Proceedings of the V Brazilian Conference on Neural Networks - V Congresso Brasileiro de Redes Neurais
pp. 205–210, April 2–5, 2001 - Rio de Janeiro - RJ - Brazil

Modelagem e Controle Neuro-Fuzzy de Sistemas Dinâmicos Não-Lineares

G. Quadrelli [1], R. Tanscheit [2], M. M. Vellasco [2]
[1] Depto. de Eng. Elétrica, UCP, CP 90.944, 25685-070 Petrópolis, RJ
[2] Depto. de Eng. Elétrica, PUC-Rio, CP 38063, 22452-970 Rio de Janeiro,RJ
E-mails: giovane.quadrelli@ucp.br, [ricardo, marley]@ele.puc-rio.br

Abstract

The main goal of this paper is to propose procedures for the modelling and control of nonlinear systems by using a neuro-fuzzy network topology. The input space of a nonlinear system is initially divided into a number of fuzzy operating regions within which reduced order models (ARMAX) are able to represent the system. The complete system model output – the global model – is obtained through the conjunction of the local models outputs by using a neuro-fuzzy network. A fuzzy adaptive learning control network with hybrid learning (self-organized learning and supervised learning), FALCON-H, is applied to control the neuro-fuzzy plant.

1. Introdução

O controle e a supervisão de processos avançados freqüentemente requerem bastante precisão na sua modelagem e representação. Muitos processos industriais exibem dinâmicas não-lineares, e isto leva a complexidades adicionais nos procedimentos de modelagem utilizados. Recentemente, redes neurais têm mostrado possuir uma boa capacidade de aproximação para uma ampla faixa de funções não-lineares e têm sido utilizadas para modelar sistemas dinâmicos não-lineares. Entretanto, resultados do treinamento das redes neurais em uma representação de "caixa preta" podem ser difíceis de interpretar. A lógica fuzzy tem sido amplamente utilizada em controle, mas, no projeto de sistemas fuzzy, é comum encontrar dificuldades na determinação das regras e das funções de pertinência.

Pode-se dizer que lógica fuzzy e redes neurais são técnicas complementares. Redes neurais extraem informação de sistemas a serem aprendidos ou controlados, enquanto a lógica fuzzy utiliza, muitas das vezes, informação lingüística ou verbal de especialistas. Uma abordagem promissora, de forma a utilizar as vantagens de ambas as técnicas e resolver os seus respectivos problemas, é combiná-las de uma forma integrada (neuro-fuzzy). O sistema integrado possui as vantagens das redes neurais – habilidade de aprendizado e de otimização e estrutura conectivista – e dos sistemas fuzzy – capacidade de incorporação de conhecimento estruturado através de regras SE-ENTÃO. Assim, pelo lado das redes neurais tem-se uma maior transparência, e, pelo lado da lógica fuzzy, a possibilidade de desenvolver métodos de ajuste automático de seus parâmetros, ou seja, uma capacidade de aprender.

O objetivo deste trabalho é apresentar procedimentos neuro-fuzzy tanto para modelar, quanto para controlar uma planta dinâmica não-linear. Além disso, é mostrada, através de uma aplicação, que a utilização simultânea da modelagem e do controle neuro-fuzzy apresenta bons resultados.

2. Modelagem neuro-fuzzy da planta

Na prática, muitos processos não-lineares são aproximados por modelos de ordem reduzida, geralmente lineares, os quais claramente representam as características subjacentes dos processos. Entretanto, estes modelos podem ser válidos somente em certas faixas específicas de operação. Quando as condições de operação mudam, podem ser necessários um modelo diferente ou mudanças nos parâmetros do modelo.

Uma abordagem para modelar processos não-lineares é dividi-los em várias faixas de operação e empregar modelos de ordem reduzida local para aproximar o processo em cada região. O modelo local aqui utilizado é o ARMAX (auto-regressivo, médias móveis com entrada exógena) .

A definição das condições de operação é freqüentemente vaga em sua natureza. Geralmente não é possível definir precisamente as regiões de operação de um processo e também podem existir intersecções entre as várias regiões de operação. Conjuntos fuzzy fornecem um meio apropriado de se definir regiões de operação. Takagi e Sugeno [1] propuseram uma abordagem de modelagem fuzzy para sistemas não-lineares, onde o espaço de entrada é dividido em várias regiões fuzzy e um modelo linear local é usado em cada região. O modelo completo de saída é obtido através da defuzzificação pelo método do centro de gravidade.

No trabalho aqui desenvolvido utiliza-se uma rede neuro-fuzzy para computar os modelos lineares locais (ARMAX) dentro de um sistema integrado, representando o sistema não-linear [2]. Esta rede neuro-fuzzy combina a capacidade de sistemas fuzzy de lidar com informações imprecisas com a capacidade das redes neurais de aprender através de exemplos.

Na rede neuro-fuzzy, um modelo fuzzy é estruturalmente mapeado em uma rede neural e a estrutura da rede é determinada pelas regras fuzzy do modelo. Conhecimento prévio a respeito do processo é

usado para iniciar a partição em várias regiões de operação e para inicializar os correspondentes pesos da rede. Dados de entrada e saída são então utilizados para treinar a rede.

2.1. Modelagem fuzzy de processos não-lineares

A operação global de um processo não-linear é dividido em várias regiões de operação locais. Em cada região local R_i um modelo de ordem reduzida linear local ARMAX representa o comportamento do processo. Este modelo não é restritivo, podendo qualquer outro modelo linear apropriado ser utilizado. Conjuntos fuzzy são usados para definir as condições de operação de forma que o modelo dinâmico de um processo não-linear possa ser descrito da seguinte forma:

R_i : SE condição de operação i ENTÃO

$$y_i^p(t) = \sum_{j=1}^{no} a_{ij} y(l-j) + \sum_{j=1}^{ni} b_{ij} u(t-j)\ ,\ i=1,2,...,nr$$

A saída do modelo final é obtida através da defuzzificação pelo método do centro de gravidade:

$$y^p(t) = \sum_{i=1}^{nr} \mu_i y_i^p(t) \Big/ \sum_{i=1}^{nr} \mu_i \qquad (1)$$

No modelo acima, y é a saída do processo, u é a entrada, y^p é a previsão da saída na i-ésima região de operação, nr é o número de regiões de operação fuzzy, ni e no são os "lags" de tempo na entrada e saída, respectivamente, μ_i é a função de pertinência do i-ésimo modelo, a_{ij} e b_{ij} são os parâmetros do modelo ARMAX e t representa um tempo discreto.

Suponha que x e y são as variáveis de processo usadas para definir as regiões de operação e que a elas são associadas os conjuntos fuzzy: "baixo", "médio" e "alto". A i-ésima região de operação pode ser definida, por exemplo, como x é alto AND y é médio. A função de pertinência para esta região de operação pode ser construída de várias formas. Duas abordagens possíveis para o cálculo de sua função de pertinência são:

$$\mu_i = \min(\mu_h(x), \mu_m(y)) \qquad (2)$$

$$\mu_i = \mu_h(x)\mu_m(y) \qquad (3)$$

Nas eqs. (2) e (3), μ_i é a função de pertinência da i-ésima região de operação, $\mu_h(x)$ é a função de pertinência de x "alto" e $\mu_m(y)$, a de y "médio".

2.2. Representação de modelos fuzzy na forma de redes neurais

O modelo fuzzy descrito acima pode ser representado por um tipo especial de topologia de rede denominada rede neuro-fuzzy, que pretende combinar as vantagens de ambas as abordagens, fuzzy e neural.

A topologia da rede neuro-fuzzy é mostrada na Figura 1. Esta contém quatro camadas: uma camada de fuzzificação, uma camada de regras, uma camada de funções e uma camada de defuzzificação. Os diferentes tipos de neurônios são analisados a seguir.

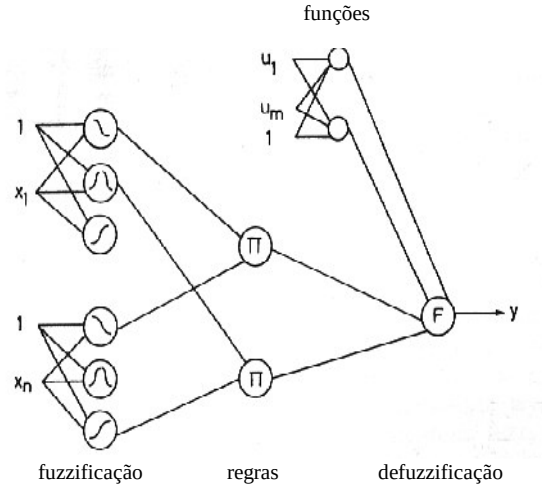

Figura 1: Topologia da rede neuro-fuzzy

As entradas da camada de fuzzificação são as variáveis de processo usadas para definir as regiões de operação. Cada uma dessas variáveis é transformada em vários conjuntos fuzzy nessa camada. Cada neurônio nesta camada corresponde a um conjunto fuzzy particular com função de pertinência dada pela saída do neurônio. Três tipos de funções de ativação são usados: a função sigmóide, a função gaussiana e a função sigmóide complementar.

Cada neurônio na camada de regras corresponde a uma região de operação do processo que está sendo modelado. Suas entradas são as ações fuzzy, as quais determinam as correspondentes regiões de operação. Sua saída é o produto de suas entradas e é a função de pertinência da região de operação correspondente. Neurônios na camada de regras implementam a intersecção fuzzy definida na eq. (2). Nesta camada não há pesos a ser estimados. O conhecimento do número de regiões de operação e de como estas são estabelecidas é usado para construir a camada de regras.

Neurônios na camada de funções implementam os modelos de ordem reduzida (aqui modelos ARMAX) nas regiões de operação fuzzy. Cada neurônio corresponde a uma região de operação particular e é um neurônio linear. Sua saída é a soma de suas entradas, as quais são variáveis de processos multiplicadas pelos respectivos pesos. Um *bias* é incluído em cada neurônio para representar o termo constante no modelo local. Os pesos na camada de funções são os parâmetros dos modelos lineares nas regiões de operação. A camada de defuzzificação realiza a defuzzificação através do centro de gravidade e fornece a saída final da rede. As entradas do neurônio de defuzzificação são funções de

pertinência das regiões de operação e saídas dos modelos locais destas regiões. A função de ativação desses neurônios é dada pela eq. (1) e novamente não há pesos nesta camada.

2.3. Treinamento da rede neuro-fuzzy

A rede neuro-fuzzy pode ser treinada através de vários métodos. Neste trabalho é utilizado o método de back-propagation. Os pesos da rede, dados pelas eqs. (4) e (5) abaixo, são ajustados por um algoritmo que minimiza a soma do quadrado dos erros, de acordo com a eq. (6).

$$\Delta\omega(k+1) = \alpha\Delta\omega(k) - \eta\delta \tag{4}$$

$$\omega(k+1) = \omega(k) + \Delta\omega(k+1) \tag{5}$$

Nas eqs. (4) e (5) acima $\omega(k)$ e $\Delta\omega(k)$ são, respectivamente, o peso e a adaptação do peso no passo k, α é o coeficiente de momentum, η é a taxa de aprendizado e δ é o gradiente da soma do erro quadrado com respeito ao peso ω. A exemplo de outros tipos de métodos de treinamento baseados em gradientes, o treinamento é finalizado quando o erro é menor do que um valor especificado.

O cálculo dos gradientes é feito pela minimização da função a seguir:

$$J = \sum_{i=1}^{N} (y_i^p - y_i)^2 \tag{6}$$

Na eq. (6) N é o número de dados, y^p é a previsão da rede e y é o valor desejado.

3. Controlador FALCON-H

O controlador FALCON (fuzzy adaptive learning control network) foi proposto por Lin e Lee (1991) para estudar estratégias de aprendizado de parâmetros híbridos [3]. O controlador FALCON é uma rede multicamadas feedforward que integra os elementos básicos e as funções de um controlador tradicional fuzzy em uma estrutura conectivista que tem habilidades de aprendizado (redes neurais). Nesta estrutura conectivista os nós de entrada e de saída representam as entradas de estado e as saídas de controle, respectivamente. Nas camadas escondidas tem-se os nós de funcionalidade, tais como funções de pertinência e regras fuzzy.

3.1. Estrutura do controlador FALCON-H

A Figura 2 mostra a estrutura do controlador FALCON-H. Associada às entradas dos nós tem-se uma função de integração *f*, que serve para combinar informações e ativações de outros nós. Esta função fornece a entrada (*net*) para este nó:

$$net_i = f(u_1^{(k)}, u_{21}^{(k)}, ..., u_{p1}^{(k)}, \omega_1^{(k)}, \omega_2^{(k)}, ..., \omega_p^{(k)}) = \\ = \text{entradas totais do nó } i \tag{7}$$

Na eq. (7), u_i = entrada, ω_i = peso, k = camada e i = nó.

A saída do nó em função de sua entrada (*net*) é dada por:

$$\text{saída} = o_i^k = a(net_i) = a(f) \tag{8}$$

onde $a(.)$ = função de ativação

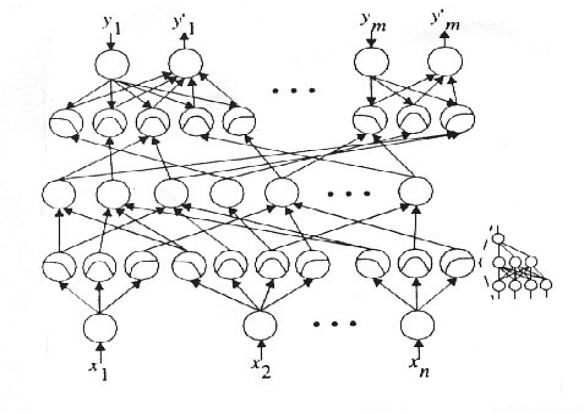

Figura 2: Estrutura do FALCON-H

3.1.1. Análise das camadas do FALCON-H

Camada 1 $\Rightarrow$ Nós lingüisticos de entrada: transmitem diretamente os valores das entradas para a segunda camada:

$$f = u_i^{(1)} \tag{9}$$

onde: $a = f$: linear e $\omega_i^{(1)} = 1$

Camada 2 $\Rightarrow$ Nós termos de entrada (nós atuando como funções de pertinência de entrada). Por exemplo, para uma função com distribuição Gaussiana:

$$f = M_{xi}^{j}(m_{ij}, \sigma_{ij}) = -(u_i^{(2)} - m_{ij})^2 / \sigma_{ij}^2 \tag{10}$$

onde $a = e^f$
m_{ij} = centro da função de pertinência
σ_{ij}^2 = largura da função de pertinência ;
o peso $\omega_{ij}^{(2)}$ pode ser interpretado como m_{ij}

Camada 3 $\Rightarrow$ Nós regras: as conexões (*links*) desta camada são usadas para representar os antecedentes das regras; representam uma operação fuzzy AND:

$$f = \min(u_1^{(3)}, u_2^{(3)}, ..., u_p^{(3)}) \tag{11}$$

onde: $a = f$ e $\omega_i^{(3)} = 1$

Camada 4 $\Rightarrow$ Nós termos de saída (nós atuando como funções de pertinência de saída); modos de operação:

- *Modo down-up*: as conexões (*links*) representam uma operação fuzzy OR para integrar as regras de disparo com o mesmo conseqüente.

$$f = \sum_{i} u_i^{(4)} \qquad (12)$$

onde: $a = \min(1, f)$ e $\omega_i^{(4)} = 1$

- *Modo up-down*: idêntico à camada 2, apenas com nós simples

Camada 5 $\Rightarrow$ Nós lingüisticos de saída:

- *Nós up-down*: usados para treinar a rede:

$$f = y_i \qquad (13)$$

onde: $a = f$: linear

- *Nós down-up*: sinal de decisão de saída; estes nós atuam como defuzzificador. Por exemplo, no caso de centro de área:

$$f = \sum \omega_{ij}^{(5)} u_{ij}^{(5)} = \sum (m_{ij}\sigma_{ij}) u_{ij}^{(5)} \qquad (14)$$

onde: $a = f \Big/ \sum_{j} \sigma_{ij} u_{ij}^{(5)}$ e $\omega_{ij}^{(5)} = m_{ij}\sigma_{ij}$

3.2. Algoritmo de aprendizado híbrido

O FALCON-H possui um algoritmo de aprendizado híbrido, consistindo de duas fases distintas: *aprendizado não-supervisionado* e *aprendizado supervisionado* (back-propagation). Este algoritmo de aprendizado híbrido apresenta melhores resultados do que um algoritmo puramente supervisionado (back-propagation), devido a uma classificação a priori dos dados de treinamento através de um campo receptivo de intersecção antes do treinamento supervisionado.

3.2.1. Aprendizado não-supervisionado

Nesta fase, a rede opera na forma *two-sided*, isto é, os nós e *links* na camada 4 estão no modo de transmissão *up-down*, de forma que os dados de entrada para treinamento e os dados de saída podem ser alimentados ao controlador FALCON-H de ambos os lados.

Primeiramente, os centros (médias) e as larguras (variâncias) das funções de pertinência são determinados, pelo aprendizado auto-organizado, através de técnicas análogas àquelas de clusterização estatística. Isto serve para alocar os recursos da rede de forma mais eficiente, através do posicionamento dos domínios das funções de pertinência cobrindo somente as regiões do espaço entrada-saída onde os dados estão presentes.

O algoritmo de Kohonen é adotado para encontrar o centro m_i da i-ésima função de pertinência de x, onde x representa qualquer uma das variáveis lingüísticas de entrada (x_1,..., x_n) ou saída (y_1,.., y_m):

$$\| x(t) - m_{closest}(t) \| = \min_{1 \le i \le k} \{ \| x(t) - m_i(t) \| \} \qquad (15)$$

$$m_{closest}(t+1) = m_{closest}(t) + \alpha(t)[x(t) - m_{closest}(t)] \qquad (16)$$

$$m_i(t+1) = m_i(t) \text{ para } m_i \neq m_{closest} \qquad (17)$$

onde $\alpha(t)$ = taxa de aprendizado escalar decrescente monotonicamente.

Uma vez determinados os centros das funções de pertinência, as suas larguras podem ser determinadas pela utilização da heurística do N-vizinho-mais-próximo (N-nearest-neighbour). Uma vez que a segunda fase de aprendizado irá ajustar os centros e larguras, estas podem simplesmente ser determinadas pelo primeiro vizinho mais próximo nesse estágio:

$$\sigma_i = | m_i - m_{closest} | / r \qquad (18)$$

onde é atribuído um valor inicial para o parâmetro de intersecção r.

Após o aprendizado competitivo ter envolvido todo o conjunto de dados de treinamento, os *links* dos pesos da camada 4 representam os esforços de existência do conseqüente da regra correspondente. Os *links* com pesos maiores são mantidos e os restantes são excluídos.

Se os *links* entre um nó regra e um nó termo são todos muito pequenos, o primeiro não tem relação (ou esta é muito pequena) com aquela variável lingüistica de saída (nó de saída); assim, todos os seus *links* podem ser deletados. Se todos os *links* entre um nó regra e os nós da camada 4 são deletados, então esta regra pode ser eliminada, pois não afeta as saídas.

Após a determinação dos conseqüentes dos nós regras, uma combinação de regras é usada para reduzir o número de regras. Os critérios para combinar um conjunto de nós regras em um nó regra simples são: que eles tenham exatamente o mesmo conseqüente, que alguns antecedentes sejam comuns para todos os nós regras do conjunto e que a união de outros antecedentes desses nós regras abranjam todo o conjunto de termos daquelas variáveis lingüisticas de entrada.

3.2.2. Aprendizado supervisionado

Nesta fase de aprendizado a rede neuro-fuzzy funciona na maneira *feedforward* , isto é, os nós e os *links* da camada 4 e 5 estão no modo de transmissão *down-up*. É então utilizado o algoritmo de back-propagation.

4. Aplicação

A técnica de modelagem e controle neuro-fuzzy foi aplicada a um processo dinâmico não-linear que relaciona o percentual de gás carbônico (%CO_2) e a taxa de um determinado gás (pés^3/min) em uma dada instalação [6], conforme mostrado na Figura 3.

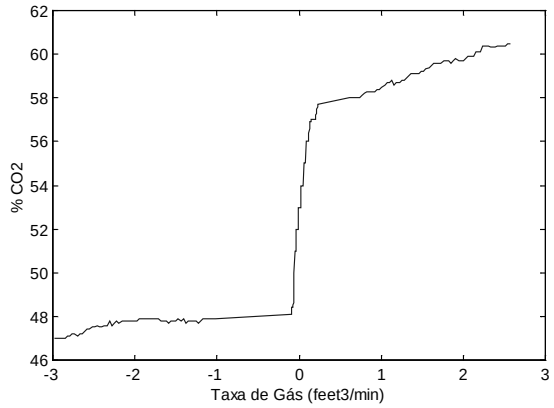

Figura 3: Relação entre %CO_2 e taxa de gás

4.1. Modelagem neuro-fuzzy

A rede neuro-fuzzy foi desenvolvida para modelar a relação entre o %CO_2 e a taxa de gás utilizada para realizar o processo.

O processo foi dividido em três regiões de operação: %CO_2 baixo, %CO_2 médio e %CO_2 alto. Foram utilizados 200 pontos para treinamento e 200 pontos para testar a rede neuro-fuzzy, com taxa de aprendizado η = 0.1 e coeficiente de momentum α = 0.2; o treinamento foi finalizado para um gradiente de erro de 10^{-3}. Obteve-se o seguinte modelo fuzzy após o treinamento:

R1: Se %CO_2 é baixo:

$$y(t) = 0.9996\,y(t-1) - 0.0170\,u(t-1)$$

R2: Se %CO_2 é médio:

$$y(t) = 1.0015\,y(t-1) - 0.1813\,u(t-1)$$

R3: Se %CO_2 é alto:

$$y(t) = 1.001\,y(t-1) - 0.0143\,u(t-1)$$

A Figura 4 apresenta o desempenho da rede neuro-fuzzy em função dos dados de teste. Para medir o desempenho da rede foi utilizado o erro percentual absoluto médio (EPAM= 1.33 %.).

4.2. Controle da planta neuro-fuzzy

O controlador FALCON-H foi utilizado para controlar a planta neuro-fuzzy, proporcionando o resultado mostrado na Figura 5. O setpoint é um degrau variando de 60% CO_2 para 50% CO_2. A estrutura do FALCON-H é a seguinte:

- Camada 1: dois nós (erro (e) e variação do erro (Δe));
- Camada 2: sete nós com funções de pertinência gaussianas para cada nó da camada 1;
- Camada 3: 49 nós de regras (AND);
- Camada 4: sete nós com funções de pertinência gaussianas;
- Camada 5: um nó de saída (ação de controle).

Foram utilizados 200 pontos para treinamento e 200 pontos para testar a rede neuro-fuzzy, com taxa de aprendizado η = 0.15, parâmetro de intersecção r = 2 e gradiente de erro = 10^{-2}.

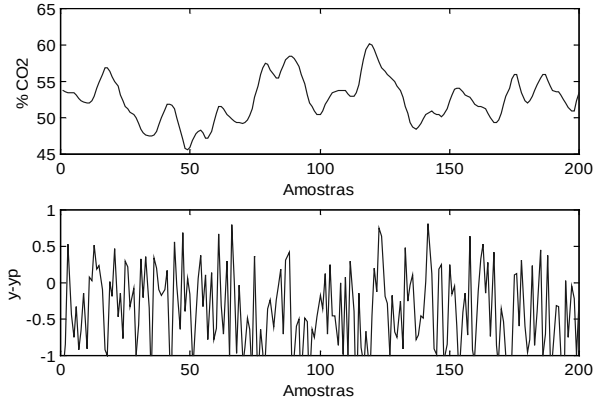

Figura 4: Desempenho da planta neuro-fuzzy

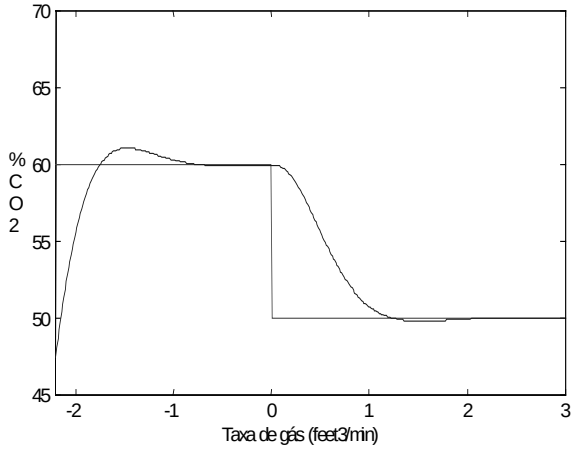

Figura 5: Controle da planta neuro-fuzzy

5. Conclusões

A abordagem neuro-fuzzy combina as vantagens das modelagens fuzzy e de redes neurais. Além disso, os modelos neuro-fuzzy são mais fáceis de interpretar do que modelos de redes neurais apenas. Uma outra vantagem é que o sistema neuro-fuzzy é capaz de criar um conjunto de regras apropriado, eliminando a eventual dificuldade de definição da base de regras. Os resultados da aplicação demonstram que a abordagem neuro-fuzzy é efetiva tanto na modelagem, quanto no

controle de um processo dinâmico não-linear. Além disso, demonstram que a utilização simultânea da modelagem e do controle através de uma abordagem neuro-fuzzy apresenta bons resultados. Portanto, as técnicas aqui apresentadas configuram-se em uma alternativa na modelagem e controle de sistemas complexos e não-lineares.

Referências

[1] T. Takagi and M. Sugeno. Fuzzy identification of systems and its aplication to modelling and control. *IEEE Transactions on Systems, Man and Cybernetics,* 15, (1): 116-132, 1985.

[2] J. Zhang and A. J. Morris. Fuzzy neural networks for nonlinear systems. Proc. *IEEE - Control Theory Applications*, 142(6): 551-561, 1995.

[3] C. T. Lin and C. S G. Lee. *Neural fuzzy systems*. Prentice Hall, 1996.

[4] J. R. Jang and C. T. Sun. Neuro-fuzzy modeling and control. *Proc. IEEE*, 83(3): 378-406, 1995.

[5] M. Figueiredo and F. Gomide. Design of fuzzy systems using neurofuzzy networks. *IEEE Trans. on Neural Networks*, 10(4): 815-827,1999.

[6] G. E. P. Box, G. M. Jenkins, G. C. Reinsel. *Time Series Analysis*. Prentice Hall, 1994.