1. Introducción

Para implementar un controlador mediante métodos de control clásico, en general es necesario conocer el modelo matemático de la planta, lo cual añade complejidad al diseño del controlador y pudiera también aumentar el tiempo de implementación del mismo.

Los métodos de control de procesos dinámicos basados en lógica difusa ofrecen ventajas importantes respecto a las técnicas de control clásico, porque no es necesario un modelado matemático de la planta por ser técnicas libres de modelo [Kosko, 1992]. Además, usando lógica difusa, es factible aprovechar el conocimiento de un experto humano sobre el proceso que se desea controlar [Jamshidi, 1993].

La implementación de un controlador difuso frecuentemente se realiza en una computadora personal debido a que ésta permite el uso de lenguajes de programación de alto nivel y programas de diseño de controladores difusos. Obviamente para ciertas aplicaciones, como lo serían controladores para aparatos domésticos e incluso algunas aplicaciones industriales, podría no ser justificable o no conveniente la inversión de una computadora. Por esa razón en aplicaciones sobre equipos o aparatos medianos o pequeños conviene utilizar controladores implementados en tarjetas autónomas basados en dispositivos microprocesadores. De aquí surge la conveniencia de aprovechar el uso de procesadores digitales de señales (DSP por sus siglas en inglés), microcontroladores, u otros dispositivos de menor capacidad y costo que una computadora personal.

Una ventaja importante de un sistema basado en lógica difusa para aplicaciones de control implantada en un dispositivo microcontrolador, es que aún funciones complicadas y lazos de control adaptable pueden ser implementados usando los recursos limitados de un microcontrolador de 8 bits de bajo costo [Microchip, 1995].

Un estudio enfocado sólo sobre lógica difusa, indicó un crecimiento anual del 20% y la posibilidad de ser una de las 10 tecnologías más exitosas del siglo 21 [Ross, 1997].

De este modo, la idea de implementar un controlador difuso en un microcontrolador resulta muy atractiva ya que el microcontrolador es un sistema digital de bajo costo, y se presta a la implementación de controladores autónomos. Sin embargo, se presentan algunas problemáticas a resolver, como las limitaciones técnicas del microcontrolador (tamaño de la memoria, rapidez de procesamiento, variedad de instrucciones, etc.), además de la manera en que el algoritmo difuso es implantado o programado en el microcontrolador (es decir las técnicas a usarse para la fusificación de las entradas, evaluación de las reglas, y defusificación de las salidas). Tal vez de todas las limitaciones, una de las más importantes se encuentra en el tamaño de la memoria que poseen, que es del orden de los kilobytes y puede resultar muy pequeña para la implementación de algoritmos difusos de control convencionales.

El algoritmo difuso mínimo [Sánchez, 1998], debido a su estructura, demanda pocos recursos del dispositivo microcontrolador en comparación a los que podría llevarse otro tipo de algoritmo difuso. Es por esto, que en este libro se desarrolla un método para la implantación de algoritmos difusos mínimos en un microcontrolador PIC16C711 de *Microchip,* y un ejemplo practico del mismo mediante el control de una planta de nivel.

Existen razones especiales para seleccionar este microcontrolador, algunas de ellas son: el contar con las características necesarias (convertidor A/D incorporado, velocidad de 20 MHz, etc.) para este tipo de implementación [Microchip, 1997] y poseer un bajo costo (aproximadamente 3 dólares estadounidenses la versión ROM y 16 dólares estadounidenses la versión EPROM). Además para este microcontrolador es posible contar con la herramienta *fuzzy*TECH-MP Explorer (que en adelante se denominará sólo como *fuzzy*TECH) para el diseño, optimización y verificación de un sistema de control basado en lógica difusa en un microcontrolador de la familia PIC16/17 de *Microchip*.

Un factor importante en la implementación de cualquier controlador es el tiempo de diseño, por lo que resulta conveniente utilizar herramientas enfocadas a su reducción.

En este libro se incluyen los diagramas electrónicos de las tarjetas electrónicas necesarias para el control de una planta de nivel ejemplo o de un sistema de dinámica similar y con el mismo número de entradas/salidas. Estas tarjetas pueden ser utilizadas para la implementación de otras aplicaciones siguiendo el método de implantación e integración descrito en este libro.

El capitulado se encuentra organizado como sigue:

En el Capítulo 2 se hace una descripción del algoritmo difuso mínimo, mostrando las ventajas que ofrece éste para su implementación en un dispositivo microcontrolador. El algoritmo se implementa y se simula en el entorno *MATLAB/FUZZY LOGIC TOOLBOX*, para obtener datos sobre su comportamiento.

En el Capítulo 3, se realiza una descripción de todo el sistema ejemplo, empezando por la planta de nivel, su modelo matemático, características físicas, hasta llegar a la implementación de ésta en el entorno de simulación *MATLAB/SIMULINK*. Posteriormente, a este modelo de la planta de nivel se le integra el algoritmo de control difuso mínimo descrito en el Capítulo 2, se simula en dicho entorno, obteniendo gráficas de desempeño del controlador ajustado y finalmente se registran los resultados cualitativos del sistema.

En el Capítulo 4, se describe de manera breve la implementación del algoritmo difuso mínimo en el microcontrolador haciendo mayor énfasis en el funcionamiento del código en ensamblador para el algoritmo difuso implementado. Se explica la técnica empleada para la implementación e integración del algoritmo difuso mínimo junto con las demás rutinas en ensamblador. Posteriormente, se expone el funcionamiento de cada una de estas rutinas mediante diagramas de flujo. Mediante la simulación del código en lenguaje ensamblador con la herramienta de desarrollo MPLAB para microcontroladores PIC, se obtienen datos sobre su comportamiento, con el propósito de compararlos con los que se encontraron en la herramienta *MATLAB/FUZZY LOGIC TOOLBOX* (Capítulo 2). Posteriormente, se muestran los resultados de las pruebas en tiempo real, para el controlador difuso autónomo que se implementó en una planta de nivel. También, se observa el desempeño de éste mediante unas gráficas obtenidas mediante una interfaz de monitoreo en tiempo real implementada en una computadora personal. Por ultimo, se comparan de manera cuantitativa los resultados de dichas pruebas con los obtenidos en la simulación en *MATLAB/SIMULINK* (Capítulo 3).

En el Capítulo 5 se explica de manera detallada la implementación del algoritmo difuso mediante la herramienta *fuzzy*TECH, desglosando cada uno de los pasos necesarios

para su realización [Microchip, 1995], así como la corrección de errores en el código y compilación de éste, para obtener finalmente el código difuso en lenguaje ensamblador y el *kernel* del sistema. También en este capitulo, se explican los procedimientos para integrar el código en lenguaje ensamblador del algoritmo difuso, con las demás rutinas necesarias de la aplicación planteada como ejemplo, así como la programación [Microchip, 1996] de todo este código en un microcontrolador PIC16C711.

En el Apéndice A se muestra el código en lenguaje ensamblador para el sistema difuso implementado en *fuzzy*TECH (archivos *min_255.asm y min_255.var*).

En el Apéndice B se presenta el código en lenguaje ensamblador para el programa principal encargado de integrar y coordinar los todos los subprogramas (archivo *main_v3.asm*).

En el Apéndice C se presenta el código en lenguaje ensamblador de los programas propios de la aplicación; es decir, de las rutinas que realizan las funciones de obtención de error, de la derivada del error, conversión analógica digital, filtrado digital de la señal del sensor y conversión a formato de 128 valores (*archivos init_v3.asm y filtro3.asm).* Así como los códigos en lenguaje ensamblador para la obtención de las ganancias G_e, g_r y G_u (archivos *mul_ge3.asm, mul_gr3.asm y mul_gu3.asm,* respectivamente) que utiliza el algoritmo difuso mínimo.

En el Apéndice D se muestra el *kernel* en lenguaje ensamblador necesario para el funcionamiento del código para el algoritmo difuso, así como el de las variables públicas y "macros" de uso global (FTPUBDEC.VAR).

En el Apéndice E se muestra el código para el programa de monitoreo en tiempo real del controlador autónomo desde una computadora personal.

En el Apéndice F se incluyen los diagramas electrónicos así como los esquemas de los circuitos impresos del controlador difuso autónomo basado en el PIC.

---

2. Algoritmo de control difuso mínimo

En este capitulo se presentan los detalles del algoritmo difuso seleccionado para ser implantado en el microcontrolador, con el propósito de entender claramente el proceso de implantación. Este algoritmo es especialmente recomendado para aplicaciones en las que los recursos del hardware son bajos, como los de un microcontrolador [Zaldívar 2000].

2.1 Descripción.

Para programar al microcontrolador, se utilizará como ejemplo el algoritmo difuso mínimo descrito en [Sánchez, 1998] ó proporcional derivativo difuso (PD difuso), que consta de sólo 4 reglas y tiene la estructura ilustrada en la Fig. 2.1. La descripción realizada en esta sección se basa en la presentación de L. A. Nuño [Nuño, 1998].

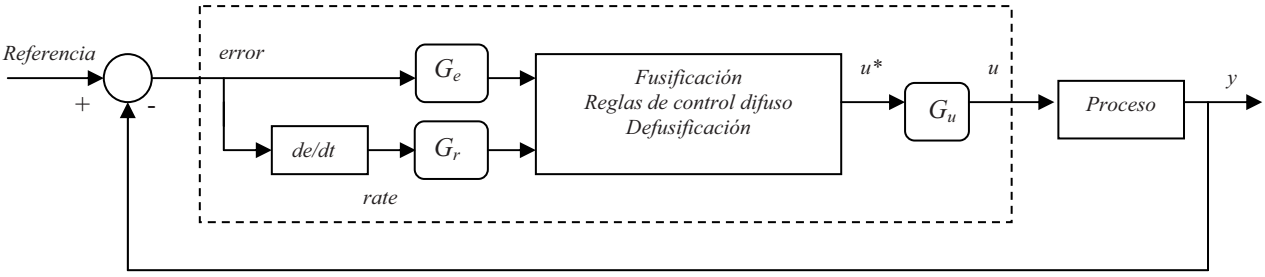

Fig. 2.1 Esquema de control difuso mínimo.

En este esquema:

$$u=G_u(u^*) \tag{2.1}$$

Las ganancias G_u, G_e y G_r se determinan por el diseñador y corresponden respectivamente a las ganancias de la *salida*, del *error* y de la derivada de ese error, que se llamará *rate*. La *u** es la salida del controlador difuso ya defusificada, es decir la salida "crisp"; este término en lo sucesivo se remplazará por "nítido".

2.1.1 Fusificación.

Como puede observarse en la Fig. 2.1, existen dos entradas al controlador, que son *error* y *rate,* que es la derivada de ese error o la velocidad con que cambia ese error. El *error* se define como:

$$error = Referencia - y \tag{2.2}$$

en donde *Referencia* es la señal a seguir por el controlador y por último la señal *y* es la salida del proceso a controlar. En cuanto a *rate* está definida como sigue:

$$rate = (error\ actual - error\ anterior) / período\ de\ muestreo \tag{2.3}$$

Tanto el error actual como el error anterior, se refieren a un error sin escalar, es decir, sin multiplicar por la ganancia G_e .

El controlador difuso tiene una sola salida *u*, la cual se utiliza para controlar el proceso.

Las funciones de membresía de las entradas y de la salida del controlador difuso se muestran en las Fig. 2.2 y 2.3, respectivamente. Como puede observarse en la Fig. 2.2, las entradas tienen dos términos lingüísticos cada una, que son: G_e**error negativo* (el cual llamaremos "*en*") y G_e**error positivo* (que llamaremos "*ep*"), para la entrada *error*, y G_r**rate negativo* (la cual llamaremos *"rn*") y G_r**rate positivo* (que llamaremos "*rp*") para la entrada *rate*, mientras que el conjunto difuso de salida (Fig. 2.3) está formado por tres términos lingüísticos, que son: *salida negativa ("on"), salida cero ("oz") y salida positiva ("op").*

Como puede observarse en la Fig. 2.2, la misma función se aplica para *error* y *rate*. Para que esto sea posible se utilizan dos factores de escalamiento G_e y G_r para *error* y *rate,* respectivamente; así entonces lo que entra al sistema difuso y se fusifica es G_e**error* y G_r**rate,* es decir *error* y *rate* escalados.

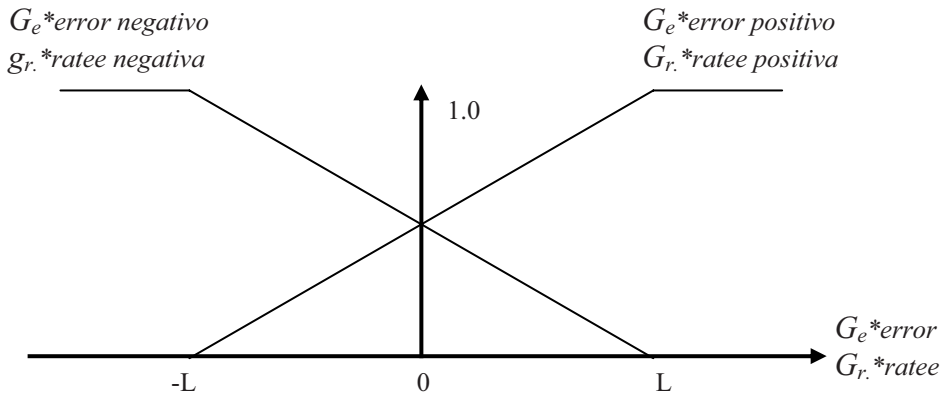

Fig. 2.2 Conjunto difuso de entrada.

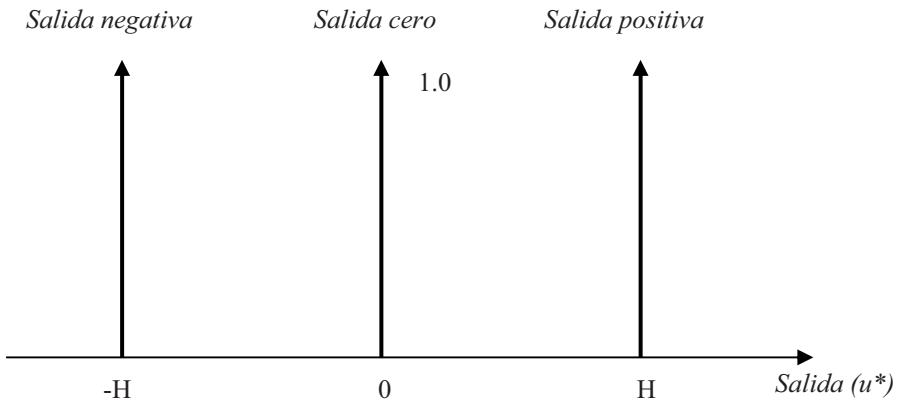

Fig. 2.3 Conjunto difuso de salida.

En las Fig. 2.2 y 2.3, *H y L* son dos constantes positivas a ser determinadas. Por conveniencia tomaremos *H=L* para reducir el número de parámetros de control a determinar.

Las funciones de membresía para las variables de entrada escaladas *error* (positivo y negativo) y *rate* (positivo y negativo) se definen mediante las siguientes expresiones [Nuño, 1998] :

$$\mu_{ep} = \frac{L + (Ge * error)}{2L} \qquad \mu_{en} = \frac{L - (Ge * error)}{2L}$$
$$\mu_{rp} = \frac{L + (Gr * rate)}{2L} \qquad \mu_{rn} = \frac{L - (Gr * rate)}{2L} \tag{2.4}$$

2.1.2 Reglas difusas.

En lo que sigue *error* corresponderá al error escalado (G_e**error*) y *rate* a la derivada escalada (G_r**rate*).

Existen cuatro reglas a evaluar en el controlador difuso mínimo [Ying, 1990] :

R1. Si *error* es ***ep*** y *rate* es ***rp*** entonces *salida* es ***op***

R2. Si *error* es ***ep*** y *rate* es ***rn*** entonces *salida* es ***oz***

R3. Si *error* es ***en*** y *rate* es ***rp*** entonces *salida* es ***oz***

R4. Si *error* es ***en*** y *rate* es ***rn*** entonces *salida* es ***on***

(2.5)

Se utiliza entonces la inferencia de *Mamdani* [Kosko, 1997] es decir :

$$min\ (a,b) \tag{2.6}$$

siendo *a* la función de membresía del antecedente y *b* la función de membresía de la acción de control.

Debido a que las reglas R2 y R3 generan la misma salida se realiza una unión de reglas; esto se lleva a cabo aplicando el operador "OR" de Lukasiewicz [Ross, 1997] definido como:

$$min\ (a+b,\ 1) \tag{2.7}$$

La conjunción "y" (dentro de cada una de las cuatro reglas 2.5) corresponde a la operación "AND" lógica de *Zadeh*, es decir:

$$\mu_A(x) \wedge \mu_B(x) = \min\{\mu_A(x), \mu_B(x)\} \tag{2.8}$$

En donde $\mu_A(x)$ y $\mu_B(x)$ son los valores de membresía de los conjuntos difusos A y B en el punto x, respectivamente, que para el caso del algoritmo difuso mínimo será: el mínimo entre el valor de la función de membresía del error escalado (*en* o *ep)* y el valor de la función de membresía de la derivada escalada (*rn* o *rp).*

El significado de las cuatro reglas puede ser explicado utilizando el problema de regulación, para una referencia constante. Dentro de este contexto se tiene que:

Para la primer regla (R1), la condición "*ep"* implica que la salida del sistema "*y*" se encuentra por debajo de la referencia y el término "*rp"* significa que la velocidad con la que cambia el sistema está disminuyendo. En este caso, la acción de control debe de ser positiva, para incrementar la salida del sistema. Para la tercer regla (R3), la condición *"en"* implica que la salida del sistema *"y"* se encuentra por encima de la referencia y el término *"rp"* significa que la velocidad con la que cambia el sistema está disminuyendo, entonces no se necesita ninguna acción de control. La segunda y cuarta regla pueden interpretarse de una forma similar.

2.1.3 Defusificación.

El método de defusificación utilizado por el algoritmo difuso mínimo, es el del centro de gravedad, que en este caso se expresa como:

$$u = \frac{-H(\mu_{R4(x)}) + 0(\mu_{R2(x)+}\mu_{R3(x)}) + H(\mu_{R1(x)})}{\mu_{R4(x)} + (\mu_{R2(x)+}\mu_{R3(x)}) + \mu_{R1(x)}} \tag{2.9}$$

Para el controlador PD difuso propuesto, los rangos de los valores de las entradas de *error* y *rate* ya escaladas se descomponen en 20 regiones adyacentes o *Input Combinations* (IC), como se muestra en la Fig. 2.4.

Si se evalúan las funciones de membresía, las 4 reglas de control, se toma a $H=L$ y se aplica la desfusificación de la ecuación 2.9 en cada una de las 20 combinaciones de entrada, se obtienen las 9 ecuaciones (2.10), las cuales determinan la señal de control *u* que se debe aplicar, dependiendo de la región en la que se encuentre. En otras palabras, para implementar el controlador difuso mínimo, tan sólo será necesario detectar la región en la que se encuentran las variables de entrada y después evaluar la ecuación correspondiente para dicha región. Como puede observarse en la parte derecha de cada una de las 9 ecuaciones se indican las regiones IC en las que actúa la ecuación correspondiente. Por ejemplo la primera ecuación actúa en las regiones *IC1, IC2, IC5, IC6.*

2.2 Implementación en *MATLAB/FUZZY LOGIC TOOLBOX.*

Debido a las ventajas que ofrece la simulación en el diseño de un controlador, se implementó el algoritmo difuso mínimo mediante la herramienta *MATLAB/FUZZY LOGIC TOOLBOX,* quedando como lo muestra la Fig. 2.5. En ésta y en las demás ventanas asociadas (Fig. 2.6, Fig. 2.7 y Fig. 2.8), se aprecian las características de dicho algoritmo. Por ejemplo, en el lado superior izquierdo de la Fig. 2.5 se observan las dos variables de entrada al sistema difuso (*error* y *rate*) después, a la mitad de la misma figura se puede observar el tipo de inferencia utilizada (*Mamdani*) y del lado superior derecho se encuentra la variable *salida*. En

la parte inferior izquierda de la misma figura se muestran los operadores seleccionados para la implementación de este algoritmo, esto es, los operadores utilizados para realizar las operaciones *And, Or, implicación, agregación*, así como también el método de defusificación. Finalmente, en la parte baja de la figura, se indica el nombre del sistema, el número de

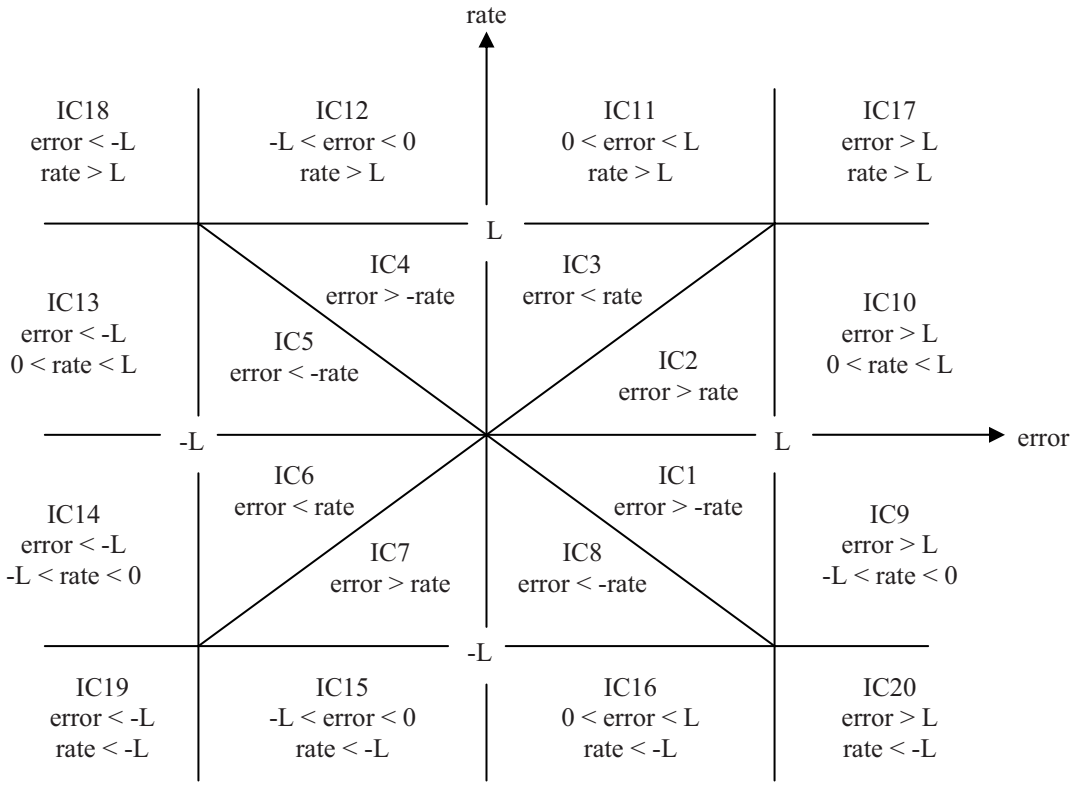

Figura 2.4 Regiones de combinación de entradas.

entradas, el número de salidas y el número de reglas.

$$
\begin{aligned}
u &= \frac{L}{2(2L - G_e|error|)}[G_e * error + G_r * rate] \qquad en\ IC1, IC2, IC5, IC6. \\
u &= \frac{L}{2(2L - G_r|rate|)}[G_e * error + G_r * rate] \qquad en\ IC3, IC4, IC7, IC8. \\
u &= \frac{1}{2}[L + G_r * rate] \qquad en\ IC9, IC10. \\
u &= \frac{1}{2}[L + G_e * error] \qquad en\ IC11, IC12.
\end{aligned}
\tag{2.10}
$$

$$u = \frac{1}{2}[-L + G_r * rate] \qquad en\ IC13, IC14.$$

$$u = \frac{1}{2}[-L + G_e * error] \qquad en\ IC15, IC16.$$

$$u = L \qquad en\ IC17.$$
$$u = -L \qquad en\ IC19.$$

$$u = 0 \qquad en\ IC18, IC20.$$

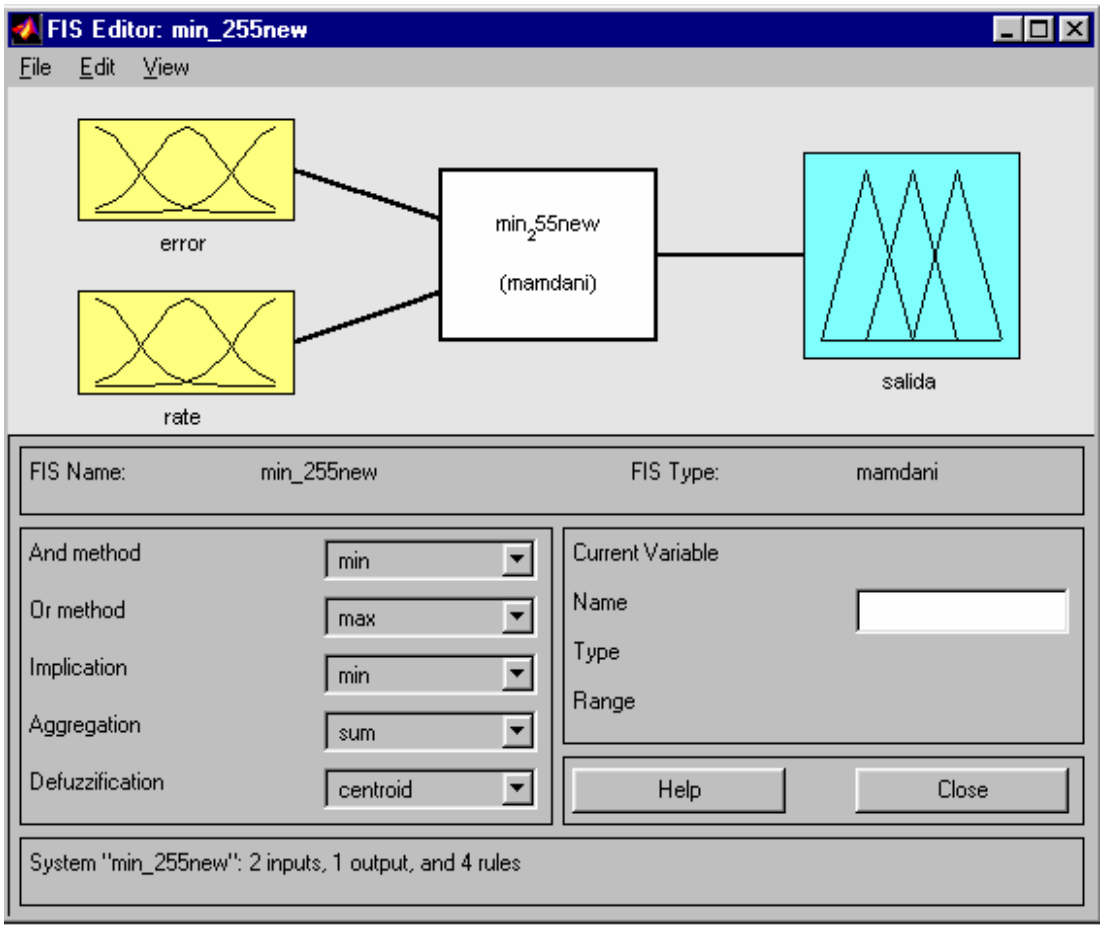

Fig. 2.5 Descripción del sistema difuso mínimo implementado en el *MATLAB/FUZZY LOGIC TOOLBOX.*

Las ganancias de entrada y salida no se implementan en la herramienta *MATLAB/FUZZY LOGIC TOOLBOX* por lo que, el comportamiento del sistema equivaldrá a uno donde las ganancias sean unitarias. Dichas ganancias se incluyeron dentro de la herramienta *MATLAB/SIMULINK* (Capítulo 3, Fig. 3.4 y Fig. 3.5).