq m(n) \leq 1$. Para las señales de entrada se definen 3 funciones de membresía para cada una, al igual que para la señal de salida. Las funciones de membresía de la señal de entrada se muestran en la figura 2.7

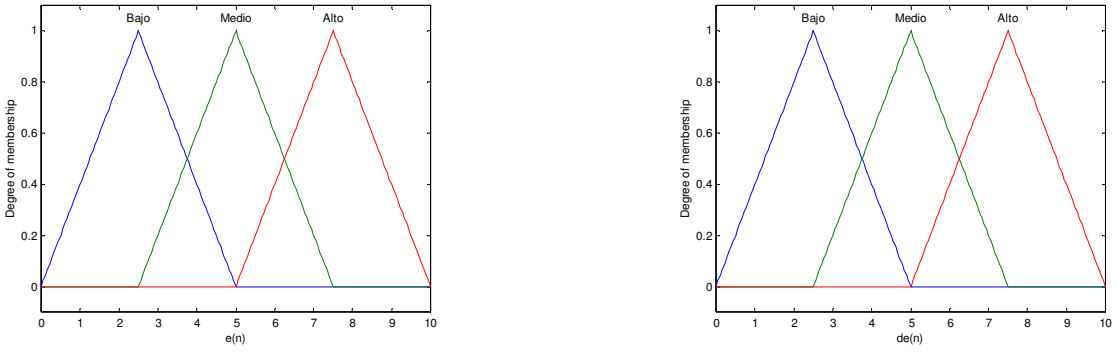

Figura 2.7 Funciones de membresía definidas para las señales de entrada al controlador difuso.

Para cada una de estas funciones de membresía se han definido etiquetas, las cuales se muestran en la parte superior de cada función de la figura 2.7. Para la señal de salida, en muchas de las aplicaciones prácticas se definen conjuntos difusos denominados "crisp" o "nítidos". Estos son conjuntos difusos que tienen un solo valor. Esto se hace para facilitar un poco la aplicación de las reglas difusas y su implementación práctica. En la figura 2.8 se muestran las funciones definidas para la señal de salida.

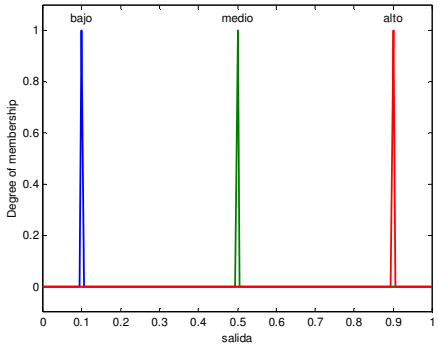

Figura 2.8 Funciones de membresía "crisp" para la salida.

- Definición de las reglas difusas

Con estas funciones de membresía y considerando alguna aplicación del sistema de control, se definen las reglas difusas (a manera de ejemplo) de la siguiente forma:

1. If (error is bajo) and (cambioerror is bajo) then (salida is bajo) (1)
2. If (error is bajo) and (cambioerror is medio) then (salida is medio) (1)
3. If (error is bajo) and (cambioerror is alto) then (salida is alto) (1)
4. If (error is medio) and (cambioerror is bajo) then (salida is bajo) (1)
5. If (error is medio) and (cambioerror is medio) then (salida is medio) (1)
6. If (error is medio) and (cambioerror is alto) then (salida is alto) (1)
7. If (error is alto) and (cambioerror is alto) then (salida is alto) (1)
8. If (error is alto) and (cambioerror is medio) then (salida is alto) (1)
9. If (error is alto) and (cambioerror is bajo) then (salida is alto) (1)

Tal vez habría que definir o explicar cuántas funciones se tienen que determiner. ¿porqué 9 y no más?

- Proceso de fuzzificación, ¿cómo es que se lleva a cabo?

- Proceso de defuzzificación

La defuzzificación es un proceso matemático utilizado para convertir un número difuso en un número real. Este es un paso necesario porque los resultados difusos generados por la inferencia difusa en las reglas difusas deberá ser de alguna manera matemáticamente combinado para generar valores simples en la salida del modelo del control difuso. Al final de cuentas los actuadores son parte de los sistemas de control que aceptan sólo valores reales.

Cada modelo y controlador difuso usa un defuzzificador, que es simplemente una fórmula matemática para realizar este proceso. Se usan diferentes tipos de defuzificadores para diferentes circunstancias. Puesto que la mayoría de los controladores difusos usan funciones "crisp" en su consecuente, la atención se concentrará en este tipo de conjuntos difusos. La fórmula utilizada para esta aplicación se puede describir de la manera siguiente. Suponga que la salida del controlador difuso es z. Suponga que se evalúan N reglas difusas en un controlador tipo Mamdami; producto del método de inferencia se generan N valores de membresía dados como $\mu_1, \ldots, \mu_N$; para N conjuntos difusos de salida. Los conjuntos difusos son diferentes de cero solamente en $z=\beta_1,\ldots, \beta_N$. El defusificador generalizado produce el siguiente resultado:

$$z = \frac{\sum_{k=1}^{N} \mu_k^{\alpha} \cdot \beta_k}{\sum_{k=1}^{N} \mu_k^{\alpha}}$$

donde α es un parámetro de diseño. ¿cómo es esto?

- Otra consideración del tipo de controlador difuso

Si la salida del controlador es *u(t)*, el controlador se llama un controlador tipo posición. Si la salida del controlador es *du/dt*, el controlador se llama tipo incremental ("*speed type*"). En el control difuso, si un controlador genera la señal *u(t)* a partir de las señales *e(t)* y *de/dt* se le denomina controlador PD difuso. Similarmente, si genera la señal *du/dt* a partir de las señales *e(t)* y *de/dt*, se le denomina controlador PI difuso. El formato general de las reglas respectivas de los controladores difusos puede ser descrito como sigue.

- Controlador difuso PI: IF *e(t)* es A y *de/dt* es B THEN *du/dt* es C
- Controlador difuso PD: IF *e(t)* es A y *de/dt* es B THEN *u(t)* es C

Con todo lo anterior, queda entonces definido el controlador difuso. En el capítulo 4 de diseño del controlador difuso específico se muestra la aplicación de los conceptos aquí descritos en el diseño del controlador específico para el sistema de nivel.

2.3.2 Control neuronal. [8]

Algunos autores [9] retrataban a las redes neuronales artificiales hace dos décadas, como una forma de magia negra alternativa a los métodos racionales de la teoría de control. El reciente y considerable interés en control neuronal ha generado un número de propuestas y de esquemas diferentes para el diseño de sistemas de control que usan redes neuronales artificiales como se verá a continuación. Para aplicaciones en las industrias de procesos, lo más relevante tiene que ver con el uso de redes neuronales como identificadores de procesos o como sistemas que optimizan el funcionamiento de un controlador. En esta sección se describirán brevemente algunas de las propuestas de las redes neuronales artificiales usadas como controladores y que son llamados neurocontroladores.

El esquema más básico de una neurona artificial se muestra en la figura 2.9.

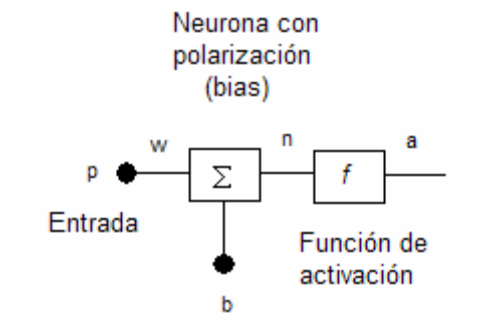

Figura 2.9 Esquema de una neurona artificial.

El valor de la salida de esta neurona está definido por la siguiente ecuación:

$$a = f(wp + b)$$

p es el valor de la señal de entrada, b es un valor conocido como polarización ("*bias*"), w es el valor conocido como peso y es el que se ajusta en el proceso de entrenamiento y finalmente *f* es una función de activación. Una red neuronal artificial es la interconexión de varias neuronas como la ilustrada en la figura anterior. Una capa de neuronas se muestra en la figura 2.10

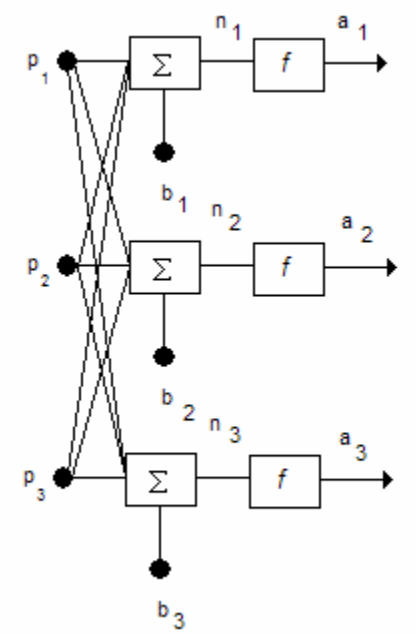

Figura 2.10 Una capa de neuronas.

Las funciones de activación más comúnmente usadas son: lineal, sigmoidal y tangente. En la figura 2.11 se muestran los tres tipos de funciones mencionadas.

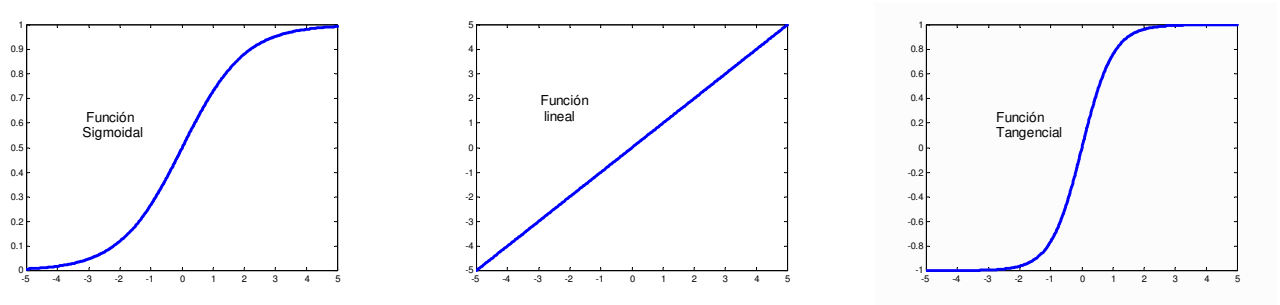

Figura 2.11 Funciones de activación comúnmente usadas.

Es importante hacer notar que una de las habilidades más importantes de la redes neuronales artificiales es su capacidad para aproximar funciones no - lineales. El entrenamiento de una red usando pares de datos (entrada – salida) de una planta o proceso con características no – lineales, se puede considerar como un problema de aproximación de una función no – lineal. Una red con alimentación hacia adelante "*feedforward*"[13] puede aproximar adecuadamente bien una función continua; la red puede tener una o varias capas ocultas y cada neurona puede utilizar, por ejemplo, la función de activación denominada sigmoidal.

En esta sección se mencionan algunos de los esquemas de diseño de un controlador basado en redes neuronales. Estos esquemas pueden ser vistos como un problema de optimización no lineal. Dependiendo de las características de la aplicación, será el algoritmo que se use.

A continuación se presentan 4 clasificaciones para el diseño de un control neuronal. Es probable que existan otras maneras de clasificar estas propuestas, sin embargo algunos de los esquemas tienen que ver con esquemas que se utilizan en la teoría del control adaptivo. Cabe aclarar que estas propuestas son mencionadas en esta parte tomando las referencias correspondientes, no son resultado de un exhaustivo estudio por lo que no serán explicadas a profundidad. Solamente se enfatiza que existen muchos esquemas de control en donde se utilizan redes neuronales. Las propuestas se pueden clasificar de la manera siguiente:

1) Control neuronal indirecto
2) Control neuronal directo
3) Otros esquemas de control neuronal
4) Control adaptivo para sistemas no lineales

En el esquema de control neuronal indirecto, una red neuronal no envía una señal de control directamente al proceso. En lugar de ello, una red neuronal se usa como un indicador de las características del proceso. Este indicador puede ser un modelo del proceso que imita el comportamiento del proceso real o un controlador autosintonizado, que produce las referencias apropiadas para el controlador basadas en el comportamiento del proceso. En esta categoría, las propuestas del control neuronal pueden ser distinguidas como sigue:

a) Control basado en un modelo de proceso generado con una red neuronal

[13] Una red "*feedforward*" es una red construida de tal forma que las salidas de las neuronas siempre son a capas subsecuentes y nunca a una capa anterior o a sí misma.

b) Control basado en un modelo inverso generado con una red neuronal
c) Desarrollo de una red neuronal autosintonizada.

En el esquema de control neuronal directo, una red neuronal se emplea como un control retroalimentado, y envía señales de control directamente al proceso. Dependiendo del concepto de diseño, la propuesta del control neuronal puede ser categorizada como:

a) Modelo de un controlador.
b) Diseño de un control neuronal libre de modelo (sin modelo).
c) Diseño de un control neuronal basado en un modelo
d) Diseño de un control neuronal basado en un modelo robusto.

En la clasificación de otros esquemas de control que usan redes neuronales se pueden considerar los siguientes:

a) Control con modelo de referencia.
b) Control con modelo interno.
c) Control predictivo.
d) Control lineal adaptivo
e) Linealización por retroalimentación.

A pesar de estas clasificaciones, un marco unificador para el control neuronal es ver el entrenamiento de una red neuronal como un problema de optimización no lineal en el cual se trata de encontrar una representación óptima de la red neuronal que minimiza una función objetivo J sobre el espacio de los pesos de la red ω (ver ecuación 2-5). Aquí, NN[14] indica que la formulación del problema de optimización involucra a la red neuronal. El rol que juega la red neuronal en la función objetivo es entonces una clave para distinguir las diferentes propuestas del diseño del control neuronal. En esta sección, la apariencia/formulación de la ecuación (2-5) tomará varias formas dependiendo de cada uno de los esquemas anteriormente mencionados, sobre todo los relacionados con las dos primeras grandes clasificaciones.

$$\text{NN: } \min_{\omega} J(\omega) \tag{2-5}$$

A continuación se describirán brevemente las propuestas de control neuronal enunciadas anteriormente.

1) Control Neuronal Indirecto [7].
    a) Control neuronal basado en un modelo neuronal.

Modelo neuronal completo. (Identificación de sistemas)

La aplicación más popular en un sistema de control de una red neuronal es usarla para modelar un proceso (SISO). Esta estructura también se conoce como "*feedforward model*", modelo hacia

[14] NN: Neural Network

adelante o modelo directo. Esta propuesta es un aprendizaje supervisado manejado por datos; la red neuronal intenta emular un proceso existente a partir de suministrarle los datos tanto de entrada como de salida del proceso (Figura 2.12).

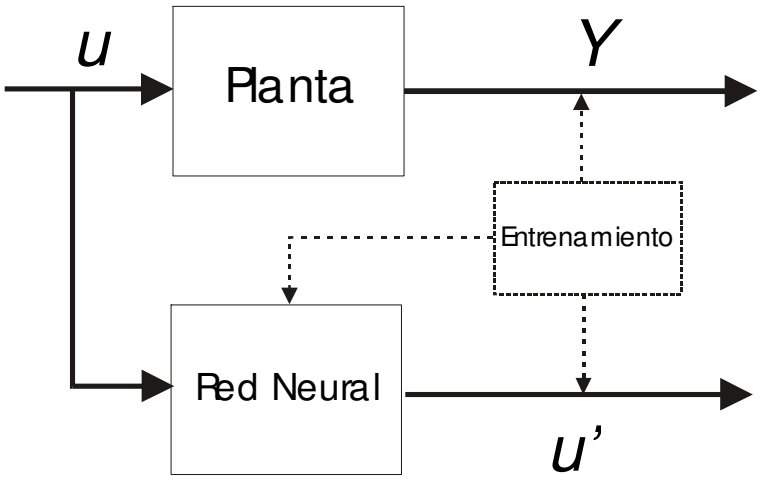

Figura 2.12. La red neuronal puede ser usada como el modelo del proceso como caja negra.

Esta propuesta genera una función de transferencia no lineal discreta en tiempo. A pesar de la estructura del modelo y de la estrategia de control, el diseño del control neuronal en este caso, puede ser descrito matemáticamente como se muestra en la ecuación 2-6:

$$\text{NN: } \min_{\omega} F\{y_p - y_n(\omega,...)\}, \tag{2-6}$$

donde y_p significa la salida de la planta o proceso, y_n la salida de la red neuronal y ω los pesos de la red neuronal. Aquí $F\{\ldots\}$ es una función que mide el funcionamiento del proceso de optimización, usualmente es la integral o la suma de los errores de predicción entre y_p y y_n. Un ejemplo típico de la ecuación 2-6 se ilustra con la ecuación 2-7:

$$NN: \min_{\omega} \sum_{t} \left| y_p(t) - y_n(t) \right|^2 ; y_n(t) = N(\omega...) \tag{2-7}$$

Una vez desarrollado el modelo del proceso utilizando la red neuronal para su identificación, puede ser implementado un diseño del controlador basado en este modelo. En este ejercicio se parte del supuesto siguiente: el modelo puede ser identificado con este tipo de estructura.

b) Red neuronal del modelo inverso.

Una red neuronal puede ser entrenada para generar el modelo inverso de la planta. La entrada a la red es la salida del proceso, y la salida de la red es la correspondiente entrada del proceso. Ver figura 2.13. En general el problema de optimización puede verse de la manera siguiente:

$$NN : \min_{w} F\left\{u^{*}_{p-1} - u_{n}(w,...)\right\},$$

(2-8)

donde u_{p-1}* es la entrada del proceso. Típicamente, el modelo inverso es un modelo estado estable/estático, que puede ser usado para control pre-alimentado. Dada una referencia deseada y* para el proceso, la señal apropiada de control u* para estado estable para este punto de referencia puede conocerse inmediatamente como:

$$u^{*} = N(y^{*},\ldots)$$

(2-9)

Obviamente un modelo inverso existe solamente cuando el proceso se comporta monotónicamente como una función directa en estado estable. Si no, esta propuesta no es aplicable.

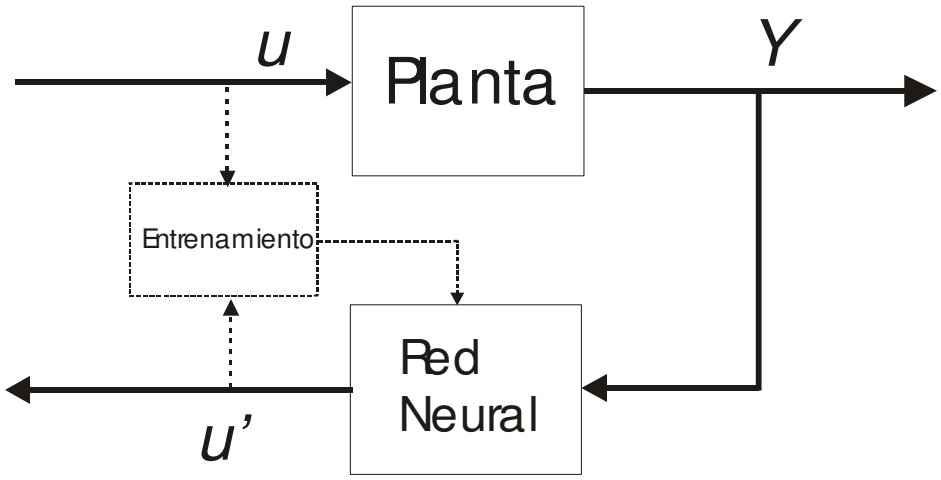

Figura 2.13. Una red neuronal en un modelo inverso.

En principio, una red neuronal de modelo inverso puede aprender la dinámica inversa bajo algunas restricciones (se requieren fase mínima[15] y causalidad). Entonces, el modelo inverso se acomoda de una forma similar a un controlador MPC[16]. En la práctica, especialmente para modelos dinámicos discretos, el modelo inverso puede no ser capaz de aprender la dinámica inversa. En muchos casos, un proceso inverso es de hecho no causal aún si el proceso se comporta monotónicamente como se mencionó anteriormente. La no causalidad de un proceso inverso puede resultar de retardos de transporte (tiempos muerto) o discretización de un proceso continuo en un sistema de datos muestreados. Aún si existe un modelo inverso, el uso de éste como controlador por retroalimentación no generará estrictamente un adecuado sistema de control.

[15] Un sistema de control se dice que es de fase mínima si no tiene ni polos ni ceros en el semiplano derecho "s". El término fase mínima se refiere al comportamiento de la fase en los diagramas de frecuencia. Un sistema de fase mínima siempre tiene un valor mínimo en su gráfica del ángulo con respecto a la frecuencia.
[16] MPC: Model Predictive Control

c) Autosintonización con una red neuronal.

Como en el caso previo donde la red neuronal puede ser usada para estimar los parámetros de un modelo conocido, también pueden ser usadas para estimar los parámetros de sintonización de un controlador cuya estructura se conoce a priori. El estimador de los parámetros de sintonización de un controlador se conoce como auto-sintonizador. El problema de optimización en este caso puede ser formulado como sigue:

$$NN : \min_{w} F\{\eta^* - \eta_n(w,\ldots)\}, \tag{2-10}$$

donde η* denota los parámetros del controlador como objetivos y η_n como los valores que predice la red neuronal. La entrada a la red puede abarcar datos de proceso muestreados o características extraídos de ella. Sin embargo, estos parámetros η no pueden ser únicamente determinados de las características del proceso. Ellos también dependen de las características deseadas del sistema de control en lazo cerrado. Usualmente, los parámetros del controlador η* son soluciones de la siguiente optimización en lazo cerrado:

$$\min_{w} F'\{y^* - y_{p/m}(u,\ldots)\}, \qquad u = C(\eta,\ldots) \tag{2-11}$$

En este caso C es un controlador con una estructura conocida. Aquí, $y_{p/m}$ denota que puede ser empleado un modelo o un proceso en el lazo cerrado de control, con el objeto de encontrar el controlador objetivo C.

De hecho, la atracción de esta propuesta consiste en que la red puede ser entrenada en simulación. El entrenamiento sobre datos del proceso actual no es necesario. Para autosintonía en lazo abierto, una simulación en lazo abierto es suficiente; de otra manera, se necesita una simulación en lazo cerrado (figura 2.14). El entrenamiento deberá conducirse sobre un espacio del modelo del proceso. Idealmente este espacio, debe cubrir el proceso que será encontrado durante la operación.

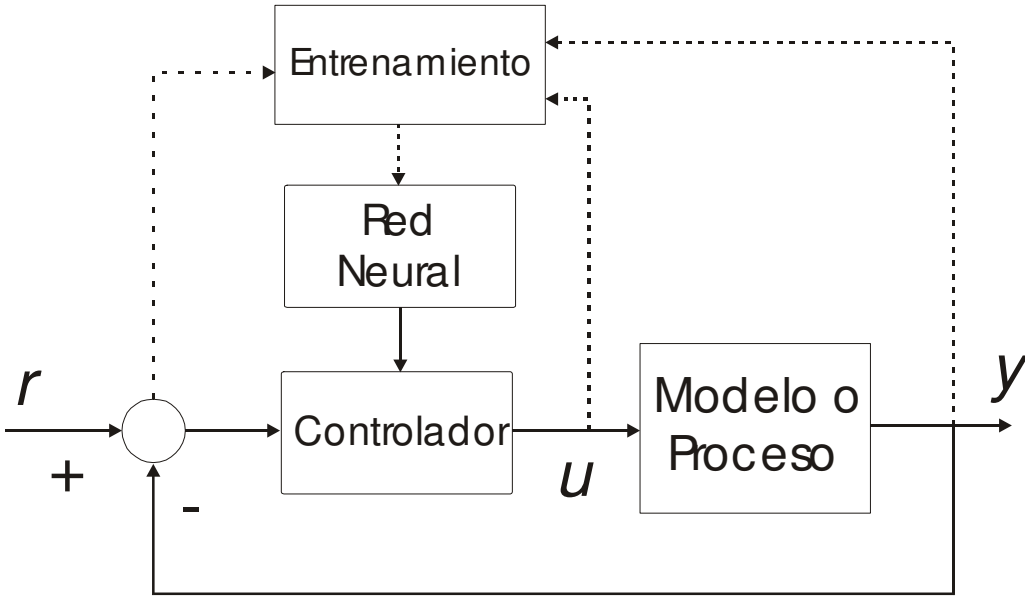

Figura 2.14. Red neuronal como autosintonizador en lazo cerrado.

La mayoría del trabajo reportado en autosintonía está dirigido a los controladores PID puesto que son los más usados en las estructuras de control en la práctica. Los valores apropiados para las ganancias del PID (K_p/ganancia proporcional, K_i (T_i)/ tiempo de reset, K_d (T_d)/ tiempo derivativo) son esenciales si el sistema de lazo cerrado está diseñado para operar de cierta manera. La sintonización de un PID es aún un proceso manual muy largo, en ocasiones basado en procedimientos heurísticos de hace medio siglo. Algunos autosintonizadores están disponibles comercialmente, pero son aún necesarias mejoras. Para el diseño de autosintonizadores basados en redes neuronales, son suficientes modelos lineales de bajo orden para la mayoría de las aplicaciones con PID.

2) Control neuronal directo.
    a) Modelado del controlador

Entre los cuatro esquemas de control neuronal directo, el desarrollo más simple para usar una red neuronal es modelar un controlador ya existente. (Figura 2.15)

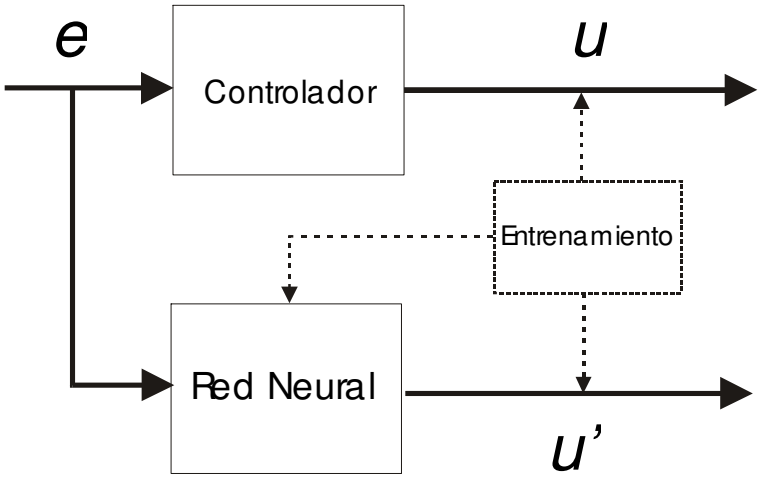

Figura 2.15. Red neuronal para modelar un controlador ya existente.

La entrada al controlador existente es la entrada de entrenamiento a la red y la salida del controlador sirve como el objetivo ("*target*"). En efecto, esta propuesta es similar a la propuesta discutida en una sección anterior, excepto que el objetivo no es el proceso sino el controlador. De cualquier manera, el diseño del control neuronal puede formularse como sigue:

$$NN : \min_{\omega} F\{u_c * \ -u_n(\omega,...)\},$$

(2-12)

donde u_c* es la salida del controlador existente C*. Usualmente, el controlador existente C* puede ser un operador humano o puede obtenerse mediante:

$$\min_{c} F'\{y * -y_{p/m}(u,...)\}, u = C(...)$$

(2-13)

Como modelo de proceso, un controlador generalmente es un sistema dinámico y casi siempre contiene integradores y diferenciadores. Si se usa una red alimentada hacia delante ("*feedforward*") para modelar el controlador existente, deberá suministrarse explícitamente a la red información dinámica, como la generada por integradores, derivadores y hasta señales de retardo. Por ejemplo, para modelar un controlador PID, una red neuronal algebraica necesita no sólo el error instantáneo entre la referencia ("*set point*") y la salida del proceso, sino también las derivadas y las integrales del error. Alternativamente, se puede entrenar a la red neuronal con una serie de aquellos errores y/o salidas del controlador en intervalos de tiempo anteriores. La propuesta posterior es similar al desarrollo de un modelo de proceso tipo ARX ("*auto regressive with exogenous inputs*"), excepto que las entradas y las salidas del proceso se reemplazan con los errores de retroalimentación y las salidas del controlador.

b) Control neuronal libre de modelo.

En ausencia de un controlador, algunos desarrolladores se han inspirado en la forma en que un operador humano aprende a "controlar y operar" un proceso con poco o nada de conocimiento detallado de la dinámica del proceso. Así han intentado diseñar controladores que por adaptación y por aprendizaje puedan resolver problemas difíciles de control en ausencia de un modelo del proceso y de un esfuerzo humano de diseño. En general, este tipo de diseño puede establecerse de la forma siguiente:

$$NN : \min_{\omega} F\{y^* - y_p(u,...)\}, u = N(\omega,...), \tag{2-14}$$

donde y_p es la salida de la planta. La característica clave de esta propuesta de control adaptivo directo es que el modelo del proceso no se conoce o no está explícitamente desarrollado durante el proceso de diseño del controlador.

Este problema de diseño del controlador es referido algunas veces como "aprendizaje por reforzamiento". Sin embargo, en este capítulo se decidió referirse a esta clase de diseño como "diseño de control neuronal libre de modelo", pues resulta más apropiado. En la figura 2.16 se muestra una representación típica de esta clase de diseño de controlador.

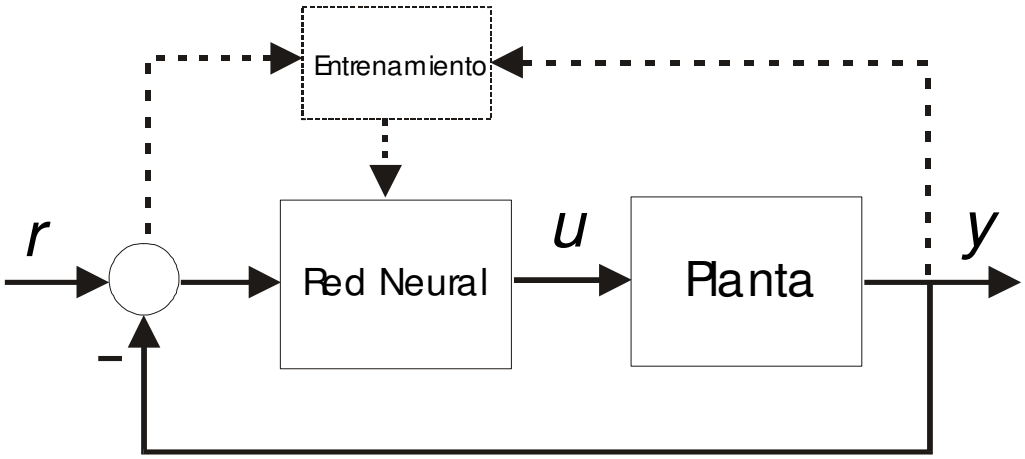

Figura 2.16. Concepto de diseño de un controlador neuronal libre de modelo.

El primer trabajo en esta área fue el algoritmo "crítico adaptivo" propuesto por Barto et all [10]. Tal algoritmo puede verse como una versión aproximada de programación dinámica. En el diseño de esta clase de control, la poca y limitada información se adopta como una indicación del criterio de funcionamiento. El problema del péndulo invertido ha llegado a ser una plataforma popular para experimentos de conceptos de diseño de controladores libres de modelo.

Además de la importancia histórica e interés intuitivo, el control neuronal adaptivo libre de modelo no es apropiado para la mayoría de las aplicaciones del mundo real. La planta normalmente se encuentra fuera de control durante el proceso de aprendizaje, y pocos procesos pueden tolerar un gran número de fallas necesarias para adaptar el controlador.

c) Control neuronal basado en un modelo.

Desde una perspectiva práctica, se preferiría que las fallas del proceso sucedieran en un medio ambiente simulado (con un modelo) más que en una planta real, ya que se pueden permitir fallas desastrosas sin que haya pérdidas sustanciales. En oposición al caso anterior, este diseño de control neuronal se denomina "control basado en un modelo". El problema se plantea de forma similar al anterior como:

$$NN : \min_{\omega} F\{y^* - y_m(u,...)\}, u = N(\omega,...), \tag{2-15}$$

Aquí, y_p de la ecuación (2-14) se reemplaza por y_m (la salida del modelo simulado). En este caso, se requiere el conocimiento del proceso de interés. Como se puede ver en la figura 2.17, un modelo reemplaza a la planta/proceso del sistema de control.

Si el modelo del proceso no está disponible, se puede primero entrenar a una segunda red neuronal para que modele la dinámica de la planta. En el curso del modelado de la planta, la planta debe operarse "normalmente" en lugar de dejarla fuera de control. Una vez terminada la fase de modelado, el modelo obtenido puede ser usado para el diseño del controlador. Si el modelo de la planta está disponible, se puede desarrollar un controlador neuronal en un ambiente de simulación en una computadora. Una vez diseñado, se puede instalar en el sistema de control real. De hecho, estos modelos no sólo han probado su efectividad en diferentes estudios, sino que ya han producido notables beneficios económicos. Estos modelos pueden ser usados tanto para un diseño fuera de línea como adaptación para sistemas en línea.

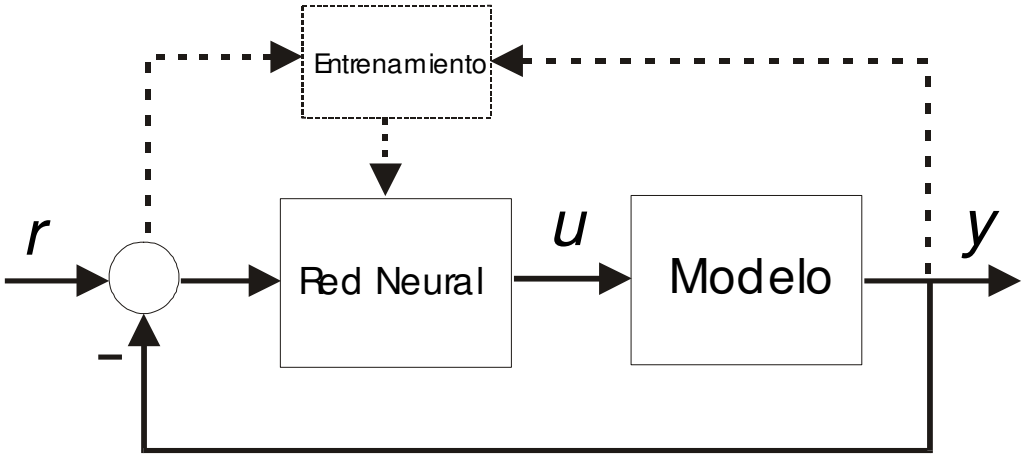

Figura 2.17. La planta se reemplaza por un modelo.

Sin embargo, la calidad del control obtenido con esta aproximación depende crucialmente de la calidad del modelo del proceso. Si un modelo no es bastante exacto, el neurocontrolador entrenado es improbable que maneje satisfactoriamente el proceso real. Sin un componente adaptivo en línea, este neurocontrolador no podrá generar buenos resultados en la planta. Un controlador que es altamente optimizado para un proceso específico, no se puede esperar que tolere desviaciones del proceso nominal con ligereza.

d) Control neuronal basado en un modelo robusto

La propuesta de neurocontrolador discutida anteriormente aún comparte un defecto común: una red neuronal debe ser entrenada para cada nueva aplicación. Es necesario el re-entrenamiento de la red aún a pequeños cambios en los criterios de control, tales como cambios en el peso relativo de la energía del control y respuesta al seguimiento, o si el controlador es aplicado a diferentes procesos muy similares. Para poder evitar tales inconvenientes, el concepto de robustez cae naturalmente hacia el diseño del controlador. En un diseño de un control neuronal basado en un modelo robusto, se consideran una familia de modelos de proceso en lugar de un solo modelo nominal (Figura 2.18).

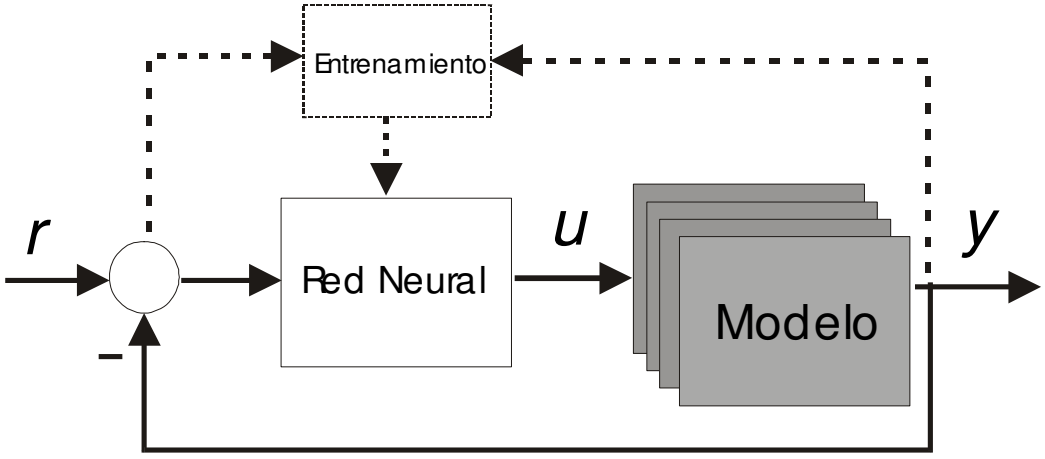

Figura 2.18. Concepto de diseño en control robusto. El controlador no se diseña exclusivamente para un proceso nominal, sino para una familia de procesos.

En ocasiones tal familia de procesos es especificada por un rango de modelos de ruido o por un rango de parámetros del proceso. El diseño de control robusto puede formularse como sigue:

$$NN : \min_{\omega} F\{y^* - y_{m_i}(u,...)\}, u = N(\omega,...), \forall m_i \in M \tag{2-16}$$

donde m_i es el i-ésimo miembro de la familia M del modelo. Idealmente, el proceso real a ser controlado debe pertenecer a esta familia, de tal forma que el controlador sea robusto no solamente para el modelo sino para el proceso real.

Se distinguen comúnmente dos aspectos para la robustez. La estabilidad robusta se refiere a un sistema de control que es estable (cualitativamente) sobre toda la familia de procesos, mientras que el funcionamiento robusto se refiere a (cuantitativamente) los criterios de funcionamiento satisfechos sobre toda la familia. Por lo general hay que sacrificar algo para conseguir la robustez. Al optimizar un controlador basado en una red neuronal sobre un modelo de proceso fijo (y exacto), se puede lograr un alto funcionamiento en tanto que el proceso permanezca invariante, sin embargo, será muy sensible a los cambios de modelo. Un procedimiento de diseño robusto, de otra manera, no alcanza el mismo nivel de funcionamiento nominal pero será

menos sensitivo a los corrimientos del proceso, perturbaciones y otras fuentes no modeladas del proceso.

3) Otros esquemas de control neuronal [11]

a) Control con modelo de referencia. [12]

En este esquema, el funcionamiento deseado del sistema de lazo cerrado se especifica a través de un modelo estable de referencia. El sistema de control intenta hacer que la salida de la planta *y(t)* coincida con la salida del modelo de referencia $y_m(t)$ de una forma asintótica, esto es:

$$\lim_{t\to\infty}\|y_m(t)-y(t)\|\le\varepsilon$$

Para alguna constante especificada $\varepsilon \geq 0$. Este esquema para sistemas no lineales se muestra en la figura 2.19. En esta estructura, el error definido se utiliza como señal de entrenamiento de la red que actúa como controlador.

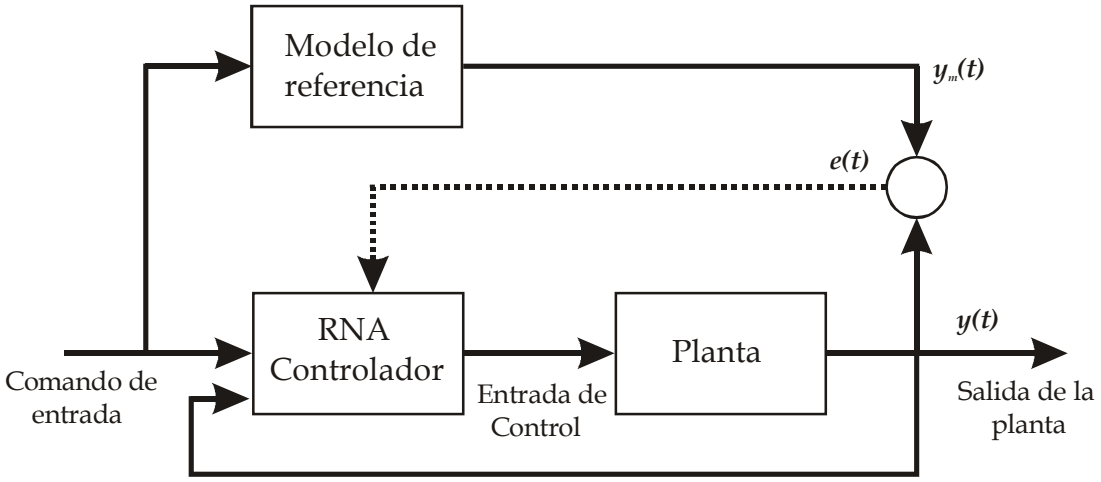

Figura 2.19 Estructura del control con modelo de referencia.

b) Control con modelo interno.

En esta estructura se usa un modelo directo y uno inverso, como elementos dentro del lazo de retroalimentación. Este esquema ha sido ampliamente estudiado y aplicado en sistemas de control de procesos [13]. En este esquema se coloca un modelo del sistema en paralelo con el sistema real. La diferencia entre las salidas del sistema y el modelo se usa para propósitos de retroalimentación, ver figura 2.20. Para el modelo del sistema se utiliza una red neuronal y para el controlador otra red obtenida con el método del modelo inverso.

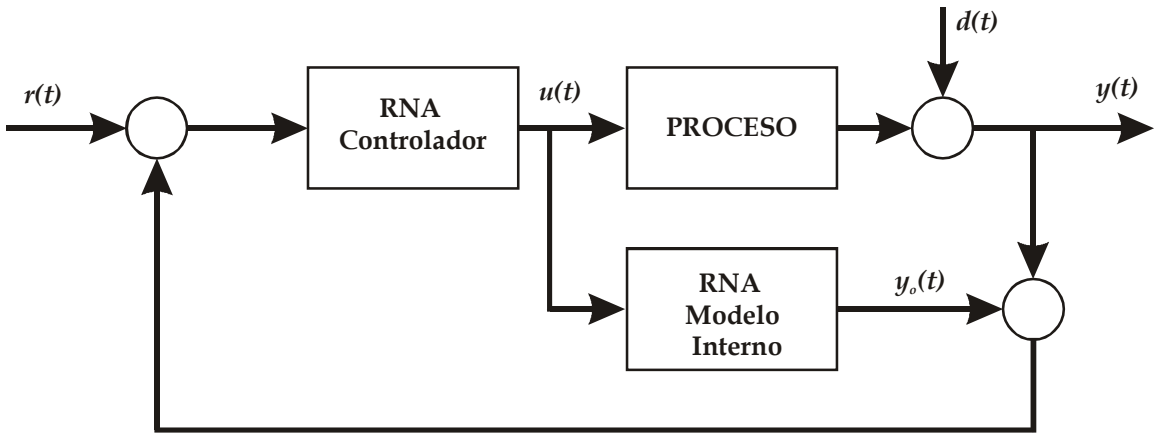

Figura 2.20 Estructura para el control con modelo interno.

c) Linealización por retroalimentación.

En el campo del control no lineal, la linealización por retroalimentación es un principio sobre el que se ha puesto mucha atención. La aplicación se restringe a cierta clase de sistemas, sin embargo esto no es muy común en la práctica. La ventaja de la linealización por retroalimentación es que el diseño puede ser usado de manera genérica, en el sentido de que el mismo principio puede ser usado para todos los sistemas del mismo tipo. Además se han desarrollado algunas extensiones que toman en cuenta posibles inexactitudes del modelo. El esquema general en donde se aplican las redes neuronales se muestra en la figura 2.21.

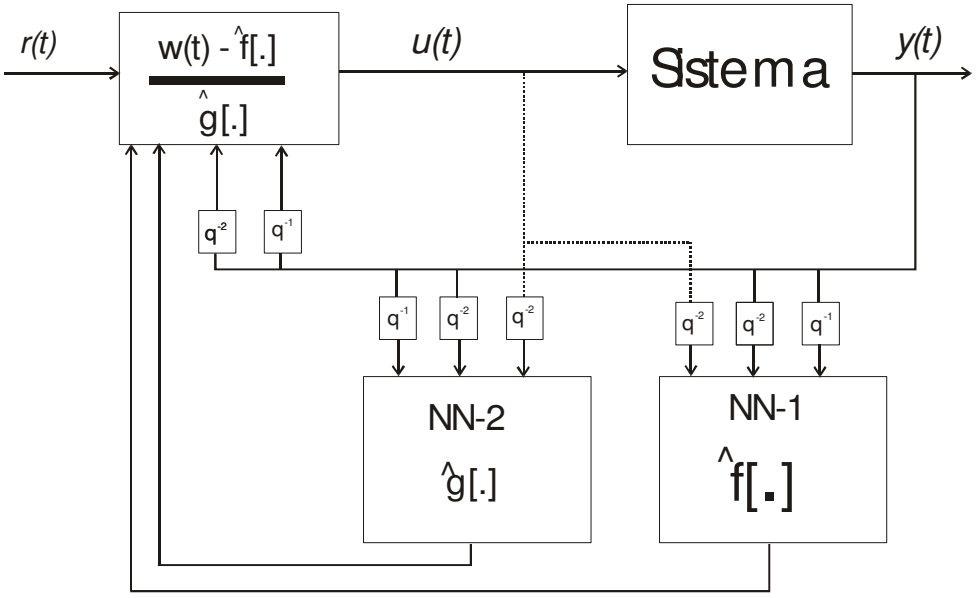

Figura 2.21 Esquema de linealización por retroalimentación.

En el esquema de la figura 2.21 se observan 4 bloques. El sistema que se desea controlar, el controlador que generará la linealización del sistema, y se observan dos redes neuronales que

van a aproximar las funciones f[ ] y $\hat{g}$[ ] que se requieren para la linealización en el controlador. Se tendrán que generar primero las dos redes neuronales. No se ahondará más en este punto.

En la tabla 2.x se presentan a continuación la lista de todos los esquemas anteriormente presentados con una breve característica de su funcionamiento.

Tabla 2.x

Esquema	Característica
1. Control neuronal indirecto	
a) Control basado en un modelo de proceso generado con una red neuronal	Diseño de un control tradicional que ha sido diseñado con un proceso modelado con redes neuronales.
b) control basado en un modelo inverso generado con una red neuronal	El control es el modelo inverso del proceso. Este modelo inverso es una red neuronal.
c) Desarrollo de una red neuronal autosintonizada.	Se ajustan los parámetros de un controlador clásico usando una red neuronal.
2. Control neuronal directo	
a) Modelo de un controlador.	Se copia el comportamiento de un controlador existente usando una red neuronal.
b) Diseño de un control neuronal libre de modelo (sin modelo).	Red neuronal como controlador, entrenado con el proceso real.
c) Diseño de un control neuronal basado en un modelo	Red neuronal diseñada primero con un modelo de un proceso.
d) Diseño de un control neuronal basado en un modelo robusto.	Red neuronal diseñada primero con un modelo de proceso y algunas variaciones del mismo.
3. Otros esquemas de control neuronal	
a) Control con modelo de referencia.	Red neuronal como controlador, pero teniendo un modelo como elemento de referencia.
b) Control con modelo interno.	Utiliza dos redes neurales, una como controlador y la otra como modelo del proceso.
c) Control predictivo.	Se usan dos redes neuronales una de las cuales es un modelo predictivo de funcionamiento.
d) Control lineal adoptivo.	¿? No entiendo bien, releer. Parecido al 2b).
e) Linealización por retroalimentación	Se usan dos redes neuronales para linealizar el comportamiento del controlador.

Estos son sólo algunos de los esquemas utilizados para el control neuronal. El tema no está agotado y es probable que esto pueda generar más trabajos de aplicación de estos esquemas o algunos otros que puedan surgir. Sería muy interesante generar aplicaciones de cada uno de ellos en un proceso real.

Todos estos esquemas son complejos para aplicar el control. Para el proceso en cuestión puede ser muy simple el control. Con una neurona se puede hacer. ¿será válido comparar el control así de simple? Habrá que dar los elementos para que de alguna manera se puedan luego implementar algunos de estos esquemas de control neural. (¿se puede aplicar los esquemas de optimización que propone Ernesto como una variante?)

2.3.3 Control Neurodifuso

En las secciones anteriores se ha descrito la forma en que se usan tanto las redes neuronales como los sistemas difusos para el control de procesos. El paso siguiente es describir ahora cómo utilizando estas dos técnicas se puede construir un sistema neurodifuso y los resultados pueden ser usados como un controlador.

Desde un primer punto de vista, un sistema neurodifuso es una combinación de redes neuronales y sistemas difusos de tal forma que la red neuronal o los algoritmos de aprendizaje de la red neuronal se usan para determinar parámetros del sistema difuso. Esto significa que la intención principal de un sistema neurodifuso es crear o mejorar un sistema difuso automáticamente por medio de los métodos o algoritmos de las redes neuronales. De manera inversa, una red neuronal difusa, es una red neuronal que usa métodos difusos para aprender más rápido o para funcionar mejor; en este caso la intención principal es mejorar la red neuronal.

Según el autor Lee [14], la combinación de las redes neuronales y la lógica difusa recoge las ventajas de cada una de ellas. Las redes neuronales proporcionan el modelo de la estructura conexionista (tolerancia a fallas y propiedades de representación distribuida) con habilidades de aprendizaje a los sistemas de lógica difusa; y la lógica difusa proporciona a las redes neuronales una armazón estructural con reglas de representación del conocimiento y razonamiento de alto nivel SI - ENTONCES *("if then rules").*

Las diferentes combinaciones de redes neuronales y lógica difusa [15] descritas por Nauck y Kruse, y también mencionadas por Lee [14], con una clasificación ligeramente diferente pero que al final de cuentas son las mismas, se pueden describir de la forma siguiente:

> ***REDES NEURONALES DIFUSAS "FUZZY"***. En este tipo de combinación se usan métodos difusos para incrementar la capacidad de aprendizaje de una red neuronal. Por ejemplo, se puede hacer variar la razón de aprendizaje ("*learning rate*") usando reglas difusas.
>
> ***SISTEMAS NEURONAL/DIFUSO"FUZZY" CONCURRENTES***. En este tipo de sistemas, una red neuronal y un sistema difuso trabajan juntos en la misma tarea sin influenciarse el uno al otro. Usualmente la red neuronal se usa para preprocesar la entrada o post-procesar la salida del sistema difuso.
>
> ***MODELOS NEURODIFUSOS COOPERATIVOS***. En este tipo de esquemas, una red neuronal se usa para determinar los parámetros de un sistema difuso (reglas, pesos y/o conjuntos difusos). Después de la fase de aprendizaje, el sistema difuso trabaja sin la red neuronal. Estos modelos de sistemas neurodifusos suelen ser de 4 clases:
>
> - Aprende conjuntos difusos fuera de línea.
> - Aprende reglas difusas fuera de línea.
> - Aprende conjuntos difusos en línea.
> - Aprende reglas difusas en línea.

MODELOS NEURODIFUSOS HÍBRIDOS. En este caso, las redes neuronales y los sistemas difusos se combinan para hacer una arquitectura homogénea. El sistema se puede interpretar como una red neuronal especial con parámetros difusos o como un sistema difuso implementado en forma distribuida en paralelo. Como aplicación de estos tipos de modelos existen ya varios algoritmos desarrollados como son: ANFIS, FuNe, ARIC, NEFCLASS, NEFCON, NEFPROX. [15].

Algunos de estos modelos usan las técnicas de aprendizaje por reforzamiento que son especialmente apropiados para tareas de control (ARIC, NEFCON), y otras son de modelos de propósito múltiple (ANFIS, FuNe, NEFCLASS, NEFPROX), que usan aprendizaje supervisado y pueden ser usados para análisis de datos.

Los dos algoritmos híbridos neurodifusos que tienen aplicaciones al control son el GARIC y el NEFCON. Considerando que existe ya trabajo hecho y experiencia con el NEFCON aquí en el ITESO, se ha decidido utilizar este último algoritmo para este proyecto de tesis. A continuación se describirá con cierto detalle el algoritmo NEFCON.

2.3.4 El algoritmo NEFCON

NEFCON, acrónimo del término "*NEuroFuzzyCONtroller*" es un modelo para controladores neurodifusos desarrollado por un grupo de investigadores[17], que está basado en la arquitectura del perceptron difuso genérico[18]. El algoritmo de aprendizaje para NEFCON está basado en la idea del aprendizaje por reforzamiento[19]. El algoritmo de aprendizaje definido para este modelo es capaz de aprender conjuntos difusos así como reglas difusas. La figura 2.22 muestra un sistema NEFCON con dos variables de entrada, una variable de salida y cinco reglas. Las variables de entrada ξ_1 y ξ_2 son las variables de estado del proceso S que se desea controlar. La salida η de NEFCON es la acción de control aplicada al sistema (que es la señal de control etiquetada como **u** en el punto 2.2. Las neuronas de la capa oculta, R_1 ... R_5, representan las reglas difusas. Por ejemplo la regla R_3 está descrita de la siguiente forma:

$$R_3\text{: Si } \xi_1 \text{ es } A_2^{(1)} \text{ y } \xi_2 \text{ es } A_2^{(2)} \text{ entonces } \eta \text{ es } B_2$$

En donde $A_2^{(1)}$, $A_2^{(2)}$ y B_2 son términos lingüísticos representados por los conjuntos difusos $\mu_2^{(1)}$, $\mu_2^{(2)}$ y ν_2.

[17] D. Nauck, R. Kruse y A. Nürnberger.
[18] Ver capítulo 9 de [15] y también: Nauck, Detlef, *A Fuzzy Perceptron as a Generic Model for Neuro-fuzzy Approach*, Proc. Fuzzy-Systeme´94. 2nd GI-Workshop, Munich, Siemens Corporation.
[19] El aprendizaje por reforzamiento tiene que ver con la idea de premiar aquella acción que se quiere reforzar y castigar aquella que no se quiere reforzar.

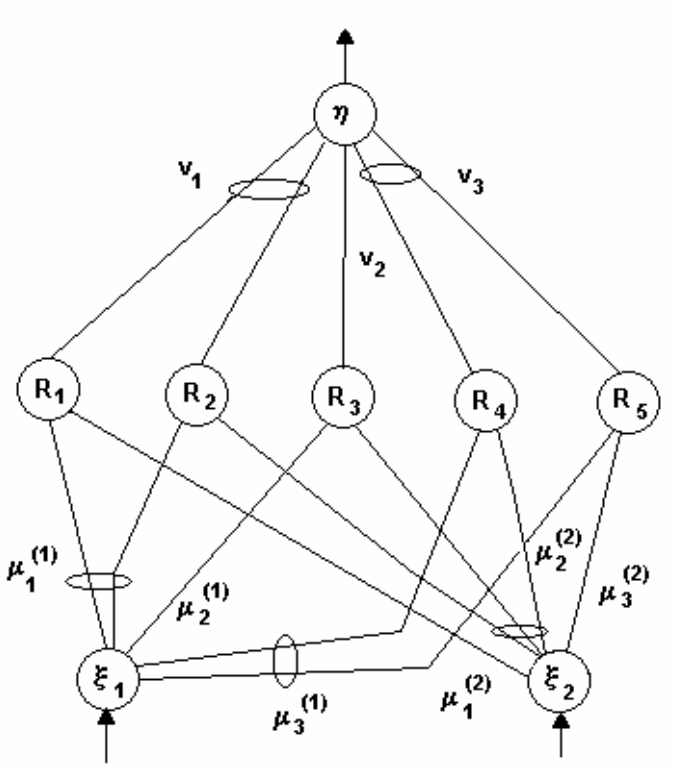

Figura 2.22. Sistema NEFCON con dos variables de entrada y cinco reglas.

Las conexiones entre neuronas en NEFCON en lugar de ser números reales (como sería en una red neuronal normal) son conjuntos difusos, además algunas conexiones siempre tienen el mismo peso (conexiones enlazadas con pesos compartidos). En la figura 2.22 las conexiones de la entrada ξ_1 a la regla R_1 y R_2 comparten el peso $\mu_1^{(1)}$ y las conexiones de R_4 y R_5 a η tienen como peso común ν_3. Esto se ilustra esquemáticamente en la figura por medio de las elipses alrededor de las conexiones. El algoritmo de aprendizaje debe tomar en cuenta esta forma de compartir pesos y realizar modificaciones idénticas en las conexiones enlazadas para asegurar la integridad de la base de reglas, según se establece en su definición original.

La idea de NEFCON es usar tanto conocimiento anterior como sea posible usando reglas conocidas a priori y usando una medición del error basada en conocimiento. Además, el algoritmo es capaz de aprender desde cero en caso de que no haya conocimiento previo. El algoritmo de aprendizaje es simple y rápido de tal forma que NEFCON puede aprender en línea. Es importante también mencionar que un sistema NEFCON es siempre interpretable como un sistema difuso.

Se pueden resumir las características de un sistema NEFCON de la forma siguiente:

- Representa un controlador difuso normal.
- Puede aprender reglas difusas en forma incremental ("*bottom – up*") o decremental ("*top – down*").

- Aprende conjuntos difusos usando una heurística simple.
- El algoritmo de aprendizaje usa mediciones de error lingüísticas que son expresadas por reglas difusas.
- El algoritmo de aprendizaje no distorsiona la semántica del controlador difuso. El sistema siempre es interpretable en términos de reglas difusas.

Si se conocen k reglas difusas para controlar un proceso S, podemos construir un sistema NEFCON como sigue:

- Para cada variable de entrada ξ_i se requiere una neurona con este nombre en la capa de entrada.
- La variable de control se representa por una neurona de salida η.
- Para cada regla difusa R_r hay una neurona en la capa oculta con este nombre. Esta neurona representa una regla.
- Cada neurona de la capa oculta con una regla R_r, se conecta a las neuronas de entrada y a la neurona de salida de acuerdo con la regla difusa que representa. El conjunto difuso $\mu_j^{(i)}$(vj) que representa el término lingüístico $A_j^{(i)}$ (Bj) en el antecedente (consecuente) de la regla Rr, se selecciona como el peso de conexión. El término lingüístico es usado para etiquetar la conexión.
- La norma t y la co-norma t para calcular las entradas de la red para las unidades (Definición 1(v)) y el procedimiento de defusificación (Definición 1(iv)) deben seleccionarse de acuerdo con el sistema difuso que será representado por el sistema NEFCON.

La base de conocimiento del sistema difuso está implícitamente dada por la estructura de la red. Las neuronas de entrada asumen la tarea de la interfaz de fusificación, la lógica de inferencia está representada por las funciones de propagación y la neurona de salida es la interfaz de defusificación.

Algoritmos de aprendizaje para NEFCON [16]

El proceso de aprendizaje para el modelo NEFCON puede ser dividido en dos fases principales. La primera fase está diseñada para aprender una base de reglas inicial, si es que no hay conocimiento previo del sistema. Además, se puede usar para completar manualmente una base de reglas definida. La segunda fase optimiza las reglas, desplazando o modificando los conjuntos difusos de las reglas. Ambas fases utilizan un error difuso '*e*', que describe la calidad del estado del sistema actual, para aprender u optimizar la base de reglas. El error difuso juega el papel del elemento crítico usado en los modelos de aprendizaje por reforzamiento. Además deberá ser conocido el signo del valor óptimo η_{opt}. El error difuso extendido E se define como:

$$E(x_1,\ldots,x_n)=\mathrm{sgn}(\eta_{opt})e(x_1,\ldots,x_n)$$

Con la entrada nítida ('crisp') $(x_1,\ldots,x_n)$.

Los algoritmos de aprendizaje actualizados de NEFCON aprenden y optimizan la base de reglas de un controlador tipo Mamdami. Los conjuntos difusos de los antecedentes y consecuentes pueden ser representados por cualquier función de membresía simétrica.

Aprendizaje de una base de reglas

Los métodos para aprender una base de reglas inicial pueden ser divididos en tres clases: métodos que inician con una base de reglas vacía, métodos que inician con una base de reglas completa (la combinación de todos los conjuntos difusos de los antecedentes con todos los conjuntos difusos de los consecuentes; todas las combinaciones posibles) y métodos que inician con una base de reglas aleatoria. En esta parte se hará alusión a las dos primeras clases solamente.

Algoritmo modificado NEFCON I (aprendizaje decremental de reglas)

El algoritmo modificado NEFCON I inicia con una base de reglas completa. La primera versión de este algoritmo fue publicado en [29]. El algoritmo puede ser dividido en dos fases que son ejecutadas durante un periodo de tiempo fijo o durante un número fijo de iteraciones. Durante la primera fase, las reglas que tienen un signo de salida diferente al del valor óptimo de la salida η_{opt} se eliminan. Durante la segunda fase, se construye una base de reglas para cada acción de control seleccionando de forma aleatoria una regla de cada grupo de reglas con idénticos antecedentes. El error de cada regla (el error total de salida de toda la red pesado por la activación de la regla individual ¿?) se acumula. Al final de la segunda fase de cada grupo de nodos de regla con idénticos antecedentes, la regla con el valor del error más pequeño permanece en la base de reglas. Todos los otros nodos de reglas se borran. Además, las reglas usadas muy rara vez también se eliminan de la base de reglas. El algoritmo original usaba funciones de membresía triangulares, mientras que la implementación mejorada soporta también funciones de membresía trapezoidales y gaussianas. Además, el algoritmo ha sido mejorado para diferentes aplicaciones.

Algoritmo bottom-up (aprendizaje de reglas incremental)

El algoritmo incremental (bottom-up) inicia con una base de reglas vacía. Debe darse una partición inicial difusa de los intervalos de entrada y de salida. El algoritmo puede ser dividido en dos fases. Durante la primera fase, los antecedentes de las reglas son determinados por la clasificación de los valores de entrada, esto es, encontrando esa función de membresía para cada variable que produce el valor de membresía más alto para el respectivo valor de entrada. Entonces el algoritmo trata de adivinar el valor de la salida derivándolo del error difuso actual. Durante la segunda fase se optimiza la base de reglas cambiando el consecuente a una función de membresía adyacente, si esto es necesario.

Esta definición genera un algoritmo de aprendizaje de reglas que es menos caro que el procedimiento de aprendizaje de reglas decremental. No es necesario manejar todas las posibles reglas a la vez, que llega a ser imposible de cualquier manera cuando hay un conjunto de variables o un conjunto de funciones de membresía. Sin embargo, el algoritmo bottom-up puede tener problemas con sistemas dinámicos complejos, debido al acercamiento heurístico de encontrar los consecuentes.

Discusión

El algoritmo de aprendizaje de reglas decremental solamente puede ser usado si hay solamente pocas variables de entrada con pocos conjuntos difusos. El tiempo y la memoria necesarios para aplicar el algoritmo crece exponencialmente con el número de variables. Para NEFCON con cuatro variables de entrada y una variable de salida, que están particionadas cada una de ellas en siete conjuntos difusos, existen entonces 7^5=16,807 reglas posibles (si es que no se usa conocimiento previo para reducir el número). Con tal número de reglas por evaluar durante el proceso de aprendizaje es probable que NEFCON no sea capaz de funcionar en tiempo real. Por lo tanto, es aconsejable usar el algoritmo de aprendizaje de reglas incremental para sistemas NEFCON más complejos.

No puede esperarse que la base de reglas encontrada por uno de los dos algoritmos se parezca a una base de reglas dada por un humano experto quien puede controlar el proceso de interés. Primero, podía haber más de una base de reglas capaz de controlar el proceso suficientemente, y segundo, como en todos los procedimientos con aprendizaje de redes neuronales, el resultado del aprendizaje depende de factores heurísticos como la razón de aprendizaje, la duración del proceso de aprendizaje y la medida del error.

La base de reglas aprendida por NEFCON no debería ser vista como una solución final, sino como una información que puede servir de base hacia una solución más completa. Es concebible que las reglas contra-intuitivas sean manualmente borradas o reemplazadas después del aprendizaje, y se continúe con la adaptación de los conjuntos difusos¿?. No es necesario iniciar el aprendizaje de los conjuntos difusos de la nada. Dos mediciones pueden hacer uso de conocimiento parcial para el proceso de aprendizaje y reducir el costo del procedimiento de aprendizaje, especialmente para el aprendizaje de reglas decremental:

- Si para un antecedente dado se conoce un consecuente, entonces una unidad de regla equivalente se pone en el sistema NEFCON, y el algoritmo de aprendizaje no permite removerla. No se crean otras reglas con este antecedente.
- Si para algunos estados solo un subconjunto de reglas posibles son necesarias a considerar, entonces solamente estas reglas se ponen en la red.

El aprendizaje de reglas incremental también se beneficia de conocimiento previo. Si las reglas se conocen por anticipado y se introducen al sistema NEFCON, entonces el aprendizaje no necesita iniciar de nada. Un usuario puede determinar que una regla dada no deba ser borrada durante el proceso de aprendizaje de reglas, o el procedimiento de aprendizaje puede tener la libertad de borrar o modificar reglas puestas con anterioridad.

Optimización de una base de reglas

Los algoritmos presentados en esta sección son diseñados para optimizar una base de reglas de un controlador difuso recorriendo y/o modificando el soporte de los conjuntos difusos. Estos no modifican las reglas o la estructura de una red determinada.

Los algoritmos presentados en las secciones siguientes están definidos para usar funciones de membresía triangulares, pero es fácilmente posible usar conjuntos difusos simétricos en los consecuentes y en los antecedentes.

Algoritmo modificado NEFCON-I

El algoritmo NEFCON-I está motivado por el algoritmo de retropropagación para el perceptron multicapa. El error difuso extendido E es utilizado para optimizar la base de reglas por "premio y castigo". Una regla es "premiada" desplazando su consecuente con un valor más alto y ampliando el soporte de los antecedentes, si es que la salida actual tiene el mismo signo que el valor óptimo de la salida η_{opt}. De otra manera, la regla es "castigada" desplazando sus consecuentes a un valor menor y reduciendo el valor de su soporte de los antecedentes.

Algoritmo modificado NEFCON-II

En contraste al algoritmo NEFCON-I, que utiliza solamente el error actual difuso E, el algoritmo NEFCON-II también hace uso de los cambios del error difuso para optimizar la base de reglas (ver sección 6). Esta es una aproximación heurística para incluir la dinámica del sistema en el proceso de optimización.

Las ideas de los algoritmos de aprendizaje presentados anteriormente son siempre las mismas: incrementar la influencia de una regla si sus puntos de acción están en la dirección correcta (premio), y decrecer su influencia si la regla se comporta en forma contraproducente (castigo). Si la base de reglas es satisfactoria, entonces una situación de castigo ocurrirá si el proceso tiene un sobreimpulso. Ambos algoritmos de aprendizaje presentados son usados solamente para adaptar las funciones de membresía, y pueden ser aplicados solamente si ya hay una base de reglas difusas. Estas reglas pueden ser transformadas directamente en un sistema NEFCON o aprendidas por los algoritmos anteriormente presentados.

El algoritmo NEFCON-II optimiza una base de reglas más específica que el algoritmo NEFCON-I, pero es más sensible a un retrazo de tiempo entre la acción de control y la respuesta del sistema. En tales casos se preferirá el algoritmo NEFCON-I.

Todos los algoritmos presentados anteriormente no están diseñados para encontrar una solución óptima para un problema de control determinado, ya que uno de los principales objetivos de nuestra investigación es desarrollar algoritmos que son capaces de determinar en línea una apropiada e interpretable base de reglas en un pequeño número de corridas de simulación. Además debe ser posible usar conocimiento previo para iniciar el proceso de aprendizaje. Esto es un contraste a las estrategias de reforzamiento "puras" o a los métodos basados en programación dinámica, que tratan de encontrar una solución óptima usando estructuras de redes neuronales. Estos métodos necesitan muchas corridas para encontrar una solución aproximada para un problema de control determinado. De otra forma, tienen la ventaja que necesitan menos información acerca del error del estado del sistema actual. Sin embargo, en muchos casos una descripción simple del error puede ser alcanzada con un pequeño esfuerzo.

NEFCON para MATLAB (brevemente)

El modelo NEFCON ha sido implementado utilizando diferentes plataformas de software; Unix, Microsoft windows y MATLAB/SIMULINK. Estos programas se pueden obtener de forma gratuita de la siguiente dirección: (http://fuzzy.cs.uni-magdeburg.de). Estos programas traen implementados algunos ejercicios y un tutorial.

Lo importante en la implementación hecha bajo MATLAB/SIMULINK fue desarrollar una herramienta interactiva para la construcción y optimización de un controlador difuso en un ambiente de simulación que pareciera ambiente industrial. Por esta razón, NEFCON puede ser aplicado fácilmente a diferentes modelos de planta. Además, la implementación habilita al usuario a utilizar conocimiento previo en el sistema, parar y resumir el proceso de aprendizaje en cualquier tiempo, así como modificar la base de reglas y los parámetros de optimización de forma interactiva. Se diseñó también una interfase gráfica para el usuario para asistirlo durante el proceso de desarrollo del controlador difuso. El controlador difuso optimizado puede ser separado del medioambiente de simulación y puede ser usado en ambientes de tiempo real.

NEFCON para MATLAB está escrito en el lenguaje de programación propio de MATLAB y es por lo tanto una plataforma independiente.

¿se requieren estas referencias?

Grantham, Walter J., Vincent, Thomas L., ***"SISTEMAS DE CONTROL MODERNO. Análisis y diseño",*** Limusa, Noriega editores, México 1998.

Tanaka,Kazuo, ***"AN INTODUCTION TO FUZZY LOGIC FOR PRACTICAL APPLICATIONS"***, Springer, USA, 1997

A.G. Barto, R.S. Sutton, and C. Anderson, Neuronlike elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, & Cibernetics, 13:835-846. 1983.

Existen dos tipos de controladores difusos, los controladores denominados tipo Mamdami y los denominados Takagi Sugeno. Los nombres de estos controladores tienen que ver con la forma en que se definen las reglas difusas.
[Ying, Hao, ***FUZZY CONTROL AND MODELING, Analytical Foundations and Applications***, IEEE Press, USA, 2000.]

---

[1] Umez-Eronini, ***"DINÁMICA DE SISTEMAS DE CONTROL"***, Thomson Learning, México, 2001.

[2] Antsaklis, Panos J., Passino, Kevin M., ***"AN INTRODUCTION TO INTELLIGENT AND AUTONOMOUS CONTROL"*** (Preface and Overview), Kluwer Academic Publishers, USA, 1993

[3] Antsaklis, Panos J. ***"INTELLIGENT CONTROL"***, Encyclopedia of Electrical and Electronics Engineering, John Wiley and Sons, Inc. 1997

[4] Zai Lu, Yong, ***"INDUSTRIAL INTELIGENT CONTROL, FUNDAMENTAL APPLICATIONS"***, John Wiley & Sons, Great Britan, 1996.

[5] Berenji, Hamid R., ***"FUZZY AN NEURONAL CONTROL"***, Cap. 9 del libro ***"AN INTRODUCTION TO INTELLIGENT AND AUTONOMOUS CONTROL"*** recopilado por Antsaklis, Panos J., Passino, Kevin M., Kluwer Academic Publishers, USA, 1993

[6] Cuevas Jiménez, Eric Valdemar, ***CONTROL NEURODIFUSO PARA UN MANIPULADOR DE DOS GRADOS DE LIBERTAD***, Tesis de Maestría, ITESO, 2000.

[7] Jager, René, ***"FUZZY LOGIC IN CONTROL"***, PhD Thesis, Technische Universiteit Delft, Germany, 1995.

[8] Omidvar, Omid; Elliot, David L.; ***"NEURONAL SYSTEMS FOR CONTROL"***; Academic Press, USA, 1997. (Ref. ITESO, 629.89 OMI)

[9] Werbos, Paul J.; ***"AN OVERVIEW OF NEURONAL NETWORKS FOR CONTROL"***. IEEE Control Systems, January 1991.

[10] Nøgaard, M.; Ravn, O.; Poulsen, N.K. and Hansen, L.K. "***NEURAL NERWORKS FOR MODELLING AN CONTROL OF DYNAMIC SYSTEMS"***, Springer Verlag, London, 2000.

[11] Neuronal Networks for Control Systems – A Survey. K.J. HUNT, D. SBARBARO, R. ZBIKOWSKI and P.J. GAWWTHROP. Automática, vol 28.no. 6. 1992. Pergamon Press Ltd., Oxford, England. (antes [12])

[12] Patiño, H.D. Liu, Derong, ***"NEURAL NETWORK-BASED MODEL REFERENCE ADAPTIVE CONTROL SYSTEM"***, IEEE Transaction on Systems, Man and Cybernetics-Part B, Vol. 30. No. 1. February 2000. (antes [15])

[13] Fink, Alexander; Nelles, Olliver; ***"NONLINEAR INTERNAL MODEL CONTROL BASED ON LOCAL LINEAR NEURAL NETWORKS"***; Proccedings of the 2001 IEEE Systems, Man, and Cybernetics Conference. (antes 16)

[14] Ling Chin Teng, C.S. George Lee, ***"NEURONAL FUZZY SYSTEMS, (A NEURO FUZZY SYNERGISM TO INTELLIGENT SYSTEMS)"***, Prentice Hall, USA, 1996. (antes 9)

[15] Detlef Nauck, Frank Klawonn and Rudolf Kruse, ***"FOUNDATIONS OF NEURO-FUZZY SYSTEMS."***, John Wiley ans sons, U.S.A. 1997. (antes 7)

[16] Nüremberger, A.; Nauck, D.; Kruse, R.; ***"NEURO – FUZZY CONTROL BASED ON THE NEFCON MODEL: RECENT DEVELOPMENTS"***. Soft computing 2 (1999) Springer Verlag 1999