

# CONCEITOS FUNDAMENTAIS DA TEORIA DE CONJUNTOS FUZZY, LÓGICA FUZZY E APLICAÇÕES

Fernando A. C. Gomide \*

Ricardo R. Gudwin \*

Ricardo Tanscheit \*\*

\* Departamento de Engenharia de Computação e Automação Industrial (DCA)  
Faculdade de Engenharia Elétrica (FEE)  
Universidade Estadual de Campinas (UNICAMP)  
CP 6101, CEP 13081-970, Campinas - SP  
e-mail : gomide@dca.fee.unicamp.br  
gudwin@dca.fee.unicamp.br

\*\* Departamento de Engenharia Elétrica e Eletrônica (DEEE)  
Universidade Federal do Maranhão (UFMA)  
Campus do Bacanga  
CEP 65085-580, São Luís - MA

**RESUMO:** A Lógica Fuzzy (Nebulosa) é a lógica que suporta os modos de raciocínio que são aproximados ao invés de exatos. Modelagem e controle fuzzy de sistemas são técnicas para o tratamento de informações qualitativas de uma forma rigorosa. Derivada do conceito de conjuntos fuzzy, a lógica fuzzy constitui a base para o desenvolvimento de métodos e algoritmos de modelagem e controle de processos, permitindo a redução da complexidade de projeto e implementação, tornando-se a solução para problemas de controle até então intratáveis por técnicas clássicas. Este trabalho apresenta uma introdução aos princípios e às idéias que fundamentam a aplicação da lógica fuzzy em sistemas inteligentes em geral, e controle de processos em particular. Hardware e ferramentas de suporte ao desenvolvimento de aplicações são também descritos. Exemplos de diferentes classes de problemas de controle são consideradas a fim de ilustrar o potencial da lógica fuzzy em aplicações práticas. Finalmente é apresentado um panorama do estado da arte atual, incluindo os principais resultados práticos em uso por vários segmentos da indústria e as tendências futuras.

## 1 - INTRODUÇÃO

A lógica fuzzy é a lógica baseada na teoria dos conjuntos fuzzy. Ela difere dos sistemas lógicos tradicionais em suas características e seus detalhes. Nesta lógica, o raciocínio exato corresponde a um caso limite do raciocínio aproximado, sendo interpretado como um processo de composição de relações nebulosas.

Na lógica fuzzy, o valor verdade de uma proposição pode ser um subconjunto fuzzy de qualquer conjunto parcialmente ordenado, ao contrário dos sistemas lógicos binários, onde o valor verdade só pode assumir dois valores: verdadeiro (1) ou falso (0). Nos sistemas lógicos multi-valores, o valor verdade de uma proposição pode ser ou um elemento de um conjunto finito, num intervalo, ou uma álgebra booleana. Na lógica nebulosa, os valores verdade são expressos linguisticamente, (e.g. : *verdade, muito verdade, não verdade, falso, muito falso, ...*), onde cada termo linguístico é interpretado como um subconjunto fuzzy do intervalo unitário.

Outras características da lógica fuzzy podem ser sumarizadas como segue : nos sistemas lógicos binários, os predicados são exatos (e.g. : *par, maior que* ), ao passo que na lógica fuzzy os predicados são nebulosos (e.g. : *alto, baixo, ...*). Nos sistemas lógicos clássicos, o modificador mais utilizado é a negação enquanto que na lógica fuzzy uma variedade de modificadores de predicados são possíveis (e.g. : *muito, mais ou menos, ...*). Estes modificadores são essenciais na geração de termos linguísticos (e.g. : *muito alto, mais ou menos perto, etc*).

Nos sistemas lógicos clássicos existem somente os quantificadores existenciais ( $\exists$ ) e universais ( $\forall$ ). A lógica fuzzy admite, em adição, uma ampla variedade de quantificadores (e.g. : *pouco, vários, usualmente, frequentemente, em torno de cinco, etc* ).

A probabilidade, no contexto da lógica clássica, é um valor numérico ou um intervalo. Na lógica nebulosa existe a opção adicional de se empregar probabilidades linguísticas (e.g. : *provável, altamente provável, improvável, etc*), interpretados como números fuzzy e manipuladas pela aritmética fuzzy (Kaufmann & Gupta , 1985). Também, em contraste com a lógica modal clássica, o conceito de possibilidade é interpretado utilizando-se subconjuntos fuzzy no universo dos reais (Zadeh, 1988).

Nas teorias de controle clássica e moderna, o primeiro passo para implementar o controle de um processo é derivar o modelo matemático que descreve o processo. O procedimento, requer que se conheça detalhadamente o processo a ser controlado, o que nem sempre é factível se o processo é muito complexo. As teorias de controle existentes se aplicam a uma grande variedade de sistemas, onde o processo é bem definido. Várias técnicas, tais como para controle linear multivariável (Doyle e Skin, 1981), estimação de estado a partir de medidas ruidosas (Anderson e Moore, 1979), controle ótimo (Sage e White, 1977), sistemas lineares estocásticos (Bertsekas, 1976), além de certas classes de problemas não-lineares determinísticos (Holtzman, 1970), foram desenvolvidas e aplicadas com sucesso em um grande número de problemas bem postulados. Entretanto, todas estas técnicas não são capazes de resolver problemas reais cuja modelagem matemática é impraticável. Por exemplo, em diversas situações um volume considerável de informações essenciais só é conhecido a priori de forma qualitativa. Do mesmo modo, critérios de desempenho só estão disponíveis em termos linguísticos. Este panorama leva a imprecisões e falta de exatidão que inviabilizam o uso da maioria das teorias utilizadas até agora.

A modelagem e o controle fuzzy (Lee, 1990) são técnicas para se manusear informações qualitativas de uma maneira rigorosa. Tais técnicas consideram o modo como a falta de exatidão e a incerteza são descritas e, fazendo isso, tornam-se suficientemente poderosas para manipular de maneira conveniente o conhecimento. A sua utilização em sistemas de controle de processos em tempo real, em computadores ou micro-controladores, é das mais convenientes, dado que, geralmente, não envolvem nenhum problema computacional sério. A teoria de modelagem e controle fuzzy trata do relacionamento entre entradas e saídas, agregando vários parâmetros de processo e de controle. Isso permite a consideração de processos complexos, de modo que os sistemas de controle resultantes proporcionam um resultado mais acurado, além de um desempenho estável e robusto. A grande simplicidade de implementação de sistemas de controle fuzzy pode reduzir a complexidade de um projeto a um ponto em que problemas anteriormente intratáveis passam agora a ser solúveis.

## 2 - CONJUNTOS FUZZY

### 2.1 - Fundamentos

Nesta seção serão apresentadas as idéias básicas sobre conjuntos e lógica fuzzy visando a modelagem e o desenvolvimento de sistemas em geral e de sistemas de controle em particular. Apesar de existir uma complexa base formal sustentando, por exemplo, seu uso na modelagem e controle de sistemas, será evidenciado aqui somente o necessário para o entendimento da teoria básica de sistemas fuzzy (Pedrycz, 1989; Yager et.al., 1987; Lee, 1990; Albertos, 1992).

Na teoria de conjuntos clássica, um elemento pertence ou não a um dado conjunto. Dado um universo  $U$  e um elemento particular  $x \in U$ , o grau de pertinência  $\mu_A(x)$  com respeito a um conjunto  $A \subseteq U$  é dado por:

$$\mu_A(x) = \begin{cases} 1 & \text{se } x \in A \\ 0 & \text{se } x \notin A \end{cases}$$

A função  $\mu_A(x) : U \rightarrow \{0,1\}$  é chamada de função característica na teoria clássica de conjuntos. Frequentemente, uma generalização desta idéia é utilizada, por exemplo, para manipulação de dados com erros limitados. Todos os números dentro de um erro percentual terão um fator de pertinência 1, tendo todos os demais um fator de pertinência 0. (veja figura 1a). Para o caso preciso, o fator de pertinência é 1 somente no número exato, sendo 0 para todos os demais (veja figura 1b).

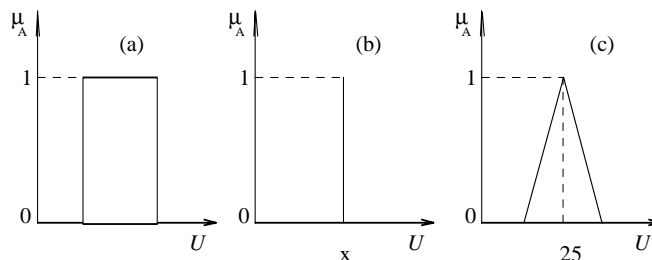


Figura 1: Funções de Pertinência

Zadeh (Zadeh, 1965) propôs uma caracterização mais ampla, na medida em que sugere que alguns elementos são mais membros de um conjunto do que outros. O fator de pertinência pode então assumir qualquer valor entre 0 e 1, sendo que o valor 0 indica uma completa exclusão e um valor 1 representa completa pertinência. Esta generalização aumenta o poder de expressão da função característica. Por exemplo, para expressar a idéia de que uma temperatura tem seu valor por volta de 25, pode-se utilizar uma função de pertinência triangular (veja figura 1c), com o pico em 25, para sugerir a idéia de que quanto mais perto o número de 25, mais ele se identifica com o conceito representado.

Formalmente, seja  $U$  uma coleção de objetos denominados genericamente por  $\{u\}$ .  $U$  é chamado de universo de discurso, podendo ser contínuo ou discreto. Um conjunto fuzzy  $A$  em um universo de discurso  $U$  é definido por uma função de pertinência  $\mu_A$  que assume valores em um intervalo  $[0,1]$ :

$$\mu_A : U \rightarrow [0,1]$$

O conjunto fuzzy  $A$  em  $U$  é, então, um conjunto de pares ordenados

$$A = \{\mu_A(u)/u\}, u \in U$$

O conjunto suporte de um conjunto fuzzy  $A$  é o sub-conjunto dos pontos  $u$  de  $U$  tal que  $\mu_A(u) > 0$ . Um conjunto fuzzy cujo conjunto suporte é um único ponto de  $U$  com  $\mu_A = 1$  é chamado de um conjunto unitário fuzzy.

## 2.2 - Definições e Operações

Considerando os conjuntos fuzzy  $A$  e  $B$  em um universo  $U$ , sejam:

Conjunto vazio:  $A = \emptyset$  se e somente se  $\forall u \in U, \mu_A(u) = 0$

Complemento  $A^c$ :  $\mu_{A^c}(u) = 1 - \mu_A(u)$

Conjuntos iguais:  $A = B$  se e somente se  $\forall u \in U, \mu_A(u) = \mu_B(u)$

$A$  subconjunto de  $B$ :  $A \subset B$  se  $\forall u \in U, \mu_A(u) \leq \mu_B(u)$

Nas operações com conjuntos fuzzy utilizam-se os conceitos de norma triangular (norma-t) e co-norma triangular (norma-s). Uma norma triangular é uma função  $\mathcal{J} : [0,1] \times [0,1] \rightarrow [0,1]$  tal que,  $\forall x, y, z, w \in [0,1]$ :

- (i)  $x \mathcal{J} w \leq y \mathcal{J} z$ , se  $x \leq y, w \leq z$
- (ii)  $x \mathcal{J} y = y \mathcal{J} x$
- (iii)  $(x \mathcal{J} y) \mathcal{J} z = x \mathcal{J} (y \mathcal{J} z)$
- (iv)  $x \mathcal{J} 0 = 0$ ;  $x \mathcal{J} 1 = x$

Uma co-norma triangular é tal que  $\mathcal{L} : [0,1] \times [0,1] \rightarrow [0,1]$ , satisfazendo as propriedades (i) a (iii) acima e ainda

- (iv)  $x \mathcal{L} 0 = x$ ;  $x \mathcal{L} 1 = 1$

Exemplos de normas-t incluem o mínimo ( $\wedge$ ) e o produto algébrico ( $\cdot$ ). Como exemplo de normas-s podem ser citados o máximo ( $\vee$ ) e a soma limitada ( $\oplus$ ) (Pedrycz, 1989).

Para representar a união e interseção de conjuntos fuzzy, utilizam-se respectivamente normas triangulares e co-normas triangulares. Assim, utilizando-se os operadores mínimo como norma-t e máximo como norma-s, têm-se:

União de  $A$  e  $B$ :  $\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x)$

Intersecção de  $A$  e  $B$ :  $\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x)$

## 2.3 - Propriedades Algébricas

Utilizando as definições de união e intersecção, implementadas pelos operadores máximo e mínimo, e complemento, é possível verificar que as seguintes propriedades algébricas de conjuntos ordinários também valem para conjuntos fuzzy:

Involução:  $(A')' = A$

Idempotência:  $A \cap A = A$  e  $A \cup A = A$

Comutatividade:  $A \cap B = B \cap A$  e  $A \cup B = B \cup A$

Associatividade:  $(A \cap B) \cap C = A \cap (B \cap C)$  e  $(A \cup B) \cup C = A \cup (B \cup C)$

Distributividade:  $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$   
 $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Lei Transitiva: se  $A \subset B$  e  $B \subset C$  então  $A \subset C$

Observando que as funções de pertinência dos conjuntos vazio e universo são definidas como sendo 0 e 1, respectivamente,  $\forall u \in U$ , as seguintes propriedades também são válidas:

$$A \cap \emptyset = \emptyset \text{ e } A \cap U = A$$
$$A \cup \emptyset = A \text{ e } A \cup U = U$$

As propriedades de conjuntos clássicos que não se verificam para conjuntos fuzzy são:

$$A \cap A' \neq \emptyset \text{ e } A \cup A' \neq U$$

## 2.4 - Relação Fuzzy

Se  $A_1, A_2, \dots, A_n$  são conjuntos fuzzy em  $U_1, U_2, \dots, U_n$  respectivamente, uma relação fuzzy n-ária é um conjunto fuzzy em  $U_1 \times U_2 \times \dots \times U_n$ , expresso da seguinte maneira:

$$R = \{ [(u_1, \dots, u_n), \mu_R(u_1, \dots, u_n)] \mid (u_1, \dots, u_n) \in U_1 \times U_2 \times \dots \times U_n \}$$

ou, mudando a notação,

$$R = \{ \mu_R(u_1, \dots, u_n) / (u_1, \dots, u_n) \}, (u_1, \dots, u_n) \in U_1 \times U_2 \times \dots \times U_n$$

Se  $R$  e  $P$  são relações fuzzy em  $U \times V$  e  $V \times W$  respectivamente, a composição de  $R$  e  $P$  é uma relação denotada por  $R \circ P$  definida a seguir:

$$R \circ P = \{ [(u, w), \sup_v (\mu_R(u, v) \mathcal{J} \mu_P(v, w))] ; u \in U, v \in V, w \in W \}$$

## 2.5 - Variáveis Linguísticas

Uma variável linguística é uma variável cujos valores são nomes de conjuntos fuzzy. Por exemplo, a temperatura de um dado processo poderia ser uma variável linguística assumindo valores *baixa*, *média*, *alta*, etc. Estes valores são descritos por intermédio de conjuntos fuzzy (vide figura 2).

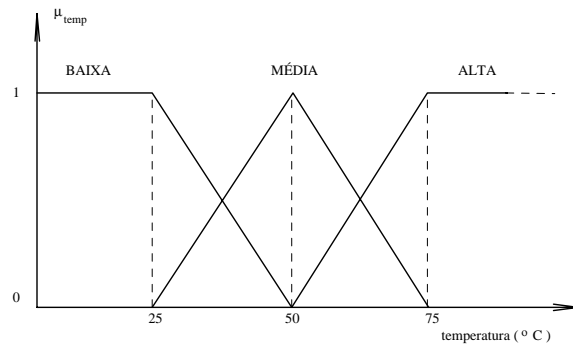


Figura 2 : Variável Linguística *Temperatura*

Generalizando, os valores de uma variável linguística podem ser sentenças em uma linguagem especificada. Para ilustrar, os valores da variável linguística temperatura poderiam ser expressos como *alta*, *não alta*, *muito alta*, *bastante alta*, *não muito alta*, *alta mas não muito alta*. Neste caso, os valores da variável são sentenças formadas a partir do nome *alta*, da negação *não*, dos conectivos *e* e *mas*, e dos modificadores *muito* e *bastante*. Para gerar estes valores compostos, utilizam-se diversos termos, que podem ser divididos nas seguintes categorias:

- (a) *Termos Primários*: nomes de conjuntos fuzzy especificados em um determinado universo (por exemplo *alto*, *baixo*, *pequeno*, *médio*, *grande*, *zero*).
- (b) *Conectivos Lógicos*: A negação *NÃO*, conectivos *E* e *OU*, e conectivos mascarados, como *mas*, *porém*.
- (c) *Modificadores*, como *muito*, *pouco*, *levemente*, *extremamente*.
- (d) *Delimitadores*, como parênteses.

Termos primários estão associados diretamente a conjuntos fuzzy. Estes, são usualmente definidos de três maneiras diferentes:

- Funções de pertinência analíticas.
- Funções de pertinência lineares por partes, resultando em formas triangulares ou trapezoidais.
- Funções de pertinência discretizadas

A negação *NÃO* e os conectivos *E* e *OU* podem ser definidos em termos das operações de complementação, intersecção e união, respectivamente. Os conectivos mascarados correspondem funcionalmente aos conectivos *E* e *OU*. Por exemplo, *quente mas não muito quente* corresponde funcionalmente a *quente e não muito quente*.

Modificadores linguísticos servem para modificar o conjunto fuzzy associado a um termo linguístico, por meio de uma composição de funções. Por exemplo, seja  $m: [0,1] \rightarrow [0,1]$  correspondente ao modificador *muito*. Se  $b: U \rightarrow [0,1]$  corresponder ao conjunto fuzzy associado ao termo linguístico *baixo*, o conjunto fuzzy associado ao termo linguístico composto *muito baixo* será  $m(b(u))$ . Note que os modificadores não são funções definidas  $U \rightarrow [0,1]$ , mas sim  $[0,1] \rightarrow [0,1]$ .

Delimitadores são utilizados para evitar ambiguidades. Por exemplo,  $\{a \text{ E } b \text{ OU } c \text{ E } d\}$  pode ser ambíguo, ou seja, pode significar  $\{ (a \text{ E } b) \text{ OU } (c \text{ E } d) \}$  ou então  $\{ a \text{ E } (b \text{ OU } c) \text{ E } d \}$ . Para determinar com precisão a ordem em que os operadores lógicos devem ser considerados, utilizam-se os delimitadores. Quando esta ordem puder ser determinada sem ambiguidade, os delimitadores podem ser omitidos.

Utilizando todos estes tipos de termos, pode-se gerar um conjunto maior de valores para uma variável linguística a partir de uma coleção de termos primários. Por exemplo, usando *MUITO* em conjunto com *NÃO*, *E*, e o termo primário *grande*, pode-se gerar os conjuntos fuzzy *MUITO grande*, *MUITO MUITO grande*, *NÃO MUITO grande*, *grande E NÃO MUITO grande*, etc.

Formalmente, uma variável linguística é caracterizada por uma quintupla  $(X, T(X), U, G, M)$ , onde:

- X : Nome da variável  
T(X) : Conjunto de termos de X, ou seja, o conjunto de nomes dos valores linguísticos de X.  
U : Universo de discurso  
G : Regra sintática (geralmente uma gramática) para gerar os valores de X como uma composição de termos de T(X), conectivos lógicos (negação, interseção e união), modificadores e delimitadores.  
M : Regra semântica, para associar a cada valor gerado por G um conjunto fuzzy em U.

A principal função das variáveis linguísticas é fornecer uma maneira sistemática para uma caracterização aproximada de fenômenos complexos ou mal definidos. Em essência, a utilização do tipo de descrição linguística empregada por seres humanos, e não de variáveis quantificadas, permite o tratamento de sistemas que são muito complexos para serem analisados através de mecanismos matemáticos convencionais.

## 2.6 - Proposições Fuzzy

Seja uma frase da forma ( $\Pi$  é A), onde  $\Pi$  é o nome de uma variável linguística e A é um subconjunto fuzzy definido no universo de discurso U de  $\Pi$ . A frase ( $\Pi$  é A) é chamada de uma proposição fuzzy. De uma maneira mais genérica, podemos ter uma proposição fuzzy n-ária, efetuando o produto cartesiano de variáveis linguísticas e utilizando relações fuzzy ao invés de conjuntos fuzzy. Sejam  $x_1, x_2, \dots, x_n$  os nomes de m variáveis linguísticas cujos universos de discurso são respectivamente  $X_1, X_2, \dots, X_n$ . Seja R uma relação fuzzy definida em  $X_1 \times X_2 \times \dots \times X_n$ . Assim, a frase

$$((x_1, x_2, \dots, x_n) \text{ é } R)$$

é uma proposição fuzzy n-ária. Observe que quando n é igual a 1, esta se reduz a uma proposição fuzzy simples.

Proposições fuzzy podem ser combinadas utilizando-se diferentes operadores, gerando novas proposições fuzzy. Exemplos de operadores incluem os conectivos lógicos E e OU, bem como o operador de implicação SE...ENTÃO. As proposições fuzzy resultantes da combinação podem ser descritas em termos de relações fuzzy. A determinação do valor desta relação fuzzy, em função dos conjuntos fuzzy de cada operando, pode ser realizada de muitas maneiras diferentes.

Por exemplo, sejam as variáveis linguísticas de nomes x e y com universos de discurso X e Y respectivamente. Sejam A e B conjuntos fuzzy definidos respectivamente em X e Y. Por fim, sejam as seguintes proposições fuzzy (x é A) e (y é B). Conectando-se as duas proposições com o operador OU, temos:

$$(x \text{ é } A) \text{ OU } (y \text{ é } B)$$

Essa combinação gerará a proposição fuzzy binária

$$((x,y) \text{ é } R_{A \text{ OU } B})$$

onde  $R_{A \text{ OU } B}$  é uma relação fuzzy em  $X \times Y$ . No caso do operador OU, a relação é determinada por uma função  $f_{OU}$ , usualmente uma norma-s:

$$R_{A \text{ OU } B} = \{ \mu_R(x,y) / (x,y) \mid \mu_R(x,y) = f_{OU}(\mu_A(x), \mu_B(y)) \}$$

Se conectarmos as proposições com o operador E, teremos:

$$(x \text{ é } A) \text{ E } (y \text{ é } B)$$

que gerará a proposição fuzzy n-ária

$$((x,y) \text{ é } R_{A \text{ E } B})$$

onde  $R_{A \text{ E } B}$  é uma relação fuzzy em  $X \times Y$ . No caso do operador E, a relação é determinada por uma função  $f_E$ , normalmente uma norma-t:

$$R_{A \text{ E } B} = \{ \mu_R(x,y) / (x,y) \mid \mu_R(x,y) = f_E(\mu_A(x), \mu_B(y)) \}$$

O operador SE...ENTÃO é também conhecido como uma declaração condicional fuzzy. Este operador descreve a dependência do valor de uma variável linguística em relação ao valor de uma outra (independente). Seu uso é dado por:

$$\text{SE } (x \text{ é } A) \text{ ENTÃO } (y \text{ é } B)$$

Esta pode ser reduzida a uma proposição fuzzy binária:

$$((x,y) \text{ é } R_{A \rightarrow B})$$

A função de pertinência  $\mu_R(x,y)$  que define a implicação pode ser obtida a partir das funções de pertinência individuais  $\mu_A(x)$  e  $\mu_B(y)$  de inúmeras maneiras (Gaines, 1976; Baldwin & Pilsworth, 1980; Yager, 1980; Mizumoto, 1982; Whalen & Schott, 1983). De um modo genérico

$$R_{A \rightarrow B} = \{ \mu_R(x,y) / (x,y) \mid \mu_R(x,y) = f_{\rightarrow}(\mu_A(x), \mu_B(y)) \}$$

Várias declarações podem ser combinadas através do conectivo OU, formando assim um algoritmo fuzzy  $R^N$ , da forma:

$$R^N : R^1 \text{ OU } R^2 \text{ OU } R^3 \text{ ..... OU } R^n$$

Se  $A^1, A^2, \dots, A^n$  são conjuntos fuzzy em X, e  $B^1, B^2, \dots, B^n$  são conjuntos fuzzy em Y, um algoritmo fuzzy é dado por:

$$\begin{aligned} ((x,y) \text{ é } R^N) : & \quad \text{SE } (x \text{ é } A^1) \text{ ENTÃO } (y \text{ é } B^1) \\ & \quad \text{OU} \\ & \quad \text{SE } (x \text{ é } A^2) \text{ ENTÃO } (y \text{ é } B^2) \\ & \quad \text{OU} \\ & \quad \cdot \\ & \quad \cdot \\ & \quad \text{SE } (x \text{ é } A^n) \text{ ENTÃO } (y \text{ é } B^n) \end{aligned}$$

$$((x,y) \text{ é } R^N) : \quad ((x,y) \text{ é } R^1) \text{ OU } ((x,y) \text{ é } R^2) \text{ OU } \dots \text{ OU } ((x,y) \text{ é } R^n)$$

Assim, supondo que a função  $f_{OU}$  é associativa, ou seja  $f_{OU}(a, f_{OU}(b,c)) = f_{OU}(f_{OU}(a,b), c) = f_{OU}(a,b,c)$ , para o algoritmo fuzzy têm-se:

$$\begin{aligned} \mu_{R^N}(x,y) &= f_{OU}(\mu_{R^1}(x,y), \mu_{R^2}(x,y), \dots, \mu_{R^n}(x,y)) \\ &= f_{OU}[f_{\rightarrow}(\mu_{A^1}(x), \mu_{B^1}(y)), f_{\rightarrow}(\mu_{A^2}(x), \mu_{B^2}(y)), \dots, f_{\rightarrow}(\mu_{A^n}(x), \mu_{B^n}(y))] \end{aligned}$$

As equações acima foram desenvolvidas para variáveis simples A e B, mas podem ser estendidas para casos de mais variáveis. Normalmente as variáveis *antecedentes* são combinadas através do conectivo E:

$$\text{SE } ((x_1 \text{ é } A_1) \text{ E } (x_2 \text{ é } A_2) \text{ E } \dots \text{ E } (x_m \text{ é } A_m)) \text{ ENTÃO } (y \text{ é } B)$$

que se reduz a:

$$((x_1, x_2, \dots, x_m, y) \text{ é } R)$$

Neste caso, supondo-se do mesmo modo que  $f_E$  é associativa têm-se:

$$\mu_R(x_1, x_2, \dots, x_m, y) = f_{\rightarrow}(f_E(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \dots, \mu_{A_m}(x_m)), \mu_B(y))$$

Consideremos agora, em detalhe, quatro das funções de implicação já utilizadas em aplicações de controle. Por uma questão de simplicidade, as sentenças serão do tipo SE A ENTÃO B.

(1) *mínimo*, proposta por Mamdani (Mamdani, 1974; Mamdani & Assilian, 1975) como simplificação de *max-min*, originariamente proposta por Zadeh, baseada em princípios de lógica:

$$\mu_{R_{A \rightarrow B}}(x,y) = \mu_A(x) \wedge \mu_B(y)$$

(2) *produto*, utilizada por Ostergaard (Holmblad & Ostergaard, 1981) em aplicações práticas:

$$\mu_{R_{A \rightarrow B}}(x,y) = \mu_A(x) \cdot \mu_B(y)$$

Em ambos os casos acima, a combinação de n declarações condicionais fuzzy é efetuada através do conectivo OU, implementado por *max* (Lembessis, 1984). Para a função de implicação *min*, por exemplo, temos:

$$\mu_{R^N}(x,y) = \bigvee_j (\mu_{A_j}(x) \wedge \mu_{B_j}(y))$$

(3) *Lukasiewicz*, baseada na lógica multivalores de Lukasiewicz:

$$\mu_{R_{A \rightarrow B}}(x,y) = 1 \wedge (1 - \mu_A(x) + \mu_B(y))$$

onde '+' representa adição algébrica.

(4) *VSS*, baseada em 'variant of the system of standard sequences' (Rescher, 1969):

$$\mu_{R_{A \rightarrow B}}(x,y) = (1 - \mu_A(x)) \vee \mu_B(y)$$

Nos casos (3) e (4) acima, a combinação de n declarações condicionais dá-se através do conectivo E, implementado pelo *mínimo*. Assim, para *VSS*:

$$\mu_{R^N}(x,y) = \bigwedge_j (1 - \mu_{A_j}(x) \vee \mu_{B_j}(y))$$

## 2.7 - Regra de Inferência Composicional

Consideremos duas proposições fuzzy:

$$\begin{aligned} ((x,y) \in R^1) : & \quad \text{SE } (x \text{ é } A) \text{ ENTÃO } (y \text{ é } B) \\ ((y,z) \in R^2) : & \quad \text{SE } (y \text{ é } B) \text{ ENTÃO } (z \text{ é } C) \end{aligned}$$

Estas duas proposições podem ser compostas de forma a resultar em:

$$((x,z) \in R^{12}) : \quad \text{SE } (x \text{ é } A) \text{ ENTÃO } (z \text{ é } C)$$

Como já vimos, a composição  $R^{12} = R^1 \circ R^2$  é determinada pela composição  $\text{sup-}f$ . Por exemplo, esta pode ser definida ou por uma regra do tipo *max-min*

$$\mu_{R^{12}}(x,z) = \bigvee_y (\mu_{R^1}(x,y) \wedge \mu_{R^2}(y,z))$$

ou por uma do tipo *max-produto*

$$\mu_{R^{12}}(x,z) = \bigvee_y (\mu_{R^1}(x,y) \cdot \mu_{R^2}(y,z))$$

Quando conjuntos fuzzy discretos são empregados, as operações acima são equivalentes a produto interno de duas matrizes, com a multiplicação e a soma substituídas pelas operações min e max, respectivamente. É razoável que se use uma das duas definições acima em correspondência com a função de implicação adotada. Assim, com as funções de implicação envolvendo somente operadores *max* e *min* (VSS e



min), emprega-se a regra composicional *max-min*. No caso de implicações que envolvem operadores aritméticos, pode-se usar a regra *max-produto*.

A questão que se coloca agora é a seguinte: dada uma relação entre 2 variáveis fuzzy, qual o consequente para um dado antecedente? Esta questão é equivalente a dizermos que temos duas proposições fuzzy, a primeira uma proposição fuzzy simples, correspondendo a um fato, e a segunda uma proposição fuzzy binária, correspondendo a uma regra fuzzy. Consideremos duas variáveis linguísticas  $x$  e  $y$ , às quais são associados os conjuntos fuzzy  $A$  e  $B$  definidos abaixo:

$$\begin{aligned} A &= \{\mu_A(x)/x \mid x \in X\} \\ B &= \{\mu_B(y)/y \mid y \in Y\} \end{aligned}$$

e a relação de implicação entre elas  $R = \{\mu_R(x,y) / (x,y)\}$ ,  $\mu_R(x,y) = f_{\rightarrow}(\mu_A(x), \mu_B(y))$  gerando a proposição fuzzy binária

$$(x,y) \in R$$

correspondendo à regra fuzzy SE  $(x \in A)$  ENTÃO  $(y \in B)$ .

Se é dada uma proposição fuzzy simples (fato)  $(x \in A')$ , onde  $A'$  é dado por

$$A' = \{\mu_{A'}(x)/x\}$$

pode-se inferir, pela regra da composição:

$$(y \in B')$$

Utilizando-se a regra (lei) de inferência max-min(Zadeh, 1973) teremos que:

$$B' = A' \circ R$$

$$B' = \left\{ \bigvee_x (\mu_{A'}(x) \wedge \mu_R(x,y)) / y \right\}$$

A interpretação gráfica desta regra é mostrada pela figura 3 abaixo, onde a intersecção é interpretada pela t-norma *min* e a projeção em  $Y$  pelo operador *max*. Note que, em síntese, esta regra é constituída de duas etapas : intersecção da extensão cilíndrica de  $A'$ ,  $\bar{A}'$  com  $R_{A \rightarrow B}$ , e a projeção desta intersecção em  $Y$ .

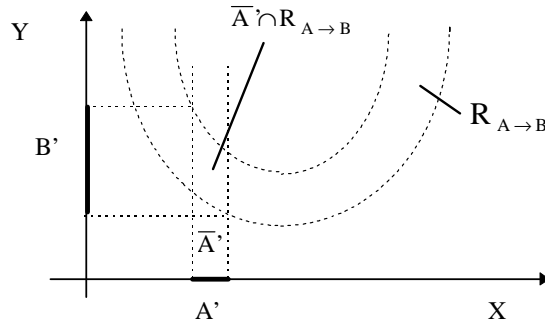


Figura 3 - Regra de Inferência Composicional

Em geral, a regra da composição é expressa por:

$$\mu_{B'}(y) = \sup_u \{\mu_{A'}(x) \mathcal{T} \mu_{R_{A \rightarrow B}}(x,y)\}$$

Quando mais de uma regra é acionada, as contribuições das diversas regras após a inferência são combinadas pelo operador de agregação ( $f_{OU}$ ). Por exemplo, supondo-se que  $B'_1, \dots, B'_n$  são os resultados

derivados das diversas regras acionadas, todos relacionados com a mesma variável linguística, o resultado combinado B' é:

$$B' = \bigcup_i^n B'_i$$

onde  $\bigcup$  representa o operador agregação (união, por exemplo)

A figura 4 ilustra o processo de inferência max-min quando existem 2 regras,  $A_i \rightarrow B_i$  e  $A_j \rightarrow B_j$ . A' é o fato de entrada, representado como um conjunto fuzzy.

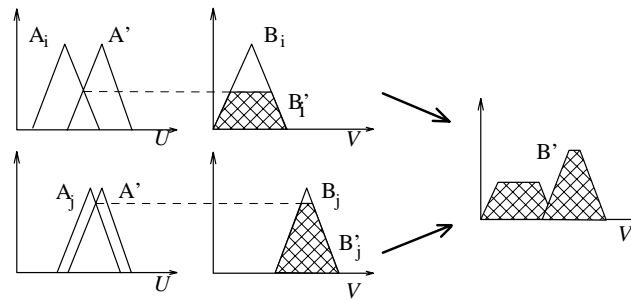


Figura 4: Mecanismo de Inferência Fuzzy

### 3 - MODELAGEM FUZZY DE SISTEMAS E PROCESSOS

Para modelar um sistema é necessário descrever o comportamento do mesmo para, por exemplo, sua análise, simulação e/ou projeto de controladores. Para os propósitos deste artigo, consideraremos três categorias de representação de sistemas : equações matemáticas, regras fuzzy (linguísticas) e redes neurais artificiais (Yamakawa, 1993).

Equações relacionais ou diferenciais descrevem a dinâmica ou a cinética de sistemas (ou o conhecimento sobre o sistema) em uma forma muito conveniente. Se a relação entre a entrada x e a saída f(x) do sistema ou a relação entre a causa x e a resposta f(x) é obtida como mostrado na figura 5, então f(x) é descrito por:

$$f(x) = (x - 3)^2$$

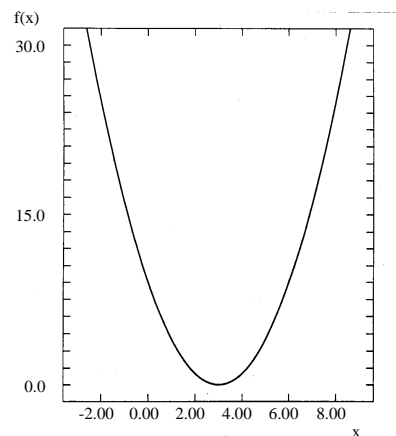


Figura 5 - Exemplo de Função f(x) descrita por Equação Matemática

A descrição de um sistema com este tipo de equação é de grande importância. É, no entanto, muito difícil de identificar a equação que descreve o sistema exatamente, especialmente em casos de sistemas complexos e com múltiplas variáveis. Em adição, é também muito difícil reescrever esta equação quando a relação entre  $x$  e  $f(x)$  varia. Portanto, esta descrição não é apropriada para a maioria de sistemas complexos tais como, sistemas não lineares e sistemas variantes no tempo. A medida que a complexidade do sistema aumenta, a possibilidade de descrever um sistema com equações matemáticas diminui (Princípio da Incompatibilidade de Zadeh).

Uma outra abordagem consiste em descrever a relação entre  $x$  e  $f(x)$  através de regras do tipo:

$$\text{Regra } i : \text{ Se } x \text{ é } A_i \text{ Então } f(x) \text{ é } B_i, \quad i = 1, \dots, N$$

onde  $x$  representa a variável independente e  $f(x)$  a variável dependente, sendo  $A_i$  e  $B_i$  constantes linguísticas e  $N$  o número de dados experimentais que descreve a função. Quando  $A_i$  e  $B_i$  são constantes linguísticas com valores numéricos exatos (como nos sistemas clássicos da inteligência artificial), teríamos o seguinte, com relação ao exemplo anterior (Figura 6).

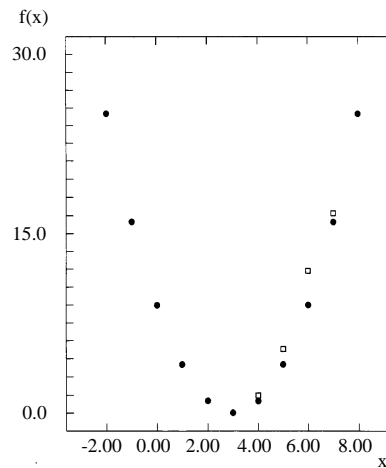


Figura 6 - Exemplo de Função  $f(x)$  descrita por Regras Linguísticas com Valores Exatos.

Regra 1	Se $x$ é -2	Então $f(x)$ é 25
Regra 2	Se $x$ é -1	Então $f(x)$ é 16
Regra 3	Se $x$ é 0	Então $f(x)$ é 9
Regra 4	Se $x$ é 1	Então $f(x)$ é 4
Regra 5	Se $x$ é 2	Então $f(x)$ é 1
Regra 6	Se $x$ é 3	Então $f(x)$ é 0
Regra 7	Se $x$ é 4	Então $f(x)$ é 1
Regra 8	Se $x$ é 5	Então $f(x)$ é 4
Regra 9	Se $x$ é 6	Então $f(x)$ é 9
Regra 10	Se $x$ é 7	Então $f(x)$ é 16
Regra 11	Se $x$ é 8	Então $f(x)$ é 25

A vantagem desta descrição é a facilidade em mudar a descrição do sistema. Por exemplo, quando somente uma variação local é verificada (indicada por quadrados na figura 6) é necessário somente modificar os valores correspondentes dos consequentes das regras (e.g. : Regra 7 : 1 para 2; Regra 8 : 4 para 6; Regra 9 : 9 para 13; Regra 10 : 16 para 18) pois as regras são independentes umas das outras. Isto mostra que a descrições na forma de regras são apropriadas para sistemas com aprendizagem, sistemas auto-organizáveis e sistemas adaptativos. Por outro lado, existem também algumas desvantagens. Quando é dado que  $x=1$  a conclusão obtida é  $f(x) = 4$  a partir do casamento entre o dado  $x=1$  com o antecedente  $x=1$  da regra 4. Este é, essencialmente, o procedimento básico de inferência em sistemas baseados em regras da inteligência artificial clássica. Contudo, se é dado que  $x = 1.5$  nada pode ser inferido a partir do conjunto de regras mencionado pois não existe nenhuma delas que possui um antecedente com  $x=1.5$ , exatamente. Isto mostra que os sistemas

clássicos binários são pouco eficientes com relação a conhecimento impreciso, com ruído ou com variação em dados de entrada, e que é necessário uma enorme quantidade de regras (base de conhecimento) para se obter um resultado ou desempenho significativo. Consequentemente, este tipo clássico de inferência demanda tempo devido a necessidade de se verificar o casamento entre um dado e uma base de regras (em geral muito grande). Em adição, este mecanismo de inferência não é apropriado para manusear possíveis regras contraditórias existentes pois, caso contrário, poderia produzir conclusões também contraditórias a partir de um mesmo dado. Estas desvantagens estão também presentes nos sistemas clássicos de inteligência artificial, baseados somente no processamento simbólico, mas não no processamento do significado dos termos linguísticos. Alternativamente, poderíamos utilizar regras do mesmo tipo anterior, mas interpretando-as como regras fuzzy. Neste caso  $A_i$  e  $B_i$  seriam termos linguísticos associados à variável  $x$ , cada um destes termos associado a um conjunto fuzzy a fim de se estabelecer seu significado.

Assim, poderíamos descrever a relação entre  $x$  e  $f(x)$  por :

Regra 1 : Se  $x$  está no entorno de -2

Então  $f(x)$  está no entorno de 25

Regra 2 : Se  $x$  está no entorno de -1

Então  $f(x)$  está no entorno de 16

...

Regra 11: Se  $x$  está no entorno de 8

Então  $f(x)$  está no entorno de 25

Neste caso, a relação precisa entre  $x$  e  $f(x)$  da figura 6 é fuzzyficada, tornando-a contínua como mostrado pela figura 7.

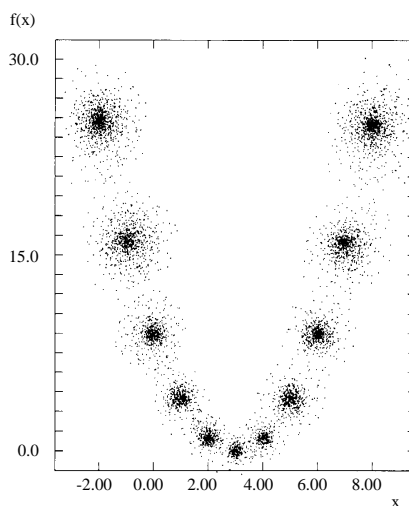


Figura 7 - Exemplo de Função  $f(x)$  descrita por Regras Linguísticas Fuzzy

Esta relação fuzzy fornece valores razoáveis para qualquer dado no universo de interesse, e.g. :  $x=-1.5$ ;  $x=3.2$ ;  $x=4.3$ , etc... através do mecanismo (regra) de inferência fuzzy e defuzzyficação. Em outras palavras, inferência fuzzy e defuzzyficação proporcionam uma forma razoável e simples de interpolação, com muito menos dados e dados inexatos. Sob este ponto de vista, a inferência fuzzy exhibe um comportamento similar àquele descrito por funções matemáticas. Ainda mais, é muito mais fácil re-elaborar regras fuzzy do que equações matemáticas quando as características do sistema ou processo são variantes. Em geral, neste caso, somente umas poucas regras são adicionadas ou revisadas independentemente das outras regras. Em contraste, no caso de equações matemáticas vários parâmetros dever ser re-avaliados simultaneamente, e.g. ordem, coeficientes, etc ... Logo, as características principais da modelagem fuzzy de sistemas e processos podem ser resumidas em:

- 1 - É apropriada para descrever sistemas complicados com uma razoável quantidade de conhecimento;
- 2 - É fácil selecionar as variáveis linguísticas e os correspondentes valores a serem usados nas regras fuzzy entre uma categoria relativamente pequena de palavras;

- 3 - A memorização do conhecimento é mais simples;
- 4 - Permite uma maior facilidade na comunicação dos modelos com projetistas e analistas devido ao uso da linguagem natural.

Note que, enquanto a inferência baseada na regra de inferência *modus ponens* nos sistemas clássicos de inteligência artificial é baseada somente em processamento simbólico, a inferência fuzzy (raciocínio aproximado) é baseada tanto no processamento simbólico quanto no processamento do significado.

Regras linguísticas (fuzzy ou não) podem representar um conhecimento explicitamente e este ser usado também explicitamente para inferência. Redes neurais artificiais, ao contrário, representam tanto o conhecimento quanto a inferência implicitamente. Portanto regras linguísticas representam conhecimento estruturado enquanto que as redes neurais representam, em geral, conhecimento não estruturado. Existem, no entanto, classes de redes neurais fuzzy que permitem a detecção do conhecimento e então sua estrutura (Figueiredo et.al., 1993).

Finalmente, outra alternativa para modelagem (e também controle) de sistemas complexos consiste em descrevê-lo por um conjunto de parâmetros, cujo exemplo típico são as redes neurais artificiais clássicas (Hecht-Nielsen, 1990). Uma rede neural é composta por um número de elementos simples interconectados, cada elemento realizando uma operação de agregação a partir de um modelo de um neurônio físico. Quando  $w_{ij}$  ( $i = 1, \dots, n$ ) é o peso atribuído ao sinal de entrada  $p_i$  para o  $j$ -ésimo neurônio e  $\theta_j$  e  $q_j$  são um limiar e o sinal de saída do  $j$ -ésimo neurônio, respectivamente, o sinal de saída é tipicamente dado por :

$$q_j = h\left(\sum_i w_{ij} \cdot p_i - \theta_j\right)$$

onde  $h$  é em geral uma função de agregação sigmoidal, e.g. :

$$h(x) = [1 + \exp(-x)]^{-1}$$

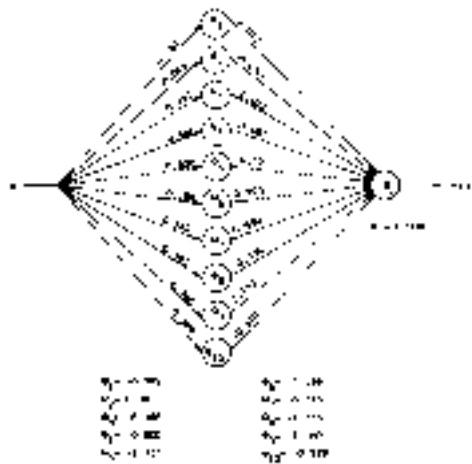


Figura 8 - Rede Neural para a Descrição da Função  $f(x)$

Utilizando-se deste modelo de neurônio, a relação existente entre  $x$  e  $f(x)$  pode ser caracterizada pela rede neural como mostrado na figura 8, onde tanto os pesos  $w_{ij}$  quanto os limiares  $\theta_j$  foram obtidos após inúmeras iterações de aprendizado utilizando-se dos algoritmos clássicos de propagação retroativa (*backpropagation*). Os dados utilizados foram aqueles correspondentes às 11 regras com termos linguísticos constantes apresentados anteriormente. Como pode ser verificado pela figura 9 a rede neural é capaz de realizar interpolações de forma similar ao caso da inferência fuzzy. Contudo é em geral muito difícil de entender, apenas pela observação do conjunto de parâmetros, como a rede neural se comporta. Em adição, existem problemas na determinação da topologia mais apropriada para o problema em questão (número de neurônios, camadas intermediárias, recursividade ou não-recursividade, etc...); bem como problemas de convergência durante o processo de aprendizagem. Se o sistema está sujeito a variações em seu

comportamento, nova aprendizagem deve, em geral, ser executada para se obter um novo conjunto de pesos e limiares correspondentes e muitas vezes os resultados não são razoáveis. Isto é, existe uma maior dificuldade em projetar sistemas para fins de modelagem (e controle) com redes neurais do que com sistemas fuzzy. Esta é uma das razões pela qual os sistemas fuzzy tem sido mais populares na solução de problemas práticos do que as redes neurais. No entanto, pesquisa recente vem gerando classes de sistemas neuro-fuzzy, onde as limitações e vantagens de ambas as abordagens são tratadas de forma a criar sistemas mais eficientes tanto do ponto de vista da solução de problemas, quanto da questão de análise e projeto dos mesmos (Pedrycz, 1992; Gomide & Rocha, 1992-b; Gomide & Rocha, 1992-c; Figueiredo et.al. 1993). Devido a limitações de espaço, este assunto (esperamos) deverá ser objeto de um artigo no futuro.

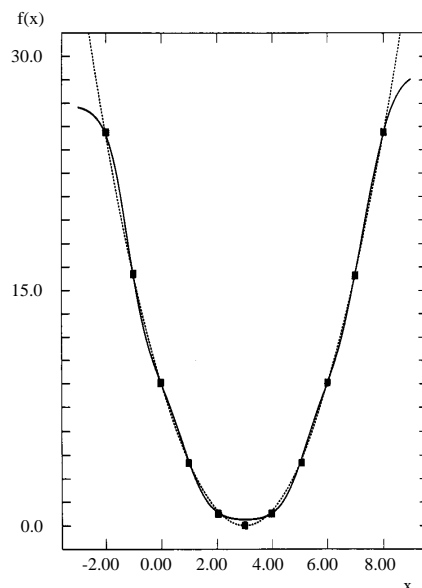


Figura 9 - Exemplo da Função  $f(x)$  descrita por uma Rede Neural

Como vimos, a representação de conhecimento por meio de regras fuzzy pode ser utilizada, também, para se modelar um processo do ponto de vista de seu conhecimento qualitativo e quantitativo.

Para concluir esta seção, apresenta-se um procedimento para a modelagem de processos. A princípio, é necessário algum conhecimento a priori do comportamento do processo. Caso este conhecimento não esteja disponível, alguns experimentos poderão fornecer um mínimo de conhecimento de modo a dar prosseguimento ao procedimento de modelagem.

Inicialmente é necessário definir quais são as variáveis físicas a serem consideradas. A partir daí, é necessário decidir:

- O número de variáveis linguísticas (V.L.)
- Os termos linguísticos utilizados para as V.L.
- Os parâmetros associados às regras

A seguir é necessário escrever um conjunto de regras que descrevam o comportamento do processo (por exemplo, a variação de uma variável observável relacionada com a saída do processo) em função das variáveis linguísticas que provocam a modificação do comportamento. As funções de pertinência são constituídas e o conjunto de regras processado, para os valores definidos das variáveis envolvidas. O modelo é testado para verificar sua fidelidade ao processo real ou não. Caso a resposta proporcionada pelo modelo não corresponda àquela do processo real, modificações no número de regras, número de antecedentes e consequentes, nas funções de pertinência e parâmetros do modelo devem ser efetuadas. Uma metodologia baseada em dados experimentais e no cálculo de relações fuzzy é descrito em (Pedrycz, 1989). Técnicas de aprendizagem baseadas em redes neuro-fuzzy e algoritmos genéticos vem sendo também frequentemente utilizadas como instrumentos de projeto automático de regras e funções de pertinência (Takagi, 1993 ; Oliveira et.al. 1994).

## 4 - SISTEMAS DE CONTROLE FUZZY

### 4.1 Definições Gerais

A idéia básica em controle fuzzy é modelar as ações a partir de conhecimento especialista, ao invés de, necessariamente, modelar o processo em si. Isso nos leva a uma abordagem diferente dos métodos convencionais de controle de processos, onde os mesmos são desenvolvidos via modelagem matemática dos processos de modo a derivar as ações de controle como função do estado do processo. A motivação para esta nova abordagem veio de casos onde o conhecimento especialista de controle era disponível, seja por meio de operadores ou de projetistas, e os modelos matemáticos envolvidos eram muito custosos, ou muito complicados para serem desenvolvidos.

A estrutura de um processo controlado por um controlador fuzzy é mostrada na figura 10, enfatizando-se seus componentes básicos: a interface de fuzzyficação, a base de conhecimento, a base de dados, o procedimento de inferência e a interface de defuzzyficação.

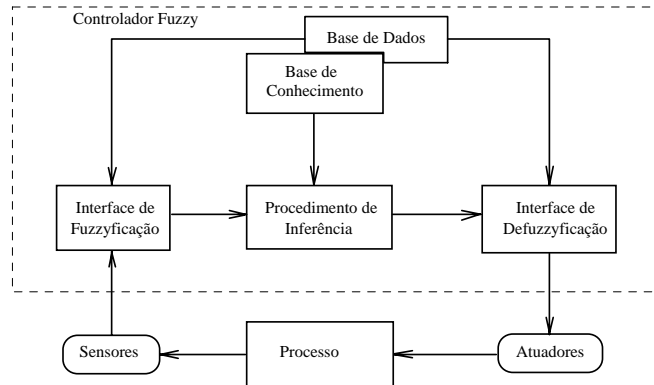


Figura 10 : Estrutura Básica de um Controlador Fuzzy

A *interface de fuzzyficação* toma os valores das variáveis de entrada, faz um escalonamento para condicionar os valores a universos de discurso normalizados e fuzzyfica os valores, transformando números em conjuntos fuzzy, de modo que possam se tornar instâncias de variáveis linguísticas. A *base de conhecimento* consiste de uma base de regras, caracterizando a estratégia de controle e suas metas. A *base de dados* armazena as definições necessárias sobre discretizações e normalizações dos universos de discurso, as partições fuzzy dos espaços de entrada e saída e as definições das funções de pertinência. O *procedimento de inferência* processa os dados fuzzy de entrada, junto com as regras, de modo a inferir as ações de controle fuzzy, aplicando o operador de implicação fuzzy e as regras de inferência da lógica fuzzy. A *interface de defuzzyficação* transforma as ações de controle fuzzy inferidas em ações de controle não-fuzzy. Em seguida, efetua um escalamento, de modo a compatibilizar os valores normalizados vindos do passo anterior com os valores dos universos de discurso reais das variáveis.

Após a inferência da ação de controle fuzzy, é necessária a determinação de uma ação de controle não fuzzy que melhor represente a decisão fuzzy, para ser efetivamente enviada ao controle. Apesar de não haver nenhum procedimento sistemático para a escolha da estratégia de defuzzyficação, as mais comuns incluem: o critério do máximo (MAX), que escolhe o ponto onde a função inferida tem seu máximo, a média dos máximos (MDM), que representa o valor médio dentre todos pontos de máximo quando existe mais de um máximo, e o método do centro de área (CDA), que retorna o centro de área da função inferida (figura 11).

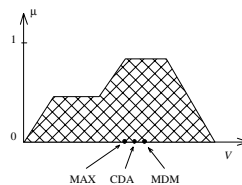


Figura 11 - Métodos de Defuzzyficação

$$CDA = \frac{\sum_{k=1}^n \mu_{B'}(v_k) \cdot v_k}{\sum_{k=1}^n \mu_{B'}(v_k)} \quad MOM = \sum_{k=1}^l \frac{v_k}{l}$$

onde  $v_k \in V^\uparrow \subseteq V$ ,  $V^\uparrow = \left\{ v_k \mid \mu_{B'}(v_k) = \max_j \mu_{B'}(v_j), v_j \in V \right\}$ ,  $l$  é o número de elementos de  $V^\uparrow$  (discretização de  $V^\uparrow$ ) e  $n$  é a discretização de  $V$

## 4.2 Controlador Fuzzy Clássico

Em um controlador fuzzy clássico, as variáveis de entrada são tipicamente o *erro*, gerado a partir da diferença entre o sinal de referência e a saída do processo, e a *variação do erro*, normalmente gerada a partir do erro. A variável de saída do controlador é normalmente a *variação no controle*. A opção por uma saída incremental, ao invés de absoluta, visa proporcionar uma maior suavidade na resposta do sistema em decorrência das ações de controle. Estabelecendo uma ligação com as seções anteriores, as variáveis fuzzy *erro* e *variação do erro* são subconjuntos fuzzy em seus respectivos universos. Elas poderiam ser definidas, por exemplo, como correspondentes aos valores reais medidos das entradas. Entretanto, um procedimento mais geral faz uso de universos discretos e normalizados, principalmente quando deseja-se a implementação do controlador em um computador digital. Nesse caso, os conjuntos correspondentes às variáveis fuzzy são definidos por conjuntos discretos.

Os universos aqui considerados para as variáveis fuzzy são discretos, finitos e normalizados. Assim, torna-se necessário utilizar fatores de escala para converter os valores reais medidos pelos sensores para universos normalizados. Além disto, deve ser efetuada uma quantização, de modo a condicionar os valores normalizados ao universo de discurso discreto. Para o caso dos valores medidos *erro* e *variação de erro*, temos (em um instante  $i$ , por exemplo):

$$e_i = (GE \times \text{erro})_{\text{quantizado}} \quad e_{ce_i} = (GCE \text{ variação do erro})_{\text{quantizado}}$$

onde  $GE$  e  $GCE$  são fatores de escala. A escolha dos fatores de escala é parte do procedimento de sintonia do controlador e esses fatores podem ser expressos por constantes ou por funções (do erro, por exemplo). Se o universo da saída do controlador também for normalizado, a exemplo das variáveis de entrada, um fator de escala  $GO$  é usado, após a defuzzificação, para mapear valores quantizados para valores reais compatíveis com a entrada do processo. Este fator de escala também deve ser ajustado de acordo com o comportamento da resposta.

A estratégia de controle é descrita por um conjunto de declarações condicionais, ou regras, linguísticas. Conforme mencionado em seções anteriores, duas entradas são consideradas para cada saída: o *erro*,  $E$ , e a *variação do erro*,  $CE$ . A saída é a *variação na saída do controlador*,  $U$ . O conjunto de regras é, então, da forma:

$$\begin{aligned} R^N: & \quad SE (E \in E^1) E (CE \in CE^1) \text{ ENTÃO } (U \in U^1) \\ & \quad OU \\ & \quad SE (E \in E^2) E (CE \in CE^2) \text{ ENTÃO } (U \in U^2) \\ & \quad OU \\ & \quad \cdot \\ & \quad \cdot \\ & \quad \cdot \\ & \quad OU \\ & \quad SE (E \in E^n) E (CE \in CE^n) \text{ ENTÃO } (U \in U^n) \end{aligned}$$

onde  $E^j$ ,  $CE^j$  e  $U^j$  são subconjuntos fuzzy dos universos  $E = \{e\}$ ,  $CE = \{ce\}$ ,  $U = \{u\}$ .

Cada regra  $R^j$  pode ser expressa, através de sua função de pertinência, como:

$$\mu_{R^j}(e, ce, u) = f_{\rightarrow} ( f_E ( \mu_{E^j}(e) ), \mu_{CE^j}(ce) ), \mu_{U^j}(u) )$$



A combinação de todas as n regras é expressa matematicamente por:

$$\mu_{R^N}(e, ce, u) = f_{OU} [ f_{\rightarrow} ( f_E ( \mu_{E1}(e) , \mu_{CE1}(ce) ) , \mu_{U1}(u) ) , f_{\rightarrow} ( f_E ( \mu_{E2}(e) , \mu_{CE2}(ce) ) , \mu_{U2}(u) ) , \dots \\ \dots f_{\rightarrow} ( f_E ( \mu_{En}(e) , \mu_{CEn}(ce) ) , \mu_{Un}(u) ) ]$$

Desta forma, a estratégia de controle é representada por uma matriz tridimensional  $\mu_{R^N}$ . O controlador é então solicitado a fornecer, a partir do valores do erro e da variação do erro, e utilizando-se da regra de inferência composicional, uma saída apropriada. Assim, em um instante  $i$ , quando as entradas determinísticas são  $e_i$  e  $ce_i$ , o conjunto fuzzy da saída será dado por:

$$U_i = (E_i \ E \ CE_i) \circ R^N$$

onde  $E_i$  e  $CE_i$  são os subconjuntos fuzzy das entradas e  $U_i$  é o subconjunto fuzzy da saída. Utilizando  $f_{OU}$  igual ao máximo,  $f_E$  igual ao mínimo e usando a regra de inferência max-min, têm-se:

$$\mu_{U_i}(u) = \bigvee_e \bigvee_{ce} ( \mu_{E_i}(e) \wedge \mu_{CE_i}(ce) \wedge \mu_{R^N}(e, ce, u) )$$

Os cálculos podem ser simplificados se considerarmos que os subconjuntos fuzzy das entradas consistem de valores reais não fuzzy ('singletons'), isto é:

$$\begin{array}{ll} \mu_{E_i}(e) = 1 & \text{se } e = e_i \\ = 0 & \text{se } e \neq e_i \end{array} \qquad \begin{array}{ll} \mu_{CE_i}(ce) = 1 & \text{se } ce = ce_i \\ = 0 & \text{se } ce \neq ce_i \end{array}$$

Com isto,

$$\mu_{U_i}(u) = \bigvee_e \bigvee_{ce} ( 1 \wedge 1 \wedge \mu_{R^N}(e, ce, u) ) \quad \text{para } e = e_i$$

$$\mu_{U_i}(u) = \bigvee_e \bigvee_{ce} ( 0 \wedge 0 \wedge \mu_{R^N}(e, ce, u) ) \quad \text{para } e \neq e_i$$

$$\text{Ou seja: } \begin{array}{ll} \mu_{U_i}(u) = \mu_{R^N}(e, ce, u) & \text{para } e = e_i \\ = 0 & \text{nos outros casos} \end{array}$$

Portanto, o conjunto fuzzy que representa a saída do controlador é dado por:

$$\mu_{U_i}(u) = f_{OU} [ f_{\rightarrow} ( \mu_{E1}(e_i) \wedge \mu_{CE1}(ce_i) , \mu_{U1}(u) ) , f_{\rightarrow} ( \mu_{E2}(e_i) \wedge \mu_{CE2}(ce_i) , \mu_{U2}(u) ) , \dots \\ \dots f_{\rightarrow} ( \mu_{En}(e_i) \wedge \mu_{CEn}(ce_i) , \mu_{Un}(u) ) ]$$

### 4.3 Algoritmo de Controle

Nesta seção é apresentada uma maneira simples de se tratar as regras, com vistas à implementação do algoritmo de controle. São também comentados aspectos de ordem prática que devem ser levados em consideração quando da implementação e utilização do algoritmo.

O exemplo visto na seção anterior pode ser resolvido de uma maneira mais eficiente se o espaço de estado  $E \times CE$  for representado por uma 'grade' e definirmos o centro da regra  $(e_R, ce_R)$  nesse espaço como o ponto onde  $\mu_{Ej}(e) = \mu_{CEj}(ce) = 1$ . Com isto, é possível utilizar um simples número para representar o conjunto fuzzy associado com o valor linguístico de uma variável em uma regra. A ação do controlador pode também ser representada da mesma forma; os valores numéricos na grade representam a posição no universo em que o valor de pertinência é 1. Assim, o armazenamento de regras torna-se bastante simples; conhecidos o conjunto suporte e o centro da regra, torna-se fácil 'recuperar' os conjuntos fuzzy correspondentes. A região de influência de cada regra é determinada por um quadrado de 5 unidades de lado e seu grau de influência dentro deste quadrado, no caso das funções de pertinência triangulares já mencionadas anteriormente, é representado, por:

	$c_{eR}$
	0,3 0,3 0,3 0,3 0,3
	0,3 0,7 0,7 0,7 0,3
$e_R$	0,3 0,7 1,0 0,7 0,3
	0,3 0,7 0,7 0,7 0,3
	0,3 0,3 0,3 0,3 0,3

A expressão  $\mu_{E1}(e_i) \wedge \mu_{CE1}(c_{ei})$ , que determina a região de influência de uma regra, descreve o quanto as entradas do controlador são compatíveis com os antecedentes de uma regra  $j$ . Esta expressão é normalmente chamada de 'degree of fulfilment' da regra  $j$  no instante  $i$ , ou DOF.

Para um ponto  $(e_i, c_{ei})$  no espaço de estado, as regras que contribuem para o cálculo da ação do controlador são aquelas cujos centros estão a uma distância  $\delta$  de  $(e_i, c_{ei})$ . Se conjuntos suporte de tamanhos diferentes forem empregados, os quadrados tornam-se paralelogramos e a distância de uma regra do ponto  $(e_i, c_{ei})$  não será uniforme em todas as direções, tornando a implementação menos simples. Dependendo do método de defuzzificação empregado, menos regras podem contribuir para o cálculo da ação do controlador. Se COG for utilizado, todas as regras dentro de uma distância  $\delta$  contribuem e devem ser incluídas nos cálculos. As regras mais próximas de  $(e_i, c_{ei})$  terão uma contribuição maior que as mais distantes, evidentemente. No entanto se MOM for empregado, somente as regras mais próximas de  $(e_i, c_{ei})$  contribuirão, e apenas elas precisam ser incluídas nos cálculos. Isto se deve ao fato de que, no método MOM, somente os picos no conjunto fuzzy da saída são relevantes. Esse método efetivamente seleciona as regras com maior DOF, ou seja, aquelas mais próximas de  $(e_i, c_{ei})$ . Em virtude desta característica, é possível criar algoritmos muito simples em que conjuntos fuzzy não são usados explicitamente no cálculo da ação do controlador.

A região de influência de uma regra dependerá da abrangência, em termos do universo considerado, dos conjuntos fuzzy das entradas. Portanto, o tamanho do conjunto suporte certamente influi no cálculo da ação de controle.

Consideremos conjuntos com a mesma abrangência mas com funções de pertinência de diferentes formas. A forma dos conjuntos dos antecedentes de cada regra afeta a determinação de DOF, que terá um efeito no consequente (ou saída) de cada regra, através do mecanismo de inferência, e no resultado final. A forma do conjunto do consequente de cada regra afeta o conjunto fuzzy da saída diretamente. No entanto, vejamos o que ocorre com a saída determinística do controlador: se MOM for usado, a forma dos conjuntos dos antecedentes não tem importância, desde que sejam simétricos e de forma aproximadamente "triangular". Só é necessário saber (através de DOF) a distância da regra de  $(e_i, c_{ei})$ ; as magnitudes absolutas dos valores da função de pertinência não são importantes; apenas as magnitudes em relação às de outra regra. Similarmente, a forma dos conjuntos do consequente também não é importante. No caso do método COG, a forma dos conjuntos suporte é importante, mesmo quando há simetria. A área sob a curva  $\mu_{U_j}(u)$  é afetada diretamente pelo consequente de cada regra e indiretamente por DOF.

Além das funções de implicação apresentadas até agora, diversas outras têm sido propostas na literatura mas geralmente considerando-as apenas sob o ponto de vista lógico (Gaines, 1976; Baldwin & Pilsworth, 1980; Yager, 1980; Mizumoto, 1982). Com o intuito de verificar também o lado prático, consideraremos alguns casos com as funções de implicação já apresentadas, em conjunto com os métodos de defuzzificação COG e MOM.

#### 1. Regra única a uma distância de $(e_i, c_{ei})$ :

As Figs. 12a, 12b e 12c mostram o que ocorre no cálculo da ação do controlador para diferentes valores de DOF. Cabe aqui um parênteses: em uma situação com várias regras, no caso das implicações *Luk.* e *VSS*, o efeito criado pelo platô que cobre todo o universo para  $DOF = 0$  é contrabalançado pela utilização do operador *min* na combinação dos diversos conjuntos fuzzy de saída de cada regra (conectivo OU). Se *max* fosse utilizado, incorreríamos no absurdo de se ter ação máxima mesmo quando a regra não apresenta contribuição alguma. Retornando à figura, com *VSS* e *Luk.* os resultados produzidos por COG e MOM são diferentes por causa do platô. Particularmente no caso de *VSS*, MOM interpreta melhor o efeito da regra, pois o método COG, em interpretando o pico mais o platô, desloca a saída  $u_i$  do valor indicado pelo centro da regra. Isto é indesejável, pois o platô possui nenhuma informação legítima. Cumpre observar que o uso da implicação *min* resulta na saída *nada*, enquanto que *Luk.* e *VSS* dão como resultado a saída *desconhecida*.



Figura 12 - Transformação da Ação da Regra para Diferentes Valores do DOF

2. Regras com ações diferentes a uma distância de  $(e_i, ce_i)$ .

Duas regras com consequentes  $U^1$  e  $U^2$  situadas a uma distância 1 de  $(e_i, ce_i)$  estão mostradas nas Figs. 13a, 13b e 13c. Com *min*, ambas as regras contribuem igualmente para o conjunto de saída e a ação interpretada é a mesma com MOM ou COG. Isto se aplica independentemente da posição de  $U^1$  e  $U^2$  nos seus universos. As implicações *VSS* e *Luk.* não são capazes de fornecer uma ação prática com significado, pois acabam por não levar em conta as contribuições individuais de cada regra. O platô, em conjunto com a combinação *min* das saídas individuais de cada regra, mascara completamente a saída de uma regra com menor prioridade ( $DOF = 0,3$ ) que porventura exista. A utilização do método COG é impraticável neste caso.

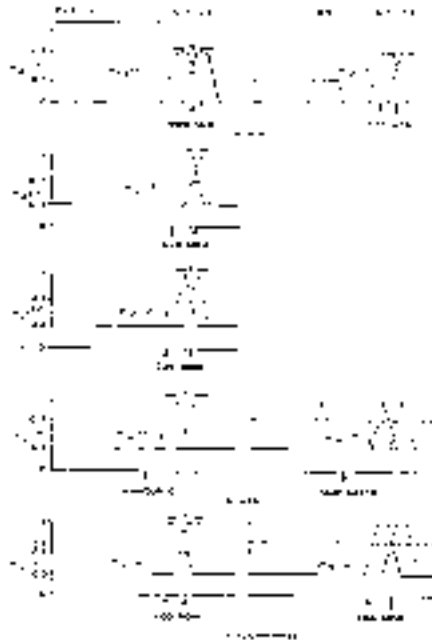


Figura 13 - Ação de Controle  $\mu_{U1}(u)$  para Duas Regras à Distância 1 de  $(e_i, ce_i)$  e uma terceira regra à distância 2

### 3. Uma regra em (ei,cei) e uma seleção de regras a outras distâncias

As várias contribuições são mostradas nas Figs. 14a, 14b e 14c abaixo. Com as implicações *VSS* e *Luk.*, o efeito do platô produzido pelas regras de menor contribuição ( $DOF = 0,7$ ) faz com que permaneça apenas a regra predominante após a combinação de todas as saídas utilizando *min* (conectivo OU). O método MOM é o mais indicado neste caso.



Figura 14 - Ação de Controle  $\mu_{U_i}(u)$  para uma regra em  $(e_i, ce_i)$ , duas regras à distância 1 e uma à distância 2

Considerando os três casos, em geral a implicação *min* é aquela que produz os melhores resultados sob o ponto de vista de controle. As implicações *VSS* e *Luk.* produzem saídas "desconhecidas" em certos casos, conforme demonstrado. Do ponto de vista lógico, as saídas são corretas; entretanto, do ponto de vista prático, é mais vantajoso produzir uma saída próxima àquela indicada por regras contraditórias do que uma saída "zero". Com as implicações *VSS* e *Luk.*, por vezes não compensa utilizar o método COG, pois os resultados produzidos podem não representar adequadamente as regras existentes (devido ao platô). Cumpre salientar que a implicação *produto* comporta-se de uma maneira semelhante à *min*. A forma dos conjuntos fuzzy da saída é similar, embora mais 'suave' do que com *min*. O uso do método COG é mais indicado neste caso.

A análise efetuada acima mostra que diversos fatores, tais como funções de implicação, métodos de defuzzificação, forma das funções de pertinência, influem no cálculo da ação de controle. Todos estes fatores devem ser considerados em conjunto; os aspectos práticos, e não apenas os lógicos, devem ser cuidadosamente considerados.

## 5 - PARÂMETROS, SINTONIZAÇÃO E MONITORAÇÃO DE CONTROLADORES FUZZY

Durante o projeto de controladores fuzzy, é necessária a definição de alguns parâmetros. Estes parâmetros são definidos a partir da experiência do projetista ou através de experimentos. Dado um processo, alguns dos parâmetros são fixos, dentro das condições normais de operação, sendo que outros precisam ser alterados de tempos em tempos. Os parâmetros fixos são chamados de parâmetros estruturais e os variáveis são chamados de parâmetros de sintonização:

- Parâmetros estruturais
  - Número de variáveis de saída
  - Número de variáveis de entrada
  - Recursos de operação sobre os dados de entrada (somadas, multiplicações, etc)
  - Variáveis linguísticas

- Funções de pertinência parametrizadas
- Intervalos de discretização e normalização
- Estrutura da base de regras
- Conjunto básico de regras

- Parâmetros de Sintonização
  - Universo de discurso das variáveis
  - Parâmetros das funções de pertinência, tais como altura, largura ou conjunto suporte
  - ganhos e offset das entradas e saídas

Certas propriedades da base de regras precisam ser testadas, dentre elas: a completude, consistência, interação e robustez. Esta última está relacionada com a sensibilidade do controle diante de algum comportamento anômalo não modelado ou ruído. Um método possível para se medir a robustez consiste em introduzir um ruído aleatório com média e variância conhecidas e observar-se a alteração dos valores das variáveis de saída.

Além destas, diversas outras decisões precisam ser tomadas, como por exemplo a especificação dos operadores para implementar o casamento (*matching*), dos conectivos *e* (*and*) e *implicação*, do operador para implementar a agregação das diversas regras e o método de defuzzyficação.

Em seguida, passa-se à etapa de sintonização do controlador. A sintonização é uma das etapas mais custosas do projeto de um controlador fuzzy. A grande flexibilidade que decorre da existência de muitos parâmetros exige um grande esforço do projetista de modo a obter o melhor desempenho do controlador. Alguns dos parâmetros podem ser alterados por mecanismos automáticos de adaptação e aprendizado. Entretanto, tais mecanismos não estão ainda bem estabelecidos na teoria de controle fuzzy, de modo que normalmente é tarefa do projetista o treinamento e a sintonização da maioria dos parâmetros envolvidos. Esta sintonização é feita por meio de busca, o que caracteriza uma atividade típica em IA. Um grande esforço tem sido devotado a esta questão na teoria clássica de controle. Recentemente, em (Ollero et.al. 1991), apresentou-se uma nova estratégia de projeto, envolvendo características de aprendizado. De acordo com o conhecimento que se tem sobre o processo a ser controlado, a técnica de projeto do controlador será mais ou menos elaborada. Tendo-se pouco conhecimento sobre o processo, deve-se adotar um projeto mais conservador, de modo a evitar um mau comportamento do controlador.

É necessário que alguns cuidados sejam observados pelo projetista. Em primeiro lugar, deve-se verificar-se se é realmente necessário o uso de um controlador fuzzy para controlar o processo. Caso o processo possa ser controlado por um controlador convencional simples, sem grandes transtornos, este poderá constituir uma boa solução. Se este não é o caso, e controladores convencionais não são apropriados devido a complexidades do processo, deve-se considerar o uso dos controladores fuzzy. Mesmo com pouco conhecimento sobre o processo, pode-se considerar o uso de um controlador fuzzy para controlá-lo. A seguir, tem-se uma proposta de metodologia de projeto, baseada nas abordagens clássicas de sintonização experimental:

- Inicie com um controlador simples, como por exemplo um que simule um controlador proporcional com:

- ganho reduzido
- as variáveis mais relevantes
- baixo número de variáveis linguísticas
- funções de pertinência não discriminantes

Aumente o ganho até que se verifiquem respostas indesejadas.

- Adicione conhecimento de acordo com a experiência que vai se adquirindo do processo
  - procure por novas variáveis linguísticas ou variáveis físicas, para evitar dificuldades de controle
  - modifique as funções de pertinência e os parâmetros do controlador fuzzy
  - adicione novos componentes ou modifique a estrutura de controle
- Valide a coerência do novo conhecimento incorporado modificando as condições de operação.

Para obter um desenvolvimento satisfatório do projeto, é necessária uma interface homem-máquina poderosa, que permita uma rápida modificação nos parâmetros utilizados, o que normalmente pode ser

encontrado em pacotes integrados. Tais pacotes incluem editores para as regras, funções de pertinência e outros parâmetros, recursos gráficos e um módulo depurador, para permitir o acompanhamento do processo de inferência e a monitoração das variáveis do processo. Tais recursos possibilitam uma grande interação com o projetista

Outro recurso extremamente valioso é a existência de um módulo supervisor que possa processar alguns índices de desempenho, além dos convencionais. Por exemplo, se uma entrada do tipo degrau é aplicada a alguma entrada, alguns índices de desempenho tais como o *overshoot*, o tempo de resposta ou o erro em regime permanente podem ser calculados. Através da análise sequenciada das amostras no tempo, pode-se detectar oscilações, desvios lentos ou algum tipo de comportamento inadequado mais raro, resposta dúbia, etc.

Os módulos supervisores podem também se encarregar de tomar algumas ações, tais como:

- **Mudanças em parâmetros** : Neste caso, uma mudança lenta e progressiva é recomendada
- **Desconsideração ou adição de regras** : Caso sejam pouco utilizadas, algumas regras poderão ser removidas. Algum tipo de indicador poderá recomendar que uma nova regra seja adicionada. Por exemplo, para reduzir erros estáticos, regras que utilizem termos concentrados em menores universos de discurso podem ser adicionadas, aumentando o ganho estático. Neste caso, o teste das ações irá confirmar a modificação ou recomendar o retorno ao conjunto de regras anterior.
- **Mudanças em variáveis** : Funções de pertinência não discriminantes poderão recomendar a desconsideração de alguns antecedentes em regras pré-estabelecidas. Um comportamento insatisfatório poderá indicar a consideração de novas variáveis.

Como anteriormente, o procedimento de verificação e teste irá confirmar se as ações de controle são aceitáveis, tendo em vista as especificações a serem atingidas.

## 6 - HARDWARE E SOFTWARE

O primeiro chip a implementar a lógica fuzzy foi desenvolvido por Togai e Watanabe nos laboratórios da AT&T em 1985. Este chip implementava uma máquina de inferência fuzzy capaz de processar 16 regras em paralelo. Além de uma memória para armazenar o conjunto de regras, implementava uma unidade de processamento de inferência, um controlador, e interfaces de entrada e saída. Em uma versão mais recente, a memória foi implementada por uma memória RAM estática, de modo que fosse possível efetuar mudanças dinâmicas no conjunto de regras. A unidade de processamento de inferência foi baseada na regra composicional de inferência "max-min". Em testes de desempenho mostrou-se que o chip realizava 250.000 FLIPS (inferências de lógica fuzzy por segundo) com um relógio de 16 MHz. Posteriormente um acelerador para operações em lógica fuzzy foi desenvolvido baseado no chip anterior. Em março de 1989, o Centro de Microeletrônica da Carolina do Norte completou com sucesso a fabricação de outra versão do chip, projetada por Watanabe, com 608.000 transistores, capaz de realizar 580.000 FLIPS.

Um controlador fuzzy de alto desempenho foi proposto por Yamakawa (1986). Composto por uma plataforma de 15 regras de controle e uma interface de saída implementando um defuzzificador pelo método do centro de área, ele manipula regras fuzzy utilizando os termos linguísticos NL (alto e negativo), NM (médio e negativo), NS (baixo e negativo), ZR (zero), PS (baixo e positivo), PM (médio e positivo) e PL (alto e positivo). Sua velocidade operacional é de 10 MFLIPS (mega FLIPS). O sistema foi testado em uma aplicação cujo objetivo é o de estabilizar um pêndulo invertido montado sobre um veículo. Pêndulos duplos com diferentes parâmetros foram também controlados. O controlador foi integrado em um chip com 40 pinos.

Yamakawa e Miki também implementaram 9 funções de uso geral em lógica fuzzy, utilizando-se o processo CMOS convencional, com um circuito em modo corrente. Posteriormente uma versão preliminar de um computador fuzzy foi proposta por Yamakawa, tendo sido implementada pela Omron. Este computador constituía-se de uma memória, um conjunto de máquinas de inferência, um bloco de operadores de máximo, um defuzzificador e uma unidade de controle. A memória fuzzy armazena a informação fuzzy, contida nas funções de pertinência. Inclui uma RAM binária, um registrador e um gerador de funções de pertinência (GFP). Este, por sua vez, consiste de uma PROM, um array de transistores e um decodificador. Cada termo linguístico é representado por um código binário, armazenado na RAM binária. As funções de pertinência correspondentes são geradas pelo GFP de acordo com estes códigos binários. A máquina de inferência aplica operações de máximos e mínimos, implementados por portas fuzzy em configuração emissor-acoplado, com circuitos em modo- tensão. As entradas são representadas por tensões analógicas nos barramentos, alimentando cada máquina de inferência em paralelo. Os resultados inferidos pelas regras são agregados pelo bloco de máximos. A saída é feita por meio de tensões analógicas.

Implementando um controlador fuzzy, o computador fuzzy é capaz de processar 10 MFLIPS. Este foi um grande passo não somente em termos de uma aplicação industrial, mas em termos de processamento de conhecimento, de um modo geral.

Mais recentemente, a Neuralogix (1991) anunciou o micro-controlador fuzzy NLX230, uma máquina de inferência VLSI configurável baseada no esquema de inferência MAX-MIN. O chip possui 16 interfaces de fuzzyficação, uma rede neural para comparações de mínimos, um comparador de máximos, memória de regras, registradores e circuitos de controle e sincronização. Apropriadamente programado, pode executar 30 MFLIPS.

Oki também desenvolveu um chip VLSI para inferência em lógica fuzzy, em um PGA cerâmico de 132 pinos. Sua arquitetura é baseada em estruturas do tipo "pipeline". Realiza 7.5 MFLIPS com no máximo 960 regras. Possui todos os circuitos necessários para uma inferência fuzzy: buffer de entrada, memória para regras e funções de pertinência, interface de fuzzyficação, circuitos MAX-MIN, defuzzyficador e registradores de saída.

A Omron desenvolveu um dos primeiros controladores fuzzy. Atualmente, anunciou uma nova geração de processadores fuzzy digitais, a família FP3000. Dotada de uma máquina de inferência de alta velocidade, processa em 650  $\mu$ s, com relógio de 24 MHz, um total de 20 regras com 5 antecedentes e 2 consequentes cada, fornecendo além disso diversos outros recursos e interfaces.

Do ponto de vista de suporte de software, um grande número de ambientes de desenvolvimento encontra-se hoje disponível no mercado. A maioria deles implementa recursos para edição de funções de pertinência, com suporte gráfico, linguagens dedicadas para descrição de regras, pré-compiladores para uma vasta gama de processadores e micro-controladores e geração de código automática para linguagens de alto nível, C por exemplo. Exemplos incluem o Togai Til Shell, da Togai Infralogic, o Cubicalc da Hyperlogic, o Manifold Editor da Fuzzy Systems Engineering, o ambiente integrado da fuzzyTECH, as ferramentas de desenvolvimento FIDE da Apronix, e a da Omrom, além do sistema de controle fuzzy da Meiden e o SDAF da HI Tecnologia. Muitos destes possibilitam a escolha de diferentes procedimentos de inferência, bem como diferentes métodos de defuzzyficação e capacidade de aprendizado.

## 7 - EXEMPLO DE UMA FERRAMENTA DE DESENVOLVIMENTO

Nesta seção, será apresentado o exemplo de uma ferramenta de desenvolvimento, utilizada para o projeto, implementação e teste de controladores fuzzy. Esta ferramenta foi apresentada em (Gudwin et.al. 1991), descrevendo seus módulos básicos, sendo colocada aqui a título de ilustração.

O SDAF (Sistema de Desenvolvimento de Aplicações Fuzzy) foi concebido como um sistema de suporte ao desenvolvimento de aplicações que utilizem a lógica fuzzy como método de construção de controladores. Uma linguagem foi implementada para que a construção de controladores seja realizada a partir de uma especificação na forma mais próxima possível da linguagem natural, viabilizando que um especialista descreva seu conhecimento de uma maneira bem amigável. Esta linguagem incorpora as inovações possibilitadas pelo uso da lógica fuzzy como instrumento, que é o uso de termos vagos e imprecisos representando conceitos. A representação e manipulação de termos deste tipo é o que diferencia basicamente o SDAF de outros *shells* para sistemas especialistas.

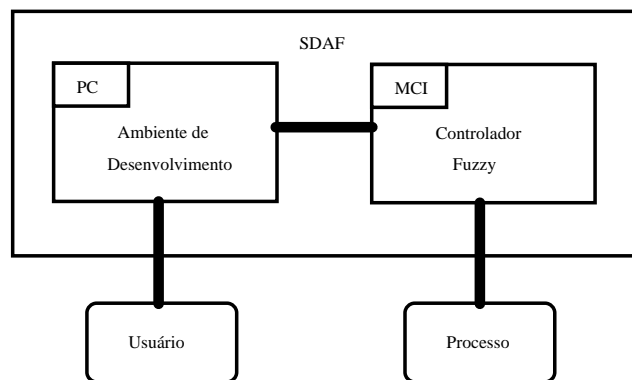


Figura 15- Componentes do SDAF

A configuração do SDAF é a de uma plataforma de desenvolvimento. Estando a base de conhecimento devidamente concluída, a mesma pode ser armazenada em uma EPROM, e um conjunto interpretador/base de conhecimento ser implementado em um hardware mais apropriado para a aplicação alvo.

O SDAF é constituído basicamente por dois módulos de hardware interconectados, operando em equipamentos distintos, e ligados ao processo a ser controlado (aplicação). São os seguintes (figura 15):

- Ambiente de Desenvolvimento - PC
- Controlador Fuzzy - Microcontrolador Industrial

### **7.1 - Módulo do Ambiente de Desenvolvimento:**

Este módulo foi projetado para operar em microcomputadores do tipo IBM/PC-XT/AT/386/486, sob sistemas operacionais compatíveis com o MS-DOS. Possui uma interface homem-máquina amigável, dotada de recursos gráficos, menus, janelas de entrada de dados, *help* sensível a contexto, operação com ou sem *mouse*, etc.

O objetivo deste módulo é permitir a integração entre o projetista e o hardware do controlador fuzzy, oferecendo uma interface homem-máquina altamente amigável. Provê, dentre outros, os seguintes recursos:

- Linguagem de Representação de Conhecimento utilizando regras fuzzy;
- Editor de funções de pertinência;
- Gerador de Código para o Controlador Fuzzy;
- Mecanismos para programação dos canais de comunicação com o processo;
- Operação remota do controlador Fuzzy;
- Depurador simbólico On line;

### **7.2 - Módulo do Controlador Fuzzy:**

Este módulo foi implementado sob uma plataforma de hardware baseada no micro-controlador industrial MCI da HI Tecnologia. A implementação deste módulo em uma outra máquina decorre da necessidade de conexão física com o processo a ser controlado. O módulo do controlador fuzzy deve suprir todos os recursos necessários para a integração com um processo real, tais como, entradas e saídas analógicas/digitais, interface com o operador e rede local de comunicação.

Opcionalmente, este módulo dispõe de uma interface de operação local, permitindo ao usuário comandar a execução do programa localmente. Existem comandos para ativar, suspender, executar passo a passo as regras, medir o tempo que o processador necessita para executar as instruções, etc.

### **7.3 - Desenvolvimento de Projetos**

O desenvolvimento de sistemas de controle baseados em lógica fuzzy no SDAF, é um processo interativo, composto das seguintes etapas:

- Identificação das variáveis de entrada/saída do processo;
- Definição das partições de cada variável linguística do sistema;
- Edição da Base de Regras Fuzzy;
- Compilação da Base de Regras;
- Edição das funções de pertinência associadas a cada termo linguístico previamente definido;
- Transferência da Base de Conhecimento do ambiente de desenvolvimento para o controlador fuzzy;
- Execução/Depuração da Base de Conhecimento;
- Análise de Desempenho do sistema;
- Alteração nas regras da base e/ou funções de pertinência, para estruturação e sintonização.

Para facilitar a programação, o SDAF permite que as variáveis linguísticas de entrada/saída do sistema, sejam convenientemente escaladas no *range* equivalente ao do sensor, na unidade de engenharia correspondente.



## 7.4 - O Editor de Funções

O SDAF dispõe de um editor gráfico que permite a edição/manipulação das funções de pertinência. As funções mais usuais, tais como, função sigmóide, triangular e gaussiana, são geradas por meio de seus parâmetros. Outras funções podem ser geradas através de técnicas de interpolação, via gráficos ou tabelas.

Uma vez definida a base de regras fuzzy, o SDAF gera, durante a compilação, um arquivo referente à base de funções, citando todos os termos linguísticos utilizados, que devem ser configurados pelo usuário. Assim, para cada termo linguístico, o usuário deverá associar uma função de pertinência, definida sobre o universo de discurso da variável linguística envolvida (figura 16).

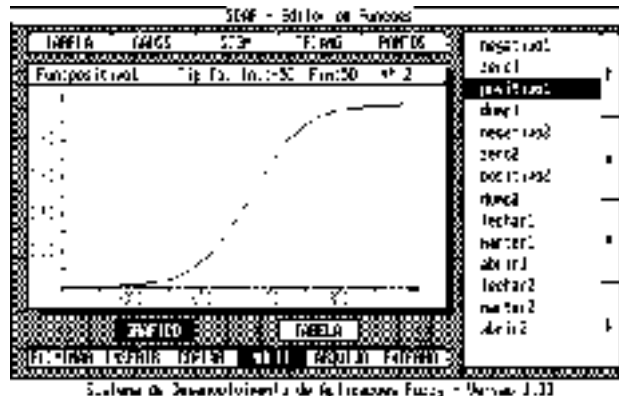


Figura 16 - Editor de Funções de Pertinência

## 7.5 - A Linguagem LEARN

O ponto chave para a compreensão do SDAF é o método de representação de conhecimento utilizado pelo sistema. Para tal, desenvolveu-se a linguagem LEARN - Linguagem de Especificação de Aplicações mediante Regras Nebulosas, utilizada pelo sistema para a implementação das bases de regras fuzzy. Esta linguagem foi especialmente concebida para ser utilizada dentro do contexto deste sistema.

A declaração das variáveis linguísticas segue a seguinte sintaxe:

$$\text{VAR Nome\_Var [ Universo\_Discurso ] = \{Conjunto\_Termos\_Linguísticos\};$$

A sintaxe das regras fuzzy na linguagem LEARN segue a seguinte forma:

$$\text{SE Condição\_1 E Condição\_2 ... ENTAO Ação;}$$

As condições são expressões simples do tipo:

$$\text{Var\_i == Termo\_Linguístico\_j}$$

E as ações:

$$\text{Var\_i = Termo\_Linguístico\_j}$$

onde Var\_i corresponde a uma variável linguística e Termo\_Linguístico\_j deve ser um dos termos linguísticos definidos na partição de Var\_i.

Um exemplo simples de uma de base de regras constituída a partir da linguagem LEARN pode ser visto na listagem 1.

```

VAR
  ANALOGICA Erro[0,100] = { Negativo, Zero, Positivo } ;
  ANALOGICA Valvula[0,200]={Diminui, Mantem,
                              Aumenta } ;

ENTRADA Erro;
SAIDA Valvula;
REGRAS
GRUPO Principal
{
  SE Erro==Negativo ENTAO Valvula=Aumenta;
  SE Erro == Zero ENTAO Valvula = Mantem;
  SE Erro == Positivo ENTAO Valvula = Diminui;
}
FimGrupo
FimRegras

```

Listagem 1 : Exemplo de Programa na Linguagem LEARN

## 7.6 - Tipos de Variáveis

As variáveis do sistema foram inicialmente concebidas para suportar aplicações de controle em tempo real, e por isso podem ser dos seguintes tipos:

**1) Variáveis Analógicas :** São variáveis numéricas de uso geral, podendo ser de entrada, de saída ou internas. Estão associadas implicitamente a variáveis linguísticas, ou seja, as medidas numéricas colhidas pelo conversor A/D são mapeadas no universo de termos linguísticos da variável (variáveis de entrada) ou então têm seus valores inferidos em termos linguísticos mapeados em termos numéricos que são enviados ao conversor D/A (variáveis de saída).

**2) Variáveis Digitais :** São variáveis lógicas de uso geral. Também podem ser de entrada, de saída ou internas. Estas variáveis são utilizadas para definir estados lógicos. Assume-se para o nível lógico 0 (*low*) o termo linguístico FALSO, e para o nível lógico 1 (*high*) o termo linguístico VERDADEIRO. Nas regras, as variáveis digitais são utilizadas de modo similar ao das variáveis analógicas, com a devida restrição nos termos linguísticos utilizados.

**3) Timers Analógicos :** São variáveis numéricas internas do sistema que possuem a característica adicional de serem atualizadas automaticamente pelo sistema, à medida que o tempo passa. Funcionam em sincronismo com o relógio de tempo real do sistema. Com este tipo de variável, é possível quantificar-se porções de tempo de definição vaga, como por exemplo "um pouco", "bastante", etc. Os timers analógicos são então ao mesmo tempo variáveis de entrada e de saída. Quando utilizados nos antecedentes das regras, avalia-se a temporização atual do timer frente aos termos linguísticos definidos. Quando utilizadas no consequente das regras, reinicializa-se o período de temporização (também em termos vagos).

**4) Timers Digitais :** São variáveis lógicas associadas a temporizadores do sistema. Ao contrário dos timers analógicos, onde deseja-se manipular um tempo definido de modo vago, os timers digitais fazem um controle bem definido do tempo transcorrido. Ao ser acionado um consequente que contenha um timer digital, a temporização do mesmo é reinicializada com o período definido na regra. A partir desta, qualquer avaliação nos antecedentes das regras surtirá um valor linguístico "tempo não expirado", até que o período de temporização se esgote, quando então a avaliação do antecedente resultará "tempo expirado". Este tipo de variáveis é útil quando se deseja o controle exato de algum tipo de temporização, como em sequenciamento de eventos, etc.

## 7.7 - Desenvolvimento de Aplicações - O Depurador

A construção de uma base de conhecimento é normalmente um processo incremental, onde o conhecimento vai sendo levantado junto ao especialista, ou pelo projetista, em etapas. A cada etapa, um novo

conjunto de regras é acrescentado à base, e o projetista do sistema, junto com o especialista, deve verificar a consistência destas regras, validando o comportamento do sistema.

Uma das técnicas na elaboração de uma base de conhecimento, que facilita a própria elicitação do mesmo e proporciona um incremento no desempenho computacional do sistema, em ambientes de tempo real, é a utilização de bases de regras estruturadas, ou modularizadas. O SDAF permite a construção de bases de regras deste tipo por meio da definição de grupos. Cada grupo deve conter portanto, uma pequena parte do conhecimento, relativa a uma determinada especialização do mesmo, utilizando um mecanismo conhecido como foco de atenção. Um dos grupos, normalmente o grupo principal, deve conter o meta-conhecimento, ou seja, o conhecimento do conteúdo dos diversos grupos, e de quando acessá-los. O SDAF permite que este acesso seja feito de duas maneiras. A primeira delas, é a execução uma única vez de um determinado grupo, processada pelo comando EXECUTA. Este tipo de acesso é feito quando se deseja apenas uma aplicação de um procedimento, de modo semelhante a uma chamada de função em programação convencional. O segundo modo de se acessar um grupo é por meio do comando CHAVEIA, quando sob determinadas condições levantadas pelo grupo, determine-se que aquele grupo já não é mais o adequado a ser processado, e com isso se deseja efetivamente chavear o processamento para outro grupo de regras, que está mais adequado à situação corrente do sistema. Para o desenvolvimento de uma base de conhecimento, o procedimento mais usual é a determinação de um grupo principal, onde sejam colocadas as diretrizes gerais de atuação, e a cada etapa, sejam acrescentados novos grupos de regras, que podem então ser analisadas pelo projetista, que passa a validar o funcionamento do grupo, efetuando as correções que forem pertinentes.

Durante este processo, é necessário que o projetista disponha de alguns recursos adicionais, as ferramentas de depuração, que o auxiliem na avaliação do comportamento do sistema. Este conjunto de recursos é reunido no SDAF no módulo depurador.

O módulo depurador do SDAF permite o controle da execução das regras, permitindo o disparo e a parada da máquina de inferência em qualquer ponto da mesma. Com isso, consegue-se executar as regras passo a passo, sendo exibidas na tela, tanto as regras com o passo sendo executado, como o estado do sistema após a execução do mesmo. Este passo pode ser a nível de ciclo, regra ou antecedente. A nível de ciclo, a máquina executa um ciclo de inferência e para, esperando um próximo comando do operador. A nível de regra a máquina executa uma determinada regra ao comando do operador, parando em seguida, e a nível de antecedentes, a máquina para após executar cada antecedente, exibindo o estado do sistema. Um outro comando auxiliar é colocado de modo a reinicializar as variáveis do sistema.

Outro recurso importante do depurador é o processo pelo qual se pode analisar o conteúdo das diversas variáveis utilizadas. Com este, permite-se verificar o estado de cada variável. Caso sejam variáveis digitais, mostra-se seu estado lógico (FALSO/VERDADEIRO). Caso sejam variáveis analógicas de entrada, mostra-se o valor medido da variável, e para cada predicado linguístico associado à mesma, como o mesmo se relaciona diante do valor medido, dando uma idéia da compatibilidade entre o valor medido e o conceito associado ao predicado linguístico. Para variáveis analógicas de saída, mostra-se além do conjunto completo de conjuntos fuzzy associados a ela, o estado do conjunto fuzzy inferido, que dará origem ao valor defuzzyficado, que será enviado ao controle.

Utilizando-se o depurador, o projetista consegue efetuar o desenvolvimento incremental do sistema, pois após a elicitação do conhecimento sobre a forma de regras e funções de pertinência, o mesmo pode ser validado executando-se o programa de regras passo a passo, o que permite a detecção de regras com comportamento inadequado, facilitando sua correção.

## 8 - EXEMPLOS DE APLICAÇÃO FUZZY

### 8.1 - Controle de Poluição em Túneis Urbanos

Descreve-se aqui, sucintamente e com objetivos meramente didáticos, o procedimento de desenvolvimento de um controlador fuzzy. O exemplo utilizado para ilustrar o procedimento é um sistema de exaustão de gás com velocidade variável, utilizado para o controle do nível de gás carbônico em túneis urbanos. O controlador possui como entradas o nível de CO<sub>2</sub> aceitável (*set point*,  $s$ ), o desvio ( $d$ ) do nível medido de CO<sub>2</sub> ( $y$ ) com relação ao *set point*, e a variação do desvio ( $\delta$ ) em instantes sucessivos de tempo.

O primeiro passo, como no caso de qualquer controlador, é definir as variáveis controladas e as variáveis de controle. Estas variáveis são as entradas e as saídas, respectivamente, do controlador. Para o sistema de exaustão, as variáveis de entrada são o desvio e a variação do desvio, sendo a variação na velocidade do exaustor a variável de saída ( $u$ ).

A seguir, determina-se, para cada variável, o universo de discurso (*range*) associado a cada variável linguística, a partição do conjunto de termos para as variáveis linguísticas e os respectivos conjuntos fuzzy. No

caso do exemplo,  $d$ ,  $\delta$  e  $u$  são as variáveis linguísticas com valores no conjunto de termos {pequeno(a), médio(a), grande}, {Negativo, Zero, Positivo}, onde os termos são, por sua vez, definidos pelos conjuntos fuzzy da figura 17. Observe que os universos de discurso correspondem ao intervalo  $[-5,5]$  para  $d$ ,  $[-1,1]$  para  $\delta$  e  $[-100,100]$  para  $u$ .

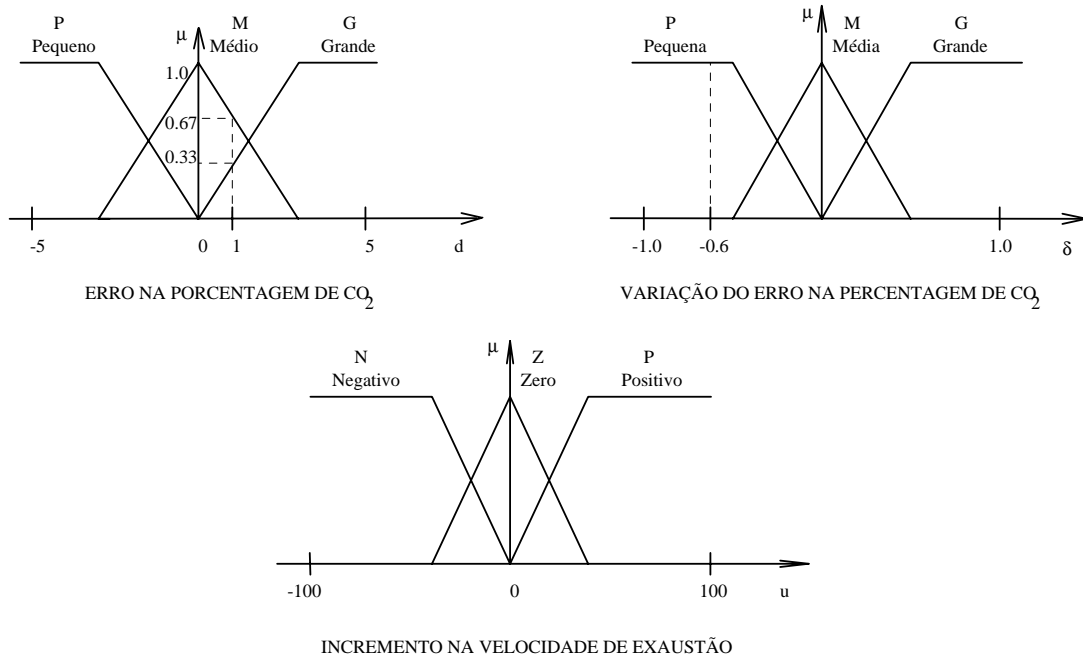


Figura 17 - Funções de Pertinência para os Termos Linguísticos

O passo final consiste em definir as regras que descrevem as ações de controle em função do erro e da variação do erro. Cada variável linguística deve ser considerada por pelo menos uma regra. A tabela 1 apresenta um conjunto possível de regras:

$\delta$  - VARIAÇÃO DO ERRO

	P	M	G
d - ERRO	N	N	Z
	M	Z	P
	G	P	P

Tabela 1- Exemplo de Regras de Controle

Na execução da estratégia de controle, representada pelo conjunto de regras da tabela 1, o primeiro passo é a classificação (*matching*) das variáveis de entrada com relação às respectivas variáveis linguísticas. Por exemplo, o valor  $d = 1$  é grande com grau 0.33 e médio com grau 0.67, enquanto que o valor  $\delta = -0.6$  é pequena com grau 1 (ver figura 17), ou seja,  $\mu_G(d) = 0.33$ ,  $\mu_M(d) = 0.67$  e  $\mu_P(\delta) = 1$ . Feita esta classificação (correspondente ao módulo de fuzzyficação), o procedimento de inferência (lógica de decisão) avalia as regras da seguinte forma. Para  $d=1$  e  $\delta = -0.6$ , as regras que se aplicam são as seguintes (tabela 1) :

$R_4$  : Se ( $d$  é Médio) e ( $\delta$  é Pequena) Então ( $u$  é Negativo)

$R_7$ : Se ( $d$  é Grande) e ( $\delta$  é Pequena) Então ( $u$  é Zero)

Como as regras possuem dois antecedentes relacionados pelo conectivo E (intersecção), definindo-se o operador intersecção como sendo  $\wedge$  (min), obtém-se como resultado da combinação dos antecedentes:

$$R_4: \min(\mu_M(d=1), \mu_P(\delta=-0.6)) = (0.67) \wedge (1) = 0.67$$

$$R_7: \min(\mu_G(d=1), \mu_P(\delta=-0.6)) = (0.33) \wedge (1) = 0.33$$

Para cada regra, o grau de ativação da ação de controle é calculada de acordo com o resultado da combinação de antecedentes. Se a regra de inferência é MAX-MIN, o resultado da inferência, para uma regra, é obtido pelo mínimo entre o grau de combinação dos antecedentes e consequentes (ação de controle), conforme ilustrado graficamente pela figura 18.

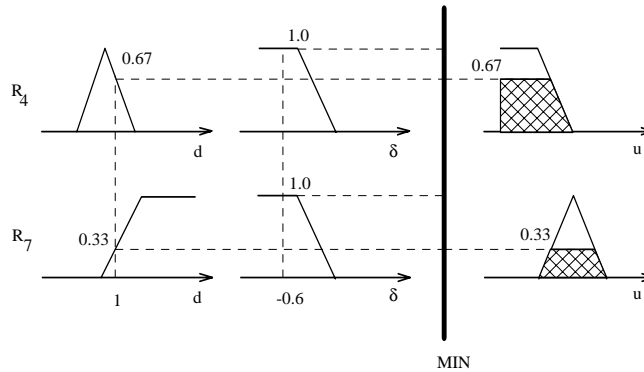


Figura 18 - Inferência : Composição MAX-MIN

Terminada a inferência, a ação final de controle é calculada a partir da união das contribuições proporcionadas por cada regra ativada. Se a união é definida como sendo o operador de agregação máximo ( $\vee$ ), o resultado será o mostrado na figura 19.

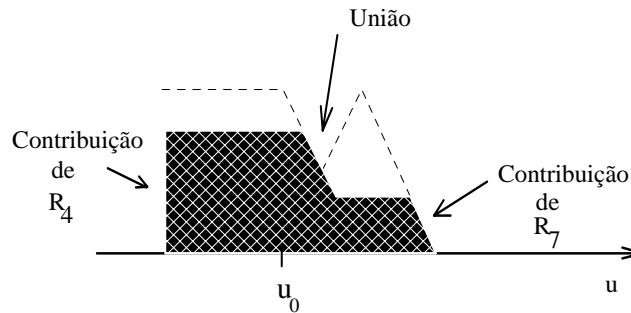


Figura 19 - Ação Final de Controle

O último passo consiste na determinação do sinal de controle ( $u_0$ ) a ser enviado ao processo (no caso, sinal proporcional ao incremento de velocidade do exaustor), obtida a partir da defuzzificação. Por exemplo, na figura acima,  $u_0$  é obtido pelo método do centro de área.

## 8.2 - Controle de Direção de Veículos Auto-Guidados

Deseja-se, nesta aplicação, controlar o movimento de um veículo auto-guido (AGV) para que o mesmo percorra uma trajetória em um ambiente, a partir do sensoreamento de sua localização (figura 20).

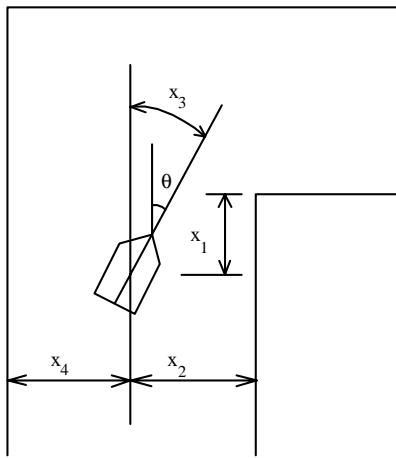


Figura 20 - Variáveis de Entrada e Saída

As variáveis de entrada do controlador são as posições  $x_1$ ,  $x_2$ , e  $x_4$  e o ângulo do veículo em relação à direção das laterais. A saída do controlador é o ângulo do eixo frontal do veículo em relação à posição de repouso (alinhada com a direção do veículo). As regras de controle são da forma:

$$R_i : \text{Se } (x_1 \in A_1^i) \text{ e } \dots \text{ e } (x_4 \in A_4^i)$$

$$\text{Então } \theta_i = p_0^i + p_1^i x_1 + p_2^i x_2 + p_3^i x_3 + p_4^i x_4$$

$$i = 1, \dots, m$$

A ação de controle é determinada por:

$$\theta = \frac{\sum_{i=1}^m w_i \theta_i}{\sum_{i=1}^m w_i}$$

onde  $w_i$  é o resultado da combinação dos antecedentes da  $i$ -ésima regra, dados os valores das variáveis de entrada. As funções de pertinência dos conjuntos fuzzy correspondentes às partições das variáveis linguísticas são mostradas na figura 21 para as variáveis  $x_1$  e  $x_3$ .

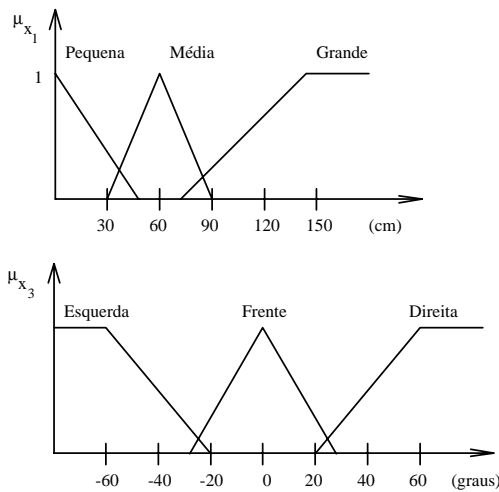


Figura 21 - Funções de Pertinência para variáveis  $x_1$  e  $x_3$

A figura 22 mostra o resultado proporcionado pelo controlador para o veículo colocado em um ambiente complexo.

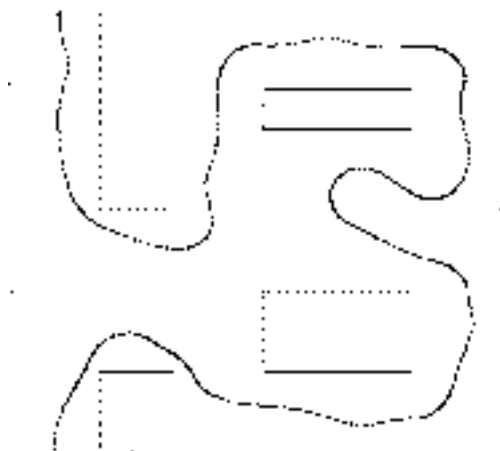


Figura 22 - Trajetória do Veículo

### 8.3 - Controle de Nível Para 2 Tanques Acoplados

Nesta aplicação, considera-se um sistema em que 2 tanques estão acoplados por meio de uma interligação (vide figura 23), sendo que os tanques são abastecidos por meio de válvulas individuais de entrada de fluido. Cada tanque tem em seu fundo, uma válvula de drenagem de abertura fixa, por onde escoo o fluido. A tarefa do controlador fuzzy é manter os níveis em ambos os tanques de acordo com os *set-points* definidos.

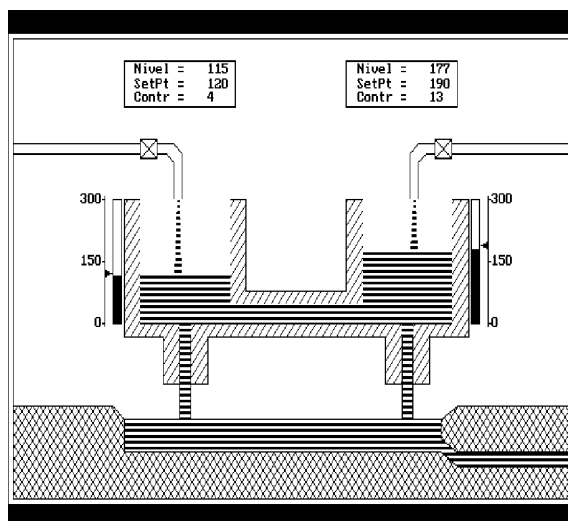


Figura 23 - Sistema de 2 Tanques Acoplados

Neste exemplo, ambos os tanques possuem uma seção horizontal com área de  $1 \times 10^5 \text{ cm}^2$ , alturas de 300 cm e suas válvulas fixas de saída possuem uma resistência à vazão de  $2.5 \times 10^{-2} \text{ cm} / \text{cm}^3 / \text{s}$ . As vazões mínimas de entrada são iguais a 0 l/s e as máximas de 40 l/s.

Para implementar o controlador fuzzy, utilizou-se como variáveis linguísticas de entrada o erro e a variação do erro entre os níveis dos tanques e seus *set-points*. Como variáveis linguísticas de saída, adotou-se os incrementos nas vazões de entrada do processo. Na figura 24 tem-se um esquema das funções de pertinência utilizadas.

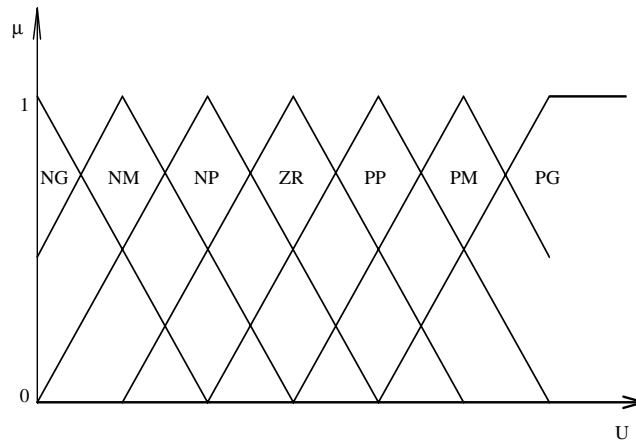


Figura 24 - Funções de Pertinência

As regras utilizadas pelo controlador foram as seguintes:

erro\derro	NG	NM	NP	ZR	PP	PM	PG
NG			PG	PG			
NM				PM			
NP	PM			PP			
ZR	PG	PM	PP	ZR	NP	NM	NG
PP				NP			NM
PM				NM			
PG				NG	NG		

Os resultados obtidos, são mostrados na figura 25 a seguir, para os dois tanques:

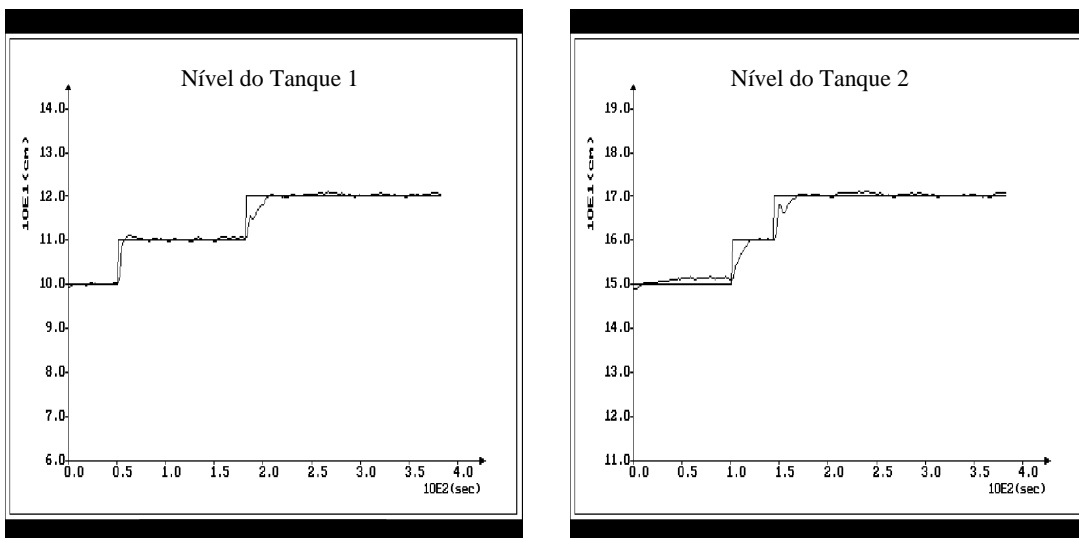


Figura 25 - Resultados para os Tanques



## 8.4 - Controle Autônomo de um Robô Móvel

Algoritmos genéticos (Goldberg, 1989) são métodos adaptativos que podem ser usados para resolver problemas de busca, otimização e aprendizagem de máquina, dentre outros. O ciclo básico do algoritmo genético é mostrado na figura 26.

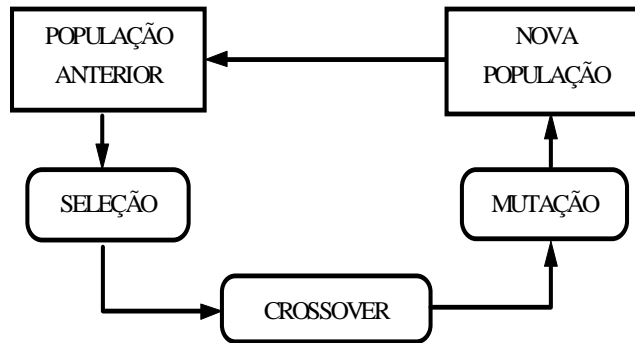


Figura 26 - Ciclo Básico de Algoritmo Genético

Um algoritmo genético é considerado neste exemplo como um método de aprendizagem de ações de controle de forma autônoma. Considera-se um controlador neural fuzzy para um robô móvel. A rede aprende a dirigir o robô móvel até que o mesmo atinja alvos predefinidos, sem colidir com obstáculos do ambiente. O conhecimento adquirido pela rede pode também ser facilmente extraído em forma de regras fuzzy. Como o algoritmo genético também determina o número de regras, as funções de pertinência dos antecedentes e os consequentes, este modelo constitui-se em um método conveniente para projetar sistemas nebulosos de forma automática (Oliveira et.al., 1994).

A rede neural fuzzy utilizada (figura 27) é baseada em um conjunto de proposições (fatos) e regras fuzzy do tipo (Figueiredo et.al. 1993):

$$\begin{array}{l}
 \text{Fato:} \quad X_1 \text{ é } A_1 \text{ e } \dots \text{ e } X_M \text{ é } A_M \\
 \text{Regra 1:} \quad \text{Se } X_1 \text{ é } A_1^1 \text{ e } \dots \text{ e } X_M \text{ é } A_M^1 \text{ Então } y \text{ é } g^1 \\
 \quad \quad \quad \vdots \\
 \text{Regra M:} \quad \text{Se } X_1 \text{ é } A_1^N \text{ e } \dots \text{ e } X_M \text{ é } A_M^N \text{ Então } y \text{ é } g^N \\
 \hline
 \text{Conclusão (açãõ):} \quad y \text{ é } g
 \end{array}$$

onde  $X_j$  é uma variável fuzzy,  $A_j$  e  $A_j^i$  são os conjuntos nebulosos associados,  $y$  é uma variável com valores  $g^j \in \mathfrak{R}$  (figura 27).

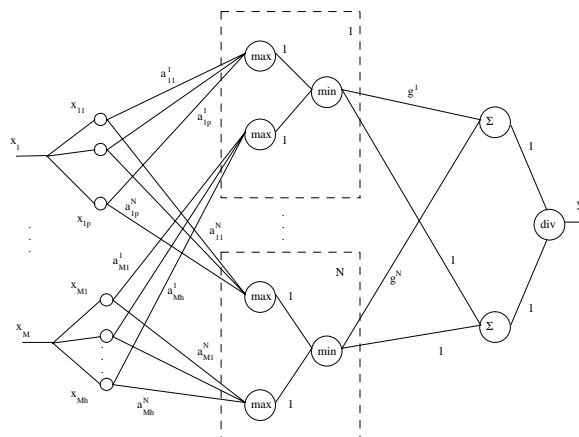


Figura 27: Controlador Neural Fuzzy

O consequente  $y$  é determinado em três etapas: casamento (*matching*), agregação de antecedentes e agregação de regras. Estas etapas são executadas por neurônios especializados do tipo min-max pois incorporam funções mais complexas para modelagem de sinapses, de agregação de entradas e de decodificação (Gomide & Rocha, 1992-c).

O algoritmo genético (GA) usado é do tipo elitista e manipula um cromossomo de formato misto inteiro/real. As características ajustadas pelo GA durante as gerações são os consequentes das regras e as funções de pertinência, representadas por formato e posição. Os consequentes das regras são codificados em genes reais, enquanto a codificação das funções de pertinência é mais elaborada: um par numérico (centro, tipo), onde centro indica a posição do ponto central da função e tipo indica o formato da função, trapezoidal ou triangular (veja tabela 2). Para uma base de conhecimento com  $n$  regras, cada uma com  $k$  antecedentes e  $f_i$  funções de pertinência para cada antecedente  $i$ , o cromossomo descrevendo esta rede neurofuzzy terá  $n + \sum_{i=1}^k f_i$  genes reais e  $\sum_{i=1}^k f_i$  genes inteiros.

Os genes relativos às funções de pertinência são interpretadas de acordo com seu próprio formato e centro e os formatos e centros das funções imediatamente anterior e posterior. Veja a Tabela 2.

O crossover sempre ocorre e seleciona apenas um ponto de permutação. A mutação possui probabilidade zero durante quase todas as gerações, ocorrendo apenas em situações especiais de convergência. Este tipo de mutação é baseada na estratégia de explosões mutantes periódicas de Kauffman.

A função objetivo é uma função do número de alvos alcançados, distância média percorrida e distância ao alvo atual ao final da simulação.

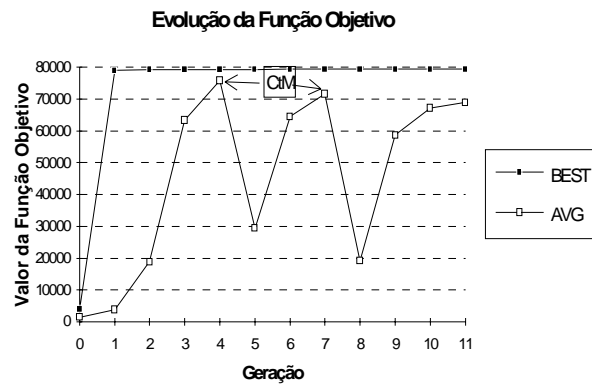


Figura 28 - Evolução da Função Objetivo

O GA descrito anteriormente utiliza durante as simulações 4 seqüências distintas de 4 obstáculos ((A,B,C,D), (B,C,D,A) etc.). A função objetivo é a média dos resultados obtidos em cada seqüência. Os resultados de simulação são mostrados nas figuras 28, 29 e 30.

Observe que, mesmo com alvos colocados em lugares diferentes, o melhor indivíduo (controlador) encontrado foi capaz de navegar pelo ambiente atingindo os alvos sem colidir.

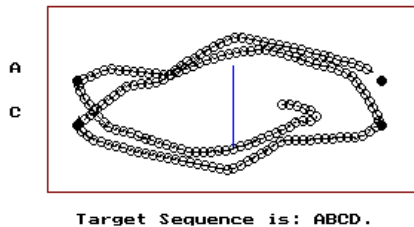


Figura 29 Simulação do Melhor Indivíduo

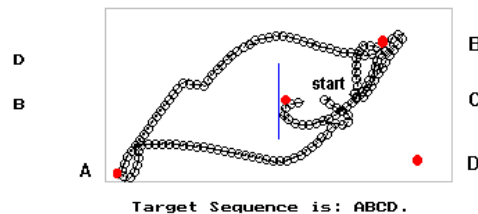


Figura 30 - Simulação com Alvos Diferentes

Formato da Função ( $i$ ) (seu ponto central é $c_i$ )	Formato da Função ( $i+1$ ) (seu ponto central é $c_{i+1}$ )	Lado direito da Função ( $i$ ) e o lado esquerdo da Função ( $i+1$ )
Triangular (Trapezoidal)	Triangular (Trapezoidal)	
Triangular (Trapezoidal)	Trapezoidal (Triangular)	

Tabela 2 : Formação das Funções de Pertinência

## 9 - CONCLUSÕES E PERSPECTIVAS

A tecnologia decorrente da lógica fuzzy tem gerado aplicações e produtos em diversas áreas. Em controle de processos industriais, área pioneira, as primeiras experiências datam de 1975 quando foi demonstrado no Queen College, Londres (Mamdani & Assilian, 1975), que um controlador fuzzy muito simples controlou eficientemente uma máquina a vapor. Na mesma época, a primeira aplicação industrial significativa foi desenvolvida pela indústria de cimento F.L.Smith Corp. da Dinamarca (Umbers & King, 1980). Neste tipo de aplicação, controladores fuzzy são usados para fornecer sinais de referência para controladores do tipo PID, ou seja, aqueles não atuam diretamente sobre a planta, mas em um nível hierárquico superior. O sucesso desta aplicação deveu-se em muito ao fato de os operadores humanos (especialistas) terem sido capazes de traduzir consistentemente, em termos de regras linguísticas, as atitudes que tomavam no ajuste dos sinais de referência para os controladores PID. Embora grande parte das aplicações considere universos discretos, similarmente ao que foi abordado, para fins de exemplo, neste texto, algumas fizeram uso de universos "contínuos" e de conjuntos fuzzy definidos por funções (contínuas) de pertinência (Kickert & Lemke, 1976), ao invés de valores de pertinência. Um controle fuzzy distinto do mostrado aqui, mas mantendo os mesmos princípios e o mesmo processo de inferência, foi empregado em controle de tráfego (Pappis & Mamdani, 1977). Em virtude da preocupação com a aquisição de regras, já mencionada anteriormente, controladores com capacidade de aprendizado foram desenvolvidos. O primeiro trabalho neste sentido (Procyk & Mamdani, 1979) deu origem ao controlador fuzzy conhecido como SOC, o qual é capaz de estabelecer sua própria estratégia de controle através da criação e modificação de regras. Para isto, é utilizada uma tabela de índice de desempenho, oriunda de regras linguísticas de caráter geral, situadas em um nível hierárquico superior às regras de controle propriamente ditas. O SOC foi aplicado a diversos processos simulados e também a um braço mecânico (Yamazaki, 1982; Scharf & Mandic, 1985; Tanscheit & Scharf, 1988). A maior dificuldade no uso deste controlador reside na sintonia dos parâmetros GE e GCE, pois, além de afetarem o controlador propriamente dito, influenciam também a criação e modificação das regras (Tanscheit & Lembessis, 1991). Um procedimento distinto, porém com o mesmo objetivo, consiste da identificação do modelo fuzzy do processo (Takagi & Sugeno, 1985; Sugeno & Tang, 1986), cujo resultado é um conjunto de regras que descrevem seu comportamento e que podem ser usadas no controle. Um terceiro procedimento visando o aprendizado de regras foi também sugerido (Van der Rhee & Lemke, 1990), mas ainda merece futuras investigações. De meados dos anos 80 em diante, o uso de controladores fuzzy em escala comercial tem mostrado um crescimento bastante significativo, principalmente no Japão. As aplicações de maior impacto comercial foram em máquinas de lavar roupa, máquinas fotográficas e na indústria automobilística. Empresas têm produzido controladores fuzzy em "chips" ou cartões para uso em microcomputadores, possibilitando sua interligação aos mais diversos processos. Existem em escala comercial interfaces que permitem que se defina graficamente conjuntos de várias formas, em universos de tamanho variável, que se escolha as funções de implicação, métodos de desnebulização, etc. Resumindo, pode-se afirmar que a utilização da Teoria de Conjuntos Fuzzy e de Lógica Fuzzy em Controle tem apresentado resultados bastante satisfatórios e muitas vezes superiores aos obtidos com outros controladores. A área de Controle Fuzzy já está bem estabelecida, inclusive com aplicações em nível industrial. Atualmente os trabalhos têm se concentrado no aprimoramento dos controladores existentes e na busca de novas estruturas, com uma preocupação em procedimentos automáticos de sintonia e em métodos de aprendizado de regras. Recentemente tem havido uma integração entre Controle Fuzzy e Redes Neurais, abrindo espaço para novas investigações.

Presentemente, uma variedade de aplicações comerciais e industriais estão disponíveis, destacando-se neste cenário o Japão e mais recentemente, os EUA e a Alemanha. Exemplos típicos incluem produtos de consumo tais como geladeiras (Sharp), ar condicionado (Mitsubishi), câmeras de vídeo (Canon, Panasonic), máquinas de lavar roupa (Sanyo), fornos de microondas (Sanyo), aspiradores de pó, etc. Na indústria automotiva destacam-se transmissões automáticas (Nissan, Lexus), injeção eletrônica, suspensão ativa, freios anti-bloqueantes. Sistemas industriais incluem controle de grupo de elevadores (Hitachi, Toshiba), veículos auto-guiados e robôs móveis (Nasa, IBM), controle de motores (Hitachi), ventilação de túneis urbanos (Toshiba), controle de tráfego urbano, controle de parada e partida de trens de metrô (Sendai, Tokio). Estas citações são, evidentemente, ilustrativas pois correntemente mais de 1000 patentes envolvendo lógica fuzzy já foram anunciadas.

No Brasil, apesar do uso e da aplicação extensiva ainda ser incipiente, várias indústrias e empresas vêm desenvolvendo produtos e serviços (Villares, IBM, Klockner & Moeller, Robertshaw, Yokogawa, HI Tecnologia).

Nos últimos dois anos o potencial de manuseio de incertezas e de controle de sistemas complexos proporcionado pela lógica fuzzy vem sendo combinado com com redes neurais artificiais as quais, por sua vez, possuem características de aprendizagem e adaptação. Esta simbiose vem gerando novas classes de sistemas e de controladores neurofuzzy, combinando assim os potenciais e as características individuais em sistemas adaptativos e inteligentes (Gomide et.al. 1992-a). Estes sistemas deverão proporcionar uma importante contribuição para os sistemas de automação e controle do futuro, principalmente em controle de processos.

**AGRADECIMENTO:** Os dois primeiros autores agradecem ao CNPq pelo apoio financeiro, respectivamente, pelo contrato n. 300729/86-3 e por bolsa de doutorado. Os autores também agradecem ao projeto ECLA-005 por seu suporte.

## REFERÊNCIAS BIBLIOGRÁFICAS

- Albertos, P. (1992). *Fuzzy Controllers - AI Techniques in Control* - Pergamon Press.
- Anderson, B.D.O e J. B. Moore (1979). *Optimal Filtering* - Prentice-Hall, N.J.
- Baldwin, J.F. & Pilsworth, B.W. (1980). Axiomatic Approach to Implication for Approximate Reasoning with Fuzzy Logic. *Fuzzy Sets and Systems*, V. 3: 193-219.
- Bertsekas, D.P. (1976). Dynamic Programming and Stochastic Control. *Mathematics in Science and Engineering*. Vol. 125, Academic Press, NY
- Doygle, J.C. e G. Skin (1981). Multivariable feedback design: concept for a classical/modern synthesis. *IEEE Trans. on Automatic Control*, vol. AC-26, pp. 4-16
- Figueiredo, M. F. Gomide ; W. Pedrycz (1993) - A Fuzzy Neural Network : Structure and Learning - *5th IFSA World Congress*, Seul, Coréia. pp 1171-1174.
- Gaines, B.R., (1976). Foundations of Fuzzy Reasoning. *International J. of Man-Machine Studies*, V. 8: 623-668.
- Goldberg, D. (1989) - *Genetic Algorithms in Search, Optimization and Machine Learning* - Addison-Wesley, MA.
- Gomide, F.; A. Rocha; P. Albertos (1992-a) Neurofuzzy Controllers - *IFAC - LCA 92*, Viena, Austria
- Gomide, F. ; A. Rocha (1992-b) A Neurofuzzy Controller ; *2nd International Conference on Fuzzy Logic and Neural Networks - IIZUKA 92* , Fukuoka, Japan.
- Gomide, F. ; A. Rocha (1992-c) Neurofuzzy Components Based on Threshold - *IFAC SICICA Symposium*, Malaga, Espanha, pp.425-430

- Gudwin, R.R ; M.A. Silva, H.J.Almeida Jr.; I.M.C.Ribeiro (1991). SDAF - Uma Ferramenta para Desenvolvimento e Teste de Sistemas utilizando Lógica Nebulosa - *Simpósio sobre Integração Computadorizada na Automação Industrial*, 18 a 20 Setembro - São Paulo-SP
- Hechst-Nielsen, R. (1990) *Neurocomputing* - Addison Wesley, N.Y.
- Holmblad, I.P. & Ostergaard, J.J., (1981). Fuzzy Logic Control: Operator Experience applied in Automatic Process Control. *FLS Review*, F.L.Smith & Co., Copenhagen, Denmark, n. 5: 11-16.,
- Hotzman, J.M. (1970) *Nonlinear System Theory* - Prentice Hall, N.J.
- Kaufmann, A. ; M. Gupta (1985) - *Introduction to Fuzzy Arithmetic* - Van Nostrand, NY.
- Kickert, W.J.M. & Van Nauta Lemke, H.R., (1976). The Application of a Fuzzy Controller in a Warm Water Plant. *Automatica*, Vol. 12: 301-308.
- Lee, C.C (1990). Fuzzy Logic in Control Systems: Fuzzy Logic Controller, part I and II. *IEEE Trans. on Systems, Man and Cybernetics*, vol. 20, pp 404-435
- Lembessis, E., (1984). Dynamic Learning Behaviour of a Rule-based Self- Organising Controller. Ph. D. Thesis, University of London.,
- Mamdani, E.H., (1974). Application of Fuzzy Algorithms for Control of Simple Dynamic Plant. *Proceedings of the IEE (Control and Science)*, V. 121: 298-316.
- Mamdani, E.H. & Assilian, S., (1975). An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *Int. Journal of Man-Machine Studies*, Vol. 7: 1-13.
- Mizumoto, M., (1982). Fuzzy Inference using max-min composition in the Compositional Rule of Inference. In: *Approximate Reasoning in Decision Analysis*, M.M. Gupta, E. Sanchez (Ed.), North-Holland: 67-76.
- Oliveira, M. ; M. Figueiredo ; F. Gomide (1994) - A Neurofuzzy Approach for Autonomous Control - *3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing, IIZUKA '94* Fukuoka, Japan.
- Ollero A., A.Garcia-Cerezo, e J.Aracil. (1991). Design of Rule Based Expert Controllers - *ECC91 European Control Conference*, Grenoble, França, pp 578-583
- Pappis, C.P. & Mamdani, E.H., (1977). A Fuzzy Logic Controller for a Traffic Junction. *IEEE Transactions on Sys., Man and Cybernetics*, V. 7: 625-629.
- Pedrycz, W. (1989). *Fuzzy Control and Fuzzy Systems*. - John Wiley and Sons Inc, NY
- Procyk, T.J. & Mamdani, E.H., (1979). A Linguistic Self-Organising Process Controller. *Automatica*, V. 15: 15-30.
- Rescher, N. (1969), *Many-valued logic*, McGraw-Hill, New York.,
- Sage, A. e C. White (1977). *Optimum Systems Control* - Prentice Hall, NJ
- Scharf, E.M. & Mandic, N.J., (1985). The Application of a Fuzzy Controller to the Control of a Multi-degree of Freedom Robot-Arm. In: *Industrial Applications of Fuzzy Control*, M. Sugeno (Ed.), North-Holland: 1-61.
- Sugeno, M. & Tang, G.T., (1986). Fuzzy Modelling and Control of Multilayer Incinerator. *Fuzzy Sets and Systems*, Vol. 18: 329-346.
- Tanscheit, R. & Scharf, E.M., (1988). Experiments with the use of a Rule-Based Self-Organising Controller for Robotics Applications. *Fuzzy Sets & Systems*, Vol. 26, n. 2: 195-21.

- Tanscheit, R. & Lembessis, E., (1991). On the Behaviour and Tuning of a Fuzzy Rule-based Self-Organising Controller. *13th IMACS World Congress on Computation and Applied Mathematics*, Dublin, Irlanda.
- Takagi, H. ; M. Lee (1993) - Neural Networks and Genetic Algorithms Approaches to Auto-Design of Fuzzy Systems - *Proc. of Fuzzy Logic in Artificial Intelligence - FLAI'93*, Linz, Austria.
- Takagi, T. & Sugeno, M., (1985). Fuzzy Identification of Systems and its Applications to Modelling and Control. *IEEE Transactions on Systems, Man & Cybernetics*, Vol. 15: 116-132.
- Umbers, I.G. & King, P.J. (1980). An Analysis of Human Decision-Making in Cement Kiln Control and the Implications for Automation. *International Journal Man Machines Studies*, vol. 12 n. 1, pp. 11-23
- Van der Rhee, F., Van Nauta Lemke, H.R., Dijkman, J.G., (1990). Knowledge Based Fuzzy Control of Systems. *IEEE Trans. Automatic Control*, V. 35: 18-155.
- Whalen, T. & Schott, B., (1983). Issues in Fuzzy Production Systems. *International Journal of Man-Machine Studies*, Vol. 19: 57-71.
- Yager, R., (1980). An Approach to Inference in Approximate Reasoning *International Journal of Man-Machine Studies*, Vol. 13: 323-338.
- Yager, R. ; S. Ovchinnikov, R.M. Tong e H. T.Nguyen (1987). *Fuzzy Sets and Applications-* Wiley Interscience, NY
- Yamakawa, T. (1993). A Fuzzy Inference Engine in Nonlinear Analog Mode and Its Application to a Fuzzy Logic Control. *IEEE Transactions on Neural Networks*, vol. 4 n. 3, May, pp. 496-522
- Yamazaki, T., (1982). An Improved Algorithm for a Self-Organising Controller. Ph. D. Thesis, Univ. London.,
- Zadeh, L. (1965). Fuzzy Sets - *Information and Control*, vol. 8, pp 338-353
- Zadeh, L. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. on Systems Man and Cybernetics*, vol SMC-3, pp 28-44
- Zadeh, L. (1988) - Fuzzy Logic - *IEEE Computer*, April, pp. 83-92