Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica

Técnicas Inteligentes Híbridas para o Controle de Sistemas Não Lineares

Marconi Câmara Rodrigues

Orientador: Prof. Dr. Fábio Meneghetti U. Araújo

Natal, RN, fevereiro de 2006

Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica

Técnicas Inteligentes Híbridas para o Controle de Sistemas Não Lineares

Marconi Câmara Rodrigues[1]

Orientador: Prof. Dr. Fábio Meneghetti U. Araújo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da UFRN (área de concentração: Automação e Sistemas) como parte dos requisitos para obtenção do título de Mestre em Ciências de Engenharia Elétrica.

Natal, RN, fevereiro de 2006

[1]Beneficiário de auxílio financeiro da CAPES-Brasil.

Divisão de Serviços Técnicos

Catalogação da publicação na fonte. UFRN / Biblioteca Central Zila Mamede

Rodrigues, Marconi Câmara.

Técnicas inteligentes híbridas para o controle de sistemas não-lineares / Marconi Câmara Rodrigues - Natal, RN, 2006.

62 f.

Orientador: Fábio Meneghetti U. Araújo.

Dissertação (Mestrado) - Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica.

1. Rede neural - Dissertação. 2. Técnicas híbridas - Dissertação. 3. Controladores inteligentes – Dissertação. 4. NEFCON – Dissertação 5. ANFIS – Dissertação. I. Araújo, Fábio Meneghetti U. II. Título.

RN/UF/BCZM CDU 004.032.26(043.3)

Técnicas Inteligentes Híbridas para o Controle de Sistemas Não Lineares

Marconi Câmara Rodrigues

Dissertação de Mestrado aprovada em 17 de fevereiro de 2006 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Fábio Meneghetti Ugulino de Araújo (orientador) DCA/UFRN

Dr. Eng. Hilton Cleber Pietrobom ..... IAE/CTA

Prof. Dr. André Laurindo Maitelli ..... DCA/UFRN

Prof. Dr. Luiz Affonso H. Guedes de Oliveira ..... DCA/UFRN

Agradecimentos

Aos meus pais, pela confiança e auxílio nas horas mais difíceis, ajudando com apoio incondicional, muitas vezes sem conhecer o que estava sendo desenvolvido.

Ao meu irmão Marcelo Câmara Rodrigues, por ser aquele em quem pude contar para um desenvolvimento acadêmico e, principalmente, pessoal. Pessoa esta, que poderei contar hoje e sempre.

Aos amigos mais próximos com características variadas, bom humor e dedicação, que todos os dias me mostravam o verdadeiro significado da palavra amizade.

À Miriam Valença Massud, que além de me aturar como amigo, não negava auxílio nos momentos de dificuldade, além de participar e proporcionar muitos momentos de felicidade.

Ao grande amigo Marcelo Borges Nogueira, por sua capacidade de sintetizar as situações mais difíceis em ações simplificadas, criando assim, uma notável facilidade de convivência.

Ao meu orientador Fábio Meneghetti e ao professor André Laurindo Maitelli, que me guiaram com sucesso na produção e elaboração deste trabalho.

À CAPES, pelo apoio financeiro.

Finalmente, a todos que de forma direta ou indireta me auxiliaram na confecção deste trabalho.

Obrigado!!!

Resumo

Neste trabalho é mostrado tanto o desenvolvimento quanto as características de algumas das principais técnicas utilizadas para o controle inteligente de sistemas. Partindo de um controlador *fuzzy* foi possível aplicar técnicas de aprendizagem, similares às utilizadas pelas Redes Neurais Artificiais (RNA's), e evoluir para os modelos *neuro-fuzzy* ANFIS e NEFCON. Estes modelos *neuro-fuzzy* foram aplicados a uma planta real do tipo *ball and beam* e tiveram tanto suas adaptações quanto seus resultados comentados. Para cada controlador desenvolvido são especificadas as variáveis de entrada, os parâmetros utilizados para a adaptação das variáveis e os algoritmos aplicados em cada um deles. Já os resultados estão voltados para a obtenção de um comparativo entre a fase inicial e a final da evolução dos controladores *neuro-fuzzy*, assim como, a aplicabilidade de cada um deles de acordo com suas características intrínsecas.

Palavras-chave: Controle, Técnicas Híbridas, Sistemas Fuzzy, Redes Neurais Artificiais, NEFCON, ANFIS.

Abstract

A neuro-fuzzy system consists of two or more control techniques in only one structure. The main characteristic of this structure is joining one or more good aspects from each technique to make a hybrid controller. This controller can be based in Fuzzy systems, artificial Neural Networks, Genetics Algorithms or reinforced learning techniques. Neuro-fuzzy systems have been shown as a promising technique in industrial applications. Two models of neuro-fuzzy systems were developed, an ANFIS model and a NEFCON model. Both models were applied to control a ball and beam system and they had their results and needed changes commented. Choose of inputs to controllers and the algorithms used to learning, among other information about the hybrid systems, were commented. The results show the changes in structure after learning and the conditions to use each one controller based on theirs characteristics.

Keywords: Control, Hybrid Systems, Fuzzy Systems, Artificial Neural Networks, NEFCON, ANFIS.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\mu_A(x)$ = Função que indica a pertinência de x no conjunto *fuzzy* A.

$\top$ = Operador t-norma

$\perp$ = Operador t-conorma

$\delta_{k,i}$ = Sinal de erro do neurônio i da camada k

K = Última camada da rede neural

$X_{k,i}$ = Saída do neurônio i da camada k

η = Coeficiente de aprendizagem

L = Pontos de treinamento

$\Delta\alpha$ = Variação para cada parâmetro ajustável

$\overline{w}_i$ = Normalização da variável w_i

$bell(x, a, b, c) = \frac{1}{1+|\frac{x-c}{a}|^{2b}}$ = Função em formato de sino

η_{otm} = Saída ótima do controlador

$[L_x^{min}, L_x^{max}]$ = Intervalo de atuação para a entrada x

$sgn(tr)$ = Sinal de tr

a, b, c = Vértices das funções de pertinência triangulares

p, q, r, s = Pesos do polinômio Sugeno de primeira ordem

d_s^l = Valor desejado para a saída

E_p^l = Erro pontual

Lista de Abreviaturas

RNA – Redes Neurais Artificiais

ANFIS – Adaptive-Network-Based Fuzzy Inference System

NEFCON – Neuro-Fuzzy Controller

PID – Proporcional-Integrativo-Derivativo

Capítulo 1

Introdução

Controladores cujas técnicas já são bem estabelecidas, como é o caso dos PID's e dos controladores baseados em realimentação de estados, possuem uma simples implementação e um baixo custo computacional. Porém, o ajuste dos seus parâmetros pode tomar um tempo considerável e seu desempenho costuma ser limitado. Algumas técnicas de ajuste automático para os parâmetros dos controladores PID foram desenvolvidas (J. Oliveira, 1994; M. Berto *et alii*, 2004; L. Coelho & A. Coelho, 1999). Essas técnicas, além de elevar o custo computacional, não tornam os controladores PID bem capacitados a resolver problemas como, por exemplo, o controle de sistemas não-lineares ou a manutenção do desempenho mesmo na presença de incertezas ou variações paramétricas.

No âmbito da necessidade de técnicas com maior abrangência na área de controle, surgiram as técnicas inteligentes, como: Redes Neurais Artificiais, Sistemas *fuzzy* ou Nebulosos, Algoritmos Genéticos e algumas outras técnicas baseadas em aprendizagem por reforço. Estas técnicas são fundamentadas em teorias evolucionistas, em teorias de como as estruturas neurais assimilam o conhecimento ou em uma lógica simples, formada por pertinências e regras. Algumas destas técnicas são utilizadas apenas para otimização, já outras podem ser utilizadas para o controle de sistemas.

Os sistemas *fuzzy* e as Redes Neurais Artificiais (RNA's) se mostram como ferramentas de grande utilidade no controle de sistemas não-lineares, com ou sem

modelos matemáticos. Estas duas técnicas serão o foco principal deste trabalho já que foi a partir de suas estruturas que se desenvolveram os controladores híbridos aqui apresentados.

As RNA's têm como objetivo tentar reproduzir a capacidade de aprender e generalizar o conhecimento da estrutura cerebral dos seres vivos. Baseando-se em estruturas simples conhecidas como neurônios artificiais, os dados se propagam de neurônio em neurônio através de sinapses, cujos pesos podem ser ajustados no decorrer do seu aprendizado (S. Haykin, 2001).

A teoria dos conjuntos *fuzzy*, que utiliza incertezas não probabilísticas, foi introduzida por Lotfi Zadeh (L. A. Zadeh, 1965) e marcou o início de uma série de modelos onde se destacam: Mamdani (E. Mamdani & S. Assilan, 1975), Sugeno (M. Sugeno & G. Kang, 1988; T. Takagi & M. Sugeno, 1985) e Tsukamoto (Y. Tsukamoto, 1979).

Cerca de trinta anos após a introdução da teoria dos conjuntos *fuzzy*, o pesquisador Jyn-Shing Roger Jang publicou um artigo no qual os parâmetros *fuzzy* são calculados através da técnica de retro-propagação do erro (*Back-propagation*), vastamente utilizada para o ajuste dos pesos sinápticos nas RNA's (J. Jang & Chuen-Tsai S., 1995). Esta técnica de associar sistema *fuzzy* com redes neurais artificiais ficou conhecida como *neuro-fuzzy* e o modelo implementado por Jang foi chamado de *Adaptive-Network-Based Fuzzy Inference System*, ou apenas, ANFIS. Outros modelos *neuro-fuzzy* também se desenvolveram, e, dentre eles, o modelo NEFCON (*Neuro-Fuzzy Controller*) se destacou por ter características de fácil implementação em tempo real. Apesar do elevado custo computacional, o desenvolvimento destas técnicas foi possível graças ao crescimento tecnológico dos últimos anos e à necessidade da indústria em produzir cada vez mais equipamentos automáticos e que minimizem a interferência humana.

Atualmente, podem ser encontrados no mercado desde eletrodomésticos até sistemas de suspensão de carros de última geração que utilizam tecnologias muito avançadas para seu controle (A. El Hajjaji & S. Bentalba, 2003; E. Purwanto *et alii*, 2001; H. Hong & L. Jian-Hua, 2003; T. Amaral & M. Crisóstomo, 2001). RNA's

podem ser usadas em servidores de e-mail para identificar mensagens eletrônicas indesejáveis ou otimizar o envio de e-mails evitando congestionamento nas redes (J. Goodman *et alii*, 2005). Para evitar possíveis erros, pode-se utilizar diferentes controladores operando ao mesmo tempo, coordenados ou não por um sistema *fuzzy*. Quando é identificada a falha em um sistema, outro assume seu posto imediatamente evitando que o problema se agrave. Uma alternativa para evitar o chaveamento entre controladores é o uso de técnicas híbridas que possuam características de vários tipos de controladores distintos de forma que, as características de um controlador supram as necessidades dos outros.

A proposta dos controladores *neuro-fuzzy* baseia-se no que foi dito até o momento. Esses controladores utilizam como estrutura básica um sistema *fuzzy*, com fácil identificação de ações e que tem carência na fase de ajuste de seus parâmetros, e RNA's que apresentam uma fase de aprendizado e em contrapartida, não permitem o fácil acesso aos dados codificados em sua estrutura.

É baseado na potencial aplicabilidade dos controladores inteligentes híbridos ANFIS e NEFCON que este trabalho foi desenvolvido, mostrando características de implementação e de uso, características intrínsecas de cada modelo, assim como vantagens e desvantagens.

O próximo capítulo deste trabalho é referente aos controladores inteligentes fundamentados nos sistemas *fuzzy* e nas RNA's, como ocorre o fluxo de dados, principais características e principais modelos. Em seguida, no capítulo 3, verificam-se as abordagens de implementação *neuro-fuzzy* propostas pelos modelos ANFIS e NEFCON. No capítulo 4 é apresentada a estrutura de controle empregada para a obtenção dos resultados da aplicação, em tempo real, das técnicas híbridas sob a planta física *ball and beam*, assim como as comparações dos resultados obtidos. No último capítulo, capítulo 5, encontram-se as conclusões referentes às técnicas apresentadas e os possíveis trabalhos futuros.

Capítulo 2

Controladores Inteligentes

2.1 Controladores *fuzzy*

Em conjuntos convencionais um elemento x ou pertence a um conjunto A ou não pertence a este conjunto, não existindo um estado intermediário em que o valor de x está parcialmente presente no conjunto A. Nos controladores *fuzzy*, as funções de pertinência são responsáveis por informar com que intensidade o elemento x é compatível com um determinado conjunto A. A partir desta característica, um elemento pode pertencer parcialmente a determinado conjunto com uma intensidade descrita pela função de pertinência.

Os controladores *fuzzy* utilizam um conjunto de regras do tipo SE <premissa> ENTÃO <conclusão> que definem ações em função de diversas faixas de valores que as variáveis de estado do problema podem assumir. Tais controladores baseiam-se na teoria dos conjuntos nebulosos, desenvolvida por Zadeh (L. A. Zadeh, 1965). Tal teoria estabelece que, um conjunto nebuloso A, do universo de discurso Ω, é definido por uma função de pertinência $\mu_A : \Omega \rightarrow [0, 1]$. Essa função associa a cada elemento de x de Ω, o grau de pertinência $\mu_A(x)$. Desta forma, a função de pertinência $\mu_A(x)$ indica o grau de compatibilidade entre x e o conceito expresso por A:

- Se $\mu_A(x) = 1$, então x é completamente compatível com A;
- Se $\mu_A(x) = 0$, então x é completamente incompatível com A;

- Se $0 < \mu_A(x) < 1$, então x é parcialmente compatível com A, com grau $\mu_A(x)$.

Para ilustrar tal conceito observe a figura 2.1, na qual a pertinência de x no conjunto nebuloso A pode ser descrita pela função expressa em (2.1).

$$\mu_A(x) = \begin{cases} x - 4, & \text{se } 4 <= x <= 5; \\ 1, & \text{se } 5 < x <= 7; \\ 8 - x, & \text{se } 7 < x <= 8; \\ 0, & \text{caso contrário.} \end{cases} \tag{2.1}$$

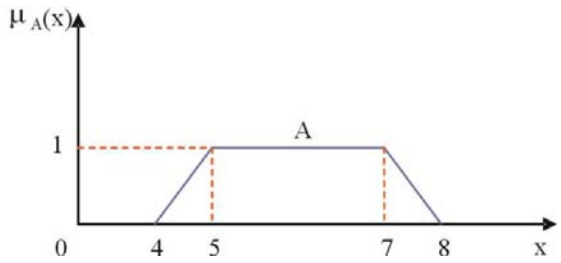

Figura 2.1: Representação de um conjunto nebuloso A.

Vale salientar que as funções de pertinência podem assumir várias formas, ficando a cargo do projetista a escolha da forma mais conveniente para sua aplicação (figura 2.2).

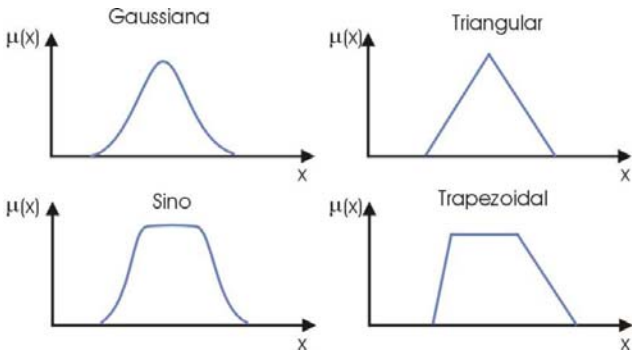

Figura 2.2: Formatos mais comuns para as funções de pertinência.

Assim como ocorre com os conjuntos clássicos, algumas operações podem ser realizadas sobre os conjuntos nebulosos. A intersecção é implementada por um

Tabela 2.1: As t-normas e t-conormas mais utilizadas.

t-norma	t-conorma	nome
$min(a, b)$	$max(a, b)$	Zadeh
$a \cdot b$	$a + b - ab$	Probabilística
$max(a + b - 1, 0)$	$min(a + b, 1)$	Lukasiewicz

conjunto de operadores denominados de t-normas ($\top$), e a união é implementado por outro conjunto denominados de t-conormas ($\perp$) ou s-normas. A tabela 2.1 indica as t-normas e t-conormas mais utilizadas.

Segundo S. Sandri & C. Correa (1999), apesar das muitas variações propostas na literatura, a representação da estrutura básica de um controlador *fuzzy* proposta por C. C. Lee (1990) (figura 2.3) é um modelo geral e suficiente para identificação dos módulos que compõem tal controlador, fornecendo, assim, uma idéia do fluxo das informações dentro do mesmo.

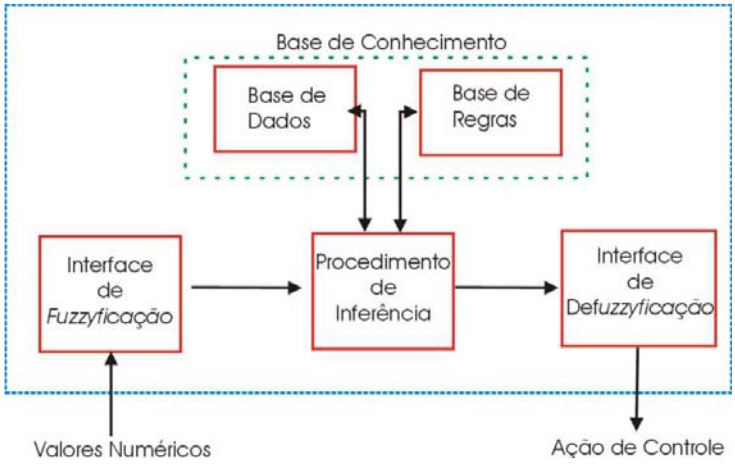

Figura 2.3: Estrutura de um controlador nebuloso.

Acompanhando o fluxo da informação em um controlador *fuzzy*, desde a entrada até a saída, podem-se constatar três etapas básicas: a *fuzzyficação*, a inferência e a *defuzzyficação*. Na etapa de *fuzzyficação*, empregando-se conjuntos de funções de pertinência, convertem-se as variáveis de entrada, oriundas do meio externo, em variáveis *fuzzy*, expressas pela pertinência em cada conjunto *fuzzy*. As variáveis *fuzzy* são submetidas a um processo de inferência com base em regras do tipo SE −

ENTÃO, gerando as saídas *fuzzy*. Na etapa de *defuzzyficação* as saídas *fuzzy*, que não têm qualquer significado físico no meio externo, são convertidas em variáveis de saída com algum significado físico.

Os modelos de inferência *fuzzy* são identificados pela forma com que é implementada a etapa de inferência. Estes modelos podem ser classificados como: clássicos ou por interpolação.

2.1.1 Modelos Clássicos

Nos modelos clássicos, a conclusão de cada regra especifica um termo nebuloso dentre um conjunto fixo de termos. Estes termos serão chamados de funções de pertinência de saída e estarão distribuídos dentro do intervalo de saída do controlador. A combinação das funções de pertinência de saída especificadas pelas regras ativas formará a saída *fuzzy* do controlador que, por sua vez, pode ser convertida em sinal de controle ao aplicar-se uma das técnicas de *defuzzyficação*. O modelo clássico mais comum é o proposto por Mamdani.

Modelo de Mamdani

O modelo de Mamdani utiliza o modelo clássico do processo de inferência *fuzzy*. Considere um sistema com duas entradas *fuzzy* x e y, e com uma saída *fuzzy* z. As duas entradas x e y irão passar por um conjunto de regras do tipo:

SE x é A_1 com μ_{x,A_1} E y é B_1 com μ_{y,B_1} ENTÃO z é C_1 com $\mu_{z,C_1} = \top(\mu_{x,A_1}, \mu_{y,B_1})$
SE x é A_2 com μ_{x,A_2} E y é B_2 com μ_{y,B_2} ENTÃO z é C_2 com $\mu_{z,C_2} = \top(\mu_{x,A_2}, \mu_{y,B_2})$;

Em que A_1, A_2, B_1, B_2, C_1 e C_2 são conjuntos nebulosos. Pode-se obter uma saída *fuzzy* aplicando o operador t-conorma,

$$z = \bot(\mu_{z,C_1}, \mu_{z,C_2});$$

A figura 2.4 ilustra este exemplo.

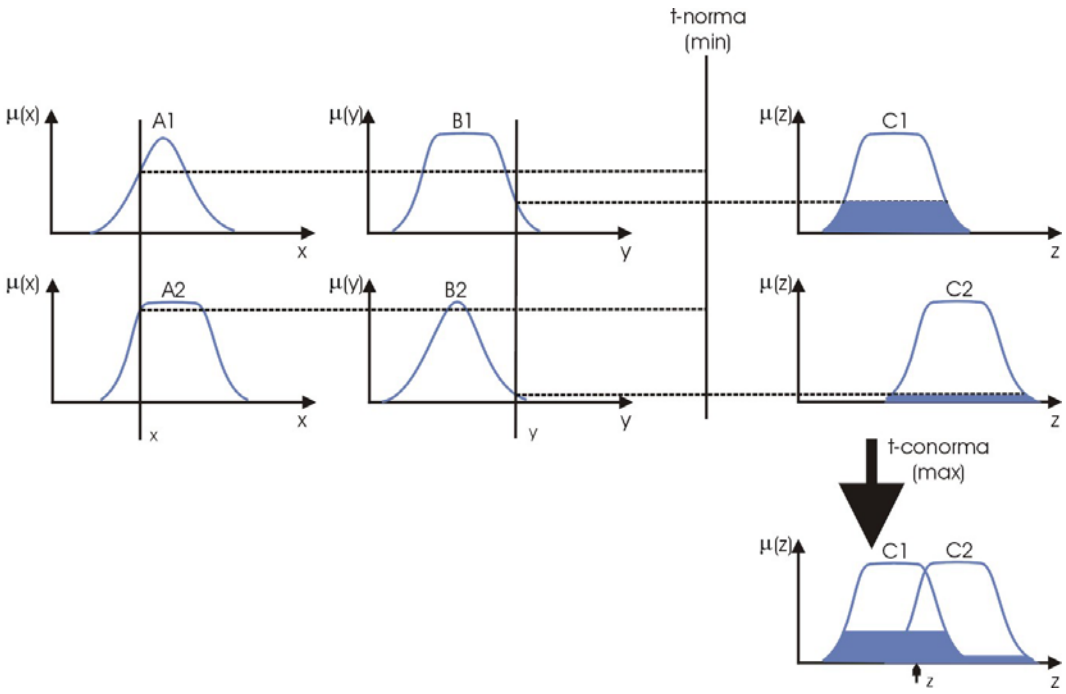

Figura 2.4: Modelo Mamdani com duas entradas x e y.

Para o modelo de Mamdani, faz-se necessário o uso de um método de *defuzzyficação*. Alguns dos principais métodos de *defuzzyficação* estão indicadas na figura 2.5.

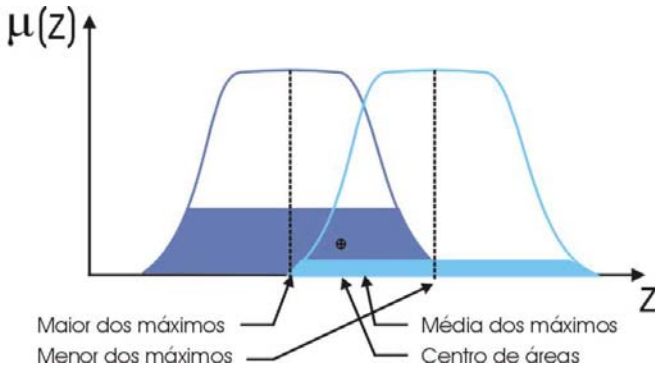

Figura 2.5: Alguns métodos de *defuzzyficação*.

2.1.2 Modelos por Interpolação

Nos modelos por interpolação a conclusão de cada regra é dada através de um polinômio ou de uma função monotônica. Uma função é dita monotônica se puder

ser classificada como crescente, estritamente crescente, decrescente ou estritamente decrescente. No modelo de Sugeno, a função é uma combinação linear das entradas, tendo como parâmetros um conjunto de constantes. Já no esquema de Tsukamoto, a função é geralmente não linear, tendo como domínio os possíveis graus de compatibilidade entre cada premissa e as entradas.

Em ambos os modelos, obtém-se, para cada regra, um único valor de controle. A ação de controle global é obtida fazendo-se uma média ponderada dos valores individuais obtidos, em que cada peso é o resultado obtido após a aplicação da t-norma ($\top$). A saída para os modelos por interpolação consiste em um valor que pode ser aplicado diretamente no sistema controlado, o que dispensa o tempo de processamento da fase de *defuzzyficação*.

Modelo de Sugeno

O modelo de Sugeno, também conhecido como modelo TSK, foi proposto por Takagi, Sugeno e Kang (ver figura 2.6) com o objetivo de desenvolver uma sistemática para geração de regras *fuzzy* a partir de um conjunto de dados de entrada e saída. A forma de uma típica regra *fuzzy* neste modelo é a seguinte:

SE x é A_1 com μ_{x,A_1} E y é B_1 com μ_{y,B_1} ENTÃO $C_1 = f_1(x, y)$
SE x é A_2 com μ_{x,A_2} E y é B_2 com μ_{y,B_2} ENTÃO $C_2 = f_2(x, y)$;

Em que $f_n(x, y)$ é um polinômio em função das variáveis x e y.

$$z = \frac{\top(\mu_{x,A_1}, \mu_{y,B_1}) \cdot C_1 + \top(\mu_{x,A_2}, \mu_{y,B_2}) \cdot C_2}{\top(\mu_{x,A_1}, \mu_{y,B_1}) + \top(\mu_{x,A_2}, \mu_{y,B_2})}; \tag{2.2}$$

Modelo de Tsukamoto

Neste modelo, os conseqüentes de cada regra *fuzzy* devem ser funções monotônicas (figura 2.7), pois elas operarão na forma inversa para a obtenção da saída do sistema.

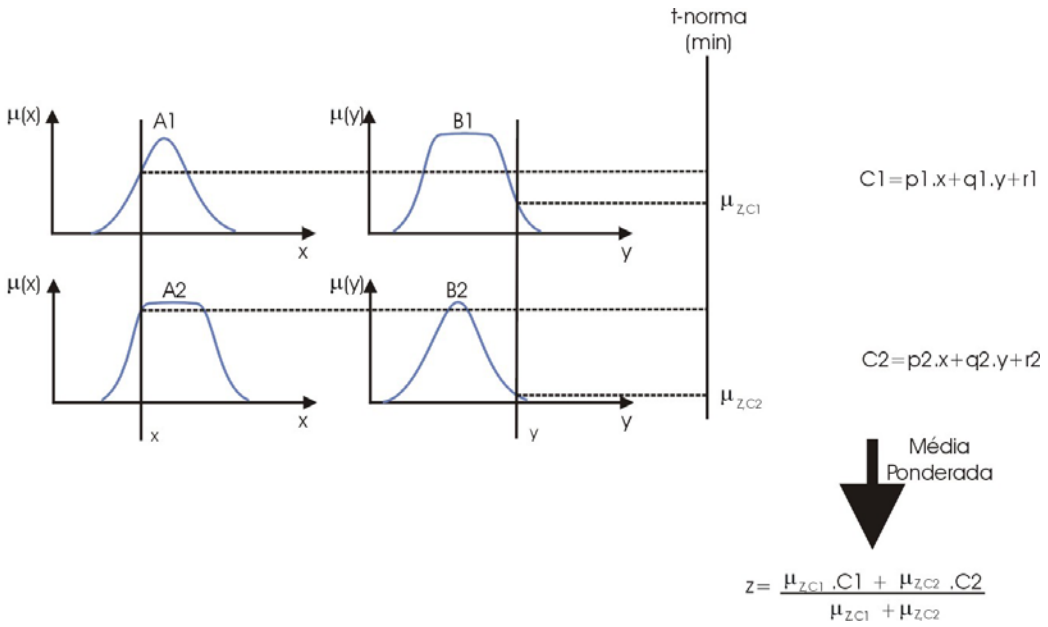

Figura 2.6: Modelo Sugeno de inferência.

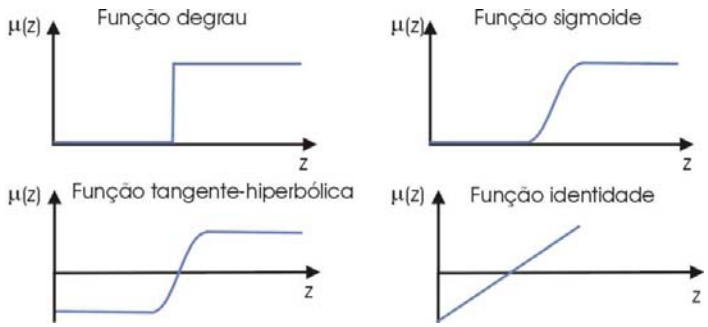

Figura 2.7: Funções de pertinência monotônicas.

SE x é A_1 com μ_{x,A_1} E y é B_1 com μ_{y,B_1} ENTÃO z é C_1 com $\mu_{z,C_1} = \top(\mu_{x,A_1}, \mu_{y,B_1})$
SE x é A_2 com μ_{x,A_2} E y é B_2 com μ_{y,B_2} ENTÃO z é C_2 com $\mu_{z,C_2} = \top(\mu_{x,A_2}, \mu_{y,B_2})$;

Pode-se obter uma saída *fuzzy* pela fórmula,

$$z = \frac{\mu_{z,C_1} \cdot f^{-1}(\mu_{z,C_1}) + \mu_{z,C_2} \cdot f^{-1}(\mu_{z,C_2})}{\mu_{z,C_1} + \mu_{z,C_2}}; \tag{2.3}$$

O esquema deste modelo pode ser visualizado na figura 2.8.

Pode-se, então, afirmar que a variação dos parâmetros das funções de pertinência

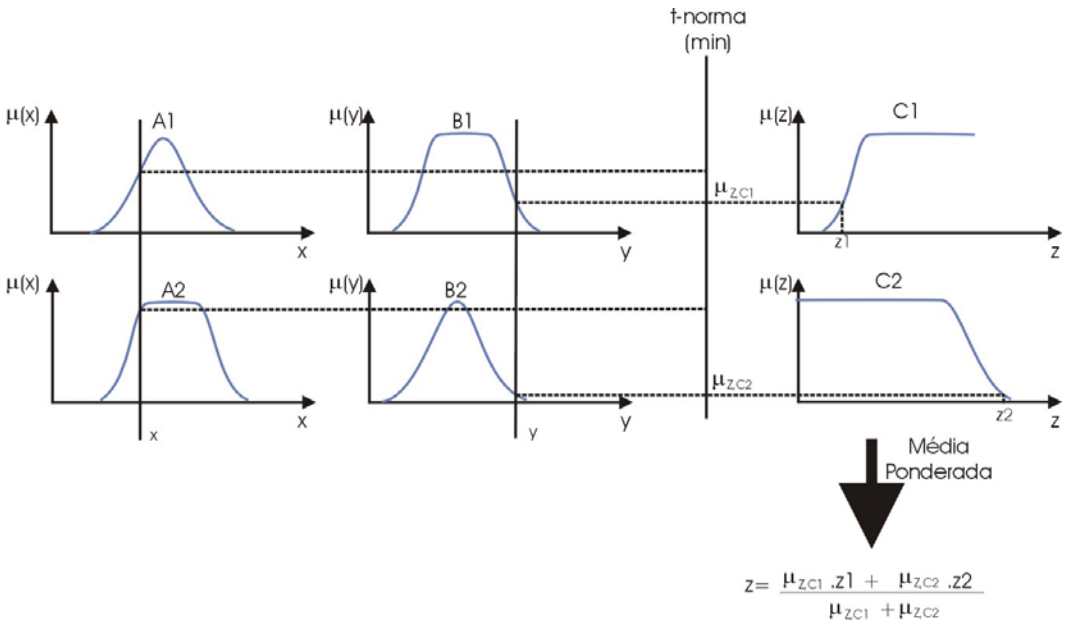

Figura 2.8: Modelo Tsukamoto de inferência.

de entrada e de saída, bem como a variação da base de regras, são os principais componentes a serem alterados nos controladores *fuzzy* para variar a relação entre as entradas e saídas destes sistemas.

2.2 Redes Neurais

As Redes Neurais Artificiais (RNA's) baseiam-se na forma de codificar o conhecimento do cérebro humano e são formadas por pequenas estruturas chamadas de neurônios. Estes neurônios, no cérebro humano, são conectados uns aos outros através de ligações axônios − dendritos com o fluxo de dados em sentido único (figura 2.9) formando uma grande rede conhecida como rede neural. No cérebro humano, a lembrança de algum fato é associada a um caminho que um impulso elétrico faz ao atravessar um certo número de neurônios. As mudanças do pulso que entrou e vai gerar uma saída, ocorre neste caminho que se define pela força das ligações entre os neurônios, também conhecida como sinapse, e por funções que podem ser associadas ao corpo celular de cada neurônio (A. Guyton & J. Hall, 1997).

Nas RNA's, as unidades de neurônios artificiais são geralmente agrupadas em ca-

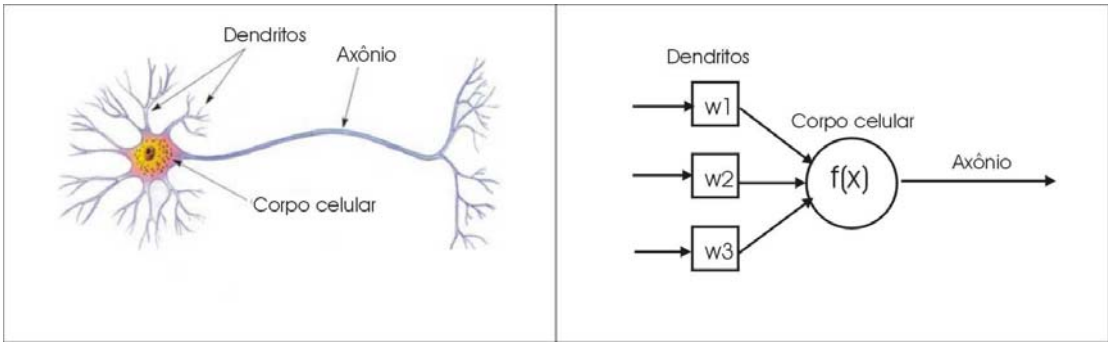

Figura 2.9: Neurônio real e a representação do neurônio artificial.

madas, rotuladas como: camada de entrada, camada(s) intermediaria(s) e camada de saída, dependendo de sua funcionalidade (figura 2.10). RNA's podem ser classificadas, com relação aos tipos de conexões entre seus neurônios, por: *feedforward* ou recorrentes. Nas RNA's *feedforward*, os dados seguem um sentido único e não são transmitidos de uma camada posterior para uma anterior. Porém, nas recorrentes, essa forma de transmissão é permitida, o que forma ciclos em sua representação gráfica.

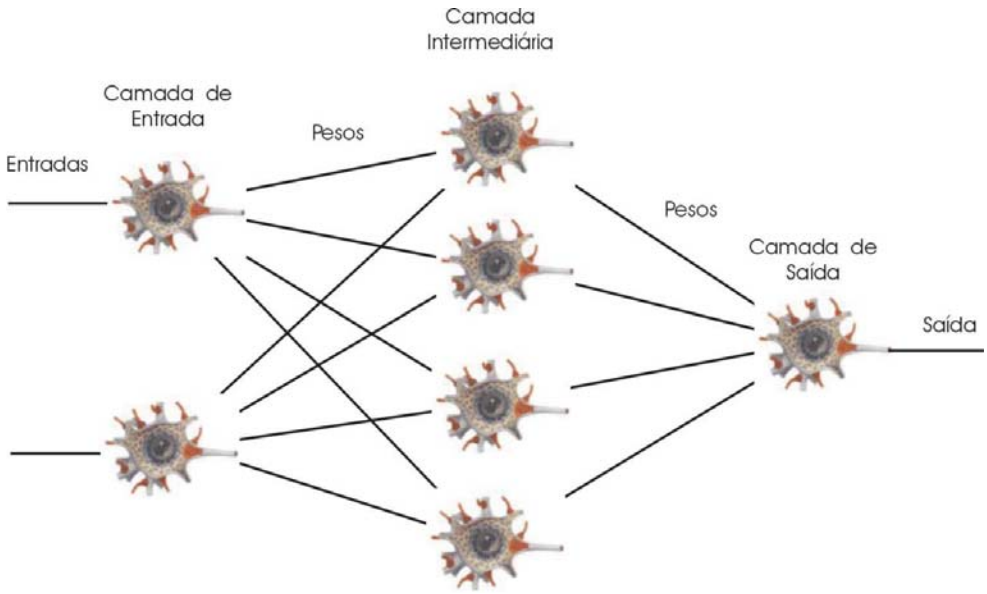

Figura 2.10: Distibruição das camadas de uma RNA.

Uma das mais importantes características das Redes Neurais Artificiais é sua capacidade de aprendizado. Os algoritmos responsáveis por modificar os pesos sinapti-

cos e os parâmetros adaptativos das funções representativas do corpo dos neurônios, nos quais vão se abstrair o aprendizado, são conhecidos como algoritmos de aprendizagem. Entre os algoritmos existentes o mais comum é o de Retropropagação do Erro (*Back-propagation*). Nesse algoritmo, existe a figura do "professor" (ou supervisor) que informa qual deve ser a saída para uma determinada entrada. Assim, encontra-se o erro entre a saída desejada e a saída da rede, e ajustam-se os pesos sinápticos e os parâmetros das funções para que esse erro seja o menor possível.

A meta de um algoritmo de aprendizagem é diminuir o erro da saída da RNA o máximo possível. Porém, com relação aos padrões fornecidos, o sucesso da aprendizagem não é garantido. Existe, por exemplo, a possibilidade de haver mínimos locais (figura 2.11). Um mínimo local pode ser interpretado como uma região onde, com o passo de aprendizagem existente, se caminharmos para a esquerda ou para direita, o erro encontrado é maior do que o erro existente. Dessa forma, o algoritmo fica "preso" nesta região e não evolui para o mínimo global.

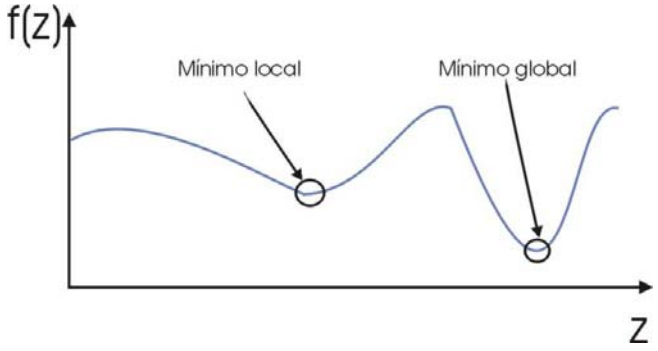

Figura 2.11: Representação simplificada de mínimo local.

2.2.1 *Back-propagation*

Considere um neurônio da k-ésima camada, com os índices como ilustra a figura 2.12. A saída $X_{k,i}$ do neurônio i desta camada é dada pela seguinte equação:

$$X_{k,i} = f_{k,i}(X_{k-1,1} \dots X_{k-1,n}, \alpha, \beta, \gamma, \dots) \tag{2.4}$$

em que $\alpha, \beta, \gamma, etc.$ São os parâmetros adaptativos do neurônio.

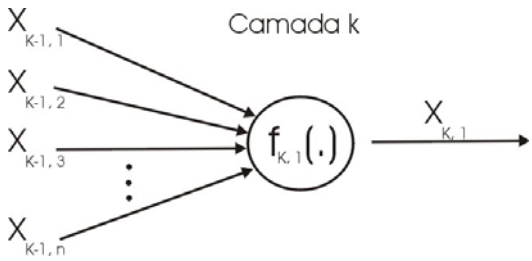

Figura 2.12: Índices associados a um neurônio da k-ésima camada.

O algoritmo *Back-propagation* se baseia na minimização, para cada ponto de treinamento l, do seguinte erro pontual:

$$E_p^l = \sum_{i=1}^{N^{out}} (d_s^l - X_{K,i})^2 \tag{2.5}$$

Suponha uma rede neural com K camadas e onde uma camada k tem $N(k)$ neurônios. Para facilitar a explicação, definiu-se o **sinal de erro**, $\delta_{k,i}$, como a derivada do erro medido, E_p, em relação a saída do neurônio i na camada k.

$$\delta_{k,i} = \frac{\partial^+ E_p}{\partial X_{k,i}} \tag{2.6}$$

Para os neurônios da camada de saída (camada K) pode-se encontrar o valor do sinal de erro diretamente por:

$$\delta_{K,i} = \frac{\partial^+ E_p}{\partial X_{K,i}} = \frac{\partial E_p}{\partial X_{K,i}} = -2(d_i - X_{K,i}) \tag{2.7}$$

Já para os neurônios das outras camadas é necessário utilizar a fórmula iterativa a seguir:

$$\delta_{k,i} = \frac{\partial^+ E_p}{\partial X_{k,i}} = \sum_{m=1}^{N(k+1)} \frac{\partial^+ E_p}{\partial X_{k+1,m}} \cdot \frac{\partial f_{k+1,m}}{\partial X_{k,i}} = \sum_{m=1}^{N(k+1)} \delta_{k+1,m} \cdot \frac{\partial f_{k+1,m}}{\partial X_{k,i}} \tag{2.8}$$

em que $0 \leq k \leq K-1$ e $N(k+1)$ é o número de neurônios da camada $k+1$.

Observe que o cálculo do sinal do erro da camada k depende do resultado do sinal de erro da camada $k+1$. Com isso, pode-se concluir que é necessário calcular

primeiro o sinal de erro da última camada e retro-propagar este sinal camada por camada, até a entrada da rede.

A variação para cada parâmetro α pode ser definida como a derivada do erro medido em relação a cada parâmetro, multiplicados pelo coeficiente de aprendizagem η.

$$\Delta\alpha_p = -\eta \cdot \frac{\partial^+ E_p}{\partial X_{k,i}} \cdot \frac{\partial f_{k,i}}{\partial \alpha} = -\eta \cdot \delta_{k,i} \cdot \frac{\partial f_{k,i}}{\partial \alpha} \tag{2.9}$$

A atualização dos parâmetros pode ocorrer ao final do processamento de todos os pontos de treinamento. Para isto, é necessário que todas as atualizações pontuais sejam somadas. Pode-se reescrever a atualização como:

$$\Delta\alpha_t = \sum_{l=1}^{L} \Delta\alpha_p^l = -\eta \cdot \sum_{l=1}^{L} \delta_{k,i}^l \cdot \frac{\partial f_{k,i}^l}{\partial \alpha} \tag{2.10}$$

em que L representa os pontos de treinamento.

Dessa forma, pode-se ajustar α utilizando:

$$\alpha(t+1) = \alpha(t) + \Delta\alpha_t(t) \tag{2.11}$$

O modelo ANFIS, apresentado no próximo capítulo, considera todos os pesos sinapticos iguais a 1 (um), de forma que as ligações entre os neurônios informam, apenas, a existência de uma conexão e o sentido do fluxo. As adaptações, promovidas pelo algoritmo *Back-propagation*, visam ajustar os parâmetros variáveis das funções associadas ao corpo celular dos neurônios nas camadas 1 e 4 deste modelo.

Capítulo 3

Controladores Inteligentes Híbridos

3.1 ANFIS

O modelo ANFIS utiliza como estrutura básica um controlador *fuzzy*, o qual pode ser interpretado como uma rede neural de cinco camadas, em que é possível aplicar técnicas de aprendizado como o *Back-propagation*, por exemplo (J. R. Jang *et alii*, 1997).

Para simplificar, considere um sistema *fuzzy* com duas entradas x e y, uma saída z e duas funções de pertinência para cada variável de entrada, resultando em quatro regras. Visualizando a figura 3.1 pode-se seguir o fluxo de dados e associar funções, que são inerentes aos neurônios, às operações referentes ao modelo *fuzzy* proposto por Sugeno. Observe que os pesos das sinapses foram definidos como 1 (um) e as variáveis ajustáveis estão distribuídas nas funções dos neurônios. Os neurônios representados por quadrados apresentam as variáveis ajustáveis para o sistema.

Definindo $X_{k,i}$ como sendo a saída do neurônio i da camada k. O fluxo dos dados pode ser analisado camada a camada, como mostrado a seguir:

- **Camada 1**: Os neurônios desta camada representam as funções de pertinência de entrada, ou seja, a fase de *fuzzyficação*. Dessa forma, os dados de saída da

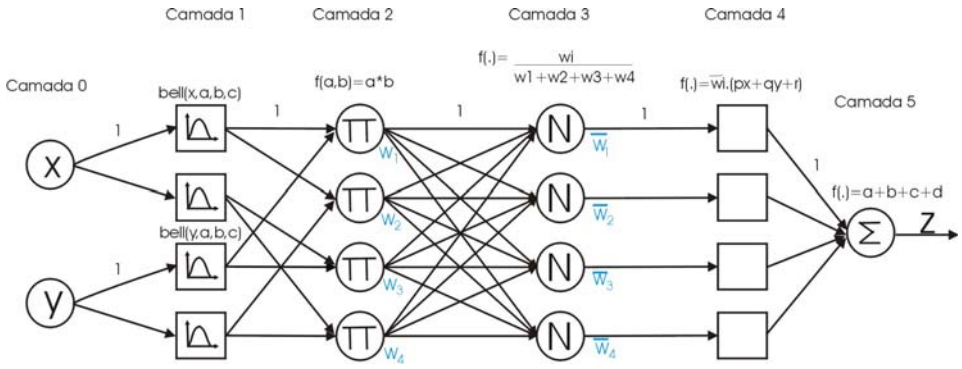

Figura 3.1: Representação do ANFIS com duas entradas x e y, e uma saída z.

camada serão as chamadas "entradas *fuzzy*".

$$\begin{aligned} X_{1,i} &= \mu_{A_i}(x) \quad para \quad i = 1, 2;\ ou \\ X_{1,i} &= \mu_{B_{i-2}}(y) \quad para \quad i = 3, 4. \end{aligned} \tag{3.1}$$

- **Camada 2**: A função para esta camada é uma t-norma (tabela 2.1). Considerando a forma probabilística, a saída $X_{2,i}$ pode ser calculada multiplicando o valor das suas entradas.

$$\begin{aligned} X_{2,1} &= w_1 = \mu_{A_1}(x) \cdot \mu_{B_1}(y) \\ X_{2,2} &= w_2 = \mu_{A_1}(x) \cdot \mu_{B_2}(y) \\ X_{2,3} &= w_3 = \mu_{A_2}(x) \cdot \mu_{B_1}(y) \\ X_{2,4} &= w_4 = \mu_{A_2}(x) \cdot \mu_{B_2}(y) \end{aligned} \tag{3.2}$$

  Note que nesta camada estão abstraídas as regras. Assim, a omissão de um dos neurônios indica a omissão de uma das regras.

- **Camada 3**: A saída desta camada será a saída dos neurônios da camada anterior, normalizadas, ou seja, a saída de cada neurônio da camada anterior dividida pela soma da saída de todos os neurônios dessa mesma camada.

$$X_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2 + w_3 + w_4}, i = 1, 2, 3, 4 \tag{3.3}$$

- **Camada 4**: A função associada aos neurônios desta camada será o polinômio $f(x, y)$, utilizado pelo modelo Sugeno, em que x e y são as entradas do sistema e p, q e r são os parâmetros ajustáveis do polinômio.

$$X_{4,i} = \overline{w}_i \cdot f_i = \overline{w}_i(p_i x + q_i y + r_i) \tag{3.4}$$

- **Camada 5**: Nesta camada ocorre o somatório das saídas dos neurônios das camadas anteriores e, desta forma, obtêm-se o sinal de controle para o sistema.

$$X_5 = \sum_i \overline{w_i} \cdot f_i = \frac{\sum_i w_i \cdot f_i}{\sum_i w_i} \tag{3.5}$$

Como já mencionado, a saída do modelo Sugeno não necessita de uma interface de *defuzzyficação*.

A partir da rede neural apresentada, pode-se identificar claramente que os neurônios que necessitam de aprendizado estão presentes nas camadas 1 (um) e 4 (quatro), pois na camada 1 estão localizadas as funções de pertinência de entrada e na camada 4, os polinômios Sugeno, que definem as implicações das regras.

O ajuste dos parâmetros de um controlador no modelo ANFIS pode ser obtido através de técnicas adaptativas como o algoritmo *Back-propagation*. Fundamentado pelas equações apresentadas no capítulo 2, é possível encontrar equações já derivadas para este modelo em especial.

3.1.1 *Back-propagation* no Modelo ANFIS

Partindo de pares de treinamento entrada-saída desejados, o erro nesta rede neural pode ser propagado para as camadas anteriores a partir da última camada até a primeira.

Para cada neurônio, de cada camada, será calculado o $\delta_{k,i}$ desta camada.

Para a última camada ($k = 5$), utiliza-se a fórmula 2.7 e obtém-se:

$$\delta_{5,1} = -2(D_l - X_{5,1}) \tag{3.6}$$

Para as camadas anteriores utiliza-se a formula 2.8, o que resulta em:

- Camada 4:

$$\delta_{4,1} = \delta_{4,2} = \delta_{4,3} = \delta_{4,4} = \delta_{5,1} \tag{3.7}$$

- Camada 3:

$$\begin{aligned}
\delta_{3,1} &= \delta_{4,1} \cdot (p_1 x + q_1 y + r_1) \\
\delta_{3,2} &= \delta_{4,2} \cdot (p_2 x + q_2 y + r_2) \\
\delta_{3,3} &= \delta_{4,3} \cdot (p_3 x + q_3 y + r_3) \\
\delta_{3,4} &= \delta_{4,4} \cdot (p_4 x + q_4 y + r_4)
\end{aligned} \tag{3.8}$$

- Camada 2:

$$\begin{aligned}
\delta_{2,1} &= \delta_{3,1} \cdot \tfrac{w_2+w_3+w_4}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,2} \cdot \tfrac{-w_2}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,3} \cdot \tfrac{-w_3}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,4} \cdot \tfrac{-w_4}{(\sum_{i=1}^{4} w_i)^2} \\
\delta_{2,2} &= \delta_{3,2} \cdot \tfrac{w_1+w_3+w_4}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,1} \cdot \tfrac{-w_1}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,3} \cdot \tfrac{-w_3}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,4} \cdot \tfrac{-w_4}{(\sum_{i=1}^{4} w_i)^2} \\
\delta_{2,3} &= \delta_{3,3} \cdot \tfrac{w_1+w_2+w_4}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,1} \cdot \tfrac{-w_1}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,2} \cdot \tfrac{-w_2}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,4} \cdot \tfrac{-w_4}{(\sum_{i=1}^{4} w_i)^2} \\
\delta_{2,4} &= \delta_{3,4} \cdot \tfrac{w_1+w_2+w_3}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,1} \cdot \tfrac{-w_1}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,2} \cdot \tfrac{-w_2}{(\sum_{i=1}^{4} w_i)^2} + \delta_{3,3} \cdot \tfrac{-w_3}{(\sum_{i=1}^{4} w_i)^2}
\end{aligned} \tag{3.9}$$

- Camada 1:

$$\begin{aligned}
\delta_{1,1} &= \delta_{2,1} \cdot \mu_{y,B_1} + \delta_{2,2} \cdot \mu_{y,B_2} \\
\delta_{1,2} &= \delta_{2,3} \cdot \mu_{y,B_1} + \delta_{2,4} \cdot \mu_{y,B_2} \\
\delta_{1,3} &= \delta_{2,1} \cdot \mu_{x,A_1} + \delta_{2,3} \cdot \mu_{x,A_2} \\
\delta_{1,4} &= \delta_{2,2} \cdot \mu_{x,A_1} + \delta_{2,4} \cdot \mu_{x,A_2}
\end{aligned} \tag{3.10}$$

Para o ajuste dos parâmetros da camada 4, é necessário derivar parcialmente o polinômio em relação às variáveis p, q e r (A. Valishvsky, 2002).

$$\frac{\partial \overline{w}(px + qy + r)}{\partial p} = \overline{w}x \tag{3.11}$$

$$\frac{\partial \overline{w}(px + qy + r)}{\partial q} = \overline{w}y \tag{3.12}$$

$$\frac{\partial \overline{w}(px+qy+r)}{\partial r} = \overline{w} \tag{3.13}$$

Da mesma forma, para o ajuste dos parâmetros da camada 1, é necessário derivar parcialmente a função de pertinência em relação as variáveis a, b e c. Considerando uma função de pertinência com a forma de sino que pode ser expressa por:

$$y = bell(x, a, b, c) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}} \tag{3.14}$$

As derivadas parciais desta função resultam em (J. R. Jang *et alii*, 1997):

$$\frac{\partial y}{\partial a} = \frac{2b}{a} \cdot y \cdot (1-y) \tag{3.15}$$

$$\frac{\partial y}{\partial b} = \begin{cases} -2 \cdot \ln|\frac{x-c}{a}| \cdot y \cdot (1-y) & se \quad x \neq c \\ 0 & se \quad x = c \end{cases} \tag{3.16}$$

$$\frac{\partial y}{\partial c} = \begin{cases} \frac{2b}{x-c} \cdot y \cdot (1-y) & se \quad x \neq c \\ 0 & se \quad x = c \end{cases} \tag{3.17}$$

A variação dos parâmetros será de:

$$\Delta\alpha_{k,j} = -\eta \cdot \sum_{l=1}^{L} \delta_{k,j}^{l} \cdot \frac{\partial f_{k,l}^{l}}{\partial \alpha} \tag{3.18}$$

em que α representa as variáveis p, q, r, a, b e c.

Optou-se por utilizar uma técnica para aumentar a velocidade de convergência das redes neurais chamada de η-adaptativo (ou η-variável) pelo fato dessa técnica ter sido utilizada com sucesso em alguns trabalhos com implementações práticas (O. Filho, 2004).

3.1.2 η-Adaptativo

Esta técnica consiste em variar a constante de aprendizado η (eta) durante a etapa de treinamento da rede, buscando otimizar o processo de adaptação dos

Tabela 3.1: Código do algoritmo η-adaptativo.

```
01- Iniciar eta com valor pequeno (entre 0,000001 e 0,01, por exemplo);
02- Iniciar erro_aux1 = 0;
03- erro_aux2 = E (E = Erro quadrático);
04- Se erro_aux2 > 1.04*erro_aux1 Então
05-     eta = 0.7*eta;
06- Senão
07-     Se erro_aux2 <= erro_aux1 Então
08-         eta = 1.05*eta;
09-     Fim Se;
10- Fim Senão;
11- erro_aux1 = erro_aux2;
12- Voltar para linha 3.
```

parâmetros, como também evitar a parada em um mínimo local.

A rotina da tabela 3.1 mostra como essa técnica funciona.

Os fatores $1,04$, $1,05$ e $0,7$ usados no algoritmo do η-adaptativo são sugeridos por (J. Rezende & A. Maitelli, 1999a; J. Rezende & A. Maitelli, 1999b).

3.2 NEFCON

O NEFCON é um modelo de controlador *neuro-fuzzy* baseado na arquitetura genérica de uma RNA, mais especificamente de uma rede perceptron de três camadas (figura 3.2). Fazendo-se um paralelo entre as redes perceptron de três camadas, e os sistemas *fuzzy*, têm-se que (A. Nürnberger *et alii*, 1999):

- **Camada 0**: a camada de entrada recebe os valores de entrada e está conectada à segunda camada através de sinapses que contém pesos. Estes pesos estão relacionados com os parâmetros das funções de pertinência das entradas (*fuzzyficação*), gerando assim uma entrada para a próxima camada, correspondente às variáveis *fuzzyficadas*.

- **Camada 1**: Nesta camada (camada intermediária) ocorre o procedimento de inferência em que, cada neurônio representa uma regra *fuzzy* e a função de

ativação é uma t-norma.

- **Camada 2**: A camada 2 é responsável pela t-conorma e encerra o processo aplicando uma técnica de *defuzzyficação*. O algoritmo proposto por A. Nürnberger *et alii* (1999), sugere o uso do método média dos máximos para a etapa de *defuzzyficação*, o que reduz as funções de pertinência de saída a simples impulsos localizados nos máximos das mesmas.

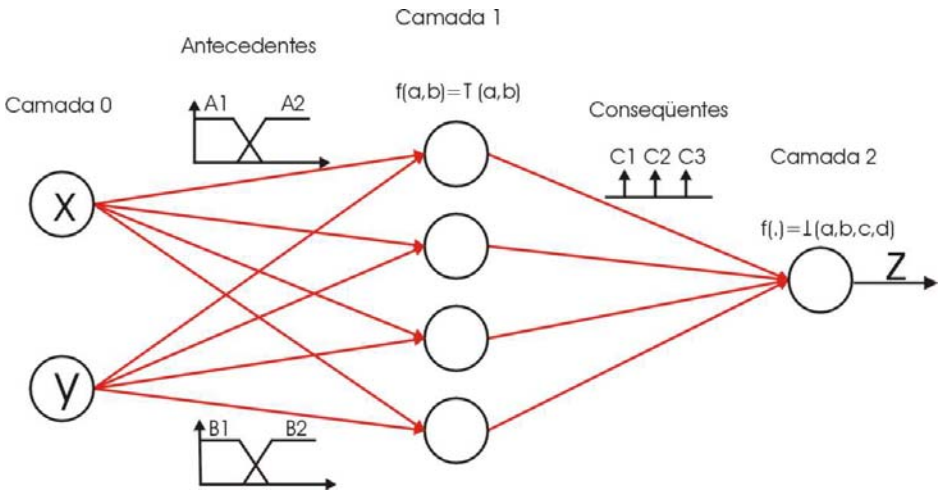

Figura 3.2: Representação do NEFCON com duas entradas x e y, e uma saída z.

O algoritmo de aprendizagem do NEFCON é baseado na idéia de Aprendizagem por Reforço (R. Sutton & A. Barto, 1998). Tanto o erro *fuzzy*, que é dado pela distância entre o valor de uma entrada do controlador *fuzzy* e o seu valor desejado, quanto o sinal da saída ótima para o controlador (η_{otm}) podem fazer o papel de crítico e, nesse caso, devem ser conhecidos.

Para alguns sistemas, o sinal ótimo de controle pode ser calculado simplesmente usando a diferença entre a referência do sistema e o estado atual da planta a ser controlada. Em casos mais complicados é possível utilizar regras simples para atingir resultados satisfatórios.

Considerando um sistema *Ball and Beam* composto por uma trave de referência com uma bola de referência e uma trave controlada por um servo-motor que movimenta outra bola. O objetivo é fazer com que a bola sobre a trave controlada

fique na mesma posição da bola da trave de referência (o sistema *Ball and Beam* será mostrado com detalhes no capítulo 4), pode-se expressar as regras da seguinte maneira:

- η_{otm} é zero quando a velocidade da bola controlada é zero e o erro ($valordesejado - valorobtido$) também é zero ou o erro é diferente de zero mas o valor da velocidade da bola indica um movimento em direção à referência.
- η_{otm} é positivo quando o erro é positivo e a bola se afasta da referência.
- η_{otm} é negativo quando o erro é negativo e a bola se afasta da referência.

O algoritmo que corresponde à técnica NEFCON foi dividido em duas partes principais, uma para a criação e otimização das regras e outra para otimização das funções de pertinência para as regras existentes (M. Rodrigues *et alii*, 2004).

Na primeira parte, o algoritmo recebe uma base de regras vazia e a partir das entradas, verifica qual a melhor saída criando uma regra equivalente. Se uma regra já existir, o algoritmo verifica se essa é a melhor opção, podendo modificá-la se achar necessário.

Na segunda parte, atua-se nas funções de pertinência, de modo que o aprendizado ocorra de forma similar a alguns algoritmos de aprendizagem por reforço. Para cada estado (cada função de pertinência ativa no momento), se há contribuição para a redução do erro, então este estado terá sua probabilidade de ocorrência aumentada, ou seja, a função de pertinência passará a englobar uma área maior e, dessa forma, a chance de ser ativada novamente é maior que no instante anterior. Caso a função de pertinência contribua para o aumento do erro, esta função terá sua probabilidade reduzida, ou seja, a função de pertinência engloba um espaço menor, sendo mais difícil desta função de pertinência se tornar ativa na iteração posterior (A. Nürnberger *et alii*, 1999).

Observe que na segunda parte do algoritmo o ajuste deve ocorrer sempre que existir erro, já que este foi considerado como parâmetro para a saída ótima. Também é possível utilizar as regras da saída ótima para que o ajuste ocorra somente quando a

saída ótima for positiva ou negativa. Porém, esta alteração não se tornou necessária para o sistema *ball and beam*.

Uma base de regras não vazia pode ser introduzida no sistema para que seja otimizada pelo algoritmo. Da mesma forma, o algoritmo pode ser usado para melhorar funções de pertinência previamente estabelecidas que representem os conhecimentos já assimilados sobre o controle.

Algumas mudanças com relação à proposta original do modelo NEFCON (A. Nürnberger *et alii*, 1999) foram introduzidas com o objetivo de satisfazer necessidades específicas deste trabalho. A seguir, na medida em que detalhes da implementação são apresentados, são destacadas as principais alterações.

3.2.1 Geração e Otimização de Regras

Como condição inicial para esta fase é fornecido ao controlador tanto um intervalo de atuação para cada uma das variáveis de entrada quanto um intervalo para a saída. Estes intervalos de atuação funcionam como referência para possibilitar a comparação entre variáveis de grandezas diferentes. O algoritmo original sugere apenas o fornecimento do intervalo de atuação de saída e, desta forma, cria uma regra que atua no sentido oposto ao crescimento do erro *fuzzy*. A técnica empregada, porém, visa não apenas reduzir o erro do sistema controlado, mas optar entre as entradas do sistema e reduzir o erro *fuzzy* daquela que estiver com maior discrepância em relação ao valor desejado.

As variáveis de entrada do controlador são encontradas e a regra ativada com maior intensidade (maior μ) é criada, ou seja, recebe um valor correspondente a uma das funções de pertinência de saída. Uma regra para o modelo NEFCON é dita ativa quando recebe um valor de pertinência diferente de zero ($0 < \mu <= 1$) após a aplicação do operador t-norma.

O valor assumido para a regra criada tenderá a diminuir o valor da entrada que estiver com o maior erro. O erro entre as variáveis de entrada poderá ser encontrado e comparado se os intervalos de atuação de cada variável de entrada

forem normalizados (figura 3.3).

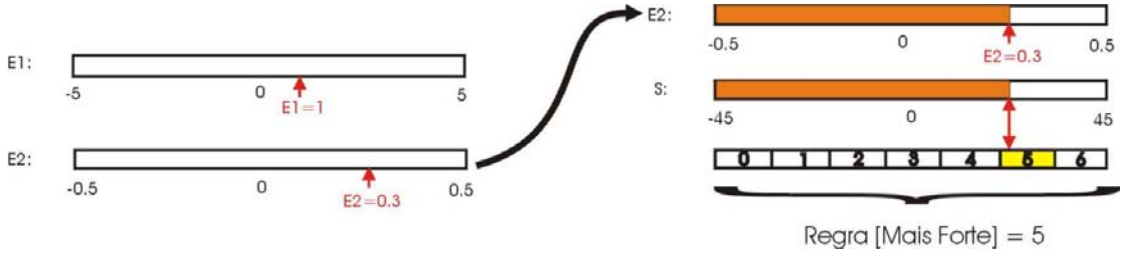

Figura 3.3: Modelo de otimização e criação de regras pelo modelo NEFCON adaptado.

A intensidade da nova regra é calculada através da conversão do valor de erro do intervalo de entrada com maior erro para o intervalo de saída. Dessa forma, a função de pertinência de saída que possuir valor mais adequado será utilizada.

Assim, pode-se escrever:

1. Considerando um sistema com as entradas $(x_1, \ldots, x_n)$ e $([L_1^{min}, L_1^{max}], \ldots, [L_n^{min}, L_n^{max}])$ definido pelo usuário como sendo o valor mínimo e o máximo para cada x_i, ou seja, o intervalo de atuação de cada entrada x_i.

2. Encontre o vetor $(\mu_1^{max}, \ldots, \mu_n^{max})$ em que cada μ_i^{max} é o maior valor de μ no conjunto *fuzzy* x_i. Esse vetor corresponde à regra que atuará com maior intensidade no sistema.

3. Para cada entrada x_i subtraia o valor desejado para esta entrada e divida o módulo do resultado pelo respectivo "intervalo de atuação" $(L_i^{max} - L_i^{min})$ da entrada referente, gerando um vetor $(x_1^{norm}, \ldots, x_n^{norm})$.

$$x_i^{norm} = \frac{|x_i - x_i^{desejado}|}{L_i^{max} - L_i^{min}} \tag{3.19}$$

   O maior valor deste vetor corresponde ao maior erro em relação ao valor desejado. Esta entrada será chamada de x_r e o seu intervalo $[L_r^{min}, L_r^{max}]$

4. A regra a ser criada será dada por:

$$regra(\mu_1^{max}, \ldots, \mu_n^{max}) = \frac{x_r - L_r^{min}}{L_r^{max} - L_r^{min}} \cdot k \tag{3.20}$$

Em que, k é o número de funções de pertinência de saída. Assim,

$$\begin{cases} regra = 0 & \rightarrow & \text{primeira função de pertinência} \\ regra = 1 & \rightarrow & \text{segunda função de pertinência} \\ \vdots & & \vdots \\ regra = n-1 & \rightarrow & \text{ultima função de pertinência} \end{cases} \tag{3.21}$$

É importante observar que o valor de *regra* tem que ser um número inteiro e, para isso, é necessário desconsiderar as casas decimais. Outro ponto importante é que as funções de pertinência de saída, no início da execução, estejam espalhadas de forma simétrica dentro de seu intervalo.

Diferente do sugerido pelo algoritmo base (A. Nürnberger *et alii*, 1999), uma regra criada anteriormente pode ser modificada a qualquer instante se uma regra diferente for encontrada para a ocasião. Também, não há necessidade de modificação na estrutura das funções de pertinência, já que estas irão ser tratadas em uma fase posterior.

3.2.2 Otimização de Funções de Pertinência

Após as principais regras serem criadas, determina-se o sinal de controle, aplicando-o na planta para obter um novo estado. O erro da planta E é encontrado de acordo com esse novo estado. Então, estima-se a contribuição, tr, de cada regra para a saída e calcula-se o erro Er para cada unidade de regra, de acordo com a seguinte equação:

$$Er = \mu \cdot E \cdot sgn(tr) \tag{3.22}$$

em que:

$sgn(tr)$ = Sinal de tr

De posse desses dados, e baseado na figura 3.4, pode-se representar as modificações nos conseqüentes por:

$$\Delta b_i = \eta \cdot \mu \cdot E \tag{3.23}$$

e as modificações para os antecedentes por:

$$\Delta a_j^{(i)} = -\eta \cdot Er \cdot (b_j^{(i)} - a_j^{(i)}) \tag{3.24}$$

$$\Delta c_j^{(i)} = \eta \cdot Er \cdot (c_j^{(i)} - b_j^{(i)}) \tag{3.25}$$

em que:

η = Coeficiente de aprendizagem

a, b, c = Vértices das funções de pertinência

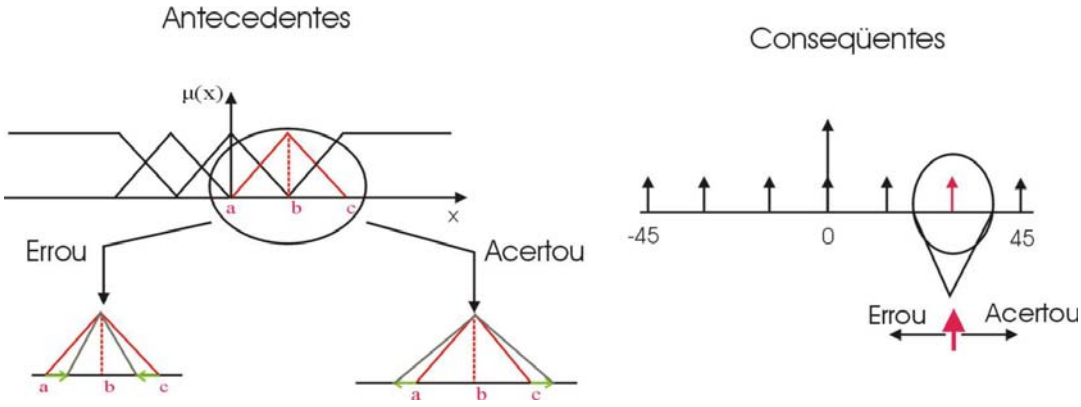

Figura 3.4: Otimização das funções de pertinência pelo modelo NEFCON adaptado.

É fácil observar que o algoritmo não altera a posição da função de pertinência dos antecedentes. O que é feito é apenas aumentar ou diminuir sua base de forma proporcional para ambos os lados, ou seja, caso a função de pertinência tenha uma contribuição positiva, esta terá maior probabilidade de ocorrer novamente.

Algumas restrições foram inseridas para evitar sobreposição de mais de duas funções de pertinência e para evitar o surgimento de lacunas entre estas. Para isso, deve-se garantir uma sobreposição de zero a 50%, ou seja, os vértices de um triân-

gulo devem estar contidos no intervalo correspondente a meia base dos triângulos vizinhos.

Capítulo 4

Resultados Experimentais

Os algoritmos de controle foram implementados em linguagem C/C++ e executados a partir de um microcomputador do tipo PC dotado de uma placa MultiQ-3 da Quanser Consulting para aquisição de dados e controle, sendo responsável pela comunicação entre o PC e a planta experimental *ball and beam*, também da Quanser Consulting.

4.1 O Sistema *Ball and Beam*

O sistema *ball and beam*, no qual os controladores foram testados, é composto basicamente por um sistema trave-bola, um servo-motor com uma caixa redutora e uma régua com bola de referência. Existem três sensores, um para medir a posição da bola de referência, outro para a posição da bola a ser controlada e um para medir a posição angular do servo-motor (figura 4.1).

Filtros de primeira ordem foram introduzidos, digitalmente, ao procedimento de leitura dos sensores para que o ruído, inerente a esse tipo de sensor, seja minimizado, como sugere o manual do projetista (J. Apkarian, 1995).

O objetivo do controlador é fazer com que a bola colocada sobre a trave siga a trajetória especificada pela bola de referência. Para tanto, é projetado um sistema de controle que envia um sinal em forma de tensão para o servo-motor que ao mover-se levanta ou abaixa uma das extremidades da trave, fazendo assim com que a bola

Figura 4.1: Localização dos sensores no sistema *ball and beam*.

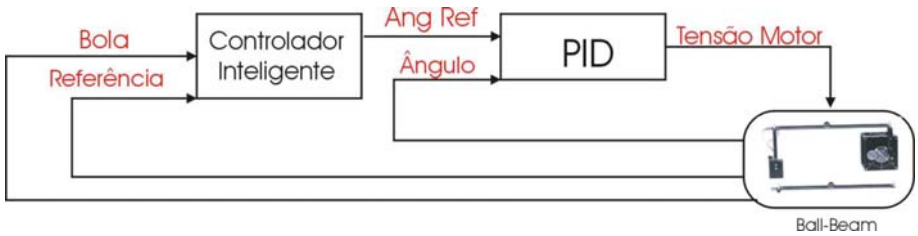

Figura 4.2: Fluxograma ilustrando as malhas de controle do *ball and beam*.

se mova.

O sistema de controle projetado para o *ball and beam* encontra-se dividido em duas malhas (figura 4.2): uma externa, onde um controlador inteligente fica responsável por receber a posição da bola a ser controlada e a posição da bola de referência e, a partir daí, fornecer o ângulo de referência para a segunda malha, onde um controlador PID gera um sinal de controle necessário para que o servo-motor posicione-se de acordo com o ângulo de referência fornecido pelo primeiro controlador.

A utilização do controlador inteligente para a substituição das duas malhas de controle da planta ainda consiste em um desafio considerável. Desta forma, optou-se pela utilização das técnicas híbridas para substituir apenas a malha externa, já que a malha interna funciona suficientemente bem com o PID projetado e a malha externa constitui um desafio suficiente para o proposto.

4.2 Resultados com o ANFIS

Como já mencionado, o modelo ANFIS será otimizado através do *Back-propagation*. Dessa forma, é necessária a obtenção de pares (pontos) de treinamento. Para a obtenção dos pontos de treinamento, utilizou-se um controlador PID (Proporcional Integrativo Derivativo) substituindo o controlador inteligente da figura 4.2. A escolha de três entradas para o modelo ANFIS se baseou na característica do controlador a ser copiado, no caso o PID. As três entradas consideradas foram os erros: atual e anterior da bola a ser controlada em relação à bola de referência, e o ângulo de referência anterior.

Após a etapa de captação dos pontos de treinamento, passou-se para a etapa do treinamento do modelo ANFIS. O algoritmo de treinamento utilizou os pares entrada-saída obtidos com o controlador PID para adaptar os parâmetros ajustáveis do sistema.

Visto como um sistema *fuzzy*, o sistema possui 5 funções de pertinência em forma de sino para cada variável de entrada. Desta forma, o cruzamento entre as funções de pertinência formam 125 possíveis regras que referenciam polinômios de primeira ordem do tipo $px + qy + rz + s$, em função das variáveis de entrada x = erro atual, y = erro anterior e z = ângulo de referência anterior. As variáveis p, q, r e s são os parâmetros ajustáveis dos polinômios e todos foram iniciados com valor igual a 0 (zero).

O coeficiente de aprendizagem para este sistema foi iniciado com $\eta = 10^{-6}$, os valores iniciais para as funções de pertinências foram uniformemente distribuídos dentro de intervalos e o período de amostragem considerado foi igual a 0,05 segundos. A figura 4.3 ilustra as funções de pertinência em seu estado inicial em que o primeiro gráfico representa o erro *fuzzy* anterior e atual, e o segundo o ângulo de referência anterior.

O valor inicial para a variável η assim como os intervalos nos quais as funções de pertinência foram distribuídas, constituíram uma das maiores dificuldades na implementação do modelo. Quando o valor do η era elevado (cerca de 10^{-4} para

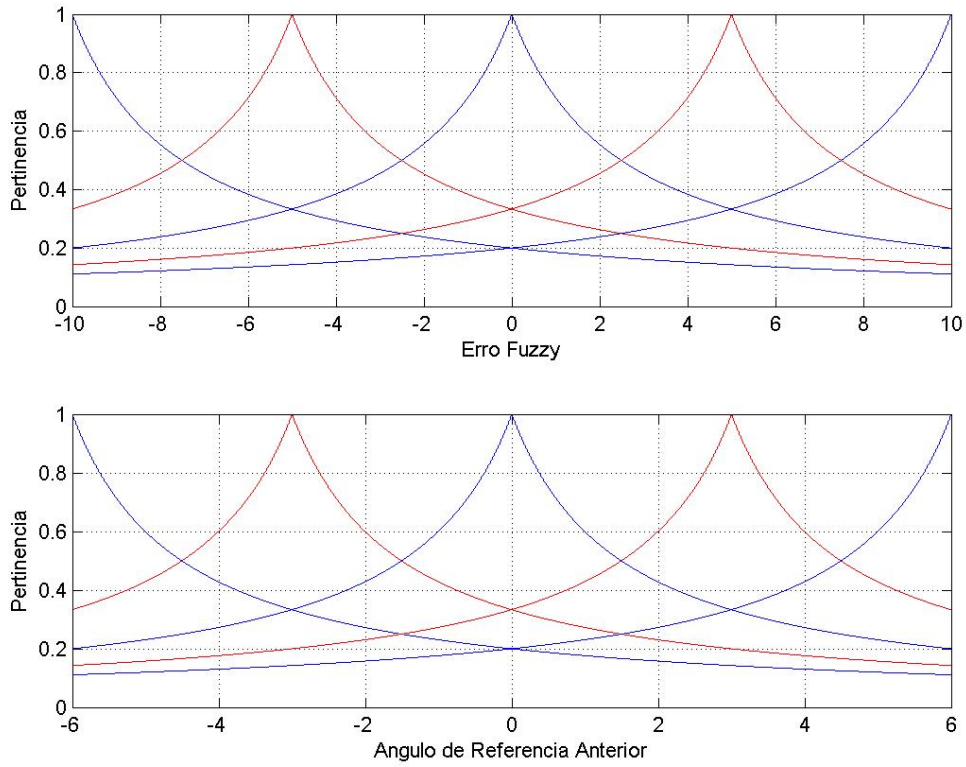

Figura 4.3: Estado inicial das funções de pertinência.

este modelo) ou quando os intervalos eram muito distantes da realidade do sistema a ser controlado, o programa tornava-se instável.

Fazendo uma associação com o sistema *fuzzy*, pode-se deduzir que a rede neural equivalente terá o seguinte número de neurônios em cada camada: 3 - camada 0; 15 - camada 1; 125 - camadas 2, 3 e 4; 1 - camada 5.

A figura 4.4 mostra como foi feita a inserção do ANFIS nas malhas de controle para a etapa de aprendizagem do modelo. Nessa figura também é possível identificar de onde foram obtidos os pontos entrada-saída usados no treinamento.

Para o treinamento foram coletados 800 pontos a partir do PID da malha externa do sistema de controle que atuava sobre o *ball and beam*. A saída desejada da rede, em função de cada ponto de treinamento, pode ser vista na figura 4.5.

O gráfico na figura 4.6 ilustra tanto a evolução do η quanto o erro quadrático do sistema para 3000 épocas de um conjunto de treinamento bem sucedido. Uma

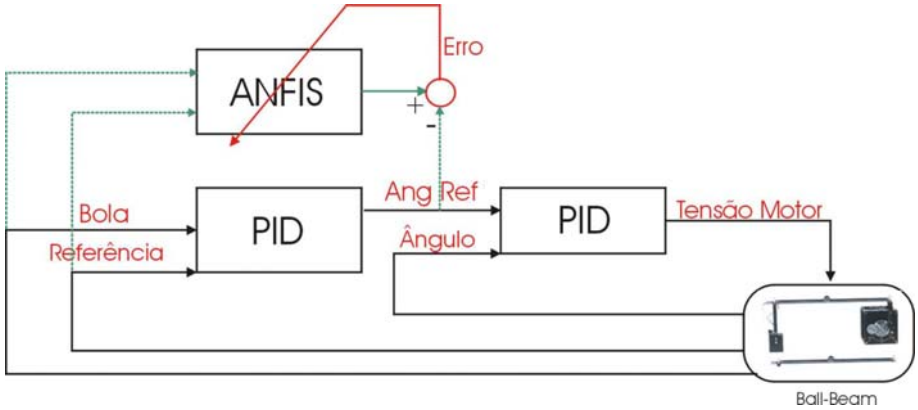

Figura 4.4: Fluxograma ilustrando as malhas no aprendizado do modelo ANFIS.

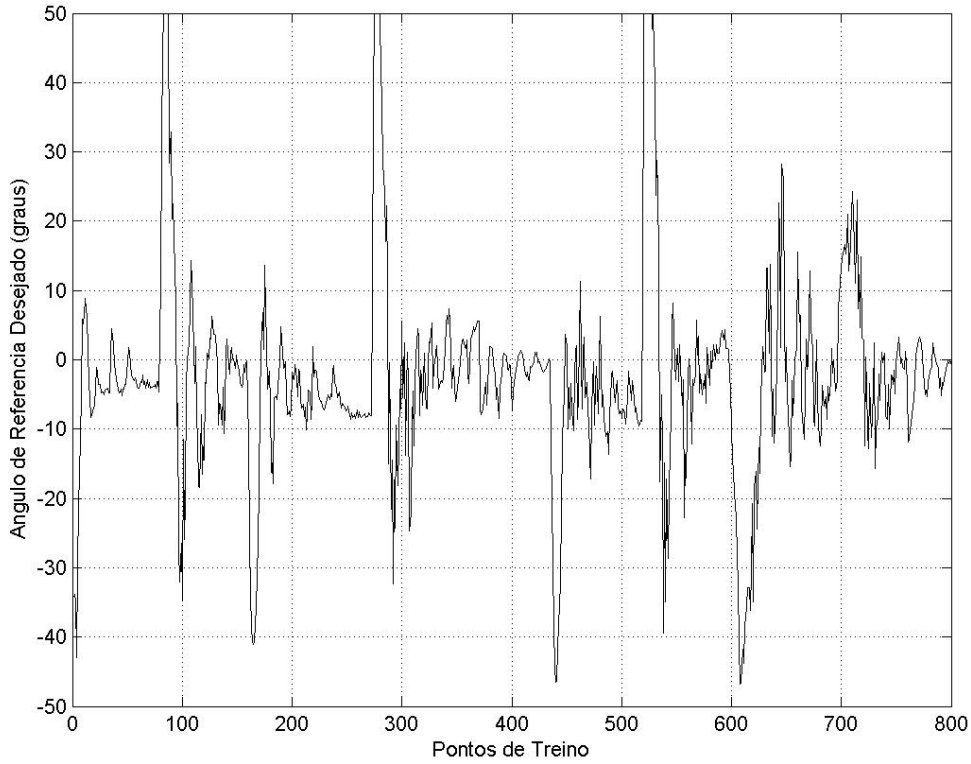

Figura 4.5: Pontos de treinamento obtidos para o ANFIS.

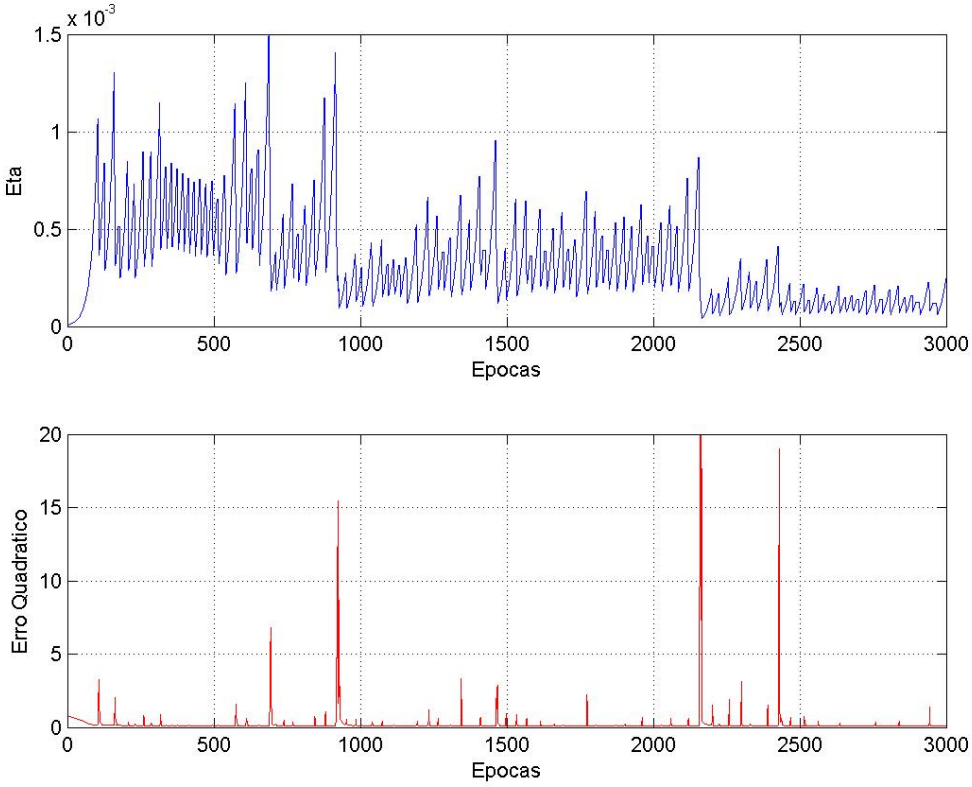

Figura 4.6: Evolução do coeficiente de aprendizagem η e do erro quadrático durante a aprendizagem.

época foi definida como sendo a passagem do algoritmo de aprendizagem por todos os 800 pontos de treinamento.

Após a etapa de treinamento que durou cerca de 22 horas, encontrou-se um erro quadrático de aproximadamente 0.035 em 160.000 épocas, o que, considerando que os valores usados no treinamento não foram normalizados, aproximou bastante os valores obtidos pelo controlador aos valores de aprendizado (figura 4.7). Como pode ser observado na figura 4.8 as funções de pertinência para as três entradas sofreram algumas modificações.

Para uma efetiva validação do aprendizado, utilizou-se o controlador treinado na planta *ball and beam* para uma seqüência de pontos de referência pré-estabelecidos, o que dispensou o uso da trave de referência. O efeito do controlador na planta pode ser observado na figura 4.9. Observe que o controlador *neuro-fuzzy* segue a referência

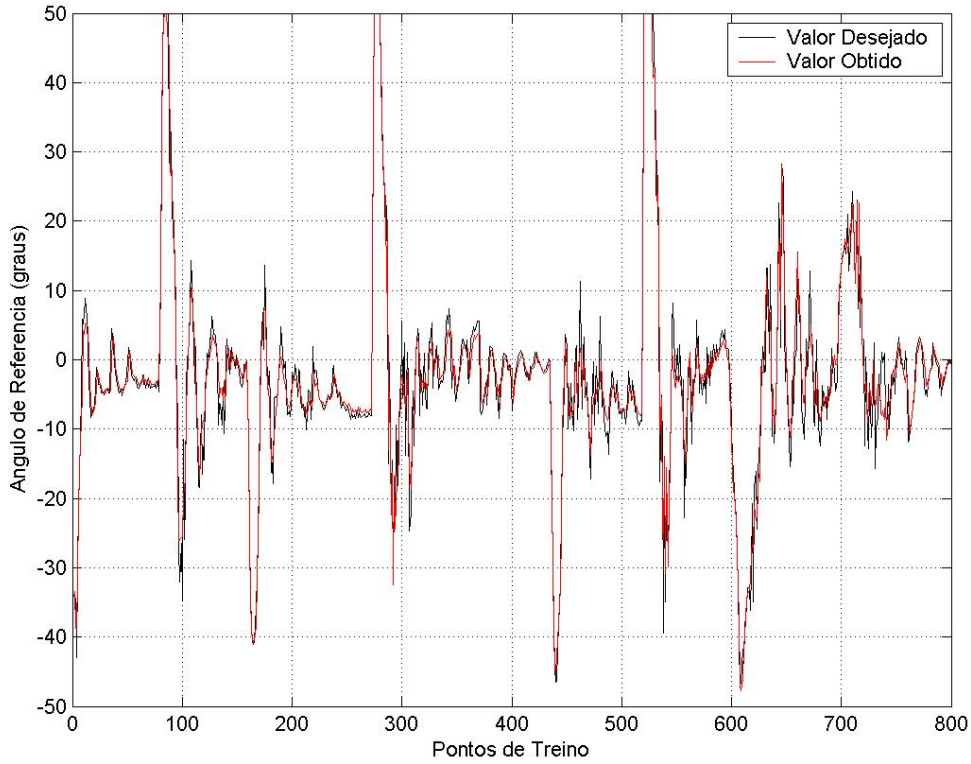

Figura 4.7: Valores obtidos pelo controlador em relação aos valores de aprendizado.

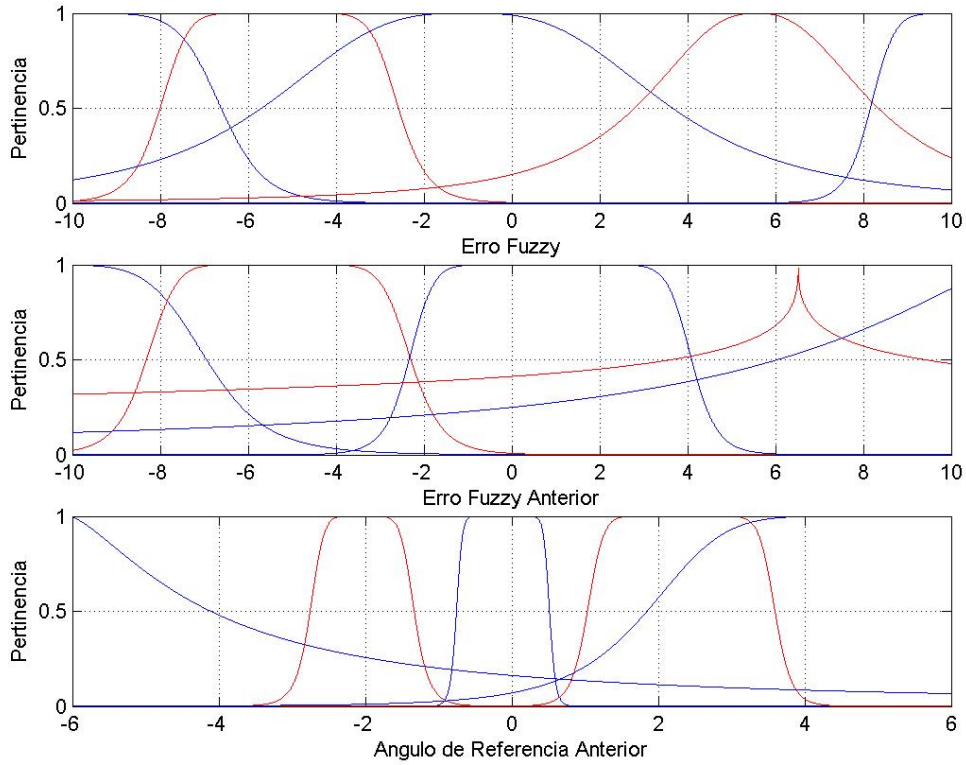

Figura 4.8: Funções de pertinência após a etapa de aprendizado.

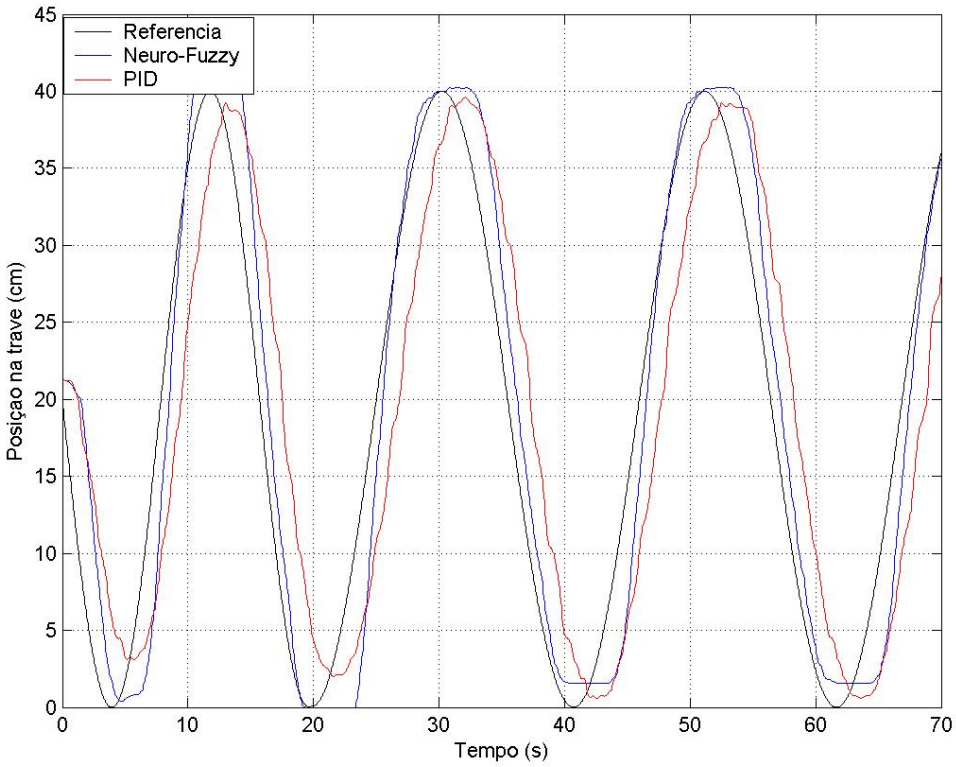

Figura 4.9: Controlador *neuro-fuzzy* de modelo ANFIS treinado e aplicado ao *ball and beam*.

assim como o controlador PID e às vezes, de forma mais precisa. Porém, em baixas velocidades (extremos superior e inferior dos gráficos) a precisão é reduzida por interferência do atrito seco. Analisando o sinal de saída destes controladores, ou seja, analisando o ângulo de referência fornecido pelo PID e pelo *neuro-fuzzy* (figura 4.10), observa-se que os valores obtidos pelo controlador *neuro-fuzzy* oscilam menos, o que nos leva a crer que o processo de aprendizagem também atuou como um filtro sobre o sinal de saída.

Para as figuras 4.11 e 4.12 a referência recebeu alguns degraus de amplitude variada. Note que nos degraus a referência não é seguida com muita perfeição pelo controlador *neuro-fuzzy*, porém a trave recebe o ângulo que, em teoria, levaria a bola a para a referência. Este fato se deve à interferência do atrito seco presente em toda superfície física da planta. O PID por oscilar mais, consegue evitar a interferência

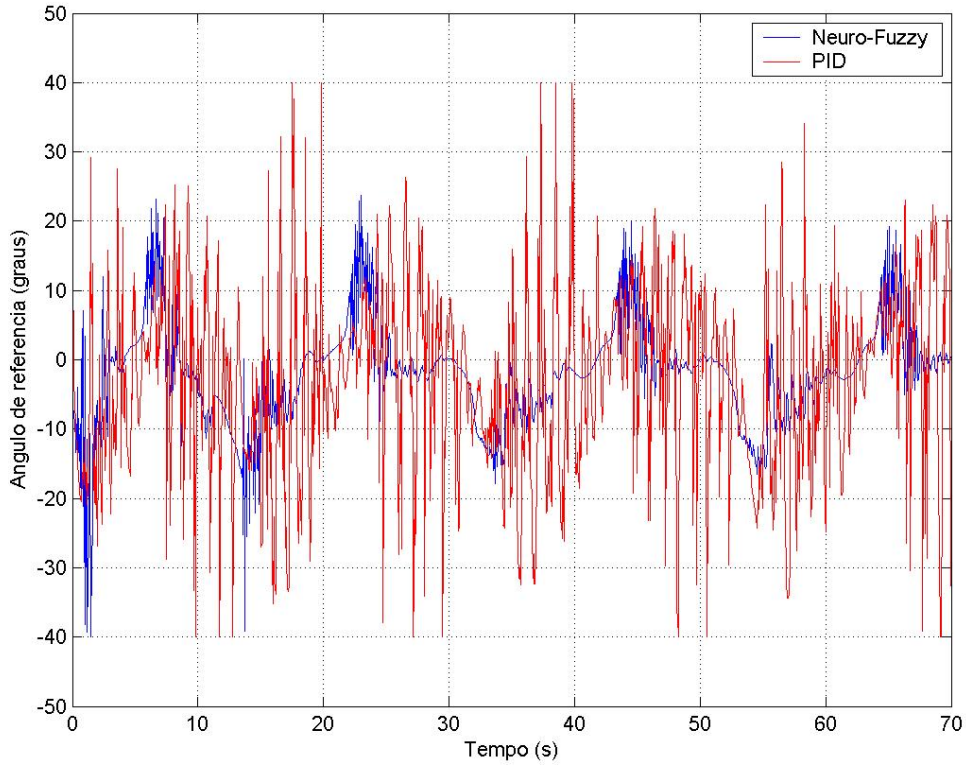

Figura 4.10: Ângulos de referencia fornecidos pelos controladores PID e *neuro-fuzzy*.

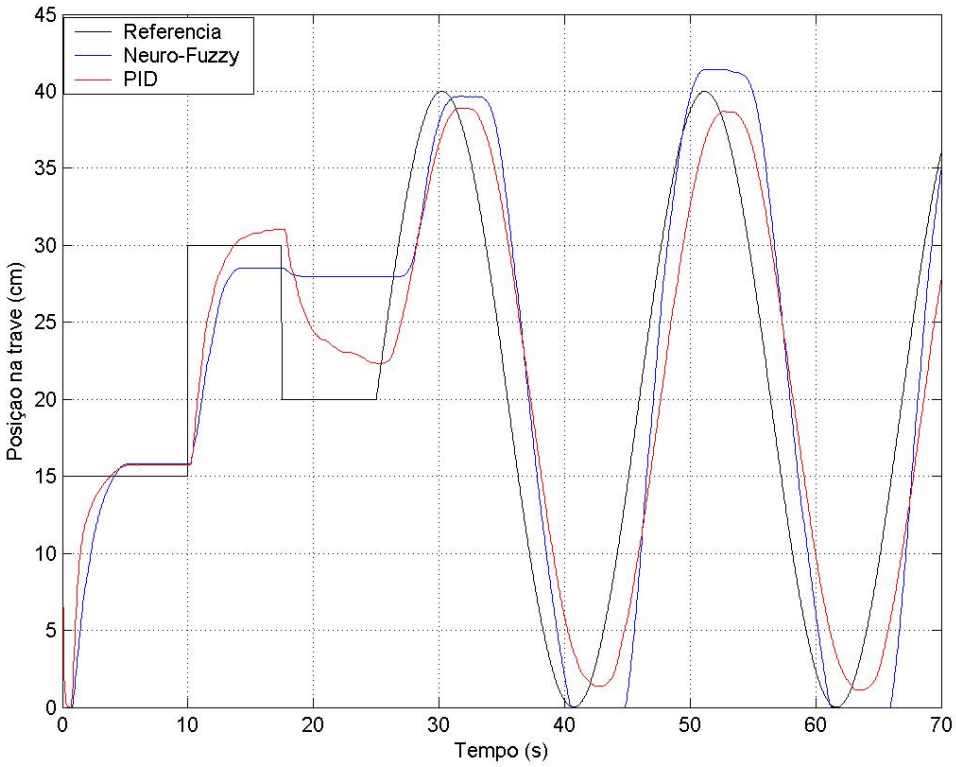

Figura 4.11: Variação da posição da bola de referência em degraus simulados.

do atrito seco. Em contrapartida, desgasta mais as engrenagens assim como outras partes físicas da planta. Outro fator que deve ser considerado é que os pontos de treino foram obtidos a partir da planta utilizando a trave de referência. Desta forma, os pontos de treinamento obtidos não possuíam a característica de um degrau como o simulado pela referência da figura 4.11, pois o filtro presente no sensor de leitura da trave de referência impossibilita esse tipo de oscilação brusca.

4.3 Resultados com o NEFCON

O controlador *neuro-fuzzy* com características baseadas no modelo NEFCON foi implementado e apresenta as seguintes particularidades:

- Existem duas entradas, com seus intervalos divididos entre 5 funções de pertinência triangulares para cada uma.

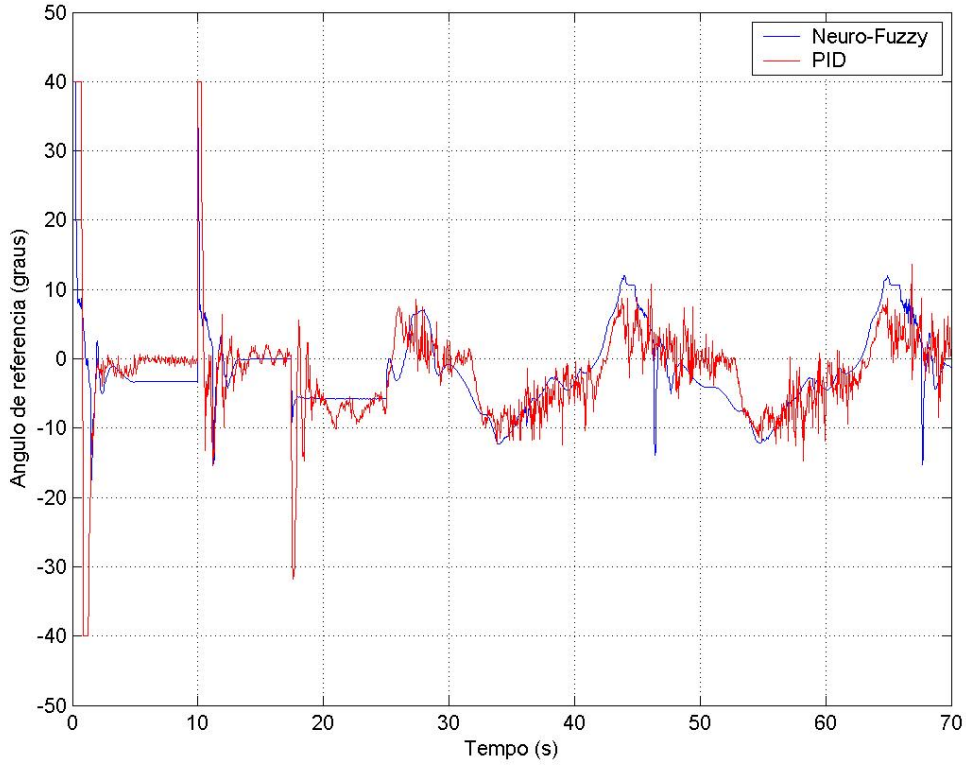

Figura 4.12: Ângulos de referencia fornecidos pelos controladores PID e *neuro-fuzzy* quando degraus são aplicados.

Tabela 4.1: Regras inicias para o controlador *neuro-fuzzy*

		Variação do Erro				
	-	NG	NP	ZE	PP	PG
E	NG	-	-	-	-	-
R	NP	-	-	-	-	-
R	ZE	-	-	ZE	-	-
O	PP	-	-	-	-	-
	PG	-	-	-	-	-

- Para uma única saída (o ângulo de referência), o intervalo foi dividido por 7 funções de pertinência.
- As funções de pertinência de entrada são iniciadas com 50% de sobreposição e simetricamente divididas dentro do seu intervalo de atuação o qual é fornecido pelo projetista.
- Como conseqüência do numero de funções de pertinência para cada entrada, a base de regras pode ter um máximo de 25 regras ativas, ou seja, regras com implicações correspondentes a funções de pertinência de saída.
- A constante de aprendizagem para o controlador foi considerada igual a 10^{-3}.
- Período de amostragem para este modelo foi de 0,02 segundos. Este valor deve-se ao fato do modelo NEFCON possuir um custo computacional inferior ao do modelo ANFIS.

Definiram-se como entradas para o sistema o erro de posição da bola e sua variação. O valor desejado para ambas entradas foi definido como zero e a faixa de execução utilizada para estas variáveis foi de -20 a 20 cm para o erro e de -0,6 a 0,6 cm/s para a variação do erro. Com relação a variável de saída, o ângulo de referência pôde variar de -45 a 45 graus. A figura 4.13 ilustra tanto o estado inicial das funções de pertinência de entrada quanto das de saída. A base de regras inicial foi considerada vazia, exceto pela regra central que recebeu o valor de saída equivalente ao ângulo de referência zero (tabela 4.1).

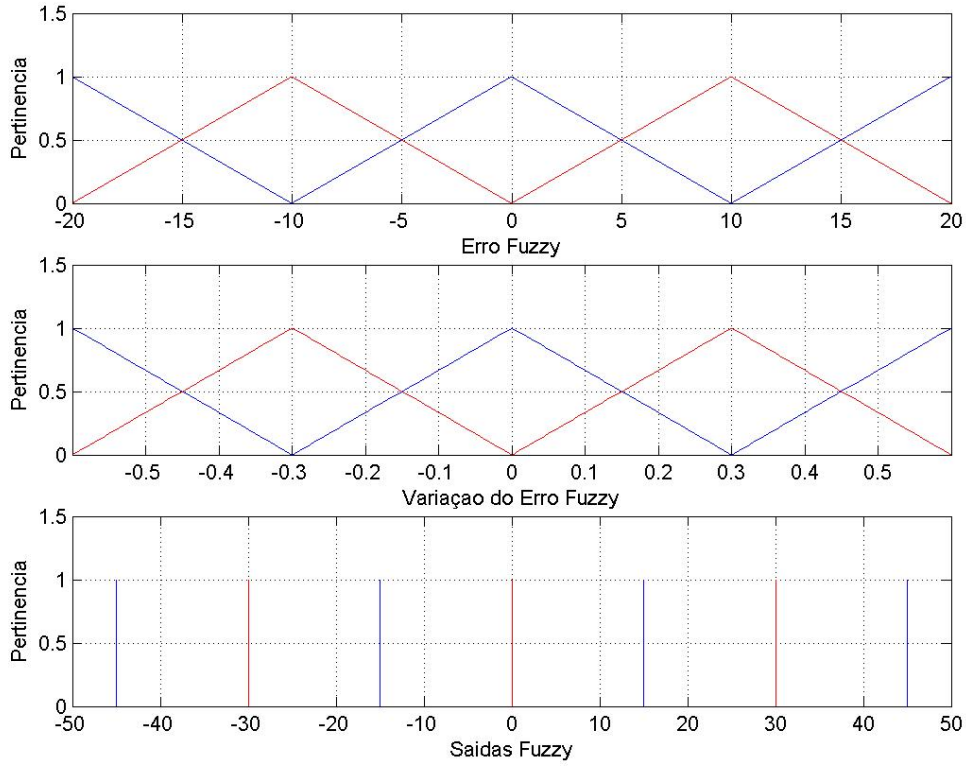

Figura 4.13: Estado inicial das funções de pertinência de entrada e de saída.

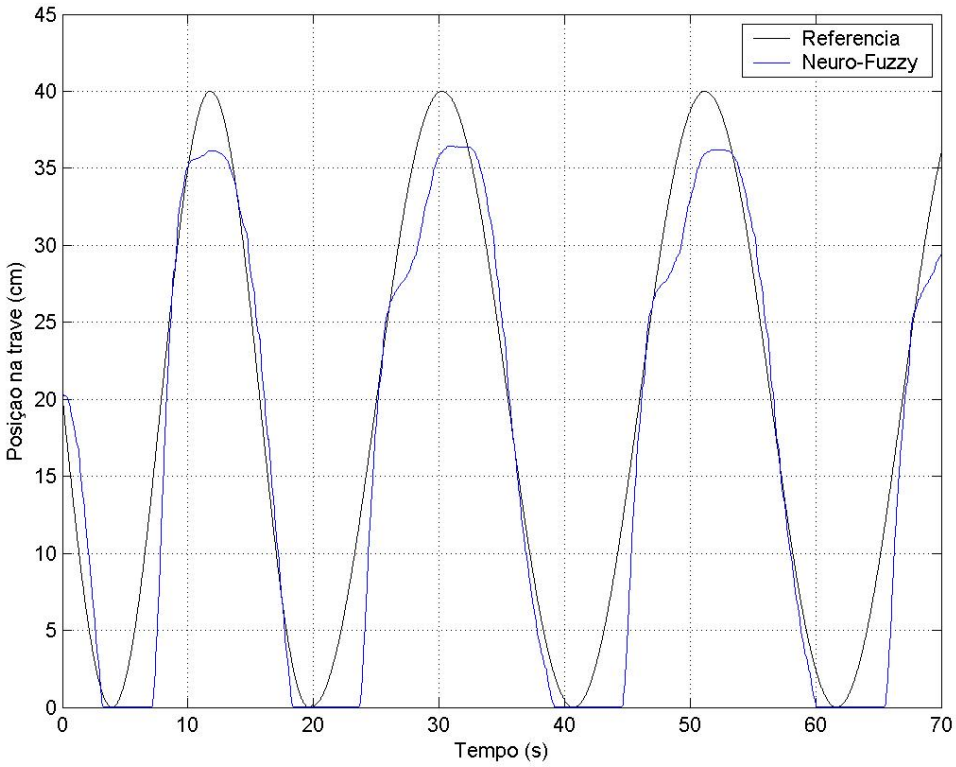

Figura 4.14: Etapa de criação de regras do algoritmo NEFCON em tempo real.

Como já mencionado, o NEFCON tem sua fase de aprendizagem em tempo real. Para mostrar o desenvolvimento desta aprendizagem enquanto o algoritmo tenta controlar a planta, utilizou-se uma seqüência de aprendizagem sem a barra de referência. A etapa de criação de regras foi a primeira a entrar em operação e seu resultado pode ser analisado na figura 4.14. Observe que após 25 segundos, o ângulo de referência obtido tende a parar a bola que se encontra em uma velocidade elevada (figura 4.15). Observe ainda que a bola não alcança a posição 40 na trave e em compensação, atinge a posição zero com extrema facilidade. Isto se deve ao fato de o ângulo zero fornecido pelo controlador não corresponder ao ângulo em que a bola tende a ficar parada, ou seja, uma situação em que é necessário calibrar o ângulo de saída. A calibração do controlador ocorre na outra etapa do algoritmo, na fase de otimização de funções de pertinência.

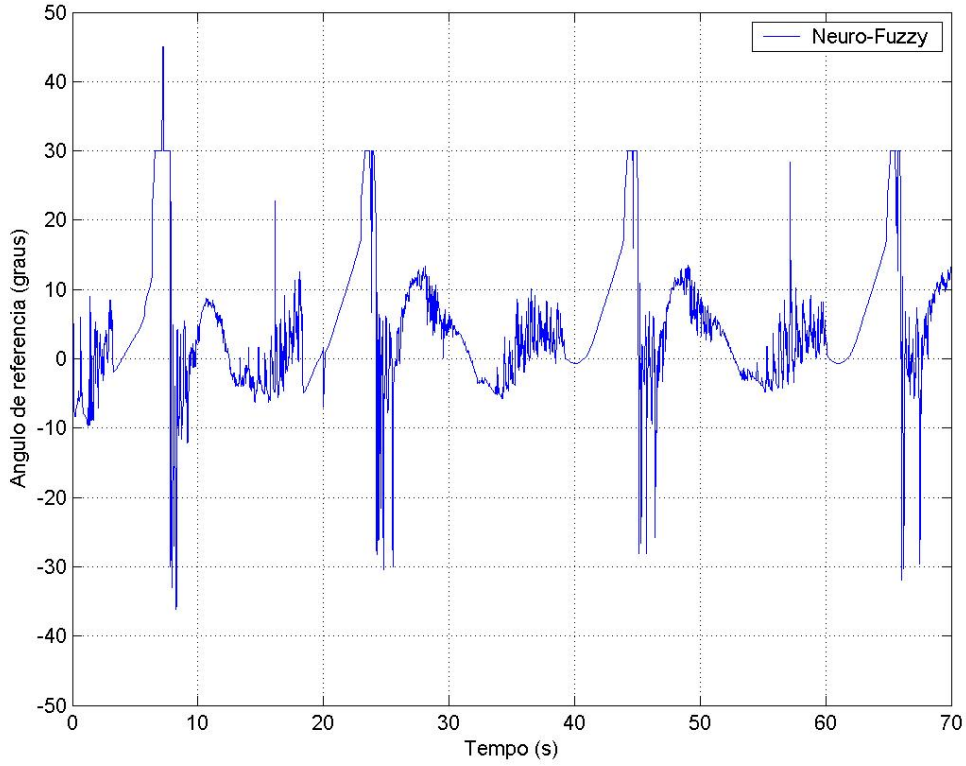

Figura 4.15: Ângulo de referência fornecido pelo controlador durante a etapa de criação de regras.

Tabela 4.2: Regras otimizadas encontradas na fase de criação de regras para o controlador *neuro-fuzzy*

		Variação do Erro				
		NG	NP	ZE	PP	PG
E	NG	-	-	-	-	-
R	NP	-	-	NP	PP	-
R	ZE	NG	NP	ZE	PP	-
O	PP	NG	PP	PP	PM	-
	PG	-	-	-	-	-

Tabela 4.3: Regras encontradas após 35 segundos de criação de regras

		Variação do Erro				
		NG	NP	ZE	PP	PG
E	NG	-	-	-	-	-
R	NP	-	-	NP	-	-
R	ZE	NG	NP	ZE	PG	-
O	PP	NG	NP	PP	PG	-
	PG	-	-	-	-	-

A base de regras otimizada após o a aplicação da fase da aprendizagem é apresentada na tabela 4.2.

A segunda parte do algoritmo funciona como um ajuste fino dos parâmetros *fuzzy* por só mover as funções de pertinência em busca de um ajuste ótimo. Considerando o sistema novamente a partir do estado inicial, aplicaram-se as duas partes do algoritmo para a mesma seqüência de referências, considerando agora a fase de criação de regras até os 35 segundos e, em seguida, a fase de otimização de funções de pertinência. A figura 4.16 mostra o desenvolvimento do sistema durante a aprendizagem. As regras criadas até os 35 segundos estão apresentadas na tabela 4.3.

Observe que existem regras diferentes criadas nesta nova aplicação. Esta situação é possível graças à política empregada pelo algoritmo de se adaptar as condições atuais da planta e desta forma, criar uma base de regras atual baseada na dinâmica da planta no momento em que entra em operação.

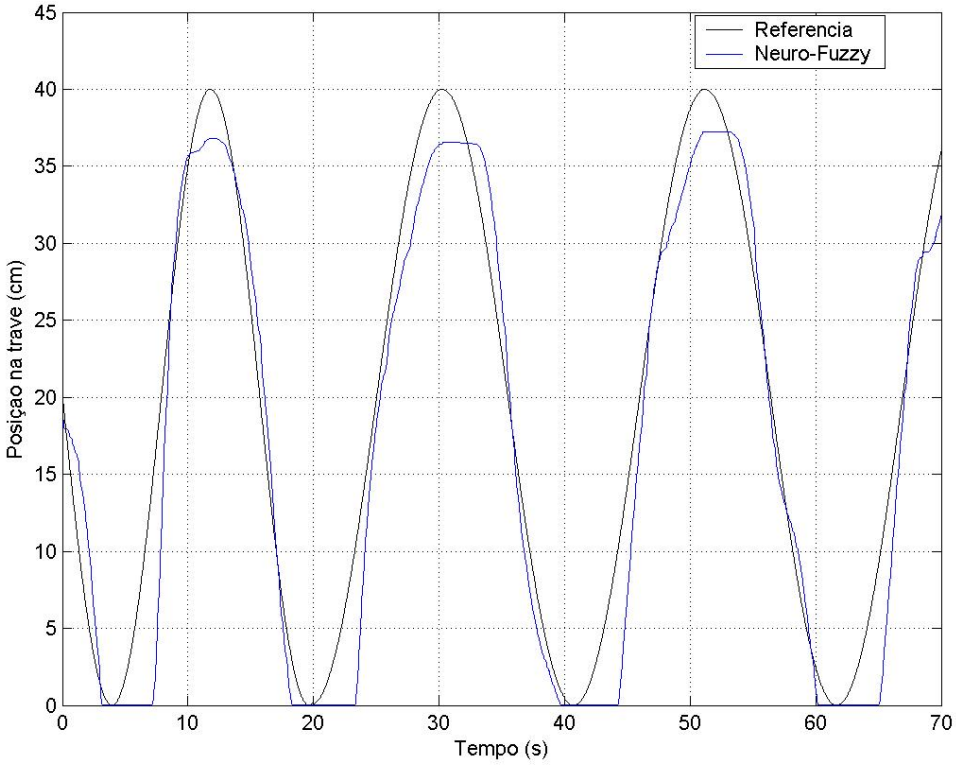

Figura 4.16: Etapas de criação de regras até 35 segundos quando a otimização de funções de pertinência do controlador começa a atuar.

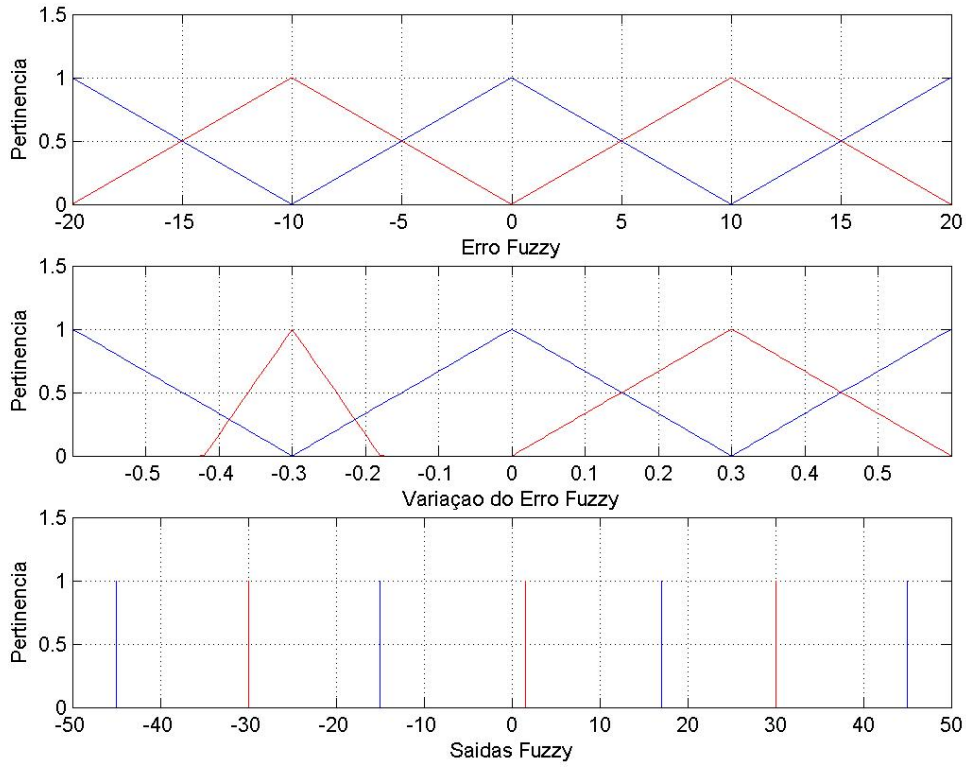

Figura 4.17: Modificação das funções de pertinência com a finalidade do ajuste fino do controlador.

Analisando as funções de pertinências (figura 4.17) é possível observar a compensação fornecida pelo algoritmo para a função de pertinência de saída com a finalidade de calibrar o ângulo zero da planta, assim como, causar um ajuste fino do controlador.

Depois de treinado, o controlador foi aplicado novamente à mesma seqüência de referências. O resultado pode ser analisado na figura 4.18. Observe que a referência passou a ser perseguida com maior fidelidade o que reduziu o erro de forma geral.

Atribuindo uma nova seqüência de referências, aplicando o algoritmo de criação de regras nos primeiros 35 segundo e, em seguida, aplicando a otimização das funções de pertinência, obtivemos um novo conjunto de regras e funções de pertinência ilustradas na tabela 4.4 e na figura 4.19, respectivamente.

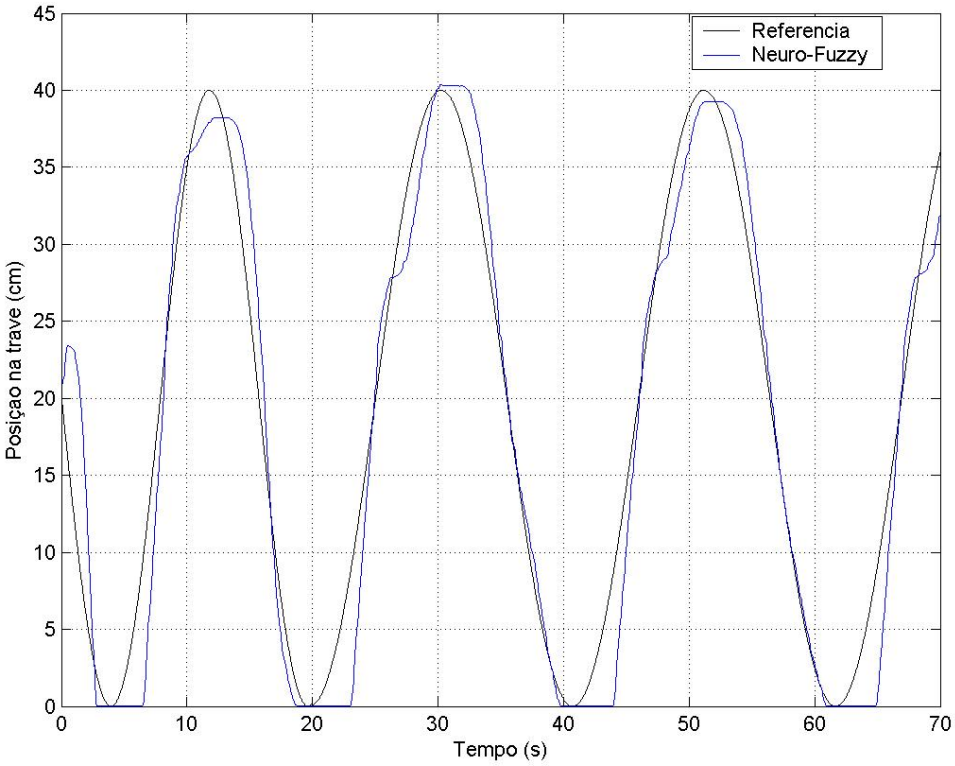

Figura 4.18: Saída obtida após o treinamento do controlador.

Tabela 4.4: Regras encontradas após 35 segundos de criação de regras

		Variação do Erro				
		NG	NP	ZE	PP	PG
E	NG	-	-	-	-	-
R	NP	NG	-	NM	PM	PG
R	ZE	NG	NP	ZE	PP	-
O	PP	NG	NP	PP	PG	PG
	PG	-	-	-	-	-

Tabela 4.5: Regras obtidas após o treinamento utilizando a trave de referência

		Variação do Erro				
		NG	NP	ZE	PP	PG
E	NG	-	-	-	-	-
R	NP	NG	NP	NP	PP	PG
R	ZE	NG	NP	ZE	PP	PG
O	PP	NG	NM	PP	PM	PG
	PG	NG	PG	PG	PG	-

Uma situação de assimetria ocorreu na função de pertinência central (correspondente à pertinência ZE) da segunda entrada (Variação do erro) do controlador *neuro-fuzzy*. Mesmo sabendo que o algoritmo tende a ajustar os dois lados simetricamente, as restrições a formação de lacunas entre as funções de pertinência possibilita essa condição de assimetria.

O resultado obtido e os ângulos de referência fornecidos pelo controlador podem ser vistos nas figuras 4.20 e 4.21, respectivamente.

O ultimo teste foi feito com a referência obtida a partir da trave de referência. Após cerca de 70 segundos de treinamento, considerando criação de regras e otimização de funções de pertinência, chegou-se à base de regras da tabela 4.5 e às funções de pertinência da figura 4.22.

A dinâmica da planta e os ângulos de referência fornecidos pelo controlador estão ilustrados nas figuras 4.23 e 4.24.

Os gráficos das figuras 4.25 e 4.26 mostram o comportamento da planta para os três controladores apresentados. Estes gráficos não podem ser utilizados para

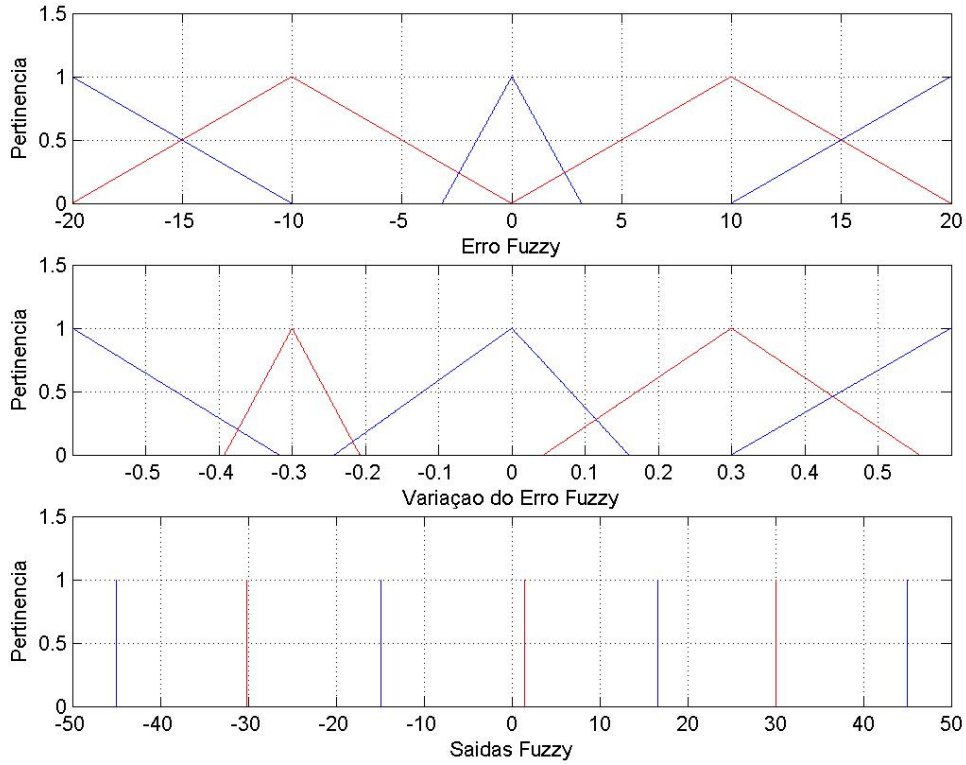

Figura 4.19: Modificação das funções de pertinência com a finalidade de ajuste fino do controlador.

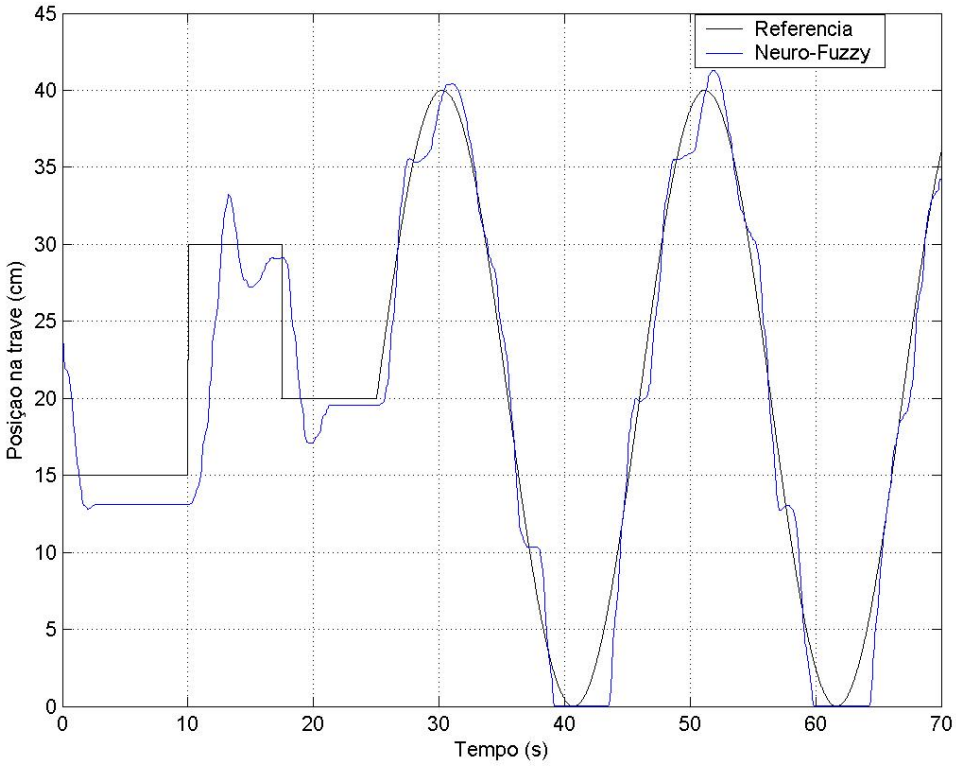

Figura 4.20: Comportamento da planta para nova seqüência de referências.

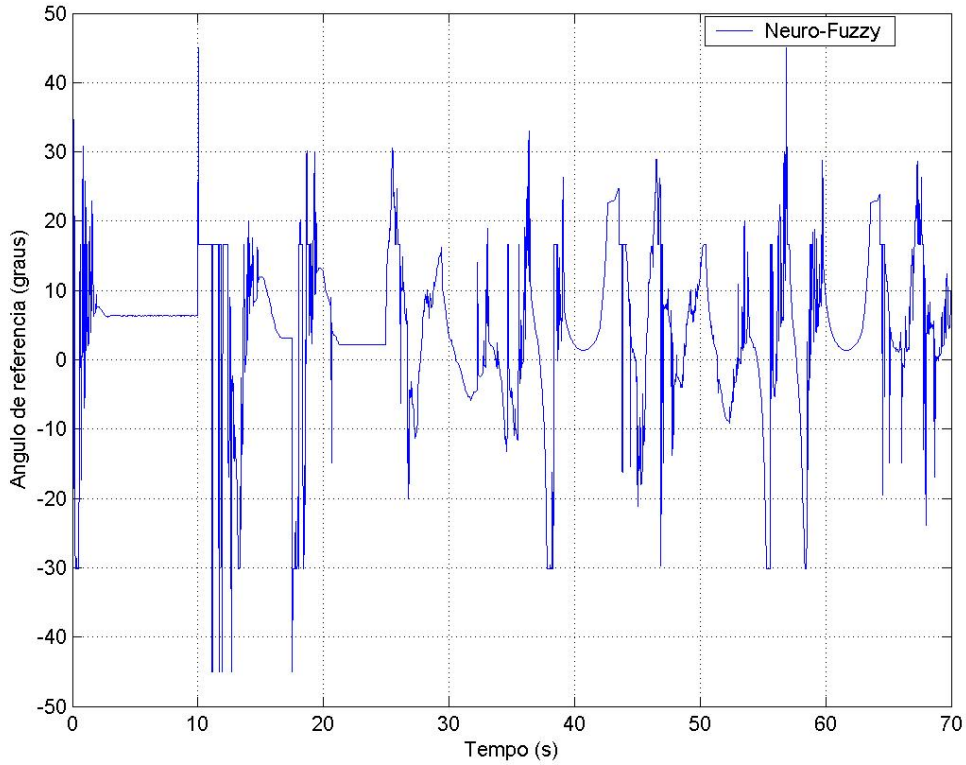

Figura 4.21: Ângulos de referência fornecidos pelo controlador para nova seqüência de referências.

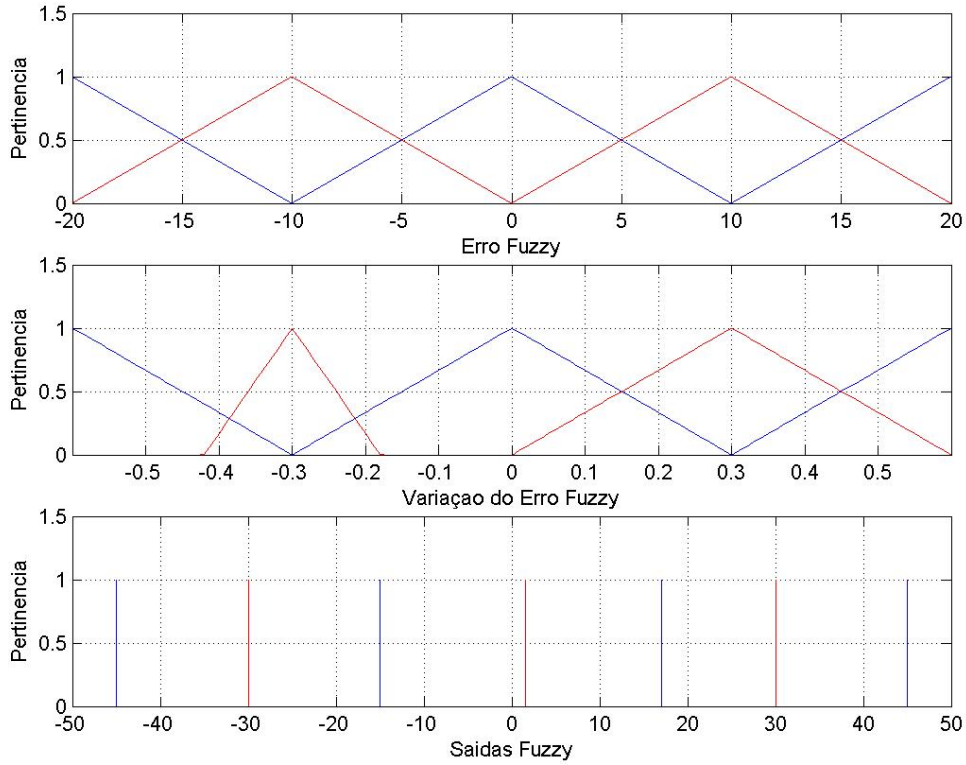

Figura 4.22: Funções de pertinência obtidas após o treinamento utilizando a trave de referência.

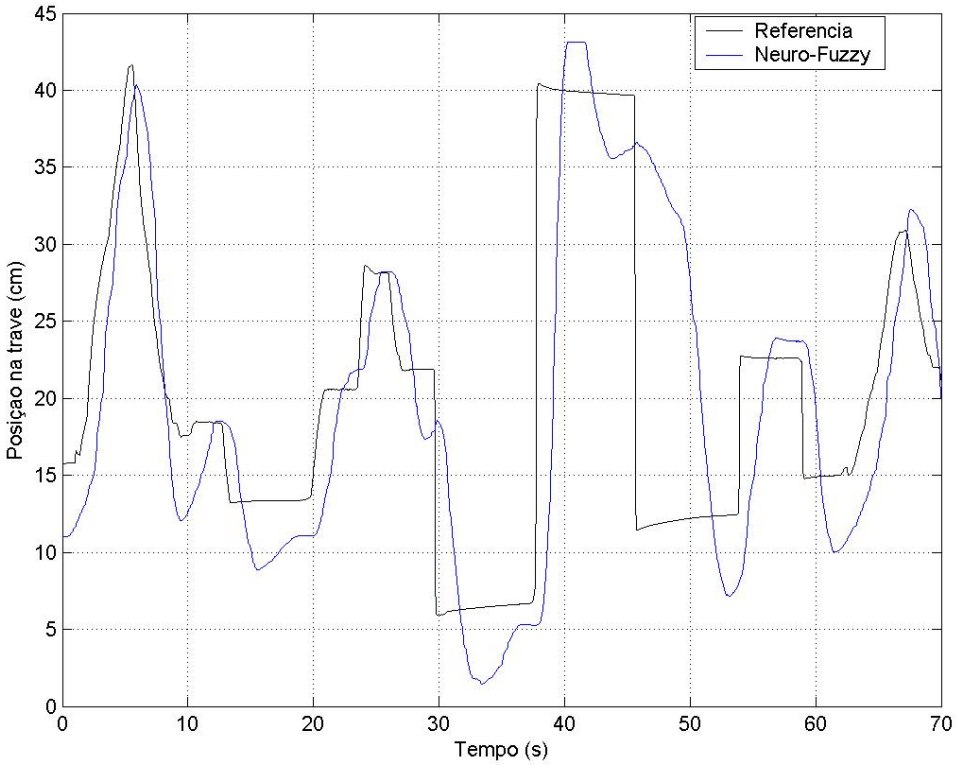

Figura 4.23: Comportamento da planta para nova seqüência de referências.

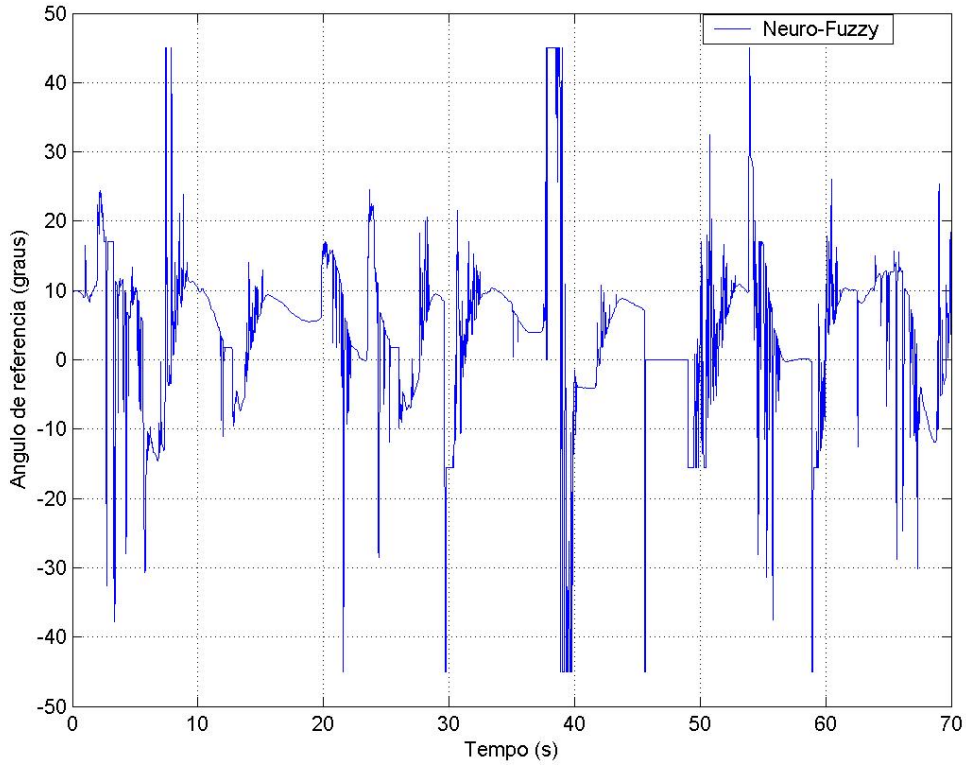

Figura 4.24: Ângulos de referência fornecidos pelo controlador utilizando a trave de referência.

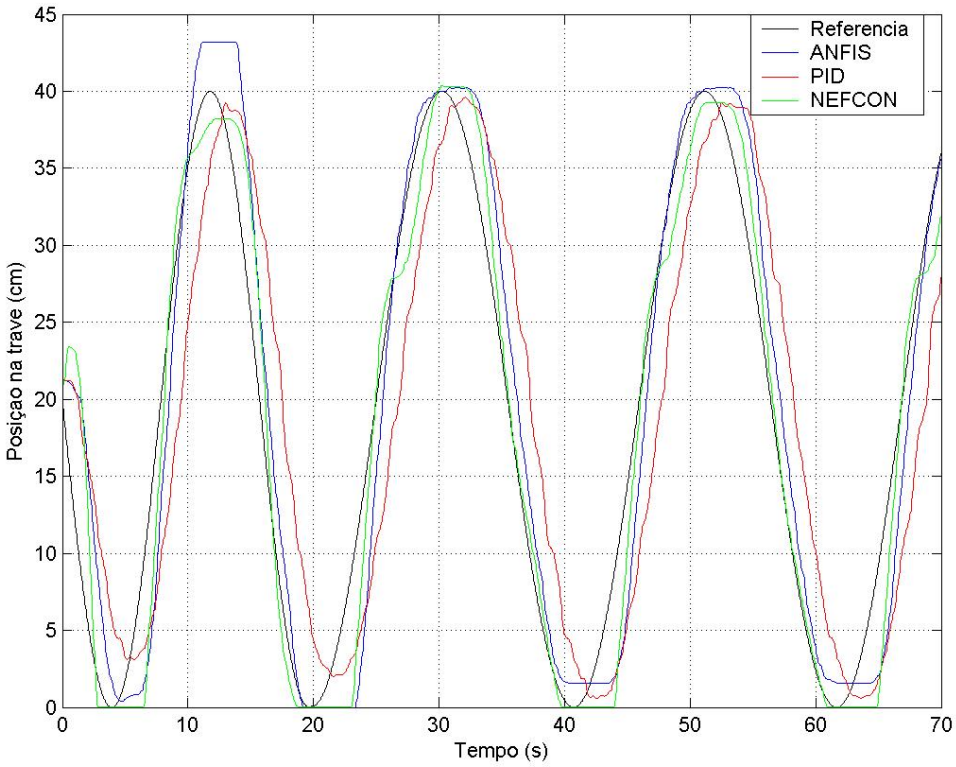

Figura 4.25: Comportamento da planta para os controladores projetados e referência em forma de senóide.

comparar as técnicas desenvolvidas já que o controlador ANFIS copiou o controlador PID e este não é ótimo, já o controlador NEFCON é ótimo pela sua própria estrutura.

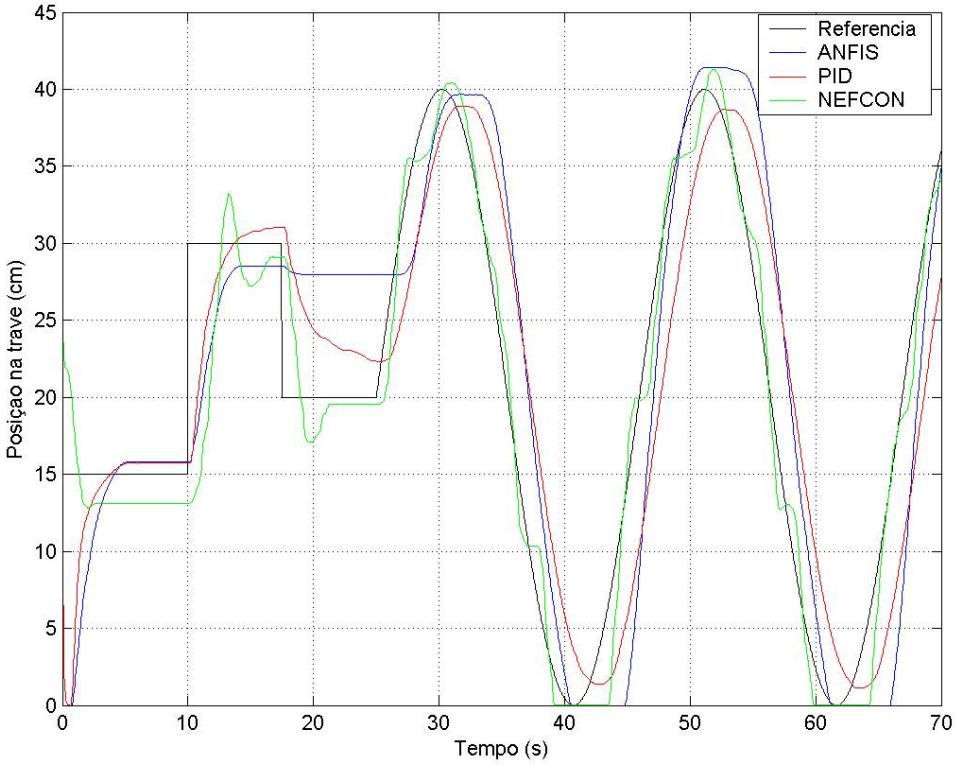

Figura 4.26: Comportamento da planta para os controladores projetados e referência em forma de degrau-senóide.

Capítulo 5

Conclusões

A união de diferentes ferramentas de inteligência artificial tem possibilitado a proposta de técnicas híbridas para o controle de plantas e processos complexos, mesmo em casos não lineares. Estes controladores híbridos têm se consolidado como uma importante área de pesquisa e desenvolvimento, pois em um único controlador podem ser encontradas características favoráveis de duas ou mais técnicas. Neste trabalho foram desenvolvidos dois controladores *neuro-fuzzy*, baseados em modelos distintos, para o controle do sistema *ball and beam*.

O controlador baseado no modelo ANFIS foi desenvolvido com uma aprendizagem baseada no algoritmo *Back-propagation*, usando pares de treinamento extraídos de um controlador PID, e, além de controlar o sistema *ball and beam*, eliminou as altas variações do sinal de controle na etapa de aprendizagem. Já o controlador baseado no modelo NEFCON utilizou para seu aprendizado uma técnica fundamentada em aprendizagem por reforço, com algumas alterações, e também conseguiu resultados satisfatórios, pois implementou com sucesso a tarefa de controlar o sistema *ball and beam*. Por oferecer uma interface mais simples, o modelo NEFCON apresentou uma melhor interação entre o usuário e o algoritmo para a modificação da estrutura por parte do usuário, pois neste modelo o usuário necessitava informar apenas os intervalos para as entradas e saída do controlador. Esta característica possibilitou uma aprendizagem voltada às necessidades do usuário.

A utilização de intervalos para as entradas no modelo NEFCON reforçou ainda mais a interação do usuário com o algoritmo e esta abordagem consagrou-se como a principal contribuição deste trabalho para a implementação deste modelo.

As duas técnicas inteligentes híbridas foram empregadas com sucesso sobre a planta. Contudo, o valor inicial para algumas variáveis ajustáveis de ambos controladores necessitou de uma escolha em razão da sua atuação e não foram arbitrárias, pois levavam à não convergência dos algoritmos utilizados.

A possibilidade de inserção de uma tabela de regras, ou mesmo, funções de pertinências pré-determinadas para os controladores híbridos desenvolvidos, é uma característica que compatibiliza estes controladores com os controladores que já vêm sendo usados no mercado, permitindo a migração suave das técnicas convencionais para estas novas técnicas.

Técnicas de otimização como algoritmos genéticos, colônias de formigas, entre outros, são mais algumas técnicas que podem promover o aperfeiçoamento ainda maior dos modelos híbridos, e poderão inserir, no futuro, características como mutação ou a evolução das várias formas associadas às estruturas. Outro fator que poderá ser analisado é a compatibilidade dos modelos desenvolvidos, a fim de aplicar ambos a uma mesma planta, de forma que, estes modelos colaborem ou concorram entre si.

Alterações nas estruturas dos modelos desenvolvidos, como os formatos das funções de pertinências, t-normas e t-conormas, poderiam ser analisadas e, desta forma, fornecer um estudo mais preciso a respeito de ambas as técnicas.

Referências Bibliográficas

A. C. Guyton & J. E. Hall, *Tratamento de Fisiologia Médica*, 9^0 edição, Ed.Guanabara Koogan, 1997.

A. El Hajjaji & S. Bentalba, *Fuzzy Path Tracking Control for Automatic steering of vehicles*, Robotics and Autonomous Systems, Elsevier Science, pp.203-213, 2003.

A. Nürnberger, D. Nauck & R. Kruse, *Neuro-Fuzzy Control Based on the NEFCON Model: Recent Developments*, Soft Computing 2, pp.168-182, 1999.

A. Valishevsky, *Adaptive Learning Algorithm for Hybrid Fuzzy System*, [online] <`http://dssg.cs.rtu.lv/en/staff/valishevsky/home/anfis.pdf`>, University of Latvia, Department of Computer Science, 2002.

C. C. Lee, *Fuzzy logic in control systems: Fuzzy logic controller (part i)*, IEEE Transactions on Systems, Man, and Cybernetics, vol.20, no.2, pp.404-418, 1990.

E. H. Mamdani & S. Assilian, *An experiment in linguistic synthesis with a fuzzy logic controller*, Int. J. Man-Machine Studies, vol.7, pp.1-13, 1975.

E. Purwanto, S. Arifin & B-S. So, *Application of Adaptive Neuro Fuzzy Inference System on the Development of the Observer for Speed Sensor Less Introduction Motor*, IEEE Catalogue, no.01ch37239, pp.409-414, 2001.

H. Hong & L. Jian-Hua, *Fingerprint Matching Using ANFIS*, Shangai Jiaotong University, IEEE, pp.217-222, 2003.

J. A. D. Rezende & A. L. Maitelli, *Um Estudo Comparativo Entre Diferentes Técni-*

cas de Otimização do Treinamento de Neurocontroladores, Congresso Brasileiro de Redes Neurais, São José dos Campos-SP, 1999a.

J. A. D. Rezende & A. L. Maitelli, *Um Esquema de Neurocontrole com Treinamento em Tempo Real Aplicado ao Posicionamento de Um Servomotor*, Simpósio Brasileiro de Automação Inteligente, São Paulo-SP, 1999b.

J. Apkarian, *A Comprehensive and Modular Laboratory for Control System Design and Implementation*, Ed.Quanser Consulting, 1995.

J. Goodman, D. Heckerman & R. Rounthwaite, *GUERRA ANTI-SPAM*, Scientific American Brasil, no.36, pp.84-91, 2005.

J. P. B. M. Oliveira. *Review of Auto-tuning Techniques for Industrial PI Controllers*, Dissertação de Mestrado, University of Salford, 1994.

J. R. Jang & Chuen-Tsai S., *Neuro-Fuzzy Modeling and Control*, Proceedings of the IEEE, vol.83, no.3, pp.378-406, 1995.

J. R. Jang, Chuen-Tsai S. & E. Mizutani, *Neuro-Fuzzy and Soft Computing, A Computational Approach to Learn and Machine Intelligence*, Prentice-Hall, 1997.

L. A. Zadeh, *Fuzzy sets*, Information and Control, vol.8, pp.338-353, 1965.

L. S. Coelho & A. A. R. Coelho. *Algoritmos Evolutivos em Identificação e Controle de Processos: uma Visão Integrada e Perspectivas*, SBA Controle & Automação, Vol.10, no.01, 1999.

M. C. Rodrigues, A. L. Maitelli & F. M. Araújo, *Controle Neuro-Fuzzy com Treinamento em Tempo Real Aplicado a um Sistema Ball and Beam*, Congresso Brasileiro de Automática, Gramado-RS, 2004.

M. I. Berto, F. R. Sá & V. Silveira. *Avaliação de Controle PID Adaptativo para um Sistema de Aquecimento Resistivo de Água*, Ciênc. Tecnol. Aliment., Campinas, 2004.

M. Sugeno & G. T. Kang, *Structure Identification of Fuzzy Model*, Fuzzy Sets and Systems, vol.28, pp.15-33, 1988.

O. G. Filho, *Contribuições à Análise de Robustez de Sistemas de Controle Usando Redes Neurais*, Tese de Doutorado, PPgEE-UFRN, 2004.

R. S. Sutton & A. G. Barto, *Reinforcement Learning, An Introduction*, Bradford Book, 1998.

S. Haykin, *Redes Neurais Artificiais: Princípios e prática*, 2^0 edição, Ed.Bookman, 2001.

S. Sandri & C. Correa, *Lógica Nebulosa*, V Escola de Redes Neurais, ITA-SP, pp:c073-c090, 1999.

T. G. B. Amaral & M. M. Crisóstomo, *Neuro-Fuzzy Controller for Helicopter Motion Control*, International Fuzzy Sistems Conference, IEEE, pp.594-597, 2001.

T. Takagi & M. Sugeno, *Fuzzy identification of systems and its applications to modeling and control*, vol.15, pp.116-132, 1985.

Y. Tsukamoto, *An Approach to fuzzy reasoning method, Advances in Fuzzy Set Theory and applications*, Madan M. Gupta, Rammohan K. Ragade and Ronald R. Yager, Ed. Amsterdam: North-Holland, pp.137-149, 1979.