Pontifícia Universidade Católica
do Rio de Janeiro

Eleazar Cristian Mejia Sanchez

Controle por Aprendizado Acelerado e Neuro-Fuzzy de Sistemas Servo - Hidráulicos de Alta Frequência

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Mecânica da PUC - Rio.

Orientador: Prof. Marco Antonio Meggiolaro

Rio de Janeiro, Setembro de 2009

Eleazar Cristian Mejia Sanchez

Controle por Aprendizado Acelerado e Neuro-Fuzzy de Sistemas Servo - Hidráulicos de Alta Frequência

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Mecânica da PUC - Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Marco Antonio Meggiolaro
Orientador
Departamento de Engenharia Mecânica PUC - Rio

Prof. Jaime Tupiassu Pinho de Castro
Departamento de Engenharia Mecânica PUC - Rio

Prof.ª Karla Tereza Figueiredo Leite
Departamento de Engenharia Elétrica - PUC - Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico - PUC - Rio

Rio de Janeiro, Setembro 2009.

Eleazar Cristian Mejia Sanchez

Graduou-se em Engenharia Mecatrônica Universidad Nacional de Ingenieria Lima - Peru em 2005.

Ficha Catalográfica

Mejia Sanchez, Eleazar Cristian.

Controle por aprendizado acelerado e neuro-fuzzy de sistemas servo-hidráulicos de alta frequência / Eleazar Cristian Mejia Sanchez ; orientador: Marco Antonio Meggiolaro. – 2009.

130 f. : il.(color.) ; 30 cm

Dissertação (Mestrado em Engenharia Mecânica)– Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009.

Inclui bibliografia.

1. Engenharia mecânica – Teses. 2. Lógica fuzzy. 3. Sistemas de neuro-fuzzy. 4. Redes neurais. 5. Controle por aprendizado neuro-fuzzy. I. Meggiolaro, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. III. Título.

CDD: 621

Ao Senhor Jesus Cristo, meu pai Jesus Israel, minha mãe Agripina, meu irmão Ronald, e minha irmã Liliana, meus avós Marcos e Juana, e meus queridos amigos.

Agradecimentos

Ao Professor Marco Antonio Meggiolaro pela paciência e orientação durante o desenvolvimento do curso de mestrado.

Agradecemos à Professora Karla Tereza Figueiredo Leite por toda a valiosa contribuição na concepção do controle proposto.

Ao Professor Jaime Tupiassú Pinho de Castro pelas sábias sugestões.

Ao meu grande amigo Juan Gerardo Castillo Alva, pela ajuda na parte experimental em todo momento.

A Marco Perez, Jesus Leal, Jorge Hinostroza e Gilmar amigos e colegas de laboratório.

Aos meus amigos Hernan zambrano, David Achanccaray, Nilton Anchayhua, Edwin Campos, Cesar Mamani, Josue e Carlos.

Aos professores da PUC - Rio pelo ensino.

Ao Departamento de Engenharia Mecânica da PUC - Rio e seus funcionários, pela colaboração comigo.

A CNPq Conselho Nacional de Desenvolvimento Cientifico e Tecnológico pela ajuda financeira.

A todas aquelas pessoas que de alguma outra forma participaram no desenvolvimento da dissertação.

Resumo

Mejía Sánchez, Eleazar Cristian; Meggiolaro, Marco Antonio. **Controle por Aprendizado Acelerado e Neuro-Fuzzy de Sistemas Servo-Hidráulicos de Alta Frequência.** Rio de Janeiro, 2009. 130p. Dissertação de Mestrado - Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Nesta dissertação foram desenvolvidas técnicas de controle por aprendizado acelerado e Neuro-Fuzzy, aplicadas em um sistema servo-hidráulico para ensaio de fadiga. Este sistema tem o propósito de fazer ensaios em materiais para prever a resistência à fadiga dos materiais. O trabalho envolveu quatro etapas principais: levantamento bibliográfico, desenvolvimento de um controle por aprendizado acelerado, desenvolvimento de um controle por aprendizado Neuro-Fuzzy, e implementação experimental dos modelos de controle por aprendizado proposto em uma máquina de ensaios de materiais. A implementação do controle por aprendizado acelerado foi feita a partir do modelo de controle desenvolvido por Alva[5], com o objetivo de acelerar o processo de aprendizagem. Esta metodologia consiste em fazer um controle do tipo *bang-bang*, restringindo a servo-válvula a trabalhar sempre em seus limites extremos de operação, i.e., procurando mantê-la sempre completamente aberta em uma ou outra direção. Para manter a servo-válvula trabalhando em seus limites de seu funcionamento, os instantes ótimos para as reversões são obtidos pelo algoritmo de aprendizado, e armazenados em tabelas específicas para cada tipo de carregamento. Estes pontos de reversão dependem de diversos fatores, como a amplitude e carga média da solicitação, e são influenciados pela dinâmica do sistema. Na metodologia proposta, a lei de aprendizado inclui um termo de momentum que permite acelerar a aprendizagem dos valores das tabelas constantemente durante a execução dos testes, melhorando a resposta a cada evento. O desenvolvimento de um controle por aprendizado Neuro-Fuzzy foi motivado pela necessidade de ter um agente com a capacidade de aprendizado e armazenamento dos pontos ótimos de reversão. Este modelo de controle também consiste na implementação de um controle do tipo *bang-bang*, trabalhando com a servo-válvula sempre nos seus limites extremos de operação. O instante de reversão é determinado pelo sistema Neuro-Fuzzy, o qual tem como entradas a gama (dobro da amplitude) e o valor mínimo do carregamento solicitado. O processo de aprendizado é feito pelas

atualizações dos pesos do sistema Neuro-Fuzzy, baseado nos erros obtidos durante a execução dos testes, melhorando a resposta do sistema a cada evento. A validação experimental dos modelos propostos é feita em uma máquina servo-hidráulica de ensaios de fadiga. Para este fim, o algoritmo de controle proposto foi implementado em tempo real em um módulo de controle *CompactRIO* da *National Instruments*. Os testes efetuados demonstraram a eficiência da metodologia proposta.

Palavras–chave

Sistemas Servo-hidráulicos; Controle de Alta Frequência; Controle por Aprendizado Acelerado; Controle Neuro-Fuzzy; Maquina de Testes de Fadiga.

Abstract

Mejía Sánchez, Eleazar Cristian; Meggiolaro, Marco Antonio (Advisor). **Accelerated Learning and Neuro-Fuzzy Control of High Frequency Servo-Hydraulic Systems.** Rio de Janeiro, 2009. 130p. M.Sc. Dissertation – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica de Rio de Janeiro.

In this thesis, accelerated learning and Neuro-Fuzzy control techniques were developed and applied to a servo-hydraulic system used in fatigue tests. This work involved four main stages: literature review, development of an accelerated learning control, development of a Neuro-Fuzzy control, and implementation of the learning control models into a fatigue testing machine. The accelerated learning control was implemented based on a learning control developed in previous works, introducing a faster learning law. Both learning control methodologies consist on implementing a bang-bang control, forcing the servo-valve to always work in its operational limits. As the servo-valve works in its operational limits, the reversion points to achieve every peak or valley in the desired history are obtained by the learning algorithm, and stored in a specific table for each combination of minimum and mean load. The servo-valve reversion points depend on a few factors, such as alternate and mean loading components, while they are as well influenced by the system dynamics. In the proposed accelerated methodology, the learning law includes one momentum term that allows to speed up the learning process of the table cell values during the execution of the tests. The developed Neuro-Fuzzy control also consists on a bang-bang control, making the servo-valve work in its operational limits. However, here the instant of each reversion is determined by the Neuro-Fuzzy system, which has the load range and minimum load required as inputs. The learning process is made by the update of the Neuro-Fuzzy system weights, based on the errors obtained during the execution of the test.The experimental validation of the proposed models was made using a servo-hydraulic testing machine. The control algorithm was implemented in real time in a C-RIO computational system. The tests demonstrated the efficiency of the proposed methodology.

Keywords

Servo-Hydraulic Systems; High Frequency Control; Accelerated Learning Control; Neuro-Fuzzy Control; Fatigue Testing Machine.

Sumário

Lista de figuras

Lista de tabelas

1.
Introdução

1.1.Objetivo

O objetivo desta dissertação é desenvolver um sistema de controle por aprendizado acelerado e Neuro-Fuzzy baseado em técnicas de inteligência computacional para sistemas servo-hidráulicos de alta frequência. Também é avaliado o desempenho do aprendizado dos modelos propostos. Estes sistemas são implementados em uma máquina para ensaios de fadiga, e avaliado seus desempenho.

1.2.Considerações Iniciais

Os avanços tecnológicos e as inovações industriais fazem uso de sistemas hidráulicos e pneumáticos em muitas aplicações como, por exemplo, indústrias automatizadas, exploração de minérios, maquinaria pesada, ensaios de materiais, indústria aeroespacial e marítima.

Estes sistemas têm muitas vantagens como alta durabilidade e capacidade para produzir forças a altas velocidades. Infelizmente estes sistemas têm características não lineares [1], as quais surgem da compressibilidade do fluido hidráulico, atrito no cilindro hidráulico, etc.

As máquinas para ensaios de materiais são desenvolvidas com sistemas servo-hidráulicos para poderem gerar forças e torques relativamente altos, assim como altas frequências de trabalho. O propósito de fazer estes ensaios é fornecer ao projetista e pessoal de manutenção as características do material para prever a vida útil de um equipamento ou estrutura.

O estudo da fadiga é importante porque a maioria das falhas dos elementos de máquinas em serviço se deve à fadiga. E a ruptura por fadiga ocorre sem nenhum aviso prévio.

Uma máquina para ensaios de fadiga é um equipamento capaz de submeter corpos de prova a esforços cíclicos, a fim de poder prever o comportamento dos mesmos tanto em condições críticas quanto em situações normais de trabalho.

É necessário desenvolver um controle eficiente para que o sistema possa atingir frequências típicas de trabalho, que para o caso dos ensaios em metais é de até cem vezes por segundo. Além disso, o controlador tem que ser capaz de superar as variações não lineares da dinâmica do sistema.

Devido à não linearidade do sistema, o melhor desempenho não é atingido com um controle clássico, isto devido a que toda a informação do processo não é conhecida a *priori*. Uma alternativa de projeto ótimo é baseada em um controlador que seja capaz de fazer a identificação e controle de sistemas dinâmicos não lineares.

Neste trabalho foi desenvolvida a otimização da metodologia de controle de uma máquina servo-hidráulica utilizando técnicas de inteligência artificial tais como: Redes neurais, Lógica fuzzy e sistemas híbridos Neuro-Fuzzy. Foi implementado um controle por aprendizado acelerado e um controle por aprendizado Neuro-Fuzzy, e as simulações computacionais do sistema para carregamentos de amplitude constante e variável. Finalmente, a verificação experimental foi feita nas máquinas servo-hidráulicas do Laboratório de Fadiga da PUC-Rio.

1.3.Motivação

Os ensaios de materiais são realizados com o propósito de obter informação sobre os limites de tensão e de tempo de uso de uma peça ou elemento de máquina, com a finalidade de desenvolver novos tipos de materiais, processos de fabricação, ou tratamento de materiais para definir suas aplicações industriais.

Os ensaios de fadiga são um tipo de ensaio de importância fundamental nas indústrias para prever possíveis falhas, fazer um programa de manutenção, e aos fabricantes determinar os materiais adequados que possam suportar os esforços repetitivos requeridos pela tarefa do mecanismo. O ensaio de fadiga consiste em

aplicar a uma peça ou corpo de prova apropriado esforços cíclicos repetitivos, dependendo do tipo de ensaio a ser realizado; por exemplo, os ensaios de iniciação e de propagação de trincas.

Os aparelhos de ensaio de fadiga são constituídos por um sistema de aplicação de cargas, que permite alterar a intensidade e o sentido do esforço, e por um contador de número de ciclos. A Figura 1.1 apresenta um ensaio de iniciação de trincas por fadiga, o qual fornece dados quantitativos das características do material antes da ruptura.

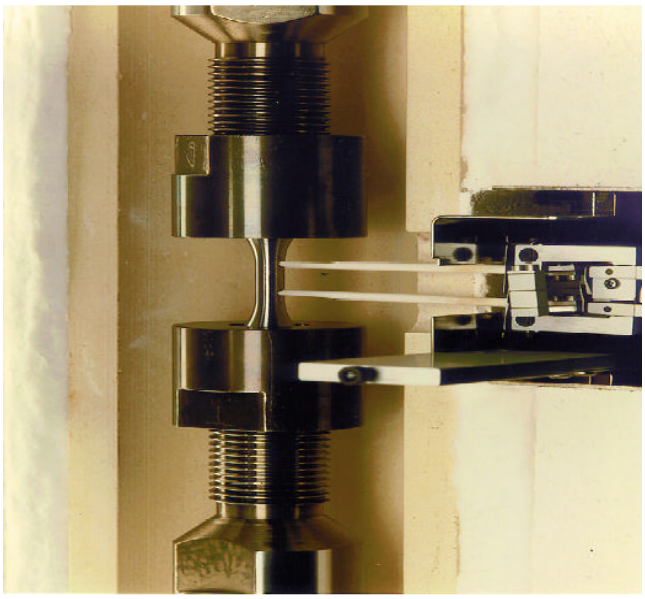


Figura 1.1. Ensaio de Fadiga.

Os ensaios de fadiga são quase independentes da frequência de trabalho, exceto em polímeros, onde a altas frequências o material é aquecido, tendo influência na resistência útil do material; mas se o corpo de prova puder ser resfriado, nesse caso a frequência não afetaria os resultados. Para um tipo de material dado e submetido a tensões alternadas, a resistência útil à fadiga basicamente só depende do número de ciclos aplicados ao material de prova, é por este motivo que os aparelhos de ensaios de materiais trabalham a altas frequências, procurando uma redução do tempo e custo dos ensaios, sem que isto interfira nos resultados.

As técnicas de inteligência artificial tais como: Redes Neurais, Lógica Fuzzy e sistemas híbridos Neuro-Fuzzy têm a capacidade de modelar uma grande

classe de plantas de dinâmica não linear com um arbitrário grau de precisão; isto permite projetar um esquema de controle ótimo aplicado a sistemas servo-hidráulicos, sem a necessidade de ter conhecimento de todo o modelo da planta.

1.4.Revisão Bibliográfica

O controle de sistemas servo-hidráulicos apresenta muitos trabalhos de pesquisa aplicados a manipuladores industriais que desempenham tarefas repetitivas. Um esquema de controle por redes neurais não requer a computação da dinâmica não-linear dos manipuladores, somente utiliza sinais medidos localmente para fazer a aprendizagem do comportamento do sistema através da atualização dos pesos da rede que minimizem o erro entre a saída desejada e real.

Nos últimos anos, sistemas baseados em redes neurais e fuzzy foram encontrando um caminho em aplicações em controle e muitas outras áreas da engenharia. Os resultados obtidos capturaram a atenção de engenheiros que trabalham com sistemas reais. Isto é devido principalmente aos resultados obtidos e freqüentemente à facilidade de implementação quando se desenvolvem sistemas de controle baseados em redes neurais e fuzzy [2].

Outro trabalho [3] apresentou o uso de um controlador por aprendizado, utilizando técnicas Neuro-Fuzzy no projeto de um controle de trajetória, modelando o atuador eletro-hidráulico.

Outra aplicação utiliza um controle híbrido adaptativo Neuro-Fuzzy por modelo de referência ("*Adaptive Neuro-Fuzzy Model Reference Controller*", *ANFMRC*) [4], e foi desenvolvido para melhorar o desempenho do controle em um sistema pneumático, este controle híbrido é combinado com um controle "*bang-bang*" aplicado quando o erro é alto, e um *ANFMRC* aplicado quando o erro é pequeno.

Na teses de dissertação de Alva [5], foi desenvolvido um controle por aprendizado, restringindo a servo-válvula a trabalhar sempre em seus limites extremos de operação "*bang-bang*". Nesse caso, os pontos de reversão da válvula sempre devem ficar entre o vale e pico de um carregamento (ou entre pico e vale),

e este parâmetro depende da amplitude e do menor valor da carga solicitada. Para que a servo-válvula trabalhe em seus limites de funcionamento, uma lei de aprendizado obtém os instantes ótimos para as reversões. E depois é armazenada em tabelas específicas para cada tipo de carregamento, e atualizada continuamente.

As principais máquinas para testes de materiais possuem sistemas servo-hidráulicos. As principais marcas que encontramos no mercado são a INSTRON e a MTS, com alguns modelos capazes de trabalhar com células de carga desde 5 kN a 500 kN, a uma frequência máxima teórica de 500 Hz (para amplitudes muito pequenas). Elas podem executar testes de tração, compressão, flexão e de fadiga. Têm a habilidade de testar os mais diversos materiais, incluindo polímeros, metais e compósitos. Todas as máquinas vêm equipadas com um controlador que usa um controle de malha fechada como se mostra na Figura 1.2.

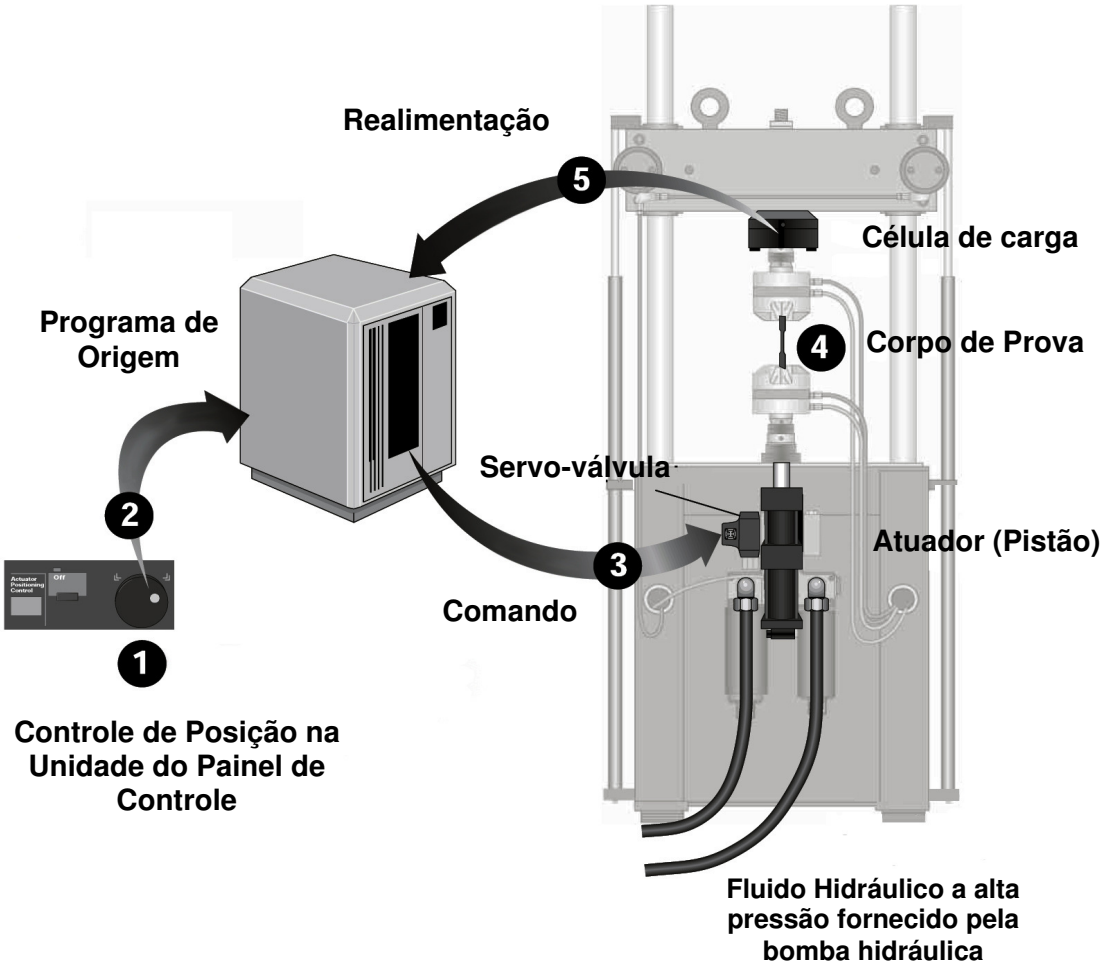


Figura 1.2. Etapas do sistema de controle das Máquinas de Ensaios do Laboratório de Fadiga da PUC-Rio [6].

1.5.Roteiro da Dissertação

Esta dissertação foi dividida em oito capítulos. A seguir estão listados os tópicos principais de cada capítulo.

- Capítulo 1: Introdução, onde se faz o resumo da pesquisa bibliográfica na qual se sustenta parte do estudo feito.

- Capítulo 2: São apresentados os conceitos básicos dos sistemas fuzzy.

- Capítulo 3: Apresentam-se os conceitos básicos das Redes neurais.

- Capítulo 4: Apresenta-se uma descrição geral dos sistemas híbridos Neuro-Fuzzy. É feita a modelagem do sistema servo-hidráulico.

- Capítulo 5: É apresentado o resumo da modelagem dos sistemas servo-hidráulico, e é feita a simulação do controle por aprendizado acelerado.

- Capítulo 6: Apresenta-se a simulação do controle por aprendizado Neuro-Fuzzy, e compara-se o desempenho do aprendizado dos modelos propostos.

- Capítulo 7: São feitos os experimentos em laboratório com a máquina servo-hidráulica, e a apresentação dos resultados.

- Capítulo 8: São apresentadas as conclusões sobre as vantagens do sistema de controle por aprendizado proposto.

2
Lógica Fuzzy

2.1. Introdução

A lógica fuzzy é uma extensão da lógica booleana, introduzida pelo Dr. Lofti Zadeh da Universidade da Califórnia / Berkeley no ano 1965. Foi desenvolvida para expressar o conceito de verdade parcial, de maneira que se possam determinar valores entre o limite "completamente verdadeiro" e "completamente falso". Isto significa que um valor lógico difuso é um valor qualquer no intervalo de 0 a 1. A lógica fuzzy torna-se importante na medida em que o mundo em que vivemos não é constituído por fatos absolutamente verdadeiros ou falsos.

Embora as técnicas de controle possam ser implementadas por modelos matemáticos determinísticos, as implementações baseada na lógica fuzzy freqüentemente têm um melhor desempenho, pelos seguintes pontos [7]:

- As estratégias de controle fuzzy imitam um comportamento baseado em regras vindas da experiência do especialista, em vez de um controle explicitamente restrito a modelos matemáticos como equações diferenciais. Portanto, é robusto em sistemas não-lineares sem requerer um modelo matemático.
- O controle fuzzy abrange um grande número de entradas, muito das quais são apenas para condições especiais. Portanto, algumas condições excepcionais (tais como alarmes) podem ser implementadas com um menor esforço computacional e flexível a modificações.
- A implementação de produtos comerciais baseados em estratégias de controle fuzzy destinadas ao mercado devem ser de custo mais baixo, freqüentemente mais eficiente e facilmente implementável em microprocessadores, em comparação a estratégias de controle

convencionais. Isto é devido a uma menor codificação e tempo computacional de execução.

A lógica fuzzy tem a capacidade de incorporar a forma humana de pensar em sistemas de controle. Dessa forma, o controlador fuzzy comporta-se conforme o raciocínio que o especialista utiliza para inferir as regras, baseadas nas informações que eles já conhecem.

A lógica fuzzy é uma variação da lógica booleana, que só apresenta os valores de "0" e "1", sem nenhum termo médio. Entretanto, na lógica fuzzy, uma premissa pode assumir valores de pertinência (grau de verdade) intermediários. Assim, é possível descrever grandezas imprecisas como: altura (alto, baixo), quantidade (muito, razoável, pouco), idade (jovem, velho), etc. Em sentido mais amplo, a lógica fuzzy é quase sinônimo da teoria de conjuntos fuzzy, uma teoria na qual os elementos têm um grau parcial de pertinência [8].

2.2. Conjuntos Fuzzy

Na teoria clássica de conjuntos, o conceito de pertinência de um elemento a um conjunto é bem definido, de maneira que para um conjunto A em um universo U, o elemento simplesmente pertence ou não pertence àquele conjunto, como se mostra na seguinte função característica [9]:

$$f_A(x) = \begin{cases} 1 & \text{Se e somente se } \; x \in A \\ 0 & \text{Se e somente se } \; x \in A \end{cases} \tag{2.1}$$

Em um sentido mais amplo, Zadeh propôs a generalização da função característica, de modo que ela possa assumir infinitos valores no intervalo [0,1] e representado por pares ordenados.

$$A = \{\mu_A(x) / x\} \quad x \in U \tag{2.2}$$

Onde: $\mu_A(x)$: Representa o grau de pertinência de x com o conjunto A

A : Conjunto fuzzy.

x : A variável de interesse.

U : Universo de discurso.

Além disso, um elemento pode pertencer a mais de um conjunto fuzzy, com diferente grau de pertinência.

Na Figura 2.1 (a) pode-se observar que se um elemento x for movido em direção aos limites do conjunto A (cor amarela), no ponto de cruzamento ocorrerá repentinamente um degrau no comportamento de sua pertinência, inicialmente de membro para não membro. Também, o grau de pertinência nos limites é indeterminado.

Por outro lado, a lógica fuzzy pode perceber as variações ocorridas nos pontos transição de uma cor para outro. Os conjuntos (faixa de cores) são facilmente representáveis através de linguagem fuzzy (Figura 2.1 b). As funções de pertinência fuzzy podem representar a variação gradual nas tonalidades. Por exemplo, o mesmo elemento x possui um grau de pertinência 0,8 na função de pertinência de cor amarela, e grau de pertinência 0,2 na função de cor vermelha. Na Figura 2.1, o eixo "X" representa o universo de discurso do elemento x e o eixo "Y" representa o grau de pertinência definido entre 0 e 1.

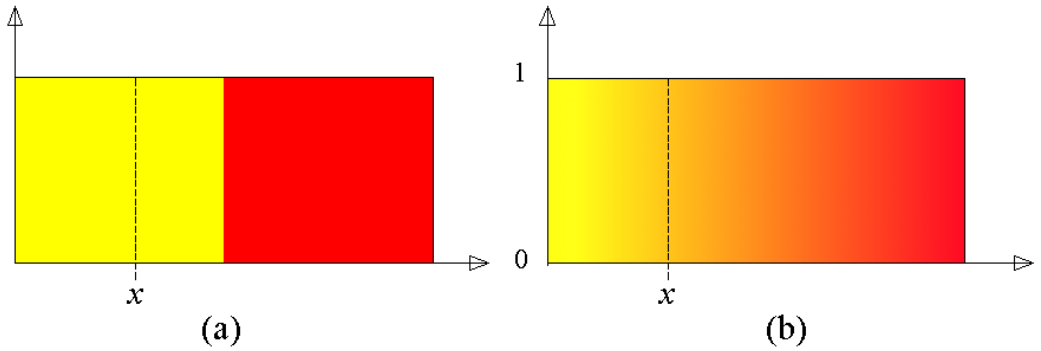


Figura 2.1. (a) lógica booleana (b) lógica fuzzy.

Neste momento, é importante definir o conceito de grau de pertinência. Ele é definido por um número no intervalo de [0,1] que determina "quanto" uma variável pertence a um determinado conjunto. Na lógica booleana, somente existem dois graus de pertinência: 0% se não pertence e 100% se pertence ao conjunto. No entanto, na lógica fuzzy existe uma faixa de valores entre 0% e 100%. O grau de pertinência é descrito pela Equação 2.2 [10].

2.2.1. Variáveis lingüísticas

A *variável lingüística* é uma variável que possui valores que não são números, mas sim palavras ou frases na linguagem natural. Elas são os nomes dos conjuntos fuzzy, os quais são representados por meio de funções de pertinência. Por exemplo, a *velocidade* de um pistão hidráulico pode ser uma variável lingüística com valores: *baixa*, *média* e *alta*, conforme mostrado na Figura 2.2.

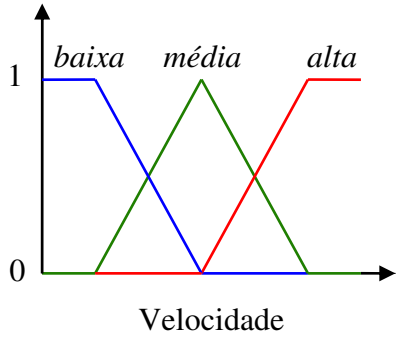


Figura 2.2. Variável lingüística

As variáveis lingüísticas têm a função de fornecer uma forma sistemática para as descrições aproximadas dos fenômenos complexos ou mal definidas, utilizando um tipo de descrição lingüística similar ao empregado pelos seres humanos. Isto permite o tratamento de sistemas muito complexos para serem analisados através de cálculos matemáticos.

Generalizando, as variáveis lingüísticas podem ser sentenças com um formato de linguagem padrão ou especificada pelo usuário, construídas a partir de termos primários (alto, baixo, grande, etc.), de modificadores (muito, pouco, levemente, etc.), de delimitadores (acima, abaixo, etc.) e conectores lógicos (não, e, e ou). Uma variável lingüística é formalmente caracterizada pela quíntupla (N, T(N), U, G, M), onde:

N : Nome da variável.

T(N) : Conjunto de nomes dos valores lingüísticos de N.

U : Universo de discurso.

G : Regra sintática para gerar os valores de N como uma composição de termos de T(N).

M : Regra semântica, para associar a cada valor gerado por G um conjunto fuzzy em U.

2.2.2. Função de Pertinência.

A função de pertinência (*FP*) é uma função numérica que atribui valores de pertinência fuzzy para valores discretos de uma variável, em seu universo de discurso [8]. Estas funções de pertinência podem ter diferentes formas, as quais dependem do critério utilizado pelo especialista para representar o contexto em que serão utilizadas. Por exemplo, considere-se a variável lingüística idade (de pessoas), constituídas pelos termos lingüísticos; T(idade) = {jovem, media, velho} correspondentes aos conjuntos fuzzy A, B e C, e cada uma definido por uma função de pertinência. Uma possível representação das funções de pertinência é mostrada na Figura 2.3.

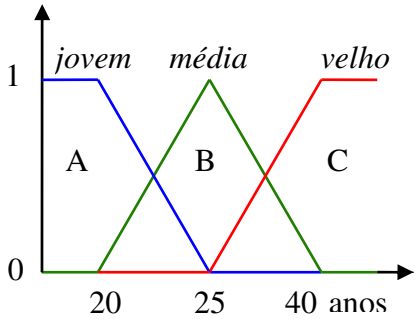


Figura 2.3. Funções de pertinência.

Na figura acima, a idade até 20 anos apresenta um grau de pertinência igual a 1 no conjunto A, à medida que a idade aumenta o grau de pertinência neste conjunto decresce; para uma idade de 25 anos é totalmente incompatível no conjunto A e compatível no conjunto B, e uma idade acima de 40 anos apresenta grau de pertinência diferente de 0 em C.

Observe-se que, na Figura 2.3, o contexto é importante na definição das funções de pertinência e de sua distribuição ao longo de seu universo de discurso, as quais são determinadas pelas noções que podem ter diferentes pessoas do contexto. Além disso, a forma (triangular, trapezoidal, gaussiana, etc.) das funções de pertinência pode ser definida baseada na experiência e perspectiva do especialista. Na prática, as formas escolhidas podem ser alteradas em função dos resultados obtidos.

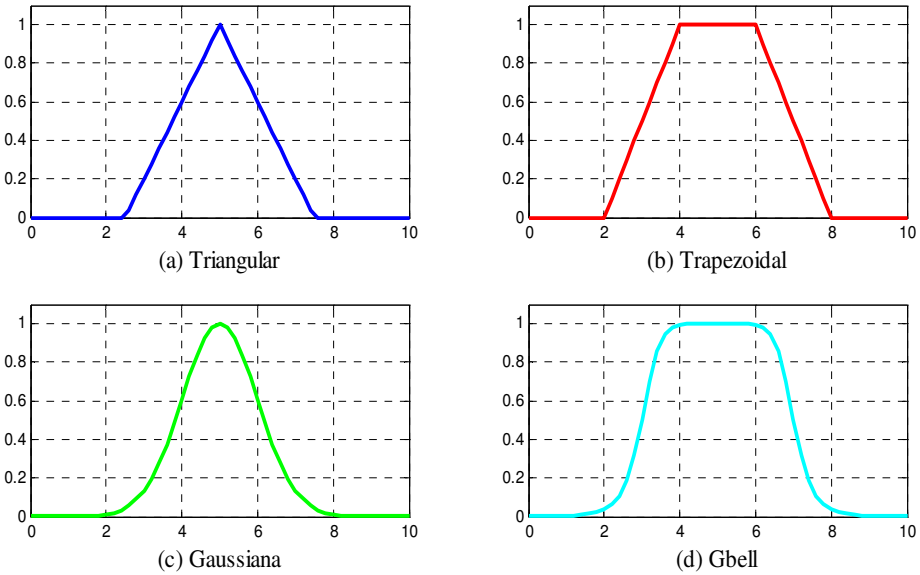


Figura 2.4. Funções de pertinência: (a) Triangular (b) Trapezoidal (c) Sigmoidal (d) função de Bell.

Algumas recomendações práticas podem ser mencionadas como segue [7].

- O número de funções de pertinência por variável deve estar entre 2 e 7; com um maior número de conjuntos, se tem uma maior precisão, mas a demanda computacional também é maior; e para valores muito maiores, não há melhorias significativas.
- Os formatos mais freqüentemente encontrados são os triangulares e trapezoidais, pela facilidade de serem gerados. Em aplicações onde se requer um desempenho suave de importância crítica, podem ser usadas funções tipo sigmóides e *spline*.
- Outro fator que afeta a precisão é o grau de superposição das funções de pertinência; experimentalmente foram determinados como adequados valores na faixa de 25 % até 75%.

2.2.3. Operações entre Conjuntos Fuzzy

As operações que podem ser feitas com conjuntos fuzzy são baseadas na teoria clássica de conjuntos. Algumas das operações elementares mais utilizadas são: união, interseção, e complemento, as quais são definidas pelas equações (2.3), (2.4) e (2.5) respectivamente e mostradas na Figura 2.4 [9, 11, 12].

União : $$\mu_{(A\cap B)}(x) = \max(\mu_{(A)}(x), \mu_{(B)}(x)) \quad (2.3)$$

Interseção : $$\mu_{(A\cap B)}(x) = \min(\mu_{(A)}(x), \mu_{(B)}(x)) \quad (2.4)$$

Complemento : $$\mu_{(\bar{A})}(x) = 1 - \mu_{(A)}(x) \quad (2.5)$$

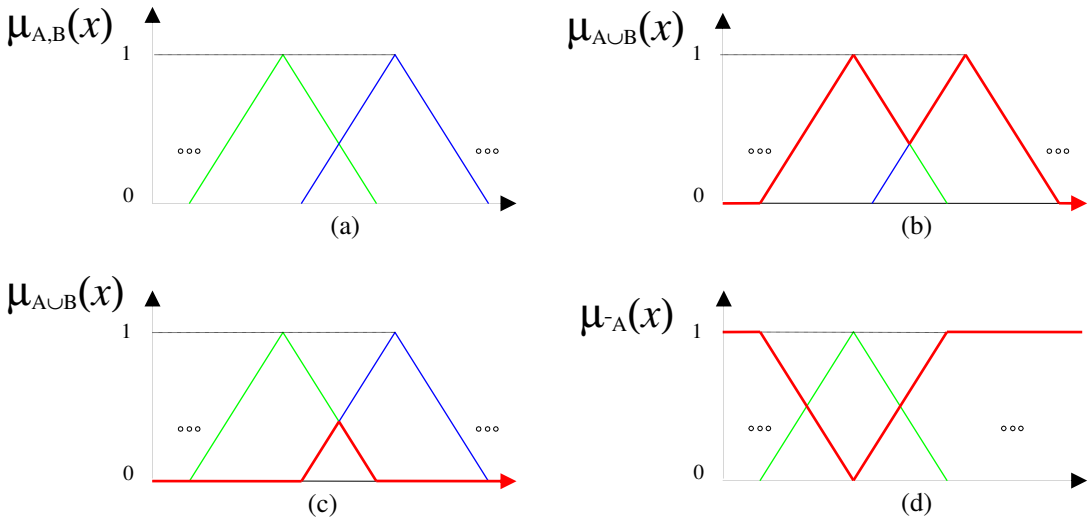


Figura 2.4. Operações entre conjuntos fuzzy: (a) Conjuntos A e B, (b) União, (c) Interseção e (d) Complemento. [2].

Embora a união e interseção possam ser descritas também por meio de outros operadores, Zadeh os representou com os operadores *min* e *max*. Alem disso, sugeriu a soma algébrica para a união fuzzy e o produto algébrico para a interseção fuzzy, como se apresenta na Equação (2.6) e (2.7); posteriormente, com o objetivo de generalizar, foram definidos operadores baseados no conceito de norma triangular (*norma-t*) e co-norma triangular (*co-norma-t* ou *norma-s*).

Soma Algébrica : $$\mu_{(A\cup B)}(x) = \mu_{(A)}(x) + \mu_{(B)}(x) - \mu_{(A)}(x).\mu_{(B)}(x) \quad (2.6)$$

Produto Algébrico : $$\mu_{(A\cap B)}(x) = \mu_{(A)}(x).\mu_{(B)}(x) \quad (2.7)$$

2.3. Sistema de Inferência Fuzzy

O sistema fuzzy é um modelo geral que permite a identificação dos módulos que compõem tal sistema, fornecendo assim a idéia do fluxo de informação dentro do mesmo. Basicamente ele é compostos por 3 etapas, como é mostrado na Figura 2.5, onde estão definidas as funções de cada uma das etapas.

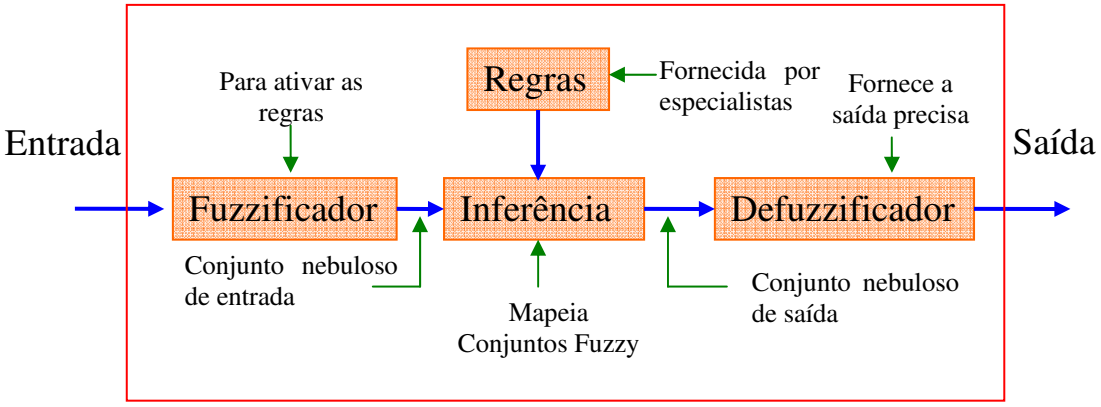


Figura 2.5. Sistema de Inferência Fuzzy.

Na figura acima, considera-se a entrada não fuzzy, e ela é resultado de medições na maioria das aplicações práticas. Portanto, é necessário efetuar-se a conversão destas entradas em uma representação conhecida como conjuntos fuzzy, o que se denomina fuzzificação. Além disso, nesta etapa ocorrem as ativações das regras para as diferentes situações.

Na segunda etapa, estabelece-se a base de regras, como a relação das variáveis de entrada e saída, as quais são obtidas pelo conhecimento e pela experiência do especialista da aplicação. Uma vez obtido o conjunto fuzzy de saída resultante do processo de inferência, é necessário efetuar a interpretação dessa informação, pois nas aplicações práticas são requeridas saídas precisas, o que é realizado na etapa de defuzzificação [9].

2.3.1.Fuzzificação

A fuzzificação é a conversão das entradas exatas (número reais) para o domínio fuzzy. O fuzzificador atribui valores lingüísticos (graus de pertinência) empregando funções de pertinência às variáveis de entrada. Isto se considera como uma etapa de pré-processamento dos sinais de entrada, reduzindo o numero

de valores a ser processado o que significa um menor esforço computacional.

2.3.2. Regras e Inferência Fuzzy

Regra fuzzy são implicações lógicas que relacionam os conjuntos fuzzy de entrada com os de saída. Geralmente são fornecidas por um especialista, em forma de sentenças lingüísticas, constituindo um aspecto fundamental no desempenho de um sistema de inferência fuzzy, como mostrado abaixo.

$$\textbf{Se } x \text{ è } A \textbf{ e } y \text{ é } B \text{, } \textbf{então } z \text{ é } C \text{.}$$

Onde, A e B são os conjuntos fuzzy de entrada, relativos à parte conhecida como antecedentes ou premissas, enquanto C é o conjunto fuzzy de saída, relativo à parte conhecida como conseqüente ou conclusão [8]. Estas regras podem ser definidas previamente ou alternativamente geradas automaticamente a partir de uma base de dados.

Na etapa de inferência, ocorrem as operações dos conjuntos propriamente ditas, como a combinação dos *antecedentes* das regras do tipo $_{SE\ -\ ENTÃO}$, gerando o conjunto de saída fuzzy. Na Figura 2.6, apresenta-se o processo de inferência fuzzy.

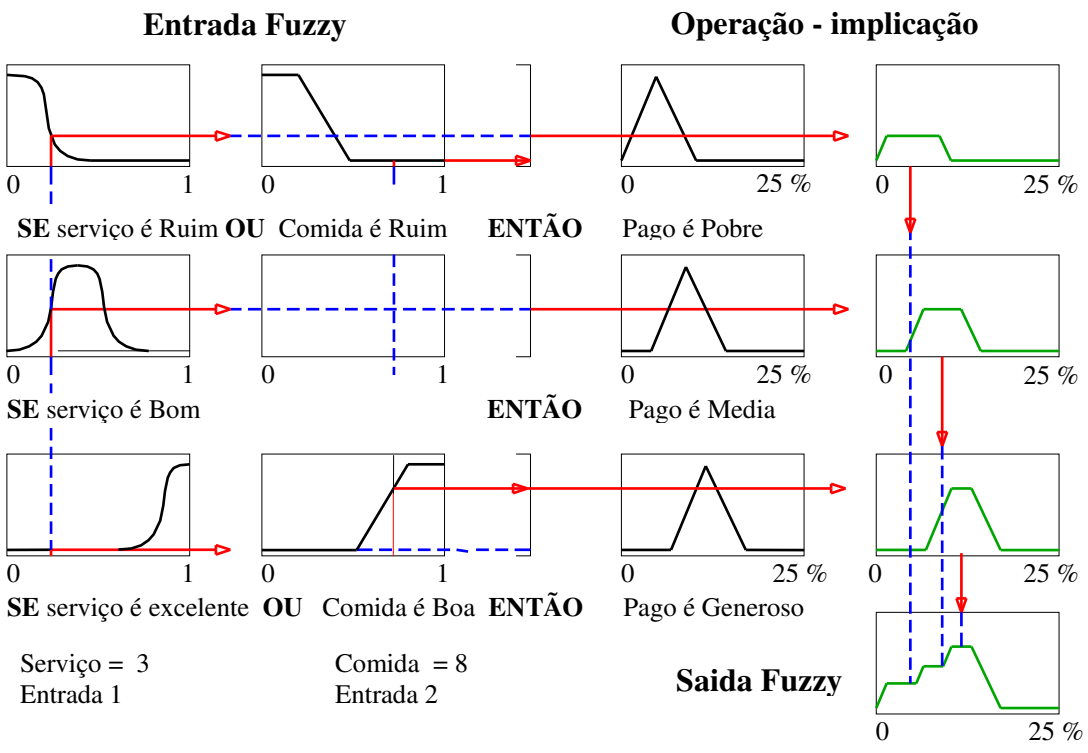


Figura 2.6. Regra e inferência fuzzy [8].

Na figura acima, mostra-se como ocorrem às operações na etapa de inferência; a entrada exata é transformada em entradas fuzzy, e com esses valores calculam-se as operações - implicação e finalmente obtém-se a saída fuzzy.

2.3.3. Defuzzificação.

No processo de defuzzificação, é efetuada a interpretação do conjunto fuzzy de saída inferida pelas regras, com o objetivo de obter um valor numérico. Isto se faz necessário porque em aplicações práticas são requeridas saídas precisas com algum significado físico. Por exemplo, no controle de uma planta utilizando um controlador fuzzy, este deve fornecer à planta sinais precisos, já que apresentar um conjunto fuzzy na entrada da planta não teria significado algum.

Na prática, o método de defuzzificação mais popular é o centro de Área (C-o-A), o qual retorna o centro da área debaixo a curva, como se mostra na Figura 2.7. Outros métodos utilizados são: Centro do Máximo (C-o-M), Média do Máximo (M-o-M) e o Maior do Máximos [13].

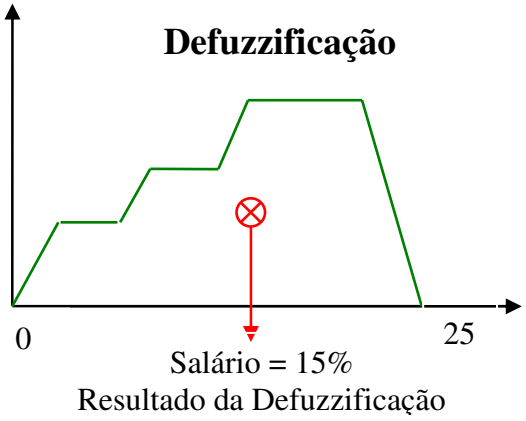


Figura 2.7. Processo de Defuzzificação.

2.4. Tipos de Sistema Fuzzy

Na literatura existem vários modelos de sistemas fuzzy, nos quais geralmente os antecedentes das regras estão formados por conjuntos fuzzy (proposições lingüísticas), e a diferença entre os modelos se dá no "conseqüente" das regras. Algum dos modelos mais conhecidos são o modelo de Mandani e o modelo de Tagaki-Sugeno.

2.4.1. Modelo de Mandani

O modelo de Mandani utiliza conjuntos fuzzy nos "conseqüentes" das regras fuzzy. Neste modelo, a saída da etapa de inferência é representada por um conjunto fuzzy, que é o resultado da agregação das saídas inferida por cada uma das regras, a qual na seguinte etapa gera uma saída exata utilizando um dos métodos de defuzzificação já mencionados.

A característica básica do modelo tipo Mandani é o fato que tanto os antecedentes como os conseqüentes são mapeados com conjuntos fuzzy. Por exemplo, uma regra típica num modelo Mandani é da seguinte forma:

SE *Erro* é "Grande" **E** a *Derivada de Erro* é "Pequena" **Então** *Torque* é "Alto".

No caso em que cada um das regras de inferência tenha mais de uma variável de entrada, é necessário aplicar uma técnica de agregação dos conjuntos antecedentes, que neste caso geralmente é dada pelas t-normas (*min* e *produto*). Além disso, nas aplicações práticas têm-se N regras na etapa de inferência, das quais são gerados N conjuntos conseqüentes, um por cada regra. Para obter o conjunto final de saída da etapa de inferência, é feita a composição dos N conjuntos conseqüentes utilizando a s-norma (*max*) [7- 10].

2.4.2. Modelo de Takagi-Sugeno

No modelo de Takagi-Sugeno (TS), o conseqüente de cada regra é representado em função das variáveis de entrada; e a saída final de todas as regras é determinada pela média ponderada das saídas geradas por cada um das regras. Nesse caso, os coeficientes de ponderação são definidos pelos graus de ativação das respectivas regras.

A fuzzificação das entradas com a aplicação dos operadores fuzzy (operação dos antecedentes) é feita de igual forma que no modelo Mandani, com a diferença que a saída é uma função linear ou constante. Uma regra típica de um modelo Sugeno é da seguinte forma:

SE $Erro = x$ **e** a $Derivada\ de\ Erro = y$ **então** *Torque* é $\tau_i = a.x + b.y + c$.

onde τ_i é o valor de saída de cada um das regras. Além disso, a partir do produto da operação do antecedente de cada regra, obtém-se um peso ω_i, o qual determina o fator de influência de cada regra no resultado final. Por exemplo, na regra mostrada acima, com o tipo de operador *AND*, com $Erro = x$ **e** $Derivada\ de\ Erro = y$ **,** o fator de influência é:

$$\omega_i = AND(f(x), g(y)) \tag{2.8}$$

onde f e g são as funções de pertinência para as entradas *Erro* e *Derivada de Erro.* A saída final é determinada pela equação:

$$out = \frac{\sum_{j=1}^{n} w_i.\tau_i}{\sum_{j=1}^{n} w_i} \tag{2.9}$$

2.5. Vantagens e Desvantagens dos Sistemas Fuzzy

As principais vantagens e desvantagens dos sistemas fuzzy, apresentadas em aplicações práticas, são as seguintes [11]:

2.5.1. Vantagens dos Sistemas Fuzzy

- A capacidade de controlar sistemas com muitas variáveis de saída utilizando um só controlador fuzzy, com um bom desempenho.
- A facilidade de utilizar expressões utilizadas na linguagem natural na elaboração das proposições lingüísticas.
- A habilidade de controlar processos com característica não-linear e de alta ordem, na qual a determinação do modelo matemático e o controle clássico do sistema são muito complexos.
- A facilidade de implementar técnicas de controle baseadas na experiência de um especialista e em aspectos intuitivos, utilizando proposições lingüísticas (regras) e entradas imprecisas.

2.5.2. Desvantagens dos Sistemas Fuzzy

Algumas das limitações que apresentam os sistemas fuzzy são as seguintes:

- A dificuldade de análise de aspectos de optimalidade, estabilidade e robustez.
- A influência da grande quantidade de parâmetros na configuração geralmente feita pelo usuário, algumas das quais são: número de funções de pertinência de cada variável, número de regras, seleção dos métodos de implicação e agregação, método de defuzzificação, assim como os parâmetros de cada função de pertinência.
- Geralmente a precisão do sistema fuzzy é limitada pela experiência do especialista na configuração dos parâmetros, a qual é determinada pelo conhecimento do processo pelo especialista.

No capítulo seguinte apresentam-se os fundamentos teóricos das redes neurais artificiais, o modelo de sua célula básica, sua arquitetura e os algoritmos de aprendizagem.

3
Redes Neurais

3.1. Introdução

As redes neurais artificiais, ou comumente conhecidas como "*Neural Networks*", foram motivadas em princípio pela extraordinária capacidade do cérebro humano para executar tarefas de grande complexidade, não lineares e com processamento paralelo da informação, tais como reconhecimento de padrões, percepção, classificação, generalização de conceitos e controle motor. Estes fatos geraram o interesse do estudo detalhado da constituição do cérebro e sua mimetização na concepção de sistemas com as capacidades acima referidas, designadas por redes neurais artificiais (*ANN*).

As redes neurais (*NN*) estão constituídas por unidades básicas independentes designadas como neurônios (processadores ou nós). Cada unidade possui ligações para outras unidades, as quais se comunicam entre si através de sinapses, formando uma rede de nós, daí o nome de rede neural.

As primeiras informações sobre redes neurais surgiram em 1943, com o primeiro modelo lógico-matemático de um neurônio biológico, que foi desenvolvido pelo neurofisiólogo Warren McCulloch, do *Instituto Tecnológico de Massachusetts*, e do matemático Walter Pitts, da *Universidade de Illinois*. Desenvolveram um modelo matemático de neurônio simulando o comportamento de uma célula nervosa, a qual consistia num modelo de resistores variáveis e amplificadores.

As redes neurais realizam o processamento de informações baseado na organização dos neurônios do cérebro, as quais têm a capacidade de aprender e tomar decisões baseadas na aprendizagem. Uma rede neural é interpretada como uma estrutura neural de organismos inteligentes, capaz de armazenar conhecimento baseado em aprendizagem e da experiência.

Uma rede neural é um processador maciçamente paralelo, distribuído, constituído por unidades de processamento simples, que têm a capacidade de armazenamento de conhecimento experimental e de torná-lo disponível para uso. As redes neurais são semelhantes ao cérebro pelos seguintes aspectos [14]:

- O conhecimento é obtido pela rede neural a partir do ambiente, através do processo de aprendizado.
- A força de conexão entre cada neurônio, conhecida como peso sináptico, é usada para armazenar o conhecimento aprendido.

3.2. Estrutura do neurônio

O neurônio é a unidade fundamental de processamento de informação para a operação de uma *NN*. Neurônios são elementos processadores interligados, trabalhando em paralelo para desempenhar uma determinada tarefa. A Figura 3.1 mostra a estrutura de um neurônio biológico, e na Figura 3.2 apresenta-se o modelo de um neurônio artificial, na qual se desenham formas básicas que constituem uma *NN*. Os elementos básicos identificados no modelo neural são: os *pesos de conexão*, o N*et* e a *Função de ativação* [15].

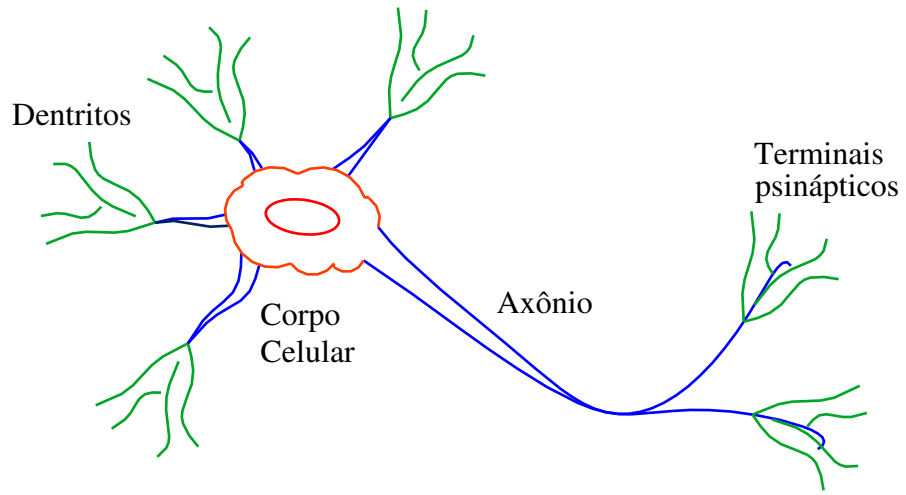


Figura 3.1. Esquema básico do neurônio biológico.

O neurônio recebe múltiplos sinais de outros neurônios através de seus dentritos, e cada um destes sinais são multiplicados pelo próprio peso da conexão. Estes sinais são adicionados no corpo celular ou função somatória, e quando este sinal composto alcança um valor umbral, um sinal potencial é enviado pelo axônio, o qual é a saída do neurônio [14].

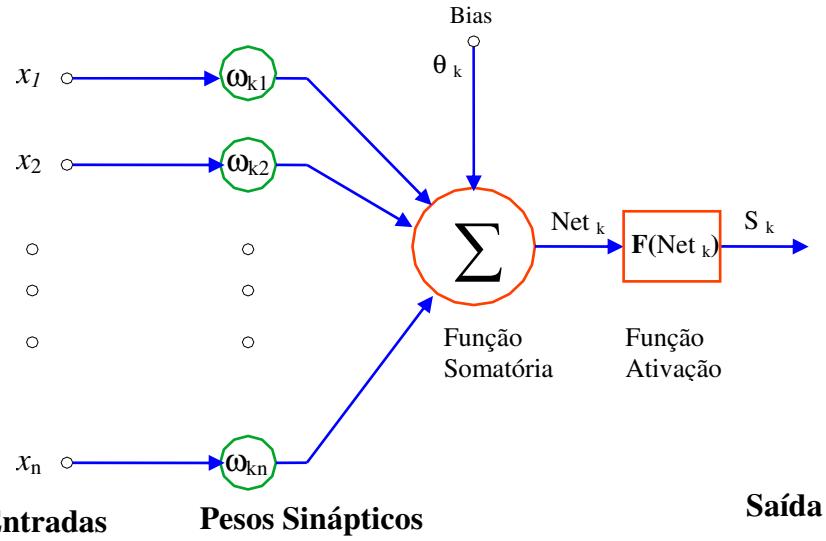


Figura 3.2. Modelo básico de um neurônio Artificial ”Perceptron”.

onde x são as entradas, ω_{kn} são os pesos ou parâmetros da rede, os quais representam a memória da rede, θ_k é conhecido como o termo polarizador (bias), Net_k representa a combinação linear dos pesos, e S_k representa a saída do neurônio.

O modelo do neurônio da Figura 3.2, além das entradas, também tem um termo constante chamado “bias” θ_k, o qual tem a função de aumentar ou diminuir a entrada Net_k da função de ativação, modificando este positiva ou negativamente.

3.2.1. Peso de conexão

O valor de sinapses, ou conexão, é caracterizado por um peso. Especificamente, o sinal de entrada x_j da sinapse j conectada para o neurônio k é multiplicado pelo peso sináptico ω_{kj}. O primeiro sub-índice faz referência ao neurônio, e o segundo faz referência à entrada.

3.2.2. Função Somatório (Net)

A *Net* é definida como uma função soma, e executa o somatório dos sinais produzidos pelo produto entre os sinais de entrada e os pesos de conexão

(sinapses) de cada neurônio. Esta operação representa uma combinação linear:

$$Net_k = \sum_{j=1}^{p} \omega_{kj}.x_j + \theta_k \tag{3.1}$$

3.2.3. Função de Ativação

A função de ativação é uma função que determina o nível de ativação do neurônio artificial, o qual limita a amplitude do sinal de saída do neurônio para uma faixa de valores finitos. Geralmente, escreve-se a faixa de amplitude normalizada da saída de um neurônio como um intervalo fechado [0,1] ou [-1,1] [14][16].

- **Função Degrau**

  Modelo proposto por McCulloch e Pits[1943], esta função de ativação modela a característica de "todo - ou- nada" do neurônio, e é expressa pela equação:

$$F(Net_k) = \begin{cases} 1 & \text{Se} \quad Net_k \geq 0 \\ 0 & \text{Se} \quad Net_k < 0 \end{cases} \tag{3.2}$$

  Na Figura 3.3(a), apresenta-se a função degrau.

- **Função de Ativação Linear**

  A função linear é apresentada na Figura 3.3(b), onde o fator de amplificação dentro da região de operação é assumido unitário. Portanto, esta função é considerada como uma aproximação para um amplificador não linear, e é apresentada pela equação:

$$F(Net_k) = \begin{cases} 1 & \text{Se} \quad 0,5 \leq Net_k \\ Net_k & \text{Se} \quad -0,5 < Net_k < 0,5 \\ 0 & \text{Se} \quad Net_k < -0,5 \end{cases} \tag{3.3}$$

- **Função de Ativação Logsig**

  A função *logsig* é a função de ativação mais comumente utilizada na construção de *ANN*. É definida como uma função estritamente crescente, que apresenta um equilíbrio entre o desempenho linear e não-linear, e definida por:

$$F(Net_k) = \frac{1}{1+e^{-a.Net_k}} \tag{3.4}$$

onde a é o parâmetro de inclinação da função *logsig* e Net_k é o valor de ativação do neurônio. Na Figura 3.3(c), apresenta-se a função *logsig*, e observa-se que à medida que o parâmetro a aumenta, tendendo-o ao infinito, esta função comporta-se como uma função de grau.

- **Função de Ativação Tansig**

  Na Figura 3.3 (d) apresenta-se a função *Tansig*. Esta função também possui a forma sigmoidal, e é a segunda função mais utilizada na construção de *ANN*. A principal diferença com relação à função *Logsig* é que pode assumir valores na faixa de [-1, 1], e é definida pela equação.

$$F(Net_k) = \frac{2}{1+e^{-a.Net_k}} - 1 \tag{3.5}$$

O parâmetro a, assim como na Equação (3.4), determina a inclinação da função.

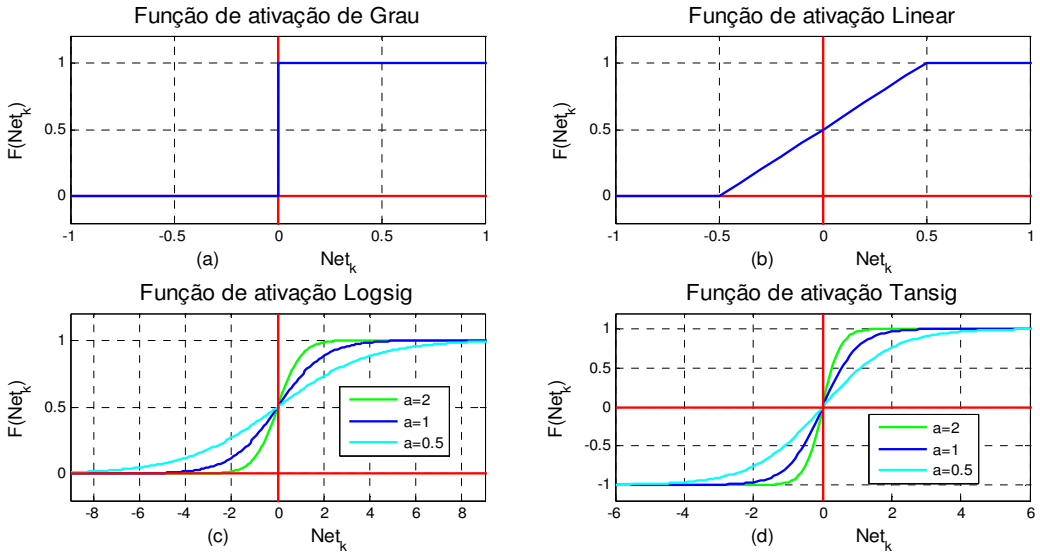


Figura 3.3. Funções de Ativação: (a) Função de Grau. (b) Função linear. (c) Função Logsig. (d) Função Tansig.

3.3. Arquitetura das Redes Neurais

A arquitetura das redes neurais artificiais pode ser formada por uma ou mais camadas de neurônios. Estas camadas estão formadas por neurônios enfileirados e ligados entre si. A entrada de uma camada intermediária " p " é a saída da camada anterior " $p-1$ " ou a camada de entrada, e a saída desta mesma camada " p " é uma entrada da camada seguinte " $p+1$ " ou a camada de saída, como se apresenta na Figura 3.4.

As redes neurais, baseadas em sua arquitetura, são classificadas em dois grupos, as redes Perceptron e as redes Multilayer Perceptron (*MLP*).

3.3.1. Rede Perceptron

O perceptron foi proposto por Rosenblatt [1962], dando continuidade ao trabalho de McCulloch-Pitts. Este pode ser considerado o primeiro modelo de redes neurais, e é capaz de aprender somente sobre problemas linearmente separáveis. Sua arquitetura consiste em uma camada de entrada e uma de saída, como se apresenta na Figura 3.4.

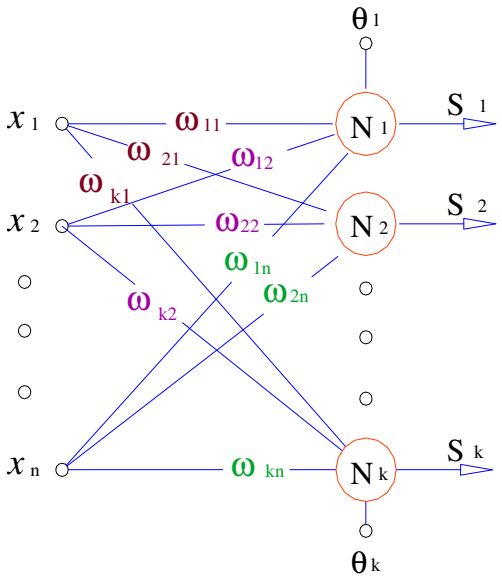


Figura 3.4. Rede perceptron com k Neurônios de saída.

onde $X=[x_1\ x_2\ \ldots x_n]$ representa a camada de entrada, N_k é o $k-ésimo$ neurônio, ω_{jk} é o peso do neurônio N_k em relação à entrada x_j, e S_k representa

a $k-ésima$ saída.

Seguindo a regra de propagação, o valor de Net do $j-ésimo$ neurônio é definido pela equação

$$Net_j = \sum_{i=1}^{k} \omega_{ji}.x_i + \theta_j \tag{3.6}$$

Geralmente, nas redes perceptron a função de ativação é a "*função Degrau*", e sua saída S_j é determinada por

$$S_j = \begin{cases} 1 & \text{Se } Net_j > 0 \\ 0 & \text{Se } Net_j \leq 0 \end{cases} \tag{3.7}$$

Durante o processo de treinamento do perceptron, busca-se encontrar um conjunto de pesos que determine uma reta que separe as diferentes classes, de maneira que a rede possa classificar corretamente as entradas apresentadas. O algoritmo de aprendizado para obter os pesos desejados é determinado de a maneira descrita a seguir.

Para um padrão de entrada, obtêm-se uma saída da rede s_j, com seu respectivo valor de saída desejado t_j. O erro na saída é definido por

$$e_j = t_j - s_j \tag{3.8}$$

Então, a variação do peso $\Delta\omega_{ji}$ é definida por

$$\Delta\omega_{ji} = \eta.x_i.e_j \tag{3.9}$$

onde η é a taxa de aprendizado.

Entretanto, os pesos são atualizados depois de cada entrada com a regra

$$\omega_{ji}^{j+1} = \omega_{ji}^{j} + \Delta\omega_{ji}^{j} = \omega_{ji}^{j} + \eta.x_i.e_j \tag{3.10}$$

O algoritmo de aprendizado aplicado em um dos neurônios da rede perceptron é o mesmo para todos os demais neurônios.

O perceptron de um único neurônio é limitado a trabalhar na classificação de padrões com apenas duas classes. Entretanto, o incremento dos neurônios na camada de saída do perceptron permite classificar padrões de mais de duas classes, sempre que as classes forem linearmente separáveis [17].

3.3.2. Rede Multilayer Perceptron

A rede multilayer perceptron (*MLP*) é considerada como uma das mais importantes *ANN*. Este tipo de rede tem sido aplicado com sucesso em diversas áreas, na solução de problemas complexos, desempenhando tarefas de classificação de padrões, reconhecimento de voz, reconhecimento de imagem, e controle. Uma vez que as redes perceptron só representam funções linearmente separáveis, um dos motivos do ressurgimento da área de redes neurais foi devido ao desenvolvimento do algoritmo *BackPropagation*. A rede MLP é uma generalização da rede perceptron estudada na seção anterior.

As *ANN* do tipo *MLP* estão constituídas de três partes: a primeira parte é denominada camada de entrada (*input layer*), a qual é constituída de um conjunto de unidades sensoriais; a segunda parte é constituída de uma ou mais camadas escondidas (*hidden layers*); e a terceira parte é constituída de uma camada denominada camada de saída (*ouput layer*). Com exceção da camada de entrada, todas as outras camadas estão constituídas por neurônios, as quais implicam em um esforço computacional.

Na Figura 3.5, apresenta-se a arquitetura de uma rede neural *MLP* com uma camada de entrada constituída de n unidades sensoriais, duas camadas escondidas constituída de i e j neurônios respectivamente, e uma camada de saída constituída de k neurônios, formando a arquitetura *MLP* (n-i-j-k).

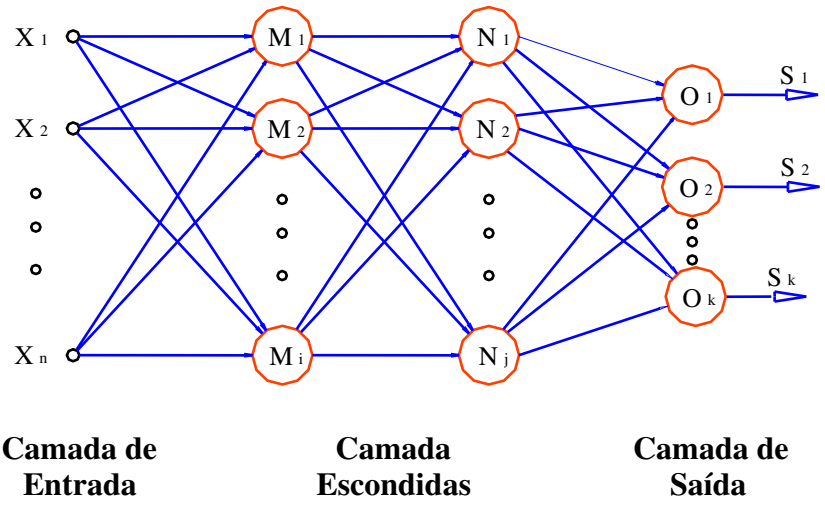


Figura 3.5. Arquitetura de uma rede neural MLP (*n-i-j-k*).

Algumas características básicas das redes *MLP* são as seguintes:

- A função de ativação de cada um dos neurônios da rede é uma *função não-linear*. Uma característica importante desta função é que precisa ser suave e diferenciável em todos os pontos. Geralmente, a função de ativação que satisfaz estes requerimentos é uma *função não-linear sigmoidal*, as mais comuns utilizadas são as *funções Logsig* e *Tansig*.
- As redes *MLP* são constituídas de uma ou mais camadas escondidas, que contém neurônios que não formam parte da camada de entrada ou saída da rede. Estes neurônios escondidos permitem que a rede possa aprender tarefas mais complexas progressivamente, pelo processo de aprendizado.
- Uma rede *MLP* é uma rede *feedforward*, onde o fluxo de dados é sempre em uma única direção. Assim, a saídas dos neurônios de uma camada qualquer se conectam unicamente às entradas dos neurônios da camada seguinte. Portanto, o sinal de entrada se propaga através da rede em um só sentido, isto é, não existe realimentação.
- As redes *MLP* apresentam um alto grau de conectividade ou podem ser completamente conectada, o que é determinado pelas sinapses da rede, caso em que um neurônio de uma camada qualquer é conectado a todos os neurônios da camada seguinte. Além disso, uma rede *MLP* pode ser parcialmente conectada, no caso em que alguma das sinapses esteja faltando. Na prática, a falta de algumas sinapses dentro da rede é representada fazendo o peso das sinapses igual a zero. Entretanto, neste estudo consideram-se redes *MLP* completamente conectadas.

A combinação destas características, junto com a capacidade de aprendizagem, produto da experiência através do processo de treinamento das redes *MLP*, determina o esforço computacional.

O número de variáveis ou nós da camada de entrada é determinado pela complexidade do problema e a dimensionalidade do espaço de observação, de

onde são gerados os sinais de entrada. O número de neurônios na camada de saída é determinado pela dimensionalidade da resposta desejada. Além disso, a configuração e modelagem de uma rede MLP têm como objetivo determinação dos seguintes aspectos:

1. A determinação do número de camadas escondidas.
2. A determinação do número de neurônios em cada uma das camadas escondida.
3. A determinação do valor dos pesos de conexão e as funções de ativação.

Existe uma variedade de metodologias para a escolha destes aspectos, mas normalmente estas são feitas de forma empírica. Na Figura 3.6, apresentam-se estes aspectos.

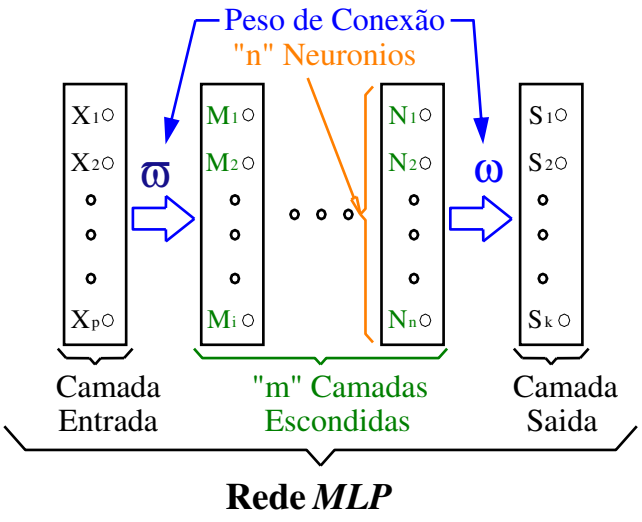


Figura 3.6. Configuração da rede MLP.

O tamanho do modelo de *ANN* é determinado pelos aspectos 1 e 2 acima, mas não há regras para a determinação de tais especificações, sendo geralmente determinados baseado na experiência do especialista. O número de camadas escondidas em uma *ANN* influi na relação entrada-saída. Isto é, uma *ANN* com um maior número de camadas escondidas tem melhor capacidade de extrair as características de algum processo aleatório desconhecido, sobre o qual a rede esta tentando adquirir conhecimento. O incremento excessivo do número de camadas escondidas e neurônios dentro da *ANN* produzem uma rede muito grande que não responderá corretamente a padrões nunca vistos. Isto é, devido à rede extrair muitas características,ela pode tentar, de forma inapropriada, modelar também o

ruído. E um número muito pequeno de camadas escondidas e neurônios produz uma *ANN* muito pequena, que não é capaz de armazenar todos os padrões necessários, a ponto de não modelar fielmente os dados. Além disso, uma rede muito grande demanda um maior esforço computacional, então se deve ter um compromisso entre *Convergência* e *Generalização*.

A *Convergência* é a capacidade da rede de aprender todos os padrões do conjunto de treinamento, a qual esta estritamente relacionada com o tamanho da rede. Entanto, a *Generalização* é a capacidade de um modelo de aprendizado responder corretamente a padrões novos que lhe são apresentados. Um modelo com boa generalização é aquele que responde corretamente a padrões contidos na base de dados, e também a padrões novos contidos na base de teste. A capacidade de generalização é o principal objetivo no processo de aprendizado.

Outro aspecto está relacionado com a determinação da função de ativação de cada um dos neurônios, e do algoritmo de treinamento utilizado para obter os pesos de conexão desejados que resolva o problema ou tarefa. Um dos algoritmos de treino que tem sido aplicado na solução de diversos problemas complexos é o algoritmo conhecido na literatura como *backpropagation*, o qual é baseado na retro-propagação do erro.

A idéia básica do algoritmo *backpropagation* foi descrita por Werbos em sua tese de doutorado na Universidade de *Harvard* em 1974. No entanto, o algoritmo de aprendizado *backpropagation* surgiu após 1985, através da publicação do livro *Parallel Distributed Processing* de Rumelhart e McClelland em 1986.

Além do algoritmo *backpropagation*, existem vários algoritmos de aprendizado, ou também chamados regras de aprendizado, algumas das quais são o algoritmo de aprendizado *Levenberg Marquardt* (utiliza uma aproximação do método de Newton), o algoritmo do gradiente descendente (usado pelo algoritmo *backpropagation*), algoritmos competitivos que se caracterizam pelas conexões laterais dos neurônios com seus vizinhos, estabelecendo uma competição entre os neurônios (usado pelas redes Hopfiel e Kohonen).

A obtenção dos pesos desejados da *ANN* é obtida mediante a atualização de pesos no processo de treinamento, o que pode ser feito de duas maneiras, *Batch* ou Incremental.

No processo de treinamento *Batch ou por ciclos*, a atualização dos pesos acontece somente após a apresentação de todos os padrões do conjunto de treinamento. Assim, todos os padrões são avaliados com a mesma configuração de pesos. Entretanto, no treinamento *Incremental (ou por Padrão)*, a atualização dos pesos é feita após a apresentação de cada novo padrão. Isto leva a que a rede aprenda melhor o último padrão apresentado, e por isso é recomendável apresentar os dados de forma aleatória.

A eficiência dos dois métodos de treinamento depende do problema em questão. O processo de treinamento (aprendizado) é mantido até que os pesos se estabilizem e o erro quadrático médio seja suficientemente pequeno, ou seja, menor que um erro admissível, de modo que o objetivo desejado seja atingido. Assim, deve-se encontrar um ponto ótimo de parada com erro mínimo e máxima capacidade de generalização.

Após o processo de treinamento, algumas vezes a *ANN* pode se especializar demasiadamente em relação aos padrões do conjunto de treinamento, gerando um problema de aprendizado conhecido como super-aprendizado ou *over-fitting*. Geralmente este problema pode ser evitado por meio de um método *Early Stopping*. Este método divide o conjunto de padrões em um novo conjunto de treinamento e validação, após cada varredura do conjunto de treinamento, a rede é avaliada com o conjunto de validação. O algoritmo pára, quando o desempenho com o teste de validação deixa de melhorar.

O procedimento de treinamento utilizado é o treinamento *Supervisionado*, a rede é treinada com um conjunto de dados de treinamento que fornece à rede os valores de entrada e seus respectivos valores de saída desejada, como apresentado na Figura 3.7.

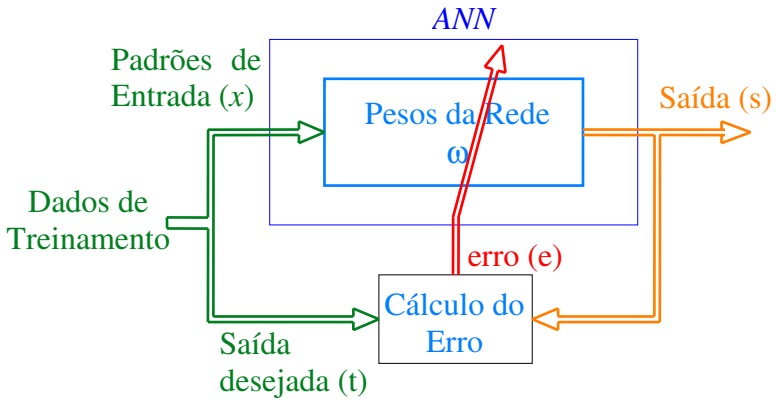


Figura 3.7. Treinamento Supervisionado.

3.4. Algoritmo Backpropagation

Backpropagation é o algoritmo de treinamento mais utilizado pelas *ANN*. Este algoritmo é de tipo supervisionado, e consiste no ajuste dos pesos da rede baseado na minimização do erro médio quadrático pelo algoritmo do gradiente descendente. Portanto, uma parte essencial do algoritmo é o cálculo das derivadas parciais do erro em relação aos parâmetros da *ANN*. O desenvolvimento deste algoritmo permitiu resolver o problema da atualização dos pesos nas camadas escondidas, e com este o ressurgimento das redes neurais.

3.4.1. Fases do Algoritmo *Backpropagation*

O algoritmo é composto de duas fases: *Propagação* e *Retropropagação*, descritos a seguir.

3.4.1.1. Propagação (*forward*)

Nesta fase, um conjunto de padrões é aplicado na camada de entrada da *ANN*, e a resposta desta é propagada como entrada na seguinte camada. Este efeito é propagado através da rede, camada a camada, até que finalmente é produzido um conjunto de saída como a resposta atual da rede. Durante esta etapa, os pesos sinápticos da rede são fixos. Observe na Figura 3.9(b) que o sinal flui através da rede no sentido direito, da esquerda para a direita. Estes sinais que se propagam de forma direta, neurônio a neurônio, através da rede desde a entrada até a saída, são denominados *Sinais Funcionais*. Esta denominação nasceu do fato de estes sinais

serem obtidos na saída de cada neurônio como uma função dos sinais de entrada de cada um. Na Figura 3.9(a), apresenta-se a fase de propagação em uma rede *MLP* (4-3-2) [14].

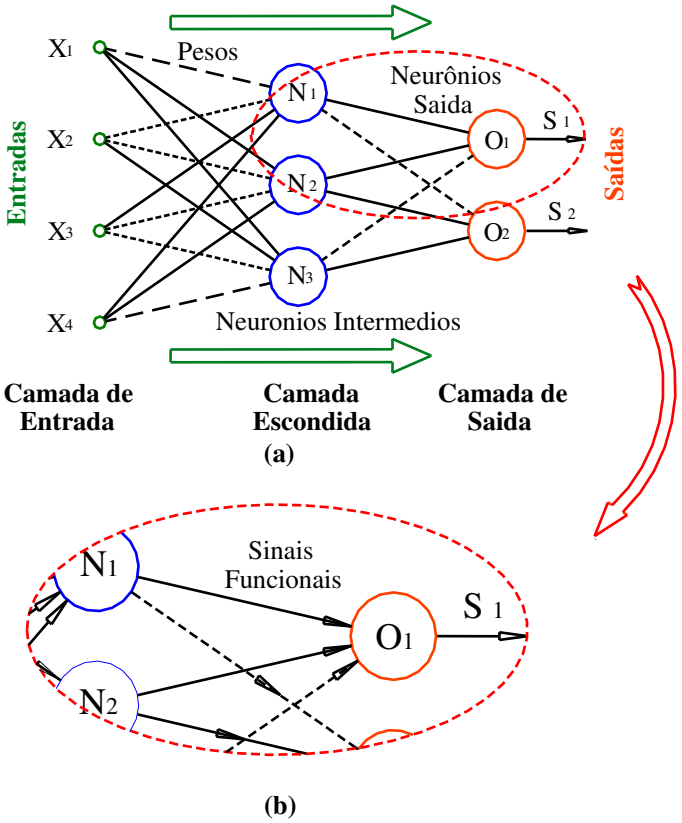


Figura 3.9. (a) Fase de propagação (b) Sinais Funcionais.

3.4.1.2. Retropropagação (*backward*)

Na fase de Retropropagação, desde a camada de saída até a camada de entrada são ajustados todos os pesos sinápticos em concordância com uma correlação do erro. Especificamente, a resposta atual da rede é subtraída de uma resposta desejada (*target*), gerando um sinal erro. Este sinal de erro novamente é retropropagado através da rede no sentido contrario ao sinal funcional, daí o nome de "*error back-propagation*".

Os pesos da *ANN* são ajustados para fazer com que a resposta atual da rede se aproxime à resposta desejada. E o erro originado em um neurônio na camada de saída da rede, que é propagado de volta através da rede, camada a camada, é denominado como sinal de erro. Este sinal é referido como sinal de erro porque o cálculo do erro em cada neurônio da rede é uma função de erro de algum tipo. Na Figura 3.10, apresenta-se a fase de retropropagação e os sinais de erro [14].

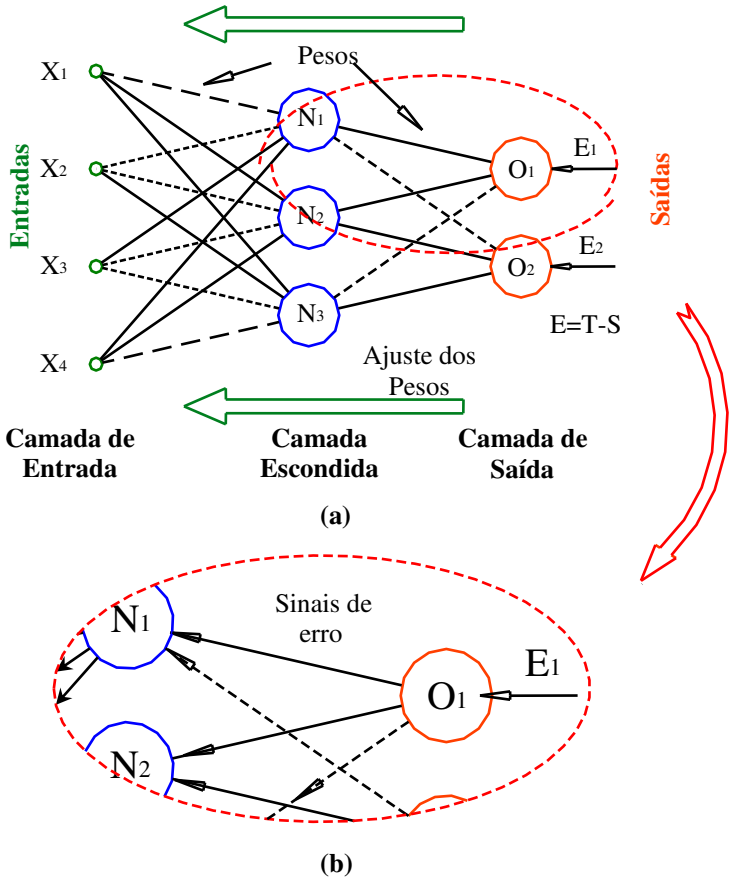


Figura 3.10. (a) Fase de Retro-propagação (b) Sinais de erro.

3.4.2. Algoritmo de Aprendizado

O algoritmo de aprendizado *backpropagation* é baseado na propagação do erro, que inicia na camada de saída e vai para as camadas anteriores da *ANN*, de acordo com o grau de participação que cada neurônio teve no nível posterior. Antes de passar à descrição do algoritmo, é conveniente fazer algumas considerações quanto às características básicas da *ANN*:

- Topologia da rede – Múltiplas camadas (*MLP*).
- Função de ativação – Função não-linear, diferenciável em todo o domínio.
- Tipo de aprendizado – supervisionado.
- Regra de propagação – $Net_j = \sum_{i=1}^{k} \omega_{ji}.x_i + \theta_j$.
- Valores de entrada – saída – binários ou contínuos.

O processo de aprendizado do algoritmo *backpropagation* tem como objetivo a atualização dos pesos sinápticos da rede, o que é feito por meio da minimização do erro quadrático médio pelo método do *Gradiente Descendente*. Baseado neste método, o fator de atualização do peso ω_{ij} relativo à entrada i do neurônio j é dado por

$$\Delta\omega_{ij} = -\eta.\frac{\partial E}{\partial \omega_{ij}} \tag{3.11}$$

onde, $\Delta\omega_{ij}$ é a variação do peso do neurônio j da conexão i, η é a taxa de aprendizado, e E é o somatório do erro quadrático médio total.

O valor do erro quadrático médio total (E) é definido pela Equação (3.12), como a soma do erro quadrático médio de todo os padrões [19].

$$E = \frac{1}{2}.\sum_{p}\sum_{i=1}^{k}(t_i^p - s_i^p)^2 \tag{3.12}$$

onde, p é o numero de padrões, k é o numero de neurônios na saída, e t_i^p e s_i^p são o valor desejado e a saída gerada pela rede para o neurônio i quando é apresentado o padrão p. Além disso, pode-se assumir sem perda de generalidade que a minimização de cada padrão vai a levar à minimização do erro total, de modo que o erro passa a ser definido por

$$E = \frac{1}{2}.\sum_{i=1}^{k}(t_i - s_i)^2 \tag{3.13}$$

3.4.2.1. Calculo da $\Delta\omega_{ij}$

Usando a regra da cadeia na Equação (3.11), pode-se expressar

$$\Delta\omega_{ij} = -\eta.\frac{\partial E}{\partial \omega_{ij}} = -\eta.\frac{\partial E}{\partial net_j}.\frac{\partial net_j}{\partial \omega_{ij}} \tag{3.14}$$

Onde, $net_j = \sum s_i.\omega_{ij} + \theta_j$, e assim

$$\frac{\partial net_j}{\partial \omega_{ij}} = S_i \tag{3.15}$$

onde, s_i é o valor de entrada recebido pela conexão i do neurônio j.

Entretanto, o outro termo da Equação (3.14) é o valor calculado do erro do neurônio j. Este termo depende da camada na qual se encontra o neurônio, e é representado por

$$e_j = -\frac{\partial E}{\partial net_j} \tag{3.16}$$

Assim, das equações acima, pode-se estabelecer que

$$\Delta\omega_{ij} = \eta.s_i.e_j \tag{3.17}$$

Entretanto, o valor do peso atualizado é definido por

$$\omega_{ij}^{t+1} = \omega_{ij}^{t} + \Delta\omega_{ij}^{t+1} = \omega_{ij}^{t} + \eta.s_i.e_j \tag{3.18}$$

onde, ω_{ij}^{t+1} é o valor de peso atualizado, ω_{ij}^{t} é o valor de peso na iteração anterior no processo de aprendizado, e $\Delta\omega_{ij}^{t}$ é a variação do peso.

3.4.2.2. Calculo do Erro (e_j)

Usando a regra da cadeia na Equação (3.16), pode-se estabelecer que:

$$e_j = -\frac{\partial E}{\partial net_j} = -\frac{\partial E}{\partial s_j}.\frac{\partial s_j}{\partial net_j} \tag{3.19}$$

onde $s_j = \varphi(net_j)$, então:

$$\varphi'(net_j) = \frac{\partial s_j}{\partial net_j} \tag{3.20}$$

Entretanto, o valor do termo $\frac{\partial E}{\partial s_j}$ depende da camada à qual pertence o neurônio j.

- **Na camada de saída**

O neurônio j pertence à camada de saída, como é apresentado na Figura 3.11. O valor de E é calculado pela Equação (3.13), de modo que o termo $\frac{\partial E}{\partial s_j}$ é

calculado por

$$\frac{\partial E}{\partial s_j} = -(t_j - s_j) \tag{3.21}$$

Assim, o valor do de e_j para o neurônio j na camada de saída é

$$e_j = (t_j - s_j).\varphi'(net_j) \tag{3.22}$$

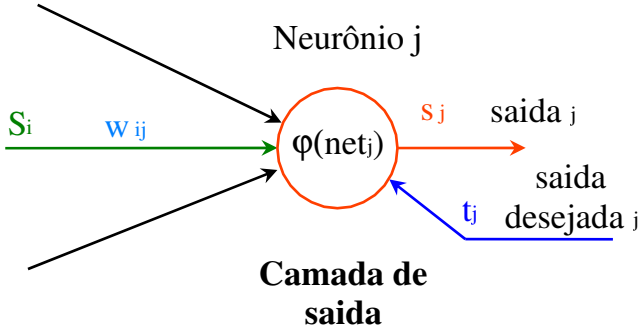


Figura 3.11. Cálculo de erro e_j na camada de saída.

- **Na camada escondida**

O caso em que o neurônio j pertence à camada escondida é apresentado na Figura 3.12.

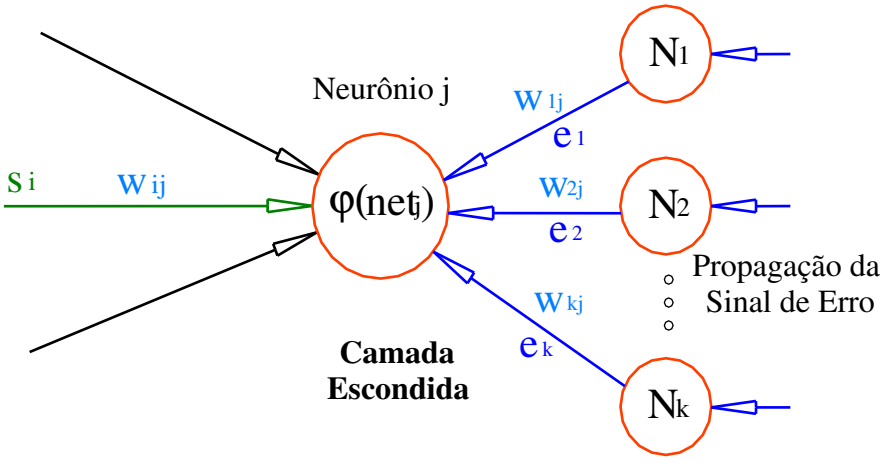


Figura 3.12. Calculo de erro e_j na camada escondida.

O valor de E para este neurônio j é calculado pela Equação (3.13), tomando em consideração a propagação do erro da camada seguinte. O termo $\frac{\partial E}{\partial s_j}$ é calculado por

$$E = \frac{1}{2}.(t_k - s_k)^2$$

$$\frac{\partial E}{\partial s_j} = -\sum_k (t_k - s_k).\frac{\partial s_k}{\partial s_j} \tag{3.23}$$

Usando a regra da cadeia na Equação (3.23), pode-se estabelecer que:

$$\frac{\partial E}{\partial s_j} = \sum_k (t_k - s_k).\frac{\partial s_k}{\partial net_k}.\frac{\partial net_k}{\partial s_j} = -\sum_k (t_k - s_k).\varphi'(net_k).\omega_{jk}$$

$$\frac{\partial E}{\partial s_j} = -\sum_k (e_k.\omega_{jk}) \tag{3.24}$$

O valor de e_j para o neurônio j na camada escondida é então

$$e_j = \sum_k (e_k.\omega_{jk}).\varphi'(net_j) \tag{3.25}$$

3.4.3. Parâmetros de Aprendizado

O algoritmo *backpropagation* faz a aprendizagem da rede através de um processo iterativo de ajuste dos pesos, seguindo uma trajetória de movimento sobre a superfície de erro no espaço de pesos sinápticos, a qual, a cada instante, segue a direção de minimização de erro, podendo chegar ao mínimo global. Alguns parâmetros que influenciam o desempenho do processo de aprendizagem são *Taxa de Aprendizado* e *Termo de Momentum*.

3.4.3.1. Taxa de Aprendizado (η)

A taxa de aprendizado (η) é um parâmetro constante no intervalo [0,1] que interfere na convergência do processo de aprendizado. A influência deste parâmetro esta relacionada à mudança nos pesos sinápticos. Assim, uma taxa de aprendizado muito pequena implica numa trajetória suave e pequenas mudanças nos pesos a cada iteração. No entanto, requer-se um tempo de treinamento muito longo, e dependendo da inicialização dos pesos é possível cair no problema de mínimo local, pois a *ANN* nesse caso não consegue calcular uma mudança nos pesos que faça a rede sair do mínimo local. Na Figura 3.13 apresenta-se o efeito produzido com uma η pequena.

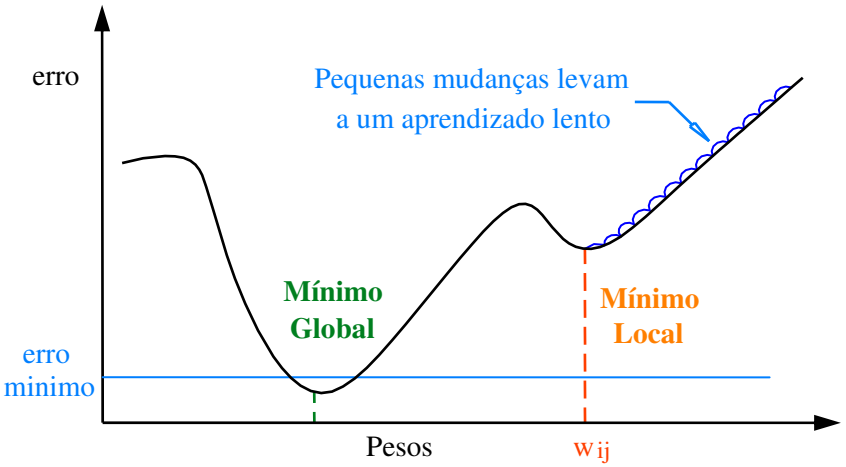


Figura 3.13. Taxa de aprendizado pequeno, com problema do mínimo local.

Entretanto, se a taxa de aprendizado for muito grande (perto de 1), ocorre uma maior mudança nos pesos, aumentando a velocidade do aprendizado, o que pode levar a oscilações em torno do mínimo global. Assim, que a taxa de aprendizado não deve ser muito pequena, nem muito grande. Na Figura 3.14, mostra-se o efeito causado por uma taxa de aprendizado muito grande [19].

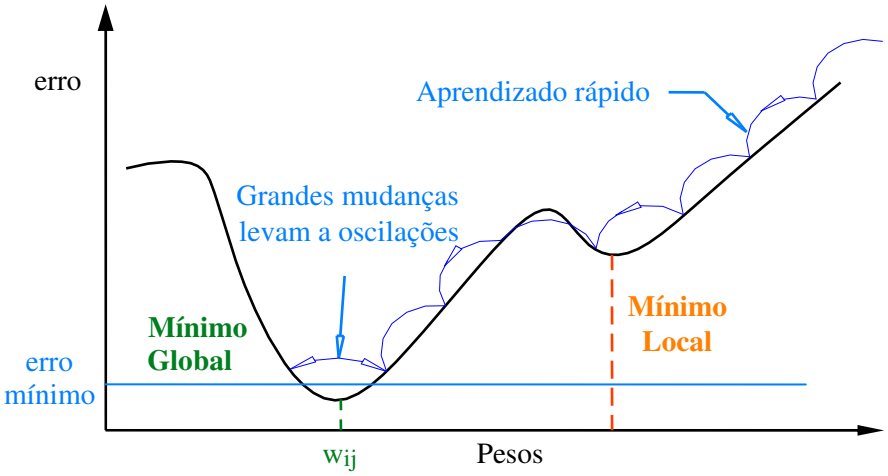


Figura 3.14. Taxa de aprendizado grande, com problema de oscilações.

O ideal seria utilizar a maior taxa de aprendizado possível que não levasse a uma oscilação, obtendo o aprendizado mais rápido. De acordo com BEALE [20], a taxa de aprendizado (η) deveria ser decrementada progressivamente, de modo que o gradiente descendente pelo qual são atualizados os pesos possa encontrar uma melhor solução. Desta forma, utiliza-se uma taxa de aprendizado (η)

adaptativa, a qual inicia com um valor grande, e que decresce exponencialmente à medida que o treinamento evolui.

3.4.3.2. Termo de Momentum (α)

Uma maneira de aumentar a taxa de aprendizado sem levar à oscilação durante a execução do algoritmo *backpropagation* está na à modificação da Equação (3.17) para incluir o termo momentum, o qual determina as mudanças passadas dos pesos. Deste modo, o termo de momentum leva em consideração o efeito das mudanças anteriores dos pesos na direção do movimento atual dos pesos. A mudança dos pesos tomando em consideração o efeito do termo de momentum é determinada por

$$\Delta\omega_{ij}^{t+1} = \eta.s_i.e_j + \alpha.\Delta\omega_{ij}^{t} \tag{3.26}$$

onde, $\Delta\omega_{ij}^{t+1}$ e $\Delta\omega_{ij}^{t}$ são a variação do peso do neurônio j em relação à conexão i no instante $t+1$ e t respectivamente, η é a taxa de aprendizado, e α é o termo de momentum. Na Figura 3.15, apresenta-se o efeito do termo de momentum na aprendizagem da rede [14], [19].

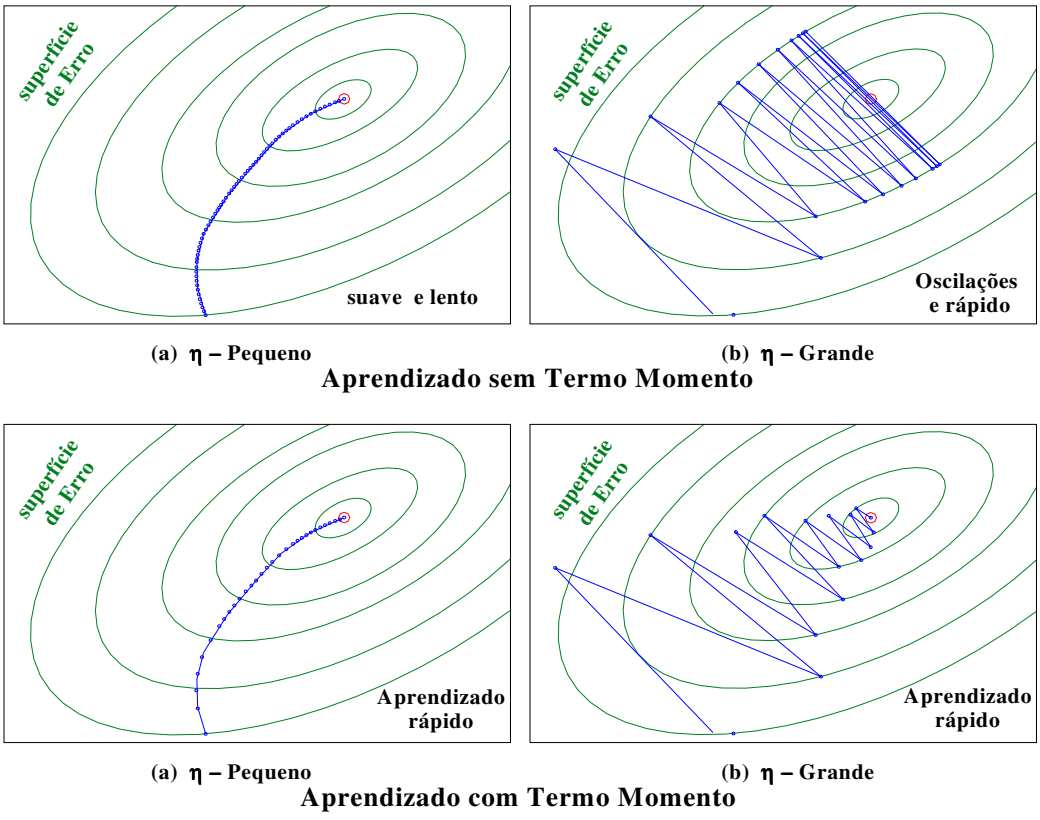


Figura 3.15. Aprendizado com Termo de Momentum.

3.5.Modelagem da *ANN*

Alguns fatores importantes para a modelagem da *ANN* que influem no desempenho das redes são:

- **Seleção de variáveis**

Na maioria das aplicações, as bases de dados contêm um grande número de atributos ou variáveis, os quais são introduzidos com o objetivo de obter um melhor desempenho. Entretanto, na maioria dos casos, muito destes dados são redundantes ou irrelevantes. Portanto, o problema está na seleção das características ou variáveis mais relevantes dentre todos os atributos da base de dados. Neste sentido, alguns dos algoritmos que possuem a capacidade de escolha de variáveis em um menor tempo e custo computacional são a correlação cruzada, a auto-correlação, o estimador por mínimos quadrados (*LSM*), e SIE (*Single Input Effectiveness*).

- **Limpeza de dados**

Geralmente, na etapa de obtenção da base de dados, suas transformações podem levar a conter valores incorretos, ausência de dados, e inconsistências. Deste modo, a limpeza de dados é importante no desempenho do modelo. Este problema é superado pelo conhecimento dos limites dos dados, a identificação de "*outliers*" baseado na visualização, e o uso de informações estatísticas para estabelecer como neutros os dados ausentes ou incorretos.

- **Representação das variáveis**

Dependendo do tipo de dado e do comportamento, as variáveis podem ser codificadas para obter um melhor aprendizado. Assim, os dados podem ser representados como discretos ou contínuos. A representação discreta transforma os dados em uma representação.

- **Normalização**

A normalização torna-se importante quando existem dados muito dispersos no domínio da variável. Nesses casos, valores muito altos podem saturar a função de ativação. Assim, uma maneira fácil de normalizar os dados consiste em somar todas as características e dividir pela soma. Outros tipos de normalização consistem na divisão pela faixa máxima de valores, ou também subtrair do valor médio e dividir pelo desvio padrão.

- **Generalização**

Um aspecto bastante importante ao longo do processo de treinamento é garantir que a rede tenha uma boa generalização. O método mais comum de se garantir uma boa generalização consiste em reservar uma parte da base de dados para validar a generalização.

3.6. Vantagens e Desvantagens das *ANN*

Algumas das principais vantagens e desvantagens, apresentada em aplicações praticas, pelas *ANN* são as seguintes:

*3.6.1.*Vantagens das *ANN*

- A capacidade de lidar com informações incompletas e ruidosas, verificando-se a capacidade da rede em fornecer uma saída satisfatória.
- Não é necessária informação sobre o ambiente a *priori*, pois o aprendizado é feito através da apresentação de padrões à rede.
- Processamento paralelo.
- Habilidade de aprender por meio de exemplos, e a capacidade de generalização a partir dos padrões de treinamento, fornecendo saídas satisfatórias a padrões novos que não foram vistos durante o treinamento.

3.6.2.Desvantagens das *ANN*

- A dificuldade de justificar o comportamento das *ANN* em determinadas situações, isto pois são consideradas como “caixas pretas”, nas quais não se sabe por que a rede chega a um determinado resultado [21].
- Nas *ANN* que utilizam o algoritmo de *backpropagation* o tempo de treinamento tende a ser muito longo. Esta é uma limitação para arquiteturas muito grande ou grande quantidade de dados de treinamento.
- A dificuldade em determinar a arquitetura ideal da *ANN*, de modo que ela seja grande o suficiente para conseguir resolver o problema, e ao mesmo tempo pequeno o suficiente para apresentar um treinamento rápido [22].
- Dependendo do tipo de aprendizagem, outra desvantagem nas *ANN* é a necessidade de ter uma base de dados para o processo de treinamento.

4.
Sistemas Neuro-Fuzzy

Neste capítulo será apresentado o sistema híbrido Neuro-Fuzzy, o qual é a combinação das técnicas de fuzzy e redes neurais. Alguns modelos Neuro-Fuzzy já desenvolvidos na literatura são descritos.

4.1. Introdução

Neste capitulo é apresentada a fundamentação de conceitos para a implementação de sistemas híbridos, tais como o sistema Neuro-Fuzzy. Nas últimas décadas, como uma alternativa aos métodos convencionais de modelagem do controle, surgiram o projeto de controle baseado em *Lógica Fuzzy (FL)* e *Redes Neurais Artificiais (ANN).* Estas duas técnicas são aplicadas com sucesso em diversas áreas nas quais o controle convencional tem falhado na modelagem do controlador.

Os sistemas fuzzy são apropriados para a modelagem de controladores a partir do conhecimento explícito fornecido pelo especialista, nos quais o desempenho do controlador depende da experiência do especialista. E as *ANN* são adequadas na criação do controle baseado no conhecimento implícito embutido em um conjunto de dados, entretanto o desempenho das redes neurais é afetado pelo ajuste de seus parâmetros (número de neurônios em cada camada, número de camadas escondidas, etc.). Portanto, muitos pesquisadores têm tentando integrar estas duas técnicas para gerar um modelo híbrido que possa aproveitar as vantagens de cada uma delas e minimizar suas deficiências. Nos últimos anos, diversos tipos de sistemas híbridos foram desenvolvidos na literatura, geralmente pela integração das técnicas de *Algoritmos Genéticos (GA)*, *ANN* e *FL.*

Neste trabalho, apresentam-se diversas arquiteturas híbridas baseadas na integração de *Sistemas Fuzzy* e *ANN.*

4.2. Sistemas Híbridos

Os sistemas híbridos são a sinergia obtida pela combinação de duas ou mais técnicas de modelagem. O foco destes sistemas está em obter um sistema mais poderoso e com menos deficiências. Dependendo da forma básica de construção, uma técnica pode ser aplicada para melhorar as deficiências do outro em um maior ou menor grau. Algumas destas formas são descritas a seguir.

4.2.1. Sistema Híbrido Seqüencial

Em um sistema híbrido seqüencial, a saída de um subsistema com "*Técnica 1*" atua como entrada de outro subsistema com "*Técnica 2*", como apresentado na Figura 4.1.

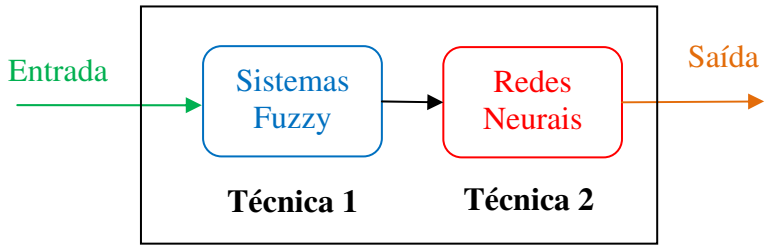


Figura 4.1. Sistema Híbrido Seqüencial.

Na figura anterior, mostra-se um sistema híbrido seqüencial constituído de duas técnicas ou subsistemas: a primeira técnica é um sistema fuzzy que trabalha como um pré-processador fuzzy dos dados, e a segunda técnica é uma *ANN*. Na literatura, freqüentemente este tipo não é considerado como sistema híbrido, por ser a forma mais fraca de hibridização.

4.2.2. Sistema Híbrido Auxiliar

Um sistema híbrido auxiliar, é constituído de dois subsistemas um principal e outro auxiliar. O subsistema principal com "*Técnica 1*" chama ao subsistema auxiliar com "*Técnica 2*" para realizar uma determinada tarefa.

No exemplo da Figura 4.2, o subsistema principal baseado em uma *ANN*, invoca um subsistema auxiliar baseado em um algoritmo genético para a otimização de seus pesos. Este tipo de sistema tem um maior grau de hibridização que o sistema híbrido seqüencial.

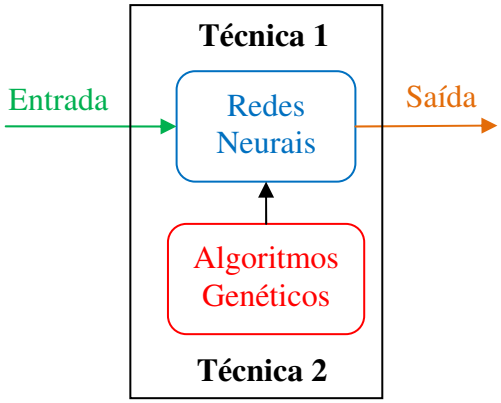


Figura 4.2. Sistema Híbrido Auxiliar.

4.2.3.Sistema Híbrido Incorporado

Na prática, os sistemas híbridos incorporados são os mais utilizados, pois apresentam o maior grau de hibridização, ao ponto que não é possível a separação entre os dois subsistemas. Nestes sistemas, o grau de hibridização é tão elevado que se pode dizer que o primeiro subsistema contém o segundo, ou vice-versa.

Na Figura 4.3, apresenta-se um exemplo de um sistema híbrido incorporado, o qual é constituído de um *Sistema Fuzzy* e uma *ANN*. Estes tipos de sistemas híbridos são conhecidos como Sistemas Neuro-Fuzzy, nos quais um sistema de inferência fuzzy é implementado conforme a estrutura paralela de uma *ANN*.

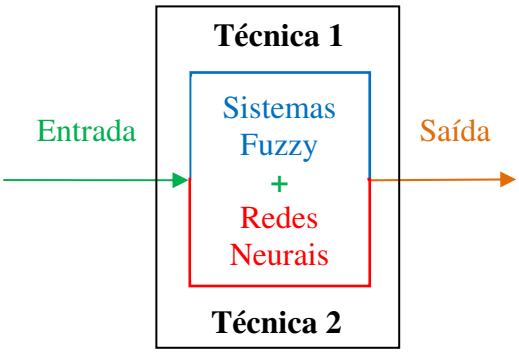


Figura 4.3. Sistema Hibrido Incorporado.

Nesta dissertação, utiliza-se o sistema híbrido incorporado Neuro-Fuzzy, pois este sistema tem recebido uma grande atenção dos pesquisadores em aplicações na área de controle, pela necessidade de controladores cada vez mais eficientes.

4.3. Sistemas Neuro-Fuzzy

Um *Sistema Neuro-Fuzzy* (*SNF*), é um tipo de sistema híbrido incorporado constituído pela combinação de duas técnicas de modelagem muito conhecidas como as *ANN* e a *FL*. Na atualidade, os *SNF* estão se tornando de grande interesse, pois trazem os benefícios tanto de *ANN* quanto de sistemas de *FL*, removendo assim as desvantagens individuais ao combinar as características comuns. Além disso, diferentes arquiteturas de *SNF* vêm sendo pesquisadas em diversas áreas de aplicação, especialmente no controle de processos [23].

A idéia básica de um *SNF* é a construção de um *Sistema de Inferência Fuzzy* (*FIS*), numa estrutura paralela distribuída de tal forma que os algoritmos de aprendizado das redes neurais possam ser aproveitados nestes sistemas híbridos para ajustar os parâmetros do *FIS*. A Figura 4.4 apresenta a estrutura de um *SNF* que é dividido em 5 camadas.

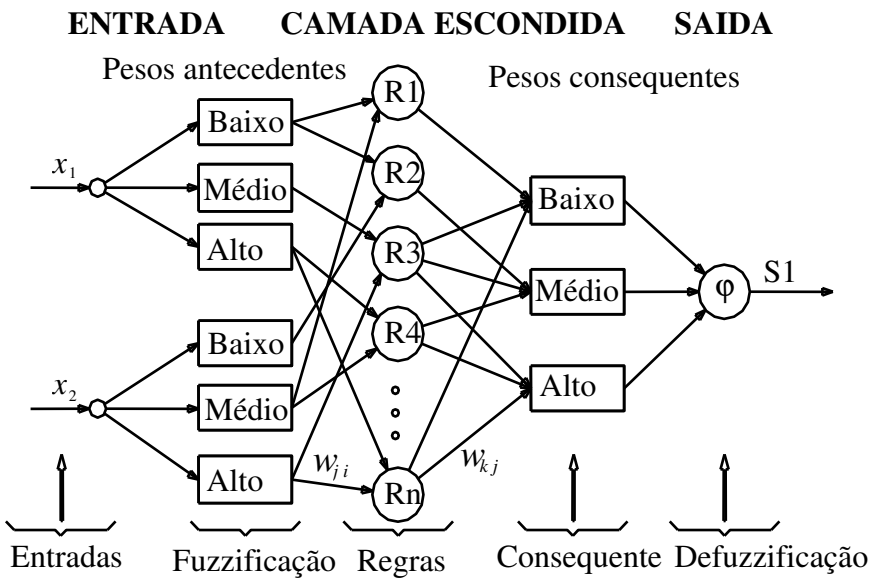


Figura 4.4. Arquitetura básica de um Sistema Neuro-Fuzzy.

A camada de entrada representa as variáveis de entrada, as quais são normalizadas e escalonadas dentro do intervalo numérico [0,1] ou [-1,1]. A segunda camada é a etapa de fuzzificação. Nesta etapa, os intervalos de cada variável de entrada são divididos em diversos níveis (Baixo, Médio e Alto), os quais indicam os pesos da rede para cada entrada. A terceira camada é definida

pelas regras do *FIS*, a camada 4 é determinada pelos conseqüentes das regras, e a camada 5 ou camada de saída é a etapa de defuzzificação, onde se calcula o valor numérico de saída.

4.3.1.Características de Sistemas Neuro-Fuzzy

Os *SNF* permitem a extração de conhecimento baseada na forma de regras de inferência fuzzy, mediante a integração do conhecimento explícito gerado pela experiência do especialista e do conhecimento implícito obtido a partir de um conjunto de dados. Assim, estes sistemas associam a capacidade de aprendizado e de tolerância a falhas das *ANN*, com a interpretabilidade dos *FIS* 's.

Devido à sua natureza dual, estes sistemas herdam as características de seus "genitores". Neste sentido, dividiram-se as características em duas categorias principais, como se apresenta na Figura 4.5.

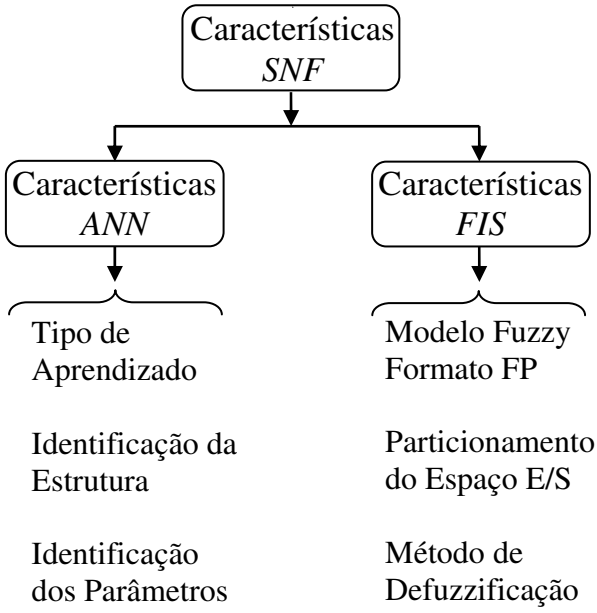


Figura 4.5. Características do Sistema Neuro-Fuzzy.

4.3.1.1.Características Fuzzy do *SNF*

As características fuzzy do *SNF* são agrupadas em 4 sub-classes, classificadas em relação aos parâmetros da estrutura fuzzy, de modo que a combinações destas características determina o modelo do *SNF*. Na Figura 4.6,

apresenta-se esta classificação.

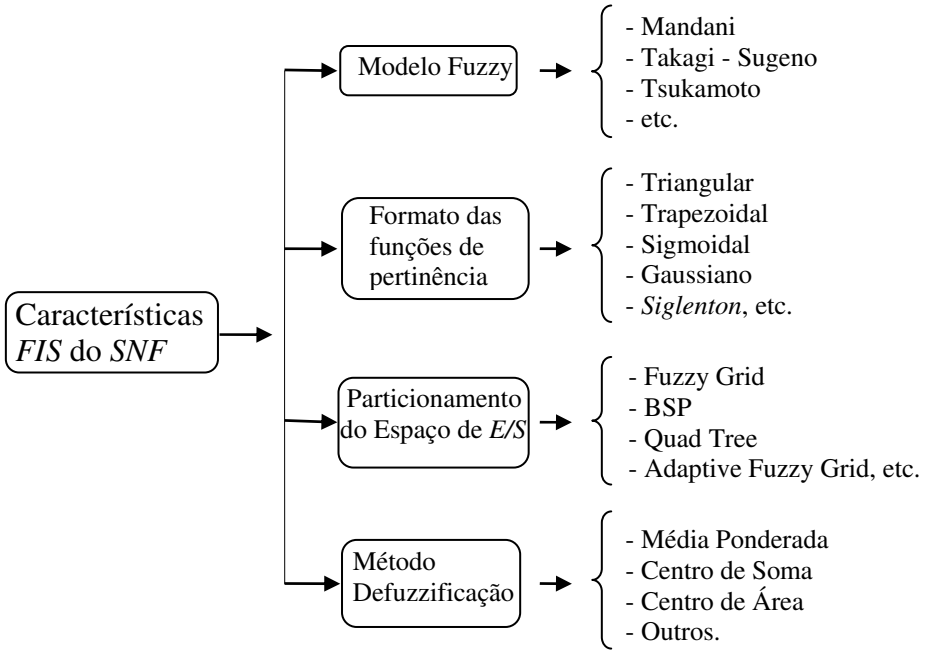


Figura 4.6. Características fuzzy do Sistema Neuro-Fuzzy.

O modelo fuzzy implementado no *SNF* determina o formato das regras fuzzy (modelo de inferência), o qual forma a parte fundamental da estrutura de conhecimento em um sistema de inferência fuzzy. Entretanto, os formatos das funções de pertinência influenciam nos graus de pertinência associados a cada variável, sendo o formato triangular e o trapezoidal os mais usados pela vantagem de serem computacionalmente simples. Assim, as funções de pertinência "*singlenton*" geralmente são utilizadas nos conseqüentes dos sistemas híbridos, isto devido à vantagem de simplificar o processo de defuzzificação do sistema fuzzy.

O particionamento do espaço das variáveis de entrada e saída (*E/S*) define internamente regiões fuzzy de espaço, os quais são relacionados através das regras fuzzy. Assim, o particionamento do espaço de saída geralmente é mais simples pois está associado aos conseqüentes das regras.

Depois de fazer as avaliações do conjunto de regras fuzzy, o valor real da saída do *SNF* é determinado pelo método de defuzzificação escolhido.

4.3.1.2. Características Neurais do SNF

As características neurais do *SNF* estão relacionadas com a capacidade de

aprendizado do sistema híbrido, estando divididas em 3 sub-classes como apresentado na Figura 4.7.

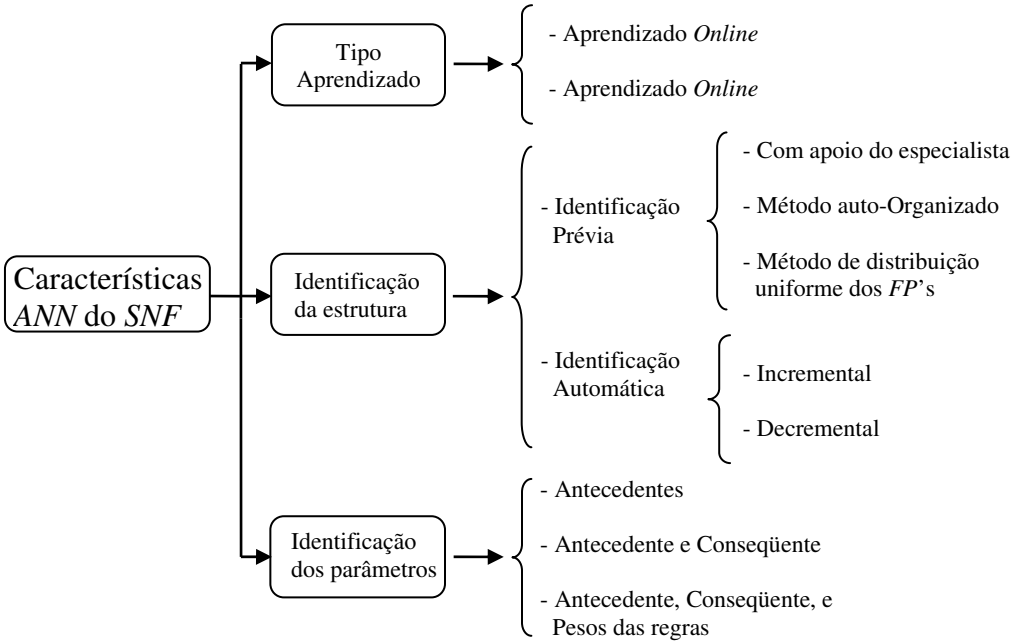


Figura 4.7. Características RNA do Sistema Neuro-Fuzzy.

O tipo de aprendizado, em relação á forma de apresentação dos padrões, divide-se em aprendizado *Off-line* e aprendizado *On-line*. No aprendizado *On-line*, a atualização dos parâmetros da estrutura é feita para cada padrão apresentado. Por outro lado, no aprendizado *Off-line* a atualização dos parâmetros é feita após a apresentação de todos os dados do conjunto de treinamento.

A identificação da estrutura de um *SNF* está relacionada com a determinação do número adequado de regras fuzzy, e de um particionamento satisfatório das E/S. Esta característica divide-se em duas categorias: *Identificação Prévia* e *Identificação Automática*. Na *Identificação Prévia*, o conhecimento da estrutura do sistema é feito *a priori* com apoio de algum especialista ou método auto-organizado, antes do início da atualização dos parâmetros. Na *Identificação Automática*, o conhecimento é obtido de forma incremental ou decremental, de modo que o sistema não precisa de conhecimento prévio.

As características do *SNF* em relação à identificação dos parâmetros são baseadas no método utilizado para o ajuste dos pesos fuzzy que definem os perfis

das *FP*'s dos antecedentes e conseqüentes das regras fuzzy. Geralmente, a maioria dos SNF apresenta identificação dos antecedentes e conseqüentes, com peso fixo de cada regra e de valor unitário. Um exemplo deste tipo de SNF são os modelos ANFIS (*Adaptative Network Based Fuzzy Inference System*). Entretanto, alguns *SNF* só apresentam ajuste nos antecedentes das regras fuzzy; um exemplo típico é o modelo NEFCLASS (*Neuro Fuzzy Classification*). Além disso, existem sistemas que apresentam ajuste no antecedente, conseqüente, e nos pesos de cada uma das regras fuzzy.

4.3.2. Modelos de Sistemas Neuro-Fuzzy

Os modelos dos *SNF* são determinados pela modelagem das características de seus "genitores". Nesta seção apresentam-se brevemente alguns dos SNF mais conhecidos: *ANFIS*, *FSOM*, *NEFCLASS*, *NEFCON* e *NFHQ*. Com isto, espera-se tornar mais clara a compreensão de um *SNF* qualquer.

4.3.2.1. Adaptive Network based Fuzzy Inference System (ANFIS)

O sistema Neuro-Fuzzy ANFIS foi criado por Roger Jang. Esta arquitetura foi usada com sucesso em aplicações de previsão e aproximação de funções. Além disso, o autor propõe algumas variações do modelo para outras aplicações, aumentando sua popularidade ao ponto de ser inserido no *software* MATLAB©. A Figura 4.8 ilustra a arquitetura *ANFIS*.

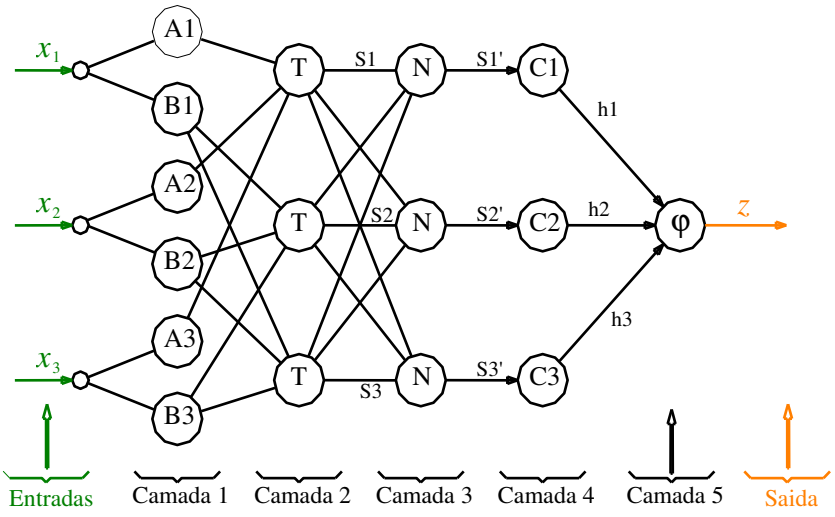


Figura 4.8. Arquitetura ANFIS.

Na figura acima, os nós que compõem uma mesma camada têm funções semelhantes, as quais são descritas a seguir.

- *Camada 1*: A saída desta camada são os graus de pertinências das entradas, baseado na premissa de cada regra. Neste caso, cada entrada apenas tem duas funções de pertinência (*Ai = Alto* e *Bi = Baixo*), podendo ser este número maior.

- *Camada 2*: Nesta camada, calcula-se o grau de pertinência ao qual é submetido o conseqüente de cada regra. Cada nó ou neurônio desta camada executa a operação de *t_norm* e corresponde a uma regra:

$$\begin{aligned} S_1 &= A_1(x_1) * A_2(x_2) * A_3(x_3) \\ S_2 &= B_1(x_1) * B_2(x_2) * A_3(x_3) \\ S_3 &= B_1(x_1) * B_2(x_2) * B_3(x_3) \end{aligned} \tag{4.1}$$

onde "*" representa a *t -norm*.

- *Camada 3*: Esta camada realiza a normalização dos graus de ativação das regras. Cada nó desta camada executa a função normalização, a qual é utilizada como um pré-processamento para a defuzzificação:

$$\begin{aligned} S_1' &= S_1 / (S_1 + S_2 + S_3) \\ S_2' &= S_2 / (S_1 + S_2 + S_3) \\ S_3' &= S_3 / (S_1 + S_2 + S_3) \end{aligned} \tag{4.2}$$

- *Camada 4*: Nesta camada, a saída de cada neurônio é calculada pelo produto da saída normalizada da camada anterior e o grau de ativação do conseqüente. Este valor de saída é dado por:

$$\begin{aligned} h_1 &= S_1'.C_1 \\ h_2 &= S_2'.C_2 \\ h_3 &= S_3'.C_3 \end{aligned} \tag{4.3}$$

onde C_i's correspondem aos valores dos conseqüentes, os quais, por exemplo, podem ser funções de pertinência "*singlenton's*".

- *Camada 5*: A saída desta camada fornece a saída precisa do sistema ANFIS. Esta etapa de defuzzificação facilita-se pelo cálculo das camadas 3 e 4, dado por:

$$Z = \frac{\sum S_i.f_i}{\sum S_i} = \sum S_i'.f_i \qquad (4.4)$$
$$Z = h_1 + h_2 + h_3$$

O espaço de particionamento das *E/S* utilizadas é do tipo *Fuzzy Grid Adaptativo*. Seu processo de aprendizado do sistema para a identificação da estrutura e parâmetros é feito em duas etapas, repetidas até atingir o critério de parada:

- *Etapa 1*: Os parâmetros dos conseqüentes são ajustados pelo método de *MQO (Mínimos Quadrados Ordinários)*, enquanto os antecedentes permanecem fixos.
- *Etapa 2*: Os parâmetros dos antecedentes são ajustados pelo método de *GD (Gradient Decrescent)*, enquanto os conseqüentes se mantêm fixos.

A idéia principal do sistema ANFIS é implementar um sistema fuzzy numa rede neural, onde geralmente as funções de pertinência utilizadas são do tipo sigmóides.

4.3.2.2. Neuro Fuzzy Classification (NEFCLASS)

Este modelo foi proposto por *Nauck* e *Kruse*, e é aplicado com sucesso basicamente em problemas de classificação. Além disso, *Nauck* desenvolveu outras duas arquiteturas, o modelo *Neuro-Fuzzy Control (NEFCON)* para aplicações em controle e o modelo *Neuro-Fuzzy Function Approximation (NEFPROX)* para aplicações em previsão e aproximação de funções. Todos estes modelos utilizam um modelo genérico denominado *Fuzzy Perceptron*. A Figura 4.9 ilustra uma arquitetura *NEFCLASS*.

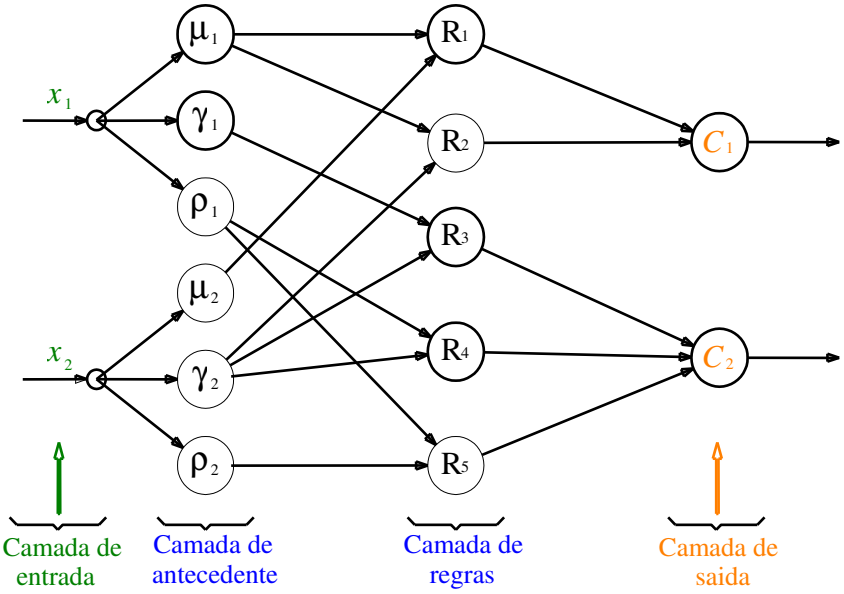


Figura 4.9. Arquitetura NEFCLASS.

Estes tipos de arquiteturas estão constituídas de 4 camadas e são descritas conforme se segue:

- *Camada de Entrada*: Esta camada só tem a função de direcionar os valores das entradas (x_1 e x_2) para as entradas das funções de pertinência dos antecedentes das regras. Estas regras fuzzy têm a forma:

  Se $x_1 \in \mu_1$ **e** $x_2 \in \mu_2$ **então** Padrão (x_1, x_2) pertence à classe *i*.

- *Camada de Antecedentes*: Esta camada fornece o grau de pertinência dos antecedentes de cada regra, e cada variável de entrada foi dividida em três funções de pertinência (*Baixo, Médio, Alto*). O particionamento do espaço de entrada utilizado nesta camada é "*Adaptative Fuzzy-Grid*".
- *Camada de Regras*: Nesta camada, executam-se a operação t-norm entre os graus de pertinência dos antecedentes de cada regra, gerando assim seu grau de ativação.
- *Camada de Saída*: Esta camada fornece a saída do sistema, obtida executando a operação *t-conorm* com os graus de ativação da camada de regras. Os pesos de interligação entre a camada de regras e a camada de saída são unitários.

O aprendizado deste modelo é feito em duas etapas:

- *Etapa 1*: Na primeira etapa, utiliza-se um algoritmo de aprendizado da base de regras, que pode ser construída de duas formas: inicializada a partir de um conhecimento prévio sobre o conjunto de padrões e depois refinada por aprendizado acrescentando ou diminuindo regras ou inicializada com um conjunto vazio de regras e acrescida por aprendizado incremental.
- *Etapa 2*: Após a criação da base de regras, nesta fase executa-se um algoritmo de aprendizado baseado no *GD(Gradient Descent)* para ajustar os parâmetros das funções de pertinências dos antecedentes das regras.

O sistema híbrido *NEFCLASS* é uma rede *Fuzzy-Perceptron* que apresenta a vantagem de interpretação da estrutura em forma de regras, e herda todas a características das *ANN* do tipo *Perceptron Multicamadas(MLP).*

4.3.2.3.Fuzzy Self-Organized Map (FSOM)

Este modelo foi desenvolvido por *Vuorimaa*, possui uma estrutura muito semelhante ao modelo *ANFIS*, como se ilustra na Fig. 4.10. Este modelo utiliza particionamento *Fuzzy-Box* no espaço de entrada.

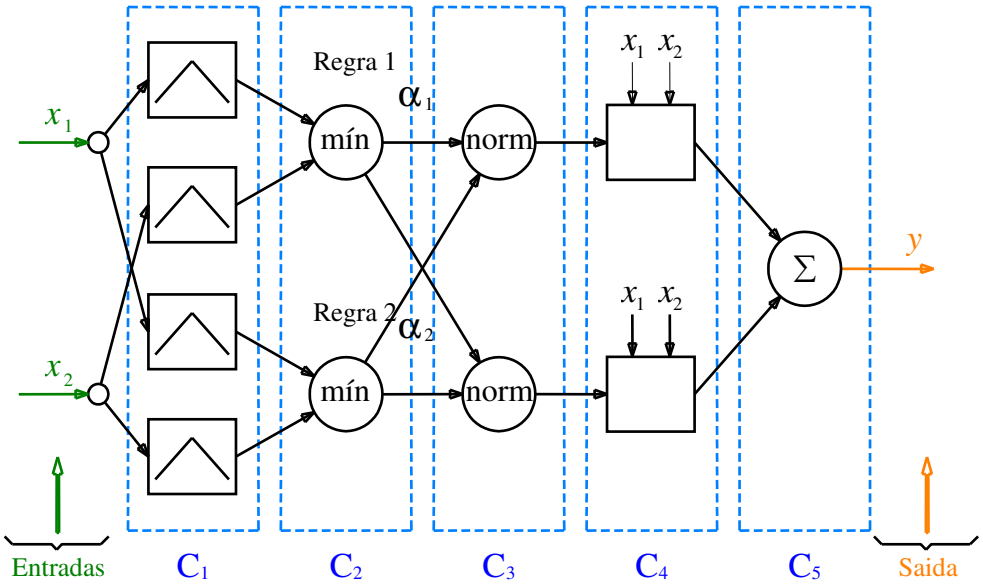


Figura 4.10. Arquitetura FSOM.

A descrição das camadas é semelhante ao modelo ANFIS, à exceção de

alguns detalhes descritos abaixo.

Na *Camada 1*, as funções de pertinência dos antecedentes são de formato triangular, e o formato da regras fuzzy são

$$\textbf{Se } x_1 \in U_{i,1} \textbf{ e } x_2 \in U_{i,2} \textbf{ então } y = S_i.$$

onde $x_1 \in U_{i,1}$ é o grau de pertinência da entrada $x_j \in U_{i,j}$ no conjunto fuzzy $U_{i,j}$.

Na camada 2, a operação *t-norm* para o cálculo do grau de pertinência das regras é

$$\alpha_i = min\left\{\mu_{U_{i,1}}(x_1) + \mu_{U_{i,2}}(x_2)\right\} \tag{4.5}$$

onde α é o grau de pertinência calculado para cada regra.

Na camada 5, o cálculo da defuzzificação é feito utilizando o método da média ponderada, e é dado por:

$$y = \frac{\sum \alpha_i . S_i}{\sum \alpha_i} = \sum \alpha_i' . S_i \tag{4.6}$$

onde S_i é o grau de ativação das funções de pertinência do conseqüente de cada regra.

O processo de aprendizado deste modelo é feito em três fases.

- *Etapa 1*: Os parâmetros dos centros C_i das funções de pertinência triangulares são auto-organizados utilizando o algoritmo *SOM* de Kohonen.
- *Etapa 2*: Os parâmetros dos conjuntos fuzzy são formados em volta aos centros obtidos na etapa 1, usando-se uma largura ω_0.
- *Etapa 3*: Nesta etapa, são ajustados os parâmetros dos antecedentes por um algoritmo supervisionado semelhante ao LVQ, o qual também é gerado por Kohonen.

Esta arquitetura pode ser usada na implementação de sistemas para aproximação de funções e controle.

4.4. Vantagens e Desvantagens dos *SNF*

Algumas das principais vantagens e desvantagens apresentadas pelos sistemas híbridos em aplicações práticas são:

4.4.1. Vantagens das *SNF*

- Os *SNF* combinam as vantagens dos sistemas fuzzy e *RNA*, mostrando um enorme potencial para aplicações que combinem conhecimento qualitativo com robustez.
- A lógica fuzzy provê uma interface de alto nível, de rápida computação e amigável para programar, permitindo que o especialista se concentre nos objetivos funcionais em vez dos detalhes matemáticos. Por outro lado, as *ANN* são convenientes para a extração de conhecimento através do aprendizado.
- O projetista não precisa ter conhecimento prévio do processo, levando a uma fácil adaptabilidade aos diferentes processos.
- Os modelos Neuro-Fuzzy, podem lidar de uma maneira melhor que as *ANN* ao problema de ruído nos dados.
- Capacidade de auto-aprendizado, auto-organização e auto-direcionamento, imitando a capacidade humana do processo de tomada de decisão.

4.4.2. Desvantagens dos *SNF*

Algumas das limitações dos *SNF* são:

- Os SNF trabalham com um reduzido número de entradas, devido ao problema de explosão combinatória das regras. Portanto, um sistema com muitas entradas demanda um maior esforço computacional.
- Limitação na construção de sua própria estrutura, devido à estrutura fixa; quando há a capacidade de alterar sua estrutura, são limitados pelo elevado número de regras.

No próximo capítulo, é apresentada uma breve descrição da modelagem do sistema servo-hidráulico de interesse desta dissertação, e o projeto do controle por aprendizado acelerado proposto.

5.
Controle por Aprendizado Acelerado e Simulação

5.1. Introdução

Neste capitulo é apresentado o projeto de um controle por aprendizado acelerado, e os resultados da simulação desenvolvida em Matlab©. Além disso, é apresentada uma breve descrição da modelagem do sistema servo-hidráulico, e o controle por aprendizado desenvolvido na dissertação de Alva [5], o qual será otimizado neste trabalho.

5.2. Modelagem do Sistema Servo-Hidráulico

Um sistema servo-hidráulico é constituído de um conjunto de componentes tais como: fonte de potência hidráulica, servo-válvula, pistão hidráulico, mangueiras, e sensores. Na Figura 5.1, apresenta-se um sistema hidráulico de uma máquina utilizada para ensaios de fadiga.

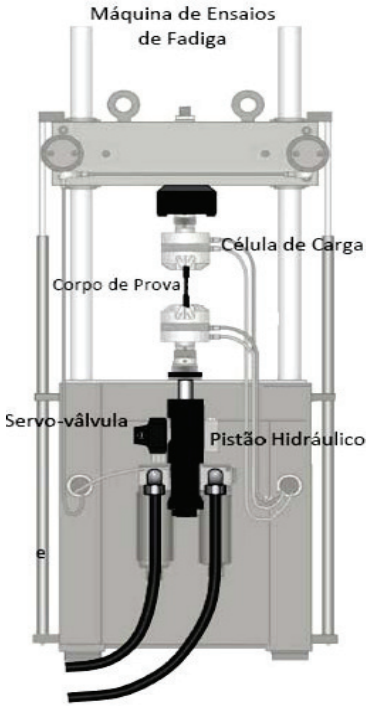


Figura 5.1. Sistema Servo-Hidráulico

Na presente dissertação, é utilizada a modelagem do sistema hidráulico de uma máquina para ensaios de fadiga desenvolvida por Alva. Este modelo é utilizado para avaliar o desempenho na simulação dos controladores propostos.

A modelagem do sistema servo-hidráulico é determinada pela dinâmica de seus componentes, os quais são descritos a seguir.

5.2.1. Modelagem da Servo-válvula

A modelagem do comportamento dinâmico da servo-válvula envolve uma variedade de parâmetros, como ilustrado na Figura 5.2. Muitos destes parâmetros são determinados de diferentes fontes de informação, fazendo com que o modelo não reflita exatamente o comportamento real.

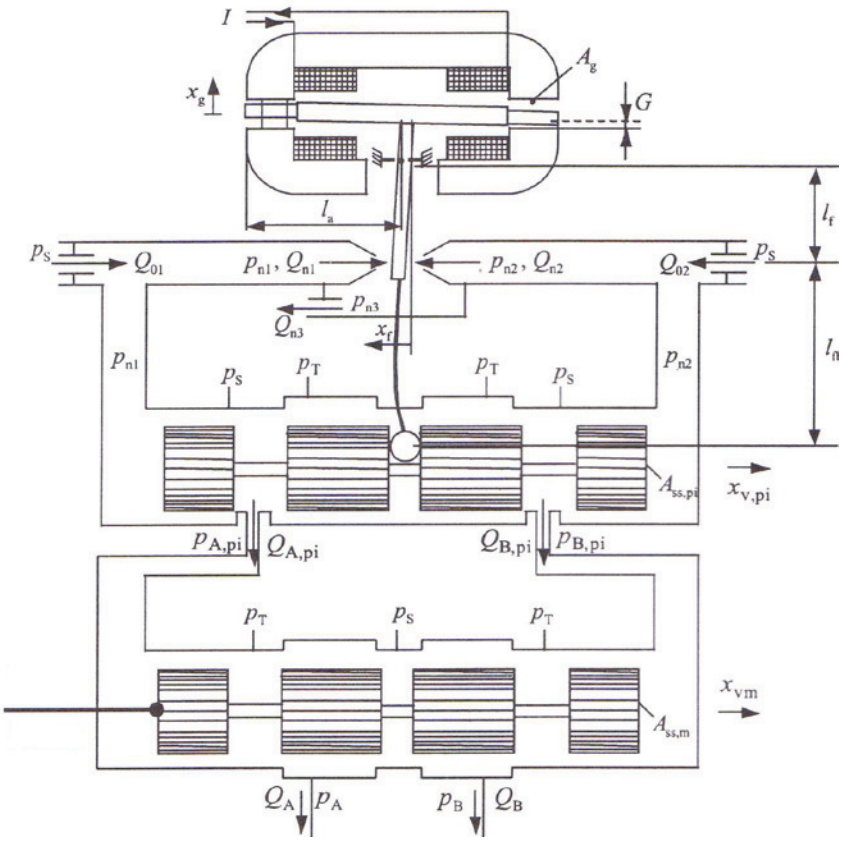


Figura 5.2. Representação esquemática da servo-válvula.

Aplicando a segunda lei de Newton, as forças sobre o carretel principal da servo-válvula podem ser obtidas:

$$m_{sm}.\ddot{x}_{vm} + F_f(\dot{x}_{vm}) = A_{ss,m}.(P_{A,pi} - P_{B,pi}) - F_{ax,m} \tag{5.1}$$

$$P_{A,pi} = \frac{E'}{V_{A,pi}}.(Q_{A,pi} + A_{ss,m}.\dot{x}_{vm}) \tag{5.2}$$

$$P_{B,pi} = \frac{E'}{V_{B,pi}}.(Q_{B,pi} + A_{ss,m}.\dot{x}_{vm}) \tag{5.3}$$

onde $m_{sm}, A_{ss,m}, \dot{x}_{vm}$ é a massa, área e velocidade do carretel principal respectivamente, $F_f(\dot{x}_{vm})$ é a força de atrito, $F_{ax,m}$ é a força de fluxo axial sobre o carretel, e $P_{A,pi}, Q_{A,pi}, P_{B,pi}, Q_{B,pi}$ são as pressões e fluxos nos pontos A e B do carretel piloto.

Como as forças de atrito e aceleração são muito menores que a força impulsora, e a relação entre a área lateral e os volumes das câmaras no carretel principal são relativamente grandes, a pressão dinâmica no carretel principal pode ser desprezada. Assim, simplificando as equações (5.1) e (5.3), obtêm-se duas relações:

$$Q_{B,pi} = A_{ss,m}\dot{x}_{vm} = -Q_{A,pi} \tag{5.4}$$

$$A_{ss,m}(P_{A,pi} - P_{B,pi}) = 0 \tag{5.5}$$

Finalmente, as equações que descrevem a dinâmica dos fluxos do atuador são:

$$\begin{aligned} Q_A = {} & C_{v1}.sg(x_{vm}).sign(P_s - P_A).\sqrt{|P_s - P_A|} \\ & -C_{v2}.sg(-x_{vm}).sign(P_A - P_T).\sqrt{|P_A - P_T|} \end{aligned} \tag{5.6}$$

$$\begin{aligned} Q_B = {} & C_{v3}.sg(-x_{vm}).sign(P_s - P_B).\sqrt{|P_s - P_B|} \\ & -C_{v4}.sg(x_{vm}).sign(P_B - P_T).\sqrt{|P_B - P_T|} \end{aligned} \tag{5.7}$$

onde Q_A, P_A, Q_B, P_B são as pressões e fluxos do carretel primário nos pontos A e B, e x_{vm} é a posição do carretel principal.

5.2.2. Modelagem do Pistão Hidráulico

A equação da dinâmica do pistão foi obtida aplicando a segunda lei de Newton nas forças do pistão, logo

$$m_t.\ddot{x}_p + F_f(\dot{x}_p) = A_p.(P_A - \alpha.P_B) - F_{ext} \tag{5.8}$$

onde m_t é massa total, constituída pela massa do pistão (m_p), e pelas massas das câmaras do cilindro e suas tubulações, dadas por $m_{A,fl}$ e $m_{B,fl}$ respectivamente, i.e.

$$m_t = m_p + m_{A,fl} + m_{B,fl} \tag{5.9}$$

A massa do fluido é determinada por

$$m_{A,fl} = \rho.[V_{PL,A} + (x_{p0} + x_p).A_p] \tag{5.10}$$

$$m_{B,fl} = \rho.[V_{PL,B} + (x_{p0} + x_p).\alpha.A_p] \tag{5.11}$$

A massa do fluido pode ser desprezada em relação à massa do pistão. Aplicando a equação de continuidade para cada uma das câmaras do cilindro, obtém-se

$$Q_A - Q_{Li} = \dot{V}_A + \frac{\dot{V}_A}{\dot{E}(P_A)}.\dot{P}_A \tag{5.12}$$

$$Q_B + Q_{Li} - Q_{Le} = \dot{V} + \frac{\dot{V}_A}{\dot{E}(P_A)}.\dot{P}_A \tag{5.13}$$

onde, V_A é o volume da câmara do pistão, V_B é o volume da câmara do anel e linhas de conexão, e Q_{Li} e Q_{Le} são os fluxos de escapamento interno e externo, respectivamente.

5.2.3. Modelos lineares

Combinando os modelos dinâmicos da servo-válvula com o modelo do pistão hidráulico, podem-se obter as funções de transferência que modelam o sistema servo hidráulico, como descrito a seguir.

5.2.3.1.Função de Transferência da Posição

A função de transferência para o controle de posição é determinado pela equação

$$G_x(S) = \frac{X_p(S)}{I(S)} = \left[\frac{K_v.K_Q}{\frac{1}{\omega_v^2}.S^2 + \frac{2.D_v}{\omega_v}.S + 1} \right] \left[\frac{\frac{A_p}{m_p}}{S^2 + 2.D_h.\omega_h.S + \omega_h^2} \right] \left[\frac{1}{S} \right] \quad (5.14)$$

onde ω_h é a frequência natural do sistema servo-hidráulico, a qual é determinada por

$$\omega_h = \sqrt{\frac{\sigma}{m_p.T_h} + \frac{A_p}{m_p}.K_d} \quad (5.15)$$

A taxa de amortecimento D_h é dada por

$$D_h = \frac{\frac{1}{T_m} + \frac{1}{T_h}}{2.\omega_h} \quad (5.16)$$

Os parâmetros T_m e K_d são determinados por

$$\begin{aligned} T_m &= \frac{m_p}{\sigma} \\ K_d &= \frac{A_p}{C_h} \end{aligned} \quad (5.17)$$

onde m_p e A_p são a massa e área do pistão, e C_h é a capacitância hidráulica.

5.2.3.2. Função de Transferência da Força

A função de transferência para o controle de força é obtida a partir da função de transferência de posição, sendo conhecida a seguinte equação:

$$P_L = \left(\frac{m_p}{A_p} + \frac{\sigma}{A_p} . \frac{1}{S} \right) \tag{5.18}$$

Usando a relação $F_L = A_P . P_L$, obtém-se:

$$G_F(S) = \frac{F_L(S)}{I(S)} = \left[\frac{K_v . K_Q}{\frac{1}{\omega_v^2} . S^2 + 2 . D_v . \omega_v . S + 1} \right] \left[\frac{A_p . \left(S + \frac{\sigma}{m_p} \right)}{S^2 + 2 . D_h . \omega_h . S + \omega_h^2} \right] \tag{5.19}$$

onde w_v é a frequência natural e D_v é a taxa de amortecimento, e os demais parâmetros forem definidos na Equação (5.14).

5.2.3.3. Função de Transferência da Deformação

Para controle de deformação em ensaios em corpos de prova, a rigidez da máquina deve ser muito maior que a dos corpos de prova testados. Assumindo-se que isto seja verdade, a função de transferência para o controle de deformação é dada por

$$G_F(S) = \frac{F_L(S)}{I(S)} = \left[\frac{K_v . K_Q}{\frac{1}{\omega_v^2} . S^2 + 2 . D_v . \omega_v . S + 1} \right] \left[\frac{A_p . \left(S + \frac{\sigma}{m_p} \right)}{S^2 + 2 . D_h . \omega_h . S + \omega_h^2} \right] \left[\frac{1}{m_p . S^2 + b . S + k} \right] \tag{5.20}$$

5.3. Controle por Aprendizado Acelerado

Na etapa de modelagem de um sistema de controle ótimo, toda a informação do sistema controlado (dinâmica da planta) é conhecida e descrita deterministicamente. O controlador ótimo é então projetado baseado em diferentes técnicas matemáticas de controle robusto. No caso em que parte ou toda a informação é descrita de maneira estatística (funções de probabilidade, distribuição de densidade de probabilidade), o processo de projeto do controlador utiliza técnicas estatísticas.

No caso em que a informação *a priori* requerida é desconhecida ou parcialmente conhecida, geralmente não é possível projetar um controlador ótimo utilizando técnicas de controle clássico. Para solucionar este problema, existem dois enfoques. Um enfoque é projetar um controlador baseado somente na quantidade de informação disponível; entretanto, a informação desconhecida é ignorada ou são assumidos alguns valores conhecidos para melhorar seu desempenho. O segundo enfoque é projetar um controlador com a capacidade de estimar a informação desconhecida ao longo de sua operação, e baseado nesta informação estimada mudar a ação de controle.

O processo de aprendizagem pode ser visto como um processo de estimação ou de aproximações sucessivas de informações desconhecidas. Na presente aplicação, esta informação representa a função de controle. Esta informação desconhecida é estimada ou aprendida pelo controlador ao longo de seu funcionamento, sendo constituídas de parâmetros ou características que descrevem uma função probabilística ou determinística. A relação entre estas funções e a lei de controle geralmente é determinada pelo projetista baseado em algum critério, como por exemplo, a otimização do desempenho. Assim, à medida que o controlador armazena mais informação dos parâmetros desconhecidos, a lei de controle é atualizada, melhorando o desempenho do sistema.

Na Figura 5.3 ilustra-se um diagrama de blocos que representa um sistema de controle por aprendizado. Neste controlador, a informação estimada é representada pela variável adimensional U_{IJ}, e armazenada na memória. Esta

variável é utilizada para mudar a ação de controle, e é atualizada após cada ciclo de operação através de uma lei de aprendizado baseada nos erros medidos.

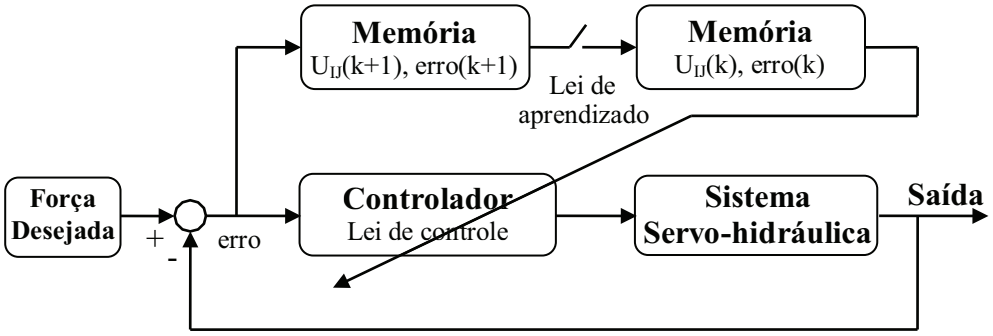


Figura 5.3. Diagrama de blocos do controle por aprendizado.

Na Figura 5.3, os valores de $U_{IJ}(k)$ e $erro(k)$ são atualizados pela chave representada, após cada ciclo, a partir dos valores $U_{IJ}(k+1)$ e $erro(k+1)$.

5.3.1. Metodologia de controle

A metodologia de controle por aprendizado acelerado (*CAL - Control for Accelerated Learning*) apresentada nesta dissertação é uma otimização da lei de controle por aprendizado desenvolvido por Alva [5], com o objetivo de obter um melhor desempenho. Este controle por aprendizado, implementado em uma máquina para ensaios de fadiga, consiste em manter uma servo-válvula trabalhando em seus limites extremos de operação, tentando sempre mantê-la completamente aberta em uma ou outra direção.

O ponto de reversão está associado ao instante no qual a servo-válvula muda de direção. Este parâmetro depende de diversos fatores, como da amplitude de carga desejada e carga média requerida, e é influenciado pelos níveis de corrente aplicada na servo-válvula, e pela zona morta causada por folgas na fixação dos corpos de prova.

Devido à dinâmica do sistema, após a reversão da servo-válvula não há uma reversão imediata do sentido de crescimento da força aplicada ao corpo de prova. Portanto, os pontos de reversão precisam ocorrer antes dos picos ou vales de força desejada. Na Figura 5.4, são apresentados exemplos de pontos de reversão.

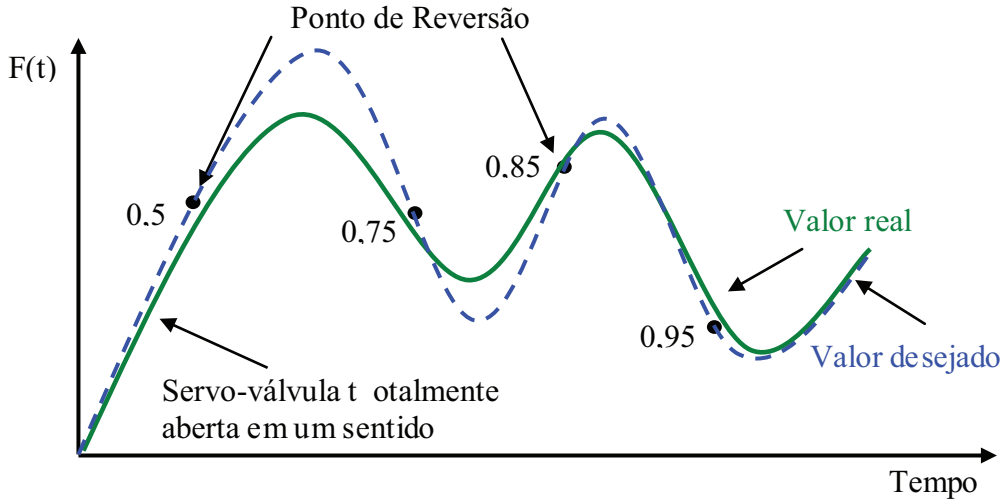


Figura 5.4. Pontos de reversão da servo-válvula.

5.3.2. Tabelas de Aprendizado

O algoritmo de controle por aprendizado acelerado tem duas tabelas, a primeira é a tabela de aprendizado que contém os valores de U_{ij}, e segunda é a tabela de momentum (ou gradientes) que contém os valores adimensionais de V_{ij}, definidos adiante. A tabela de aprendizado armazena valores adimensionais entre 0 e 1, que junto com outros parâmetros determina o valor de U_{IJ}, o qual representa o ponto de reversão como uma fração da amplitude do carregamento.

Assim, o ponto de reversão no qual a servo-válvula irá inverter seu sentido sempre se encontra antes de um pico ou de um vale, de maneira que o pico ou vale desejado é atingido pelos efeitos da dinâmica do sistema. E.g, se $U_{IJ} = 0.95$, então a mudança do sentido da servo-válvula será comandada quando 95% do trajeto entre o vale e o pico (ou entre o pico e o vale) tiver sido percorrido. Se este valor de U_{IJ} for ótimo, então espera-se que o movimento seja revertido após percorrer os 5% restantes do trajeto, devido aos efeitos da dinâmica do sistema.

Os valores de U_{ij} determinam a fração U_{IJ}, enquanto que os valores de V_{ij} permitem acelerar o processo de aprendizado. Estes dependem do valor de gama (o dobro da amplitude do carregamento) e de seu valor mínimo. Portanto, é

necessário de finir dua s t abelas pa ra U_{ij} e V_{ij} com valores ini ciais de 0, 5 e 0 respectivamente.

As co lunas da t abela de apr endizado r epresentam o v alor de ga ma d a magnitude física que se es teja co ntrolando, o nde ga ma é o do bro da am plitude desta magnitude física (que neste cas o é a força, e m kN). E e m cada linha s ão representados o s v alores mínimos, e m u m ciclo, da m esma magnitude física, como ilustrado na Tabela 1.

Colunas (gama)

Linhas (mínimo)	**-50**	**-30**	**-10**	**10**	**30**	**50**
-30	0,9810	0,9645	0,8795	0,8016	0,8742	0,9475
-15	0,9688	0,9415	0,8425	0,8245	0,8956	0,9515
0	0,9520	0,9230	U_{ij}	0,8526	0,9285	0,9756
15	0,9256	0,8910	0,7489	0,9189	0,9415	0,9889
30	0,9086	0,8723	0,6896	0,9387	0,9678	0,9915

Tabela 1. Tabela de aprendizado.

Na T abela 1, o s inal po sitivo as sociado ao v alor de ga ma indica q ue o sistema está indo de um vale para um pico, enquanto que o sinal negativo indica que o sistema está indo de um pico para um vale. Esta diferenciação è importante devido a e feitos n ão lineares, e. g. de zo na morta. A t abela de apr endizado é representada por uma matriz de dimensão especifica. Neste trabalho, assim como no t rabalho de A lva, e scolheu-se u ma matriz de 21 x 21 para e feito de comparação.

Levando e m consideração que o v alor máximo de força a lcançado pe la máquina de ensaio de fadiga estudada é 100 kN, a tabela de aprendizado utilizada tem seus limites de valores d e ga ma na faixa d e [-200,200]kN, e nquanto que o mínimo tem seus limites de valores na faixa de [-100,100]kN.

A outra tabela, que contém momentum V_{ij}, é definida do mesmo modo e com a mesma dimensão que a tabela de aprendizado. Assim, para cada valor de U_{ij} na tabela de aprendizado, há um correspondente V_{ij} na outra tabela, como é ilustrado na Tabela 2.

Tabela de Momentum — Colunas (gama)

	-50	-30	-10	10	30	50
-30	0.2810	0.058	-0.156	-0.0586	0.0451	0.0982
-15	.0854	-0.0958	-0.0015	0.0191		
0	Vij	0.0826	0.0985	0.0656		
15	.0035	-0.0919	0.1415	0.1589		
30	.0425	0.0387	-0.0678	0.0715		

Tabela de Aprendizado — Linhas (minimo)

	-50	-30	-10	10	30	50
-30	0.9810	0.9645	0.8795	0.8016	0.8742	0.9475
-15	0.9688	0.9415	0.8425	0.8245	0.8956	0.9515
0	0.9520	0.9230	Uij	0.8526	0.9285	0.9756
15	0.9256	0.8910	0.7489	0.9189	0.9415	0.9889
30	0.9086	0.8723	0.6896	0.9387	0.9678	0.9915

Tabela 2. Tabelas de aprendizado e de momentum.

Na Tabela 2, a tabela de momentum armazena valores positivos ou negativos que representam a gradiente de U_{ij}; o conhecimento destes gradientes permite acelerar o processo de aprendizagem.

5.3.3. Determinação do Valor de U_{IJ}

O valor de U_{IJ} é calculado a partir dos valores de U_{ij} armazenados na tabela de aprendizado para cada combinação de mínimo e gama. U_{ij} (com sub-índices em letras minúsculas) é definido como o elemento da tabela de aprendizado associado à linha i (para um valor mínimo igual a "min_i") e coluna j (valor de gama associada a "$gama_j$"). Assim, para um carregamento com valores de mínimo e gama que coincidam com algum valor de min_i e $gama_j$ da tabela, respectivamente, tem-se:

$$U_{IJ} = U_{ij} \tag{5.21}$$

Por exemplo, na tabela 1, para um $min_i = 15$ e um $gama_j = -10$ o valor de U_{IJ} seria $0,7489$. Na tabela de momentum, o valor do gradiente correspondente a

este carregamento utilizado no processo de aprendizado também coincidiria com V_{ij}.

No caso em que os valores de mínimo e gama do carregamento estejam entre dois valores discretos na tabela, e. g. $min_i < min < min_{i+1}$ e $gama_j < gama < gama_{j+1}$, o valor de U_{IJ} é determinado como uma interpolação dos elementos vizinhos $U_{i,j}$, $U_{i,j+1}$, $U_{i+1,j}$ e $U_{i+1,j+1}$, como ilustrado na Tabela 3.

Colunas (gama)

Linhas (mínimo)

			gama$_j$	**gama$_{j+1}$**		
	0.9810	0.9645	0.8795	0.8016	0.8742	0.9475
min $_i$	0.9688	0.9415	$U_{i,j}$	$U_{i,j+1}$	0.8956	0.9515
min $_{i+1}$	0.9520	0.9230	$U_{i+1,j}$	$U_{i+1,j+1}$	0.9285	0.9756
	0.9256	0.8910	0.7489	0.9189	0.9415	0.9889
	0.9086	0.8723	0.6896	0.9387	0.9678	0.9915

Tabela 3. Método de interpolação quando os valores de gama e mínimo estão entre duas células.

O valor interpolado de U_{IJ} é dada pela seguinte equação,

$$U_{IJ} = a + (b-a).\frac{(gama - gama_j)}{(gama_{j+1} - gama_j)} \tag{5.22}$$

onde os valores de a e b são determinados pelos elementos da vizinhança de $U_{i,j}$

$$a = U_{i,j} + (U_{i+1,j} - U_{i,j}).\frac{(min - min_i)}{(min_{i+1} - min_i)} \tag{5.23}$$

$$b = U_{i,j+1} + (U_{i+1,j+1} - U_{i,j+1}).\frac{(min - min_i)}{(min_{i+1} - min_i)} \tag{5.24}$$

As equações (5.22-5.24) permitem generalizar o cálculo de U_{IJ} para qualquer tipo de carregamento. Note que o valor de U_{IJ} obtido quando o sistema vai para um pico é em geral diferente do valor quando o sistema vai para um vale, devido à definição de gamas negativos.

Uma vez determinado o valor de U_{IJ}, o ponto de reversão da servo-válvula é determinado por

$$Ponto\_Reversão = \begin{cases} min + U_{IJ}.gama & (na\ subida) \\ (min + gama) - U_{IJ}.gama & (na\ descida) \end{cases} \tag{5.25}$$

Para cada um dos elementos da vizinhança de U_{ij}, é determinado na tabela de momentum seu respectivo valor do gradiente V_{ij}, bastando trocar U por V nas equações (5.22) a (5.24). O valor de V_{IJ} assim obtido é utilizado no processo de aprendizado para acelerar a convergência do erro, como descrito a seguir.

5.3.4. Processo de Aprendizado

No processo de aprendizagem, são feitas atualizações dos valores de U_{ij} seguindo uma lei de aprendizado. Os valores de U_{ij} e V_{ij} são atualizados em função do erro normalizado da grandeza controlada x (que pode ser e.g., força, deformação ou deslocamento), da taxa de aprendizagem, e do termo de momentum. O erro normalizado é o erro obtido entre o pico (ou vale) desejado x_d e o pico (ou vale) atingido x, dividido pela gama obtida entre o pico (ou vale) desejado x_d e o valor de pico (ou vale) atingido na reversão anterior x', ou seja:

$$erro = \frac{x_d - x}{x_d - x'} \tag{5.26}$$

Se o sistema for de um vale para um pico (subida), os valores de x e x_d estão perto do pico e x' terá sido um vale, portanto $x_d - x'$ se torna positivo.

Assim, se $x < x_d$ então o erro é positivo ($erro > 0$), apresentando um caso de *undershoot*, entanto se $x > x_d$ o erro é negativo ($erro < 0$), apresentando um caso de *overshoot*.

No caso em que o sistema for de pico para vale (descida), os valores de x e x_d estão perto do vale e x' terá sido um pico, portanto $x_d - x'$ se torna negativo. Neste caso, como o carregamento está diminuindo, se $x > x_d$ o erro é positivo ($erro > 0$), um o caso de *undershoot*, enquanto que se $x < x_d$ o erro é negativo ($erro < 0$), um caso de *overshoot*.

Assim, um erro positivo sempre está associado a um caso de *undershoot*, enquanto um erro negativo implica em um caso de *overshoot*, independentemente se o sistema sair de um pico ou vale. No caso de *overshoot*, é preciso que o ponto de reversão da servo-válvula seja antecipado, o que implica em diminuir os valores de U_{ij}. Por outro lado no caso de *undershoot* é preciso aumentar os valores de U_{ij}.

No caso em que o valor de gama e mínimo coincidem com os valores discretizados da tabela de aprendizado, de maneira que $U_{IJ} = U_{ij}$, é proposta a seguinte lei de aprendizado:

$$U_{IJ} := U_{IJ}(1 + \eta.erro + \Omega.V_{IJ}) \tag{5.27}$$

$$V_{IJ} := \eta.erro + \Omega.V_{IJ} \tag{5.28}$$

onde η é a taxa de aprendizado, Ω é o termo de momentum, e V_{IJ} é o valor obtido na tabela de momentum para a mesma fila e coluna nas quais foi obtido U_{IJ} na tabela de aprendizado.

O termo de momentum utilizado na lei de aprendizado é obtido em função do erro normalizado, resultando em

$$\Omega = \begin{cases} \dfrac{1}{1+exp^{-a(erro-b)}} & \text{se } erro \geq 0 \\ -\dfrac{1}{1+exp^{a(erro+b)}} & erro < 0 \end{cases} \tag{5.29}$$

onde a e b são constantes. Na Figura 5.5 mostra-se o termo de momentum em função do erro, para $a = 10$ e $b = 0,5$.

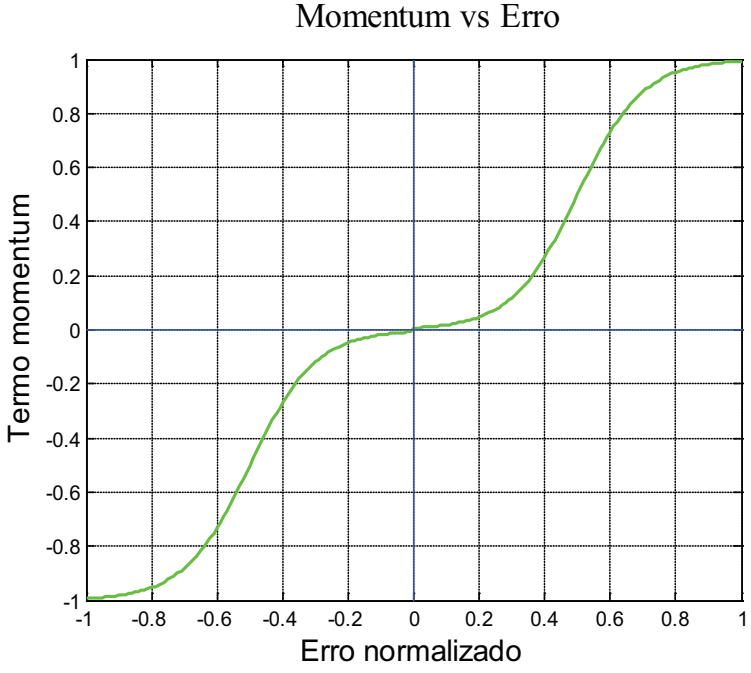


Figura 5.5. Termo de momentum em função do erro para $a = 10$ e $b = 0,5$.

Em geral, da Figura 5.4 pode-se observar que o valor do erro normalizado e do termo de momentum estão na faixa de $[-1, 1]$.

No caso em que U_{IJ} foi determinado por interpolação na tabela de aprendizado, é preciso atualizar os valores de $U_{i,j}$, $U_{i,j+1}$, $U_{i+1,j}$, $U_{i+1,j+1}$ que o geraram, e também os valores de $V_{i,j}$, $V_{i,j+1}$, $V_{i+1,j}$, $V_{i+1,j+1}$ que aceleram o processo de aprendizado. Essas atualizações são feitas em função do grau de influência que cada célula da vizinhança tem sobre U_{IJ}. Ou seja, as células da vizinhança que estão mais próximas de U_{IJ} têm um maior grau de atualização que aquelas mais distantes. Isto é ilustrado na Figura 5.6.

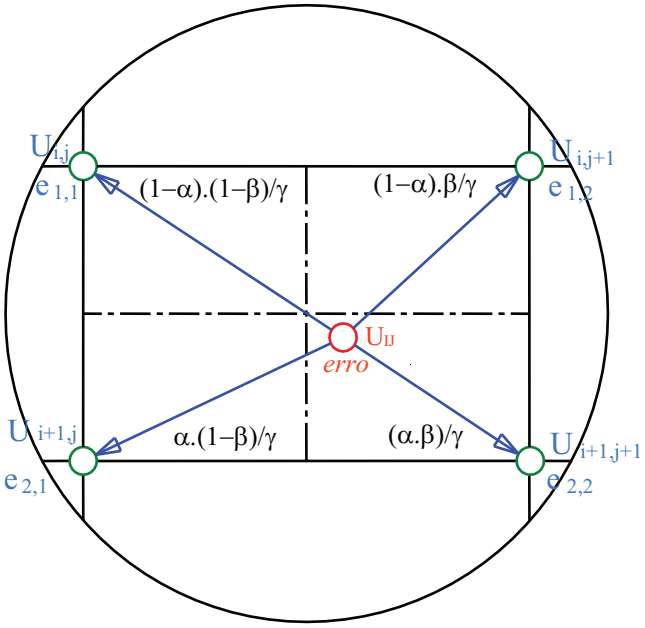


Figura 5.6. Esquema do grau de influência do erro.

O erro normalizado é distribuído nas células da vizinhança de U_{IJ} de acordo com o grau de influência. E.g, na Figura 5.6 o erro $e_{2,2}$ relativo à célula $U_{i+1,j+1}$ tem o maior grau de influência devido a que está célula esta mais perto de U_{IJ}. A variável γ é definida como o maior de todos os graus de influência, o que permite normalizar o grau de influência de maneira que a célula mais próxima receba um grau de atualização unitário.

$$\gamma := max(\ [(1-\alpha).(1-\beta)\ ,\ \alpha.(1-\beta)\ ,\ (1-\alpha).\beta\ ,\ \alpha.\beta]\ ) \tag{5.30}$$

onde

$$\alpha := \frac{min - min_i}{min_{i+1} - min_i} \quad , \quad 0 < \alpha < 1 \tag{5.31}$$

$$\beta := \frac{gama - gama_j}{gama_{j+1} - gama_j} \quad , \quad 0 < \beta < 1 \tag{5.32}$$

Assim, a distribuição do erro nas células da vizinhança é dada pelas equações:

$$e_{1,1} := \frac{(1-\alpha).(1-\beta)}{\gamma}.erro \tag{5.33}$$

$$e_{1,2} := \frac{\alpha.(1-\beta)}{\gamma}.erro \tag{5.34}$$

$$e_{2,1} := \frac{(1-\alpha).\beta}{\gamma}.erro \tag{5.35}$$

$$e_{2,2} := \frac{\alpha.\beta}{\gamma}.erro \tag{5.36}$$

O termo de momentum Ω é calculado em função do *erro* e para uma taxa de aprendizado η, as atualizações dos valores da tabela de momentum em função do erro distribuído em cada uma das células é dada pela seguinte lei de aprendizado.

$$V_{i,j} := \eta.e_{1,1} + \Omega.V_{i,j} \tag{5.37}$$

$$V_{i,j+1} := \eta.e_{1,2} + \Omega.V_{i,j+1} \tag{5.38}$$

$$V_{i+1,j} := \eta.e_{2,1} + \Omega.V_{i+1,j} \tag{5.39}$$

$$V_{i+1,j+1} := \eta.e_{2,2} + \Omega.V_{i+1,j+1} \tag{5.40}$$

As atualizações das células da tabela de aprendizado são então obtidas pela seguinte lei de aprendizado:

$$U_{i,j} := U_{i,j}.(1+V_{i,j}) \tag{5.41}$$

$$U_{i,j+1} := U_{i,j+1}.(1+V_{i,j+1}) \tag{5.42}$$

$$U_{i+1,j} := U_{i+1,j}.(1+V_{i+1,j}) \tag{5.43}$$

$$U_{i+1,j+1} := U_{i+1,j+1}.(1+V_{i+1,j+1}) \tag{5.44}$$

Note que o cálculo de U_{IJ} pelas equações (5.22) a (5.24) pode ser re-escrito em função de α e β, resultando em:

$$U_{IJ} := U_{i,j}.(1-\alpha).(1-\beta) + U_{i,j+1}.(1-\alpha).\beta + U_{i+1,j}.\alpha.(1-\beta) + U_{i+1,j+1}.\alpha.\beta\frac{\pi}{4} \tag{5.45}$$

5.3.5. Algoritmo de Aprendizado

O fluxograma do algoritmo de controle por aprendizado acelerado representado na Figura 5.7 apresenta os passos seguidos pelo fluxograma do controle desenvolvido por Alva[5], com a diferença que a lei utilizada para as atualizações de $U_{i,j}$ na tabela de aprendizado incluem a taxa η e o termo de momentum Ω, os quais permitem acelerar este processo.

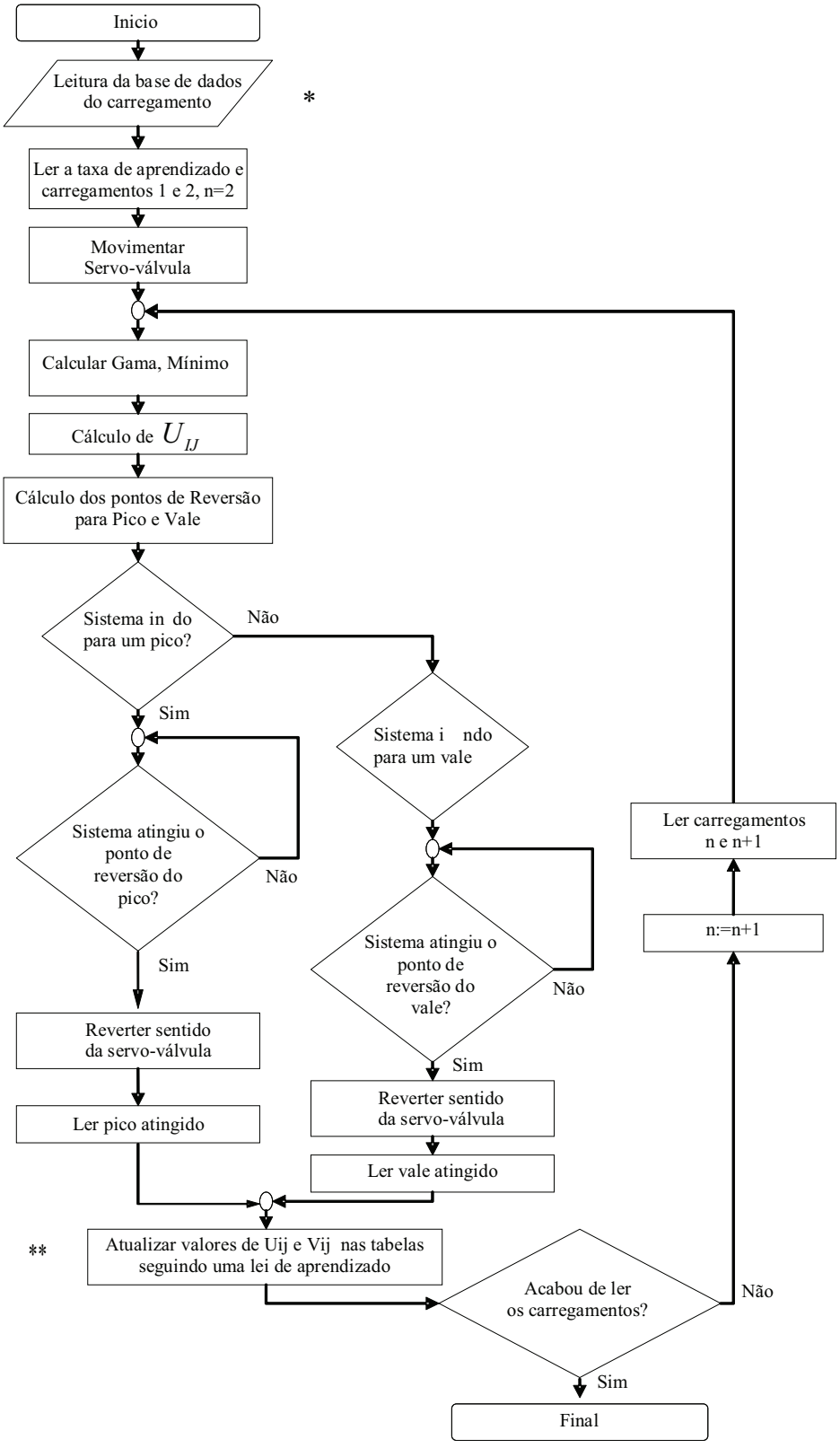


Figura 5.7. Algoritmo de controle por aprendizado acelerado.

* Os carregamentos são picos e vales requeridos para o ensaio de fadiga

** Para atualizar o valor U_{IJ} é calculado o erro do pico e/ou do vale e o termo de momentum.

5.3.6. Resultado das simulações do Algoritmo de Controle por Aprendizado Acelerado

As simulações do controle por aprendizado acelerado proposto foram feitas no software de MATLAB©. As simulações foram aplicadas à máquina servo-hidráulica modelada anteriormente. Escolheram-se para as simulações carregamentos de amplitude constante e variáveis, e com uma taxa de aprendizado $\eta = 0.95$, obtendo bons resultados, apresentados nas Figuras 5.8 e 5.9.

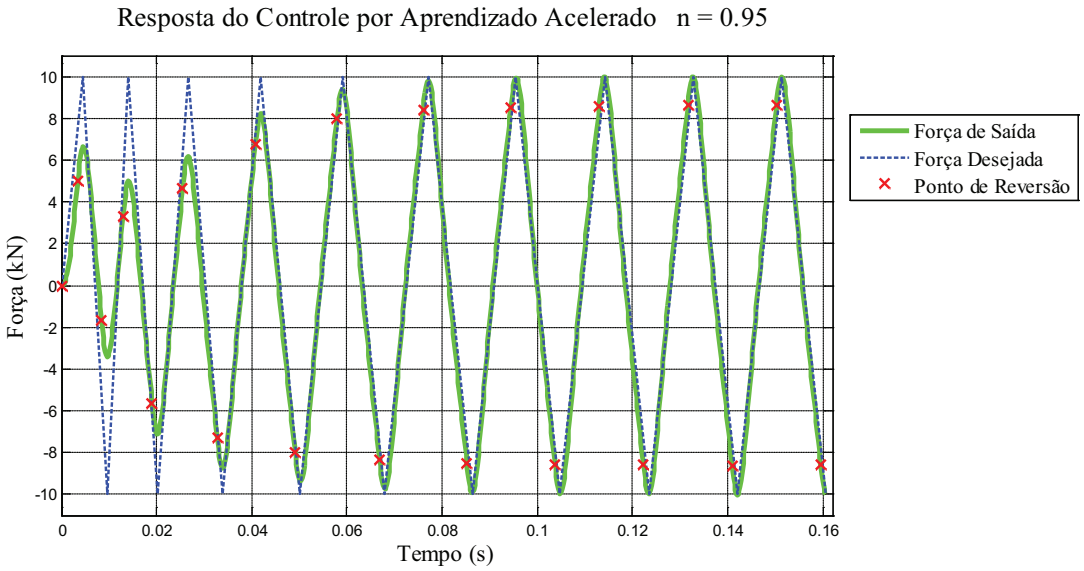


Figura 5.8. Resposta do controle por aprendizado acelerado para um carregamento de amplitude constante de $\pm$ 10 kN .

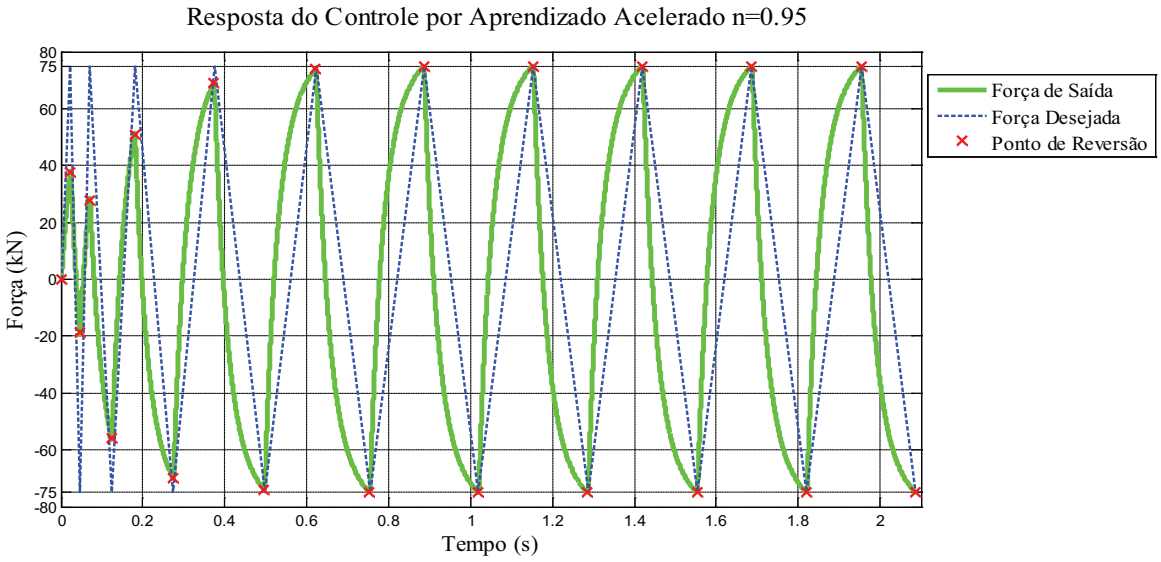


Figura 5.9. Resposta do controle por aprendizado acelerado para um carregamento de amplitude constante de $\pm$ 75 kN .

Nas Figuras 5.8 e 5.9 pode-se notar que os pontos de reversão são modificados até convergir em um valor ótimo de U_{IJ} que minimiza o erro entre os valores (de pico e vales) reais e desejados. Assim, no futuro, para carregamentos de mesma amplitude, o controlador poder responder satisfatoriamente sem a necessidade de reaprendizado, desde que não haja alterações significativas no sistema. Note que para altos carregamentos desejados, como nos ±75 kN da Figura 5.9, a forma da curva da força de saída é muito diferente da forma da curva desejada, no entanto sob o ponto de vista de ensaios de fadiga isto é irrelevante, pois somente os valores dos picos e vales atingidos é que influenciam na vida da peça.

No caso em que haja mudança nas amplitudes do carregamento, o algoritmo de aprendizado ajusta automaticamente os pontos de reversão, como é mostrado na Figura 5.10. Todos os pontos de reversão ótimos U_{IJ} obtidos ao longo do funcionamento do controlador são armazenados na tabela de aprendizado. Assim, no futuro, o sistema não tem a necessidade de reaprendizado para carregamentos que já tiverem sido apresentados anteriormente.

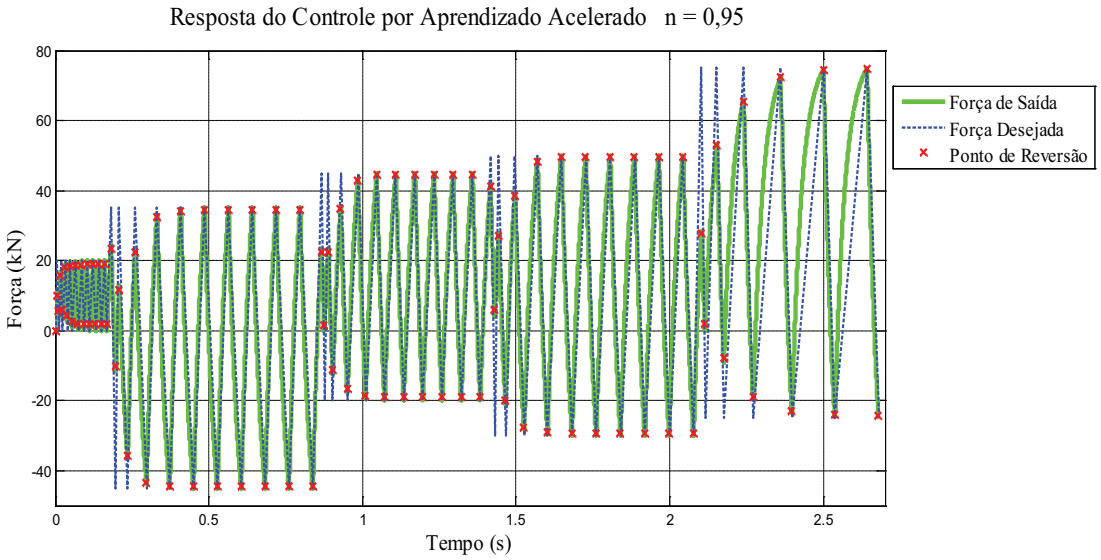


Figura 5.10. Resposta do controle por aprendizado acelerado para diferentes amplitudes de carregamento.

A Figura 5.11 apresenta a resposta do controle por aprendizado acelerado para carregamentos que já foram apresentados anteriormente. Pode-se notar que apresenta um ótimo desempenho desde o primeiro evento, sem a necessidade de um transiente de aprendizado.

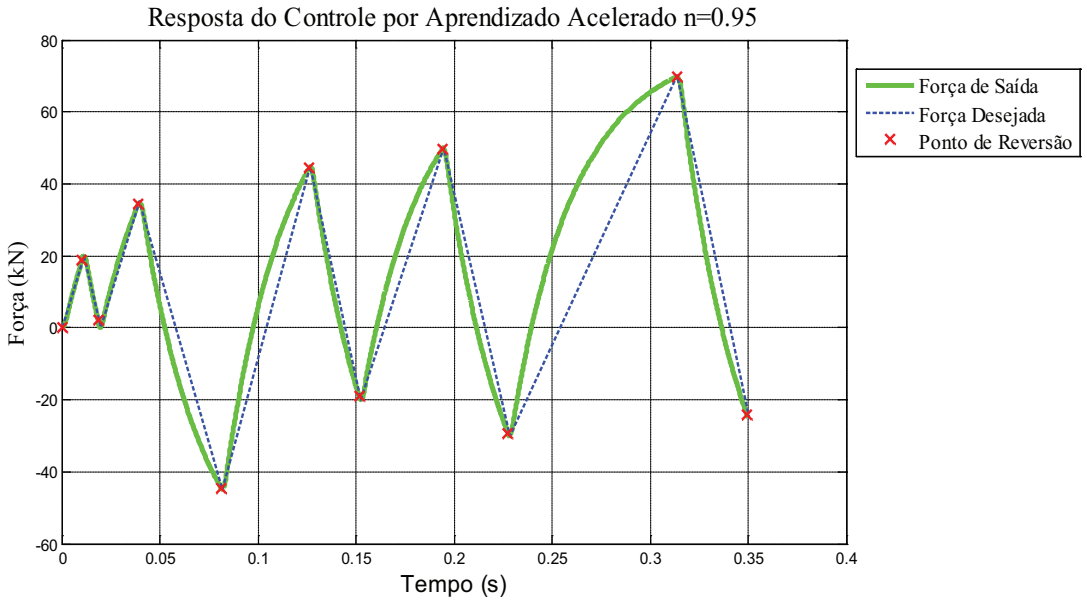
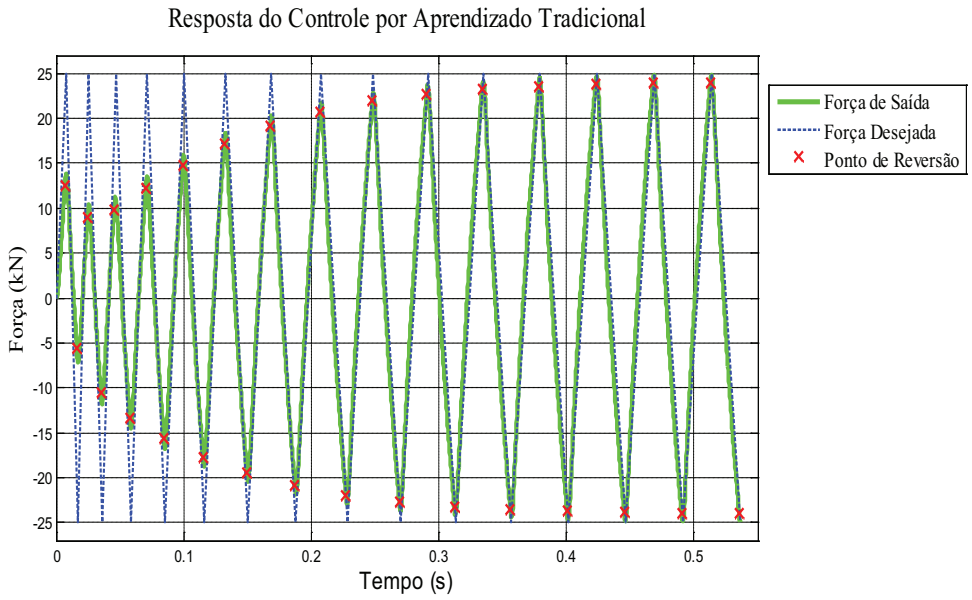


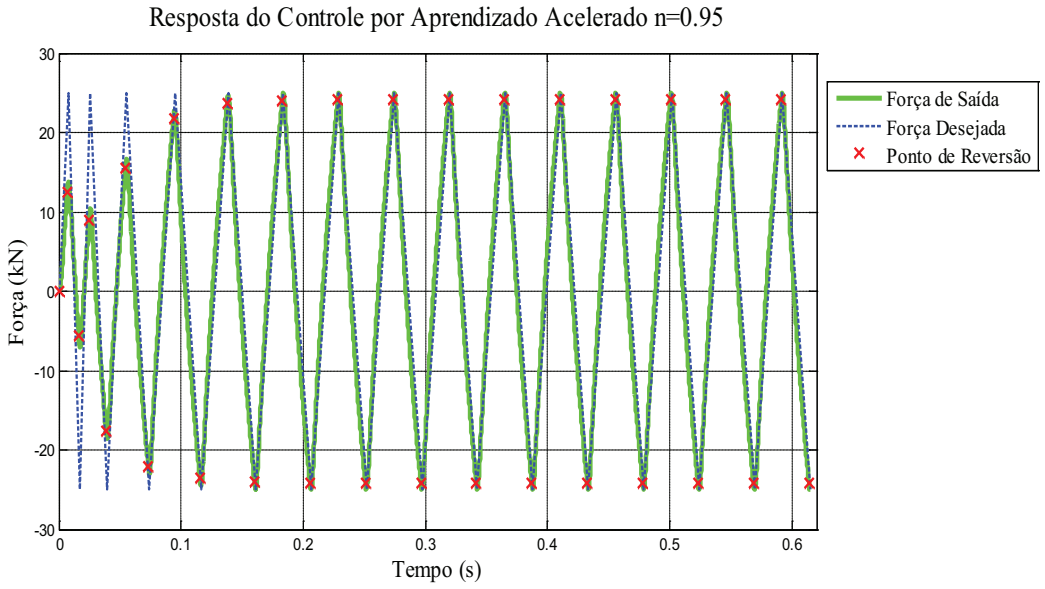
Figura 5.11. Resposta do controle por aprendizado acelerado para carregamento já apresentados.

Na figura 5.9 e 5.11 mostra-se que força de saída (“linha verde”) apresenta uma forma curva a qual é mais notória para amplitudes grandes. Entretanto a forma da curva da força de saída não é relevante nos ensaios de fadiga (sempre que não tenha envolvido efeitos de corrosão e fluência).

Na Figura 5.12 é apresentada a comparação entre o controle por aprendizado desenvolvido por Alva [5] e o controle por aprendizado acelerado proposto nesta dissertação. O controle por aprendizado acelerado tem uma maior velocidade de aprendizagem em relação ao controle por aprendizado de Alva [5].



(a)



(b)

Figura 5. 12. Resposta para um carregamento constante de ±25 kN: (a) Controle por aprendizado de Alva e (b) Controle por aprendizado acelerado.

Na Figura 5. 13 é apresentada a velocidade de convergência do erro normalizado do controle por aprendizado de Alva [5] (“linha vermelha”) e o erro normalizado do controle por aprendizado acelerado (“linha azul”). O controle por aprendizado acelerado tem uma rápida convergência do erro em relação ao controle por aprendizado de Alva, convergindo após 5 ciclos, enquanto que o controle por aprendizado de Alva convergiu após 12 ciclos.

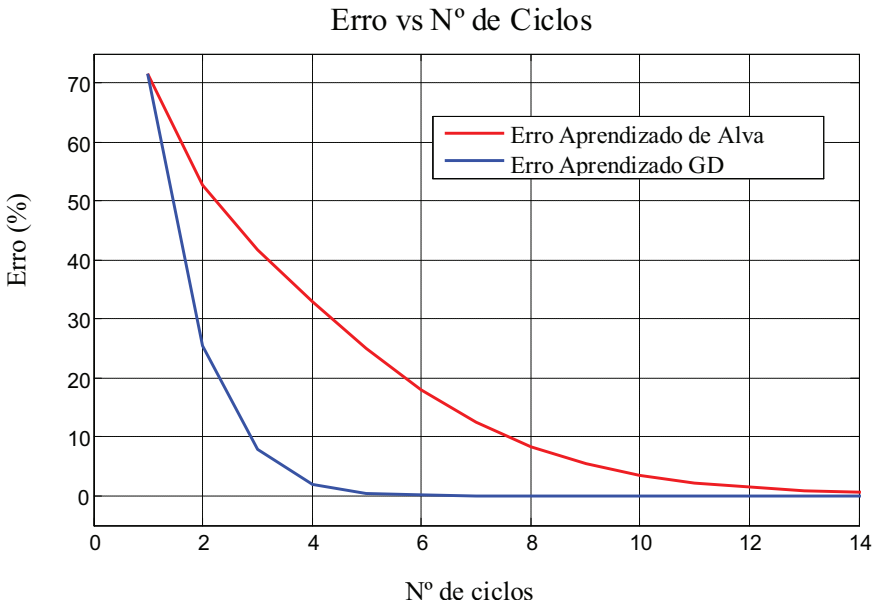


Figura 5.13. Erro do controle por aprendizado de Alva e aprendizado acelerado.

No próximo capitulo será apresentado o controle por aprendizado Neuro-Fuzzy, assim como as simulações e comparações entre os três métodos de controle descrito nesta dissertação.

6.
Controle por Aprendizado Neuro-Fuzzy

6.1. Introdução

Neste capítulo é apresentado o controle por aprendizado utilizando um sistema híbrido Neuro-Fuzzy, para o cálculo e atualização dos pontos de reversão da servo-válvula do sistema servo-hidráulico. Além disso, são apresentadas as comparações entre os modelos de controle por aprendizado desenvolvido. Todos os programas de simulação foram desenvolvidos no software MATLAB©.

6.2. Esquema do controle por aprendizado NF

Neste modelo de controle por aprendizado, a informação desconhecida é estimada ou aprendida pelo sistema Neuro-Fuzzy e fornecida ao controlador "*bang-bang*". Assim, à medida que o sistema Neuro-Fuzzy armazena mais informação, o controlador é atualizado melhorando o desempenho do sistema.

Na Figura 6.1 apresenta-se o diagrama de blocos que ilustra o controle por aprendizado Neuro-Fuzzy. Neste modelo de controle, a informação é representada pela mesma variável U_{IJ} definida anteriormente, e que nesse caso é a saída do sistema Neuro-Fuzzy. A informação que gera o valor de U_{IJ} é armazenada nos pesos da estrutura do sistema Neuro-Fuzzy. A variável U_{IJ}, utilizada para mudar a ação de controle sobre a servo-válvula, é atualizada após cada ciclo de operação através do ajuste dos pesos da estrutura Neuro-Fuzzy, utilizando um algoritmo de aprendizado baseado nos erros medidos.

O objetivo do sistema Neuro-Fuzzy é fornecer o valor de U_{IJ} ao controlador, determinando assim o ponto de reversão no qual a servo-válvula vai reverter seu sentido, de maneira que a máquina (o sistema servo-hidráulico) seja mantida trabalhando em seus limites de operação como nos controladores descritos no capítulo anterior.

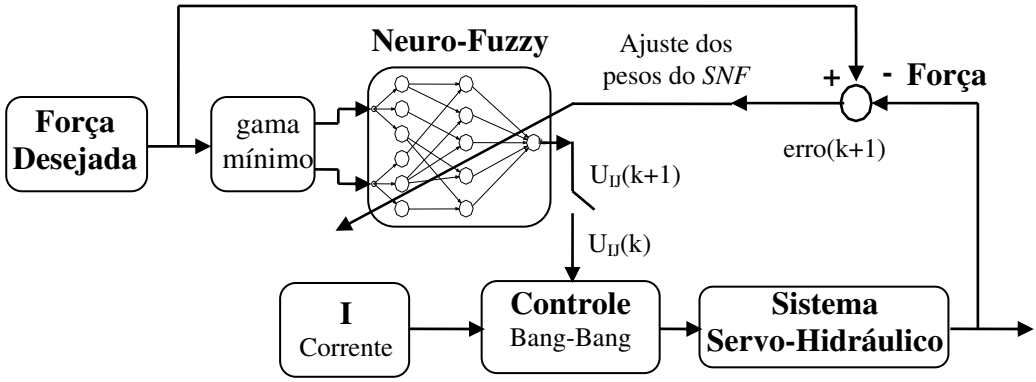


Figura 6.1. Diagrama de blocos do controle por aprendizado Neuro-Fuzzy.

Na Figura 6.1, o valor de $U_{IJ}(k)$ é atualizado pela chave com o valor de $U_{IJ}(k+1)$, e o erro normalizado $erro(k+1)$ é obtido em função da força desejada e da força real medida.

As entradas do sistema Neuro-Fuzzy são *gama* (dobro da amplitude) e *mínimo* da grandeza controlada, a saída do sistema é a variável U_{IJ}. O sistema Neuro-Fuzzy é constituído por duas camadas escondidas: a camada Fuzzy e a camada de regras. Os pesos do sistema Neuro-Fuzzy ω_j entre a camada de regras e a camada de saída são atualizados utilizando o algoritmo de aprendizado *Backpropagation*, baseado no $erro(k+1)$ a cada iteração. Na Figura 6.2 apresenta-se a estrutura do sistema Neuro-Fuzzy proposto.

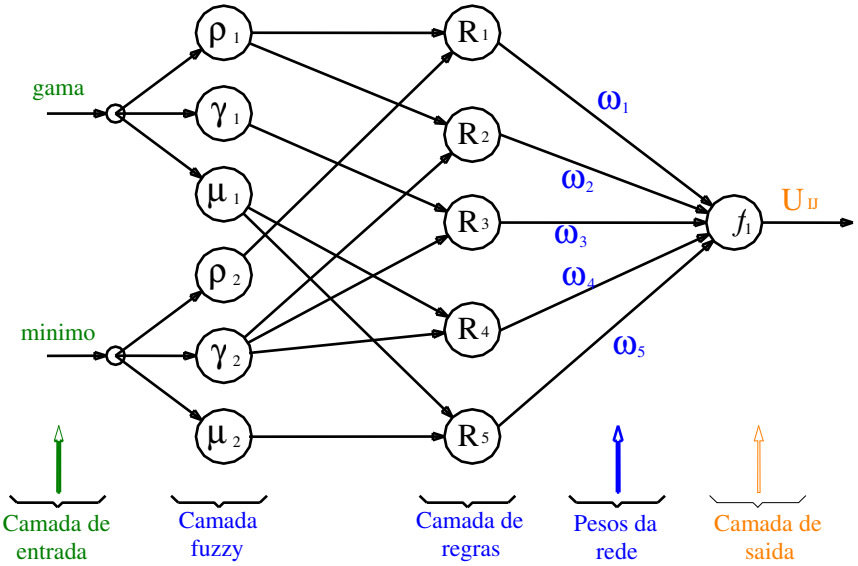


Figura 6.2. Estrutura do Sistema Neuro-Fuzzy.

6.3. Modelagem do Controle por Aprendizado Neuro-Fuzzy

O controle por aprendizado baseado em sistemas híbridos Neuro-Fuzzy (*SNF*) geralmente é constituído pela combinação de sistemas Fuzzy e Redes Neurais (*ANN*). Ele combina as vantagens das *ANN* tais como a habilidade de aprendizagem, otimização, e conexão em estrutura, com as vantagens dos sistemas Fuzzy, que usa raciocínio semelhante ao humano, com facilidade de incorporar informações de especialistas.

A modelagem deste sistema é determinada pelas configurações das características de seus "genitores", a parte Fuzzy e Neural.

6.3.1. Modelagem Fuzzy do *SNF*

A modelagem das características da parte Fuzzy é determinada pela configuração dos parâmetros das seguintes 4 categorias:

- ***Modelo Fuzzy:*** O modelo de inferência Fuzzy (formato das regras) implementado neste *SNF* foi o modelo Takagi-Sugeno, com conjuntos Fuzzy de saída do tipo "*singlenton*" para cada uma das regras. As regras fuzzy são da seguinte forma:

$$\text{Regra j : Se "}gama\text{" é } \rho_1 \text{ e "}mínimo\text{" é } \mu_2 \text{ Então } U_{IJ} = \omega_j \qquad (6.1)$$

- ***Formato das Funções de Pertinência:*** As funções de pertinência na camada Fuzzy do *SNF* são geralmente funções simétricas, tais como funções triangulares, trapezoidais, e sigmóidais. Nesta dissertação, escolhem-se 8 funções de pertinência do tipo triangular para cada uma das variáveis de entrada, pela simplicidade para sua implementação experimental. Na Figura 6.3, ilustra-se a função de pertinência triangular, cuja equação é expressa por:

$$\mu(x_i) = 1 - \frac{2|x_i - c_i|}{b_i} \qquad (6.2)$$

onde, x_i é o valor da variável de entrada, c_i é o centro do triângulo da função de pertinência, e b_i é a largura da base do triângulo.

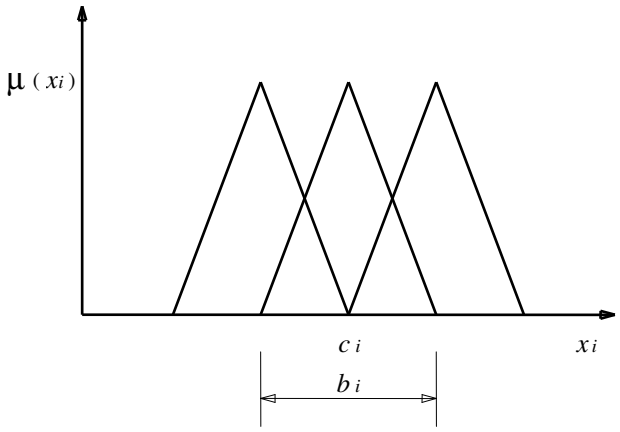


Figura 6.3. Função de pertinência triangular do *SNF*.

- ***Particionamento do espaço de E/S:*** O particionamento do espaço das variáveis de entrada e saída é do tipo fuzzy grid, o qual mapeia internamente regiões fuzzy relacionados através de suas regras. Na Figura 6.4 apresenta-se o particionamento fuzzy grid.

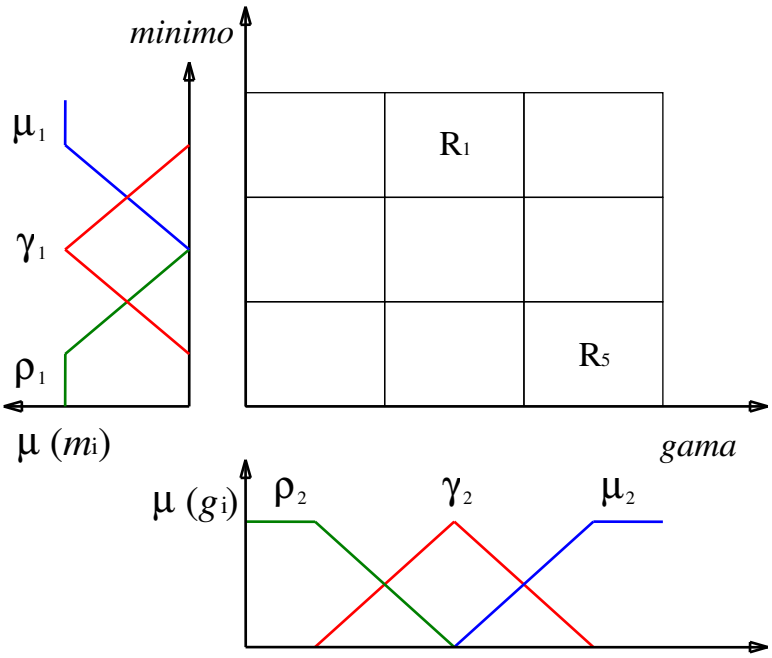


Figura 6.4. Particionamento Fuzzy Grid, onde ρ_1, γ_1, μ_1 e ρ_2, γ_2, μ_2, são os graus de pertinência dos conjuntos Fuzzy das variáveis "*minimo*" e "*gama*" respectivamente.

- ***Método de defuzzificação:*** O conseqüente das regras é uma função do tipo *"singlenton"*, e a saída do sistema, U_{IJ}, é obtida da média ponderada dos graus de disparo de cada regra. A saída U_{IJ} é calculada por:

$$U_{IJ} = f(P(\mu_i, \mu_i), \omega) \tag{6.3}$$

onde $P(\mu_i, \mu_i)$ é o nível de disparo correspondente a cada regra, e ω é o peso da estrutura (saída "*singlenton*").

6.3.2. Modelagem da parte neural

A modelagem das características da parte neural está relacionada com a capacidade de aprendizado do *SNF*, e determinada pela configuração dos parâmetros das seguintes sub-clases:

- ***Tipo de aprendizado:*** O aprendizado utilizado é do tipo *online*, isto devido a que ao longo da operação da máquina o sistema tem que ter a capacidade de mudar os pontos de reversão, de maneira que os pesos do SNF têm que ser atualizados a cada iteração.

- ***Identificação da estrutura.*** O número de regras é determinado pela combinação dos conjuntos fuzzy das variáveis de entrada.

- ***Identificação dos parâmetros.*** Neste modelo de SNF só apresentam aprendizado nos parâmetros do conseqüente de cada regra, baseado na medida do erro normalizado.

6.4. Controle por Aprendizado Neuro-Fuzzy

6.4.1. Calculo do valor de U_{IJ}

O valor da variável adimensional U_{IJ} é obtido como o resultado da avaliação do *SNF* para cada combinação de *mínimo* e *gama*. Assim, para um carregamento com valor mínimo igual a min_i, e gama igual a $gama_j$, tem-se U_{IJ} como saída do *SNF*.

O carregamento gerado pelo sistema servo-hidráulico está na faixa de $[-100,100]$kN. Assim, os valores da variável de entrada, *mínimo* e *gama* esta na faixa de $[-100,100]$ e $[-200,200]$ respectivamente. Os valores de gama são positivos quando o sistema esta indo de um vale para um pico e negativo quando vai de um pico para um vale. Estas variáveis de entradas do SNF (*mínimo* e *gama*) são normalizadas na faixa de [-1,1] utilizando a Equação 6.4.

$$x_n = \frac{2.(x - \min)}{Max - \min} - 1 \tag{6.4}$$

Onde, x_n é o valor normalizado da variável x, min e Max são os valores de mínimo e Maximo da variável x.

A normalização das variáveis de entrada, *mínimo* e *gama* e feito substituindo na Equação 6.4 e obtêm-se.

$$min_n = \frac{min}{100} \tag{6.5}$$

$$gama_n = \frac{gama}{200} \tag{6.6}$$

Depois da normalização das variáveis de entrada na camada de entrada, vide a Fig. 6.5, na camada fuzzy calcula-se o grau de pertinência com que as entradas satisfazem aos conjuntos fuzzy associado a cada entrada.

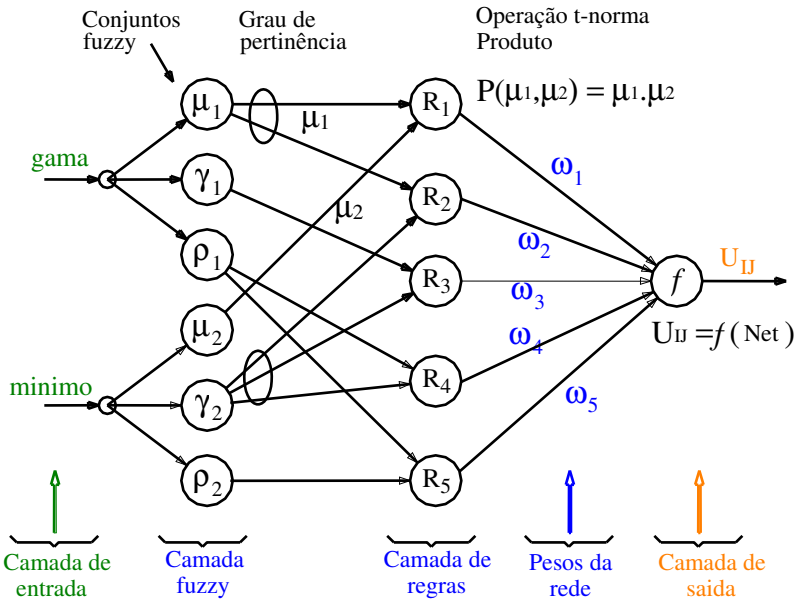


Figura 6.5. Calculo de U_{IJ} e descrição das camadas do SNF.

Na camada de regras, calcula-se o nível de disparo correspondente a cada um das regras, executando a operação *t-norm* (produto).

$$P_k(\mu_i, \mu_j) = \mu_i.\mu_j \tag{6.7}$$

Na camada de saída calcula-se o valor de U_{IJ} em função do $P_k(\mu_i, \mu_j)$ e ω_k.

$$U_{IJ} = f(Net)$$

$$Net = \frac{\sum_{k=1}^{M} P_k(\mu_i, \mu_j).\omega_k}{\sum_{k=1}^{M} P_k(\mu_i, \mu_j)} \tag{6.8}$$

Nesta dissertação, considero-se f como função de ativação linear pela facilidade na implementação experimental, mas pode-se aplicar outras funções do tipo sigmoidais (logsig ou tansig).

$$U_{IJ} = \frac{\sum_{k=1}^{M} P_k(\mu_i, \mu_j).\omega_k}{\sum_{k=1}^{M} P_k(\mu_i, \mu_j)} \tag{6.9}$$

Onde, $P_k(\mu_i, \mu_j)$ é o resultado da operação *t-norm* na camada de regras e ω_k é o peso de conexão da regra k e o neurônio de saída.

As equações apresentadas acima determinam o valor de U_{IJ} para qualquer tipo de carregamento. Uma vez calculado o valor de U_{IJ}, os pontos de reversão da servo-válvula são calculados, pela seguinte equação:

$$Ponto\_Reversão = \begin{cases} min + U_{IJ}.gama & (na\ subida) \\ (min+gama) - U_{IJ}.gama & (na\ descida) \end{cases} \tag{6.10}$$

6.4.2. Lei de aprendizado do *SNF*

O aprendizado do SNF é feito pelas atualizações dos pesos ω_k em um instante seguinte com o valor atual. Todos os valores de ω_k são inicializados com 0,5 e depois atualizados em função do erro normalizado, a taxa de aprendizado e o nível de disparo de cada um das regras. O erro normalizado é o erro obtido entre o pico (ou vale) desejado x_d e o pico (ou vale) atingido x, dividido pela *gama* obtida entre o pico (ou vale) desejado e o valor de pico (ou vale) atingido na reversão anterior x', como foi definido na Equação 5.26.

De maneira semelhante ao controle por aprendizado acelerado, se x e x_d forem picos, x' terá sido um vale. Assim, no caso que $x < x_d$ e o $erro > 0$, tem-se *undershoot*. E no caso que $erro < 0$, tem-se *overshoot*. Entretanto, se x e x_d forem vales, x' terá sido um pico. Assim, no caso que $x > x_d$, e $erro > 0$, tem-se *undershoot*. E no caso que $erro < 0$, tem-se *overshoot*.

Conclui-se assim que erros positivos estão associados ao *undershoot*, e negativos ao *overshoot*, tanto na subida (vale – pico) como na descida (pico-vale).

Pelo geral, o valor de *erro* esta na faixa de [-1,1] e o algoritmo de atualização dos pesos ω_k do *SNF* é dado pela seguinte lei de aprendizado.

$$\omega_k(t+1) = \omega_k(t) + \Delta\omega_k(t) \tag{6.11}$$

$$\Delta\omega_k(t) = \eta.erro.\frac{P_k(\mu_i,\mu_j)}{max[P_k(\mu_i,\mu_j)]} \tag{6.12}$$

Onde, $\omega_k(t)$ peso de conexão correspondente á regra k, η é a taxa de aprendizado, erro é o erro normalizado e $P_k(\mu_i,\mu_j)$ é o nível de disparo correspondente á regra k.

6.4.3. Algoritmo de controle por aprendizado Neuro-Fuzzy

O fluxograma do algoritmo de controle por aprendizado Neuro-Fuzzy apresenta os mesmos passos seguidos pelo fluxograma do controle por aprendizado acelerado, com a diferença que o valor de U_{IJ} é determinado como o valor de saída da avaliação do *SNF*, e a lei de aprendizado atualiza os pesos ω_k do *SNF*, a qual incluem uma taxa de aprendizado η e um termo de momentum Ω que permite acelerar o processo de aprendizado.

Na Figura 6.6 é apresentado o fluxograma do algoritmo de controle por aprendizado Neuro-Fuzzy.

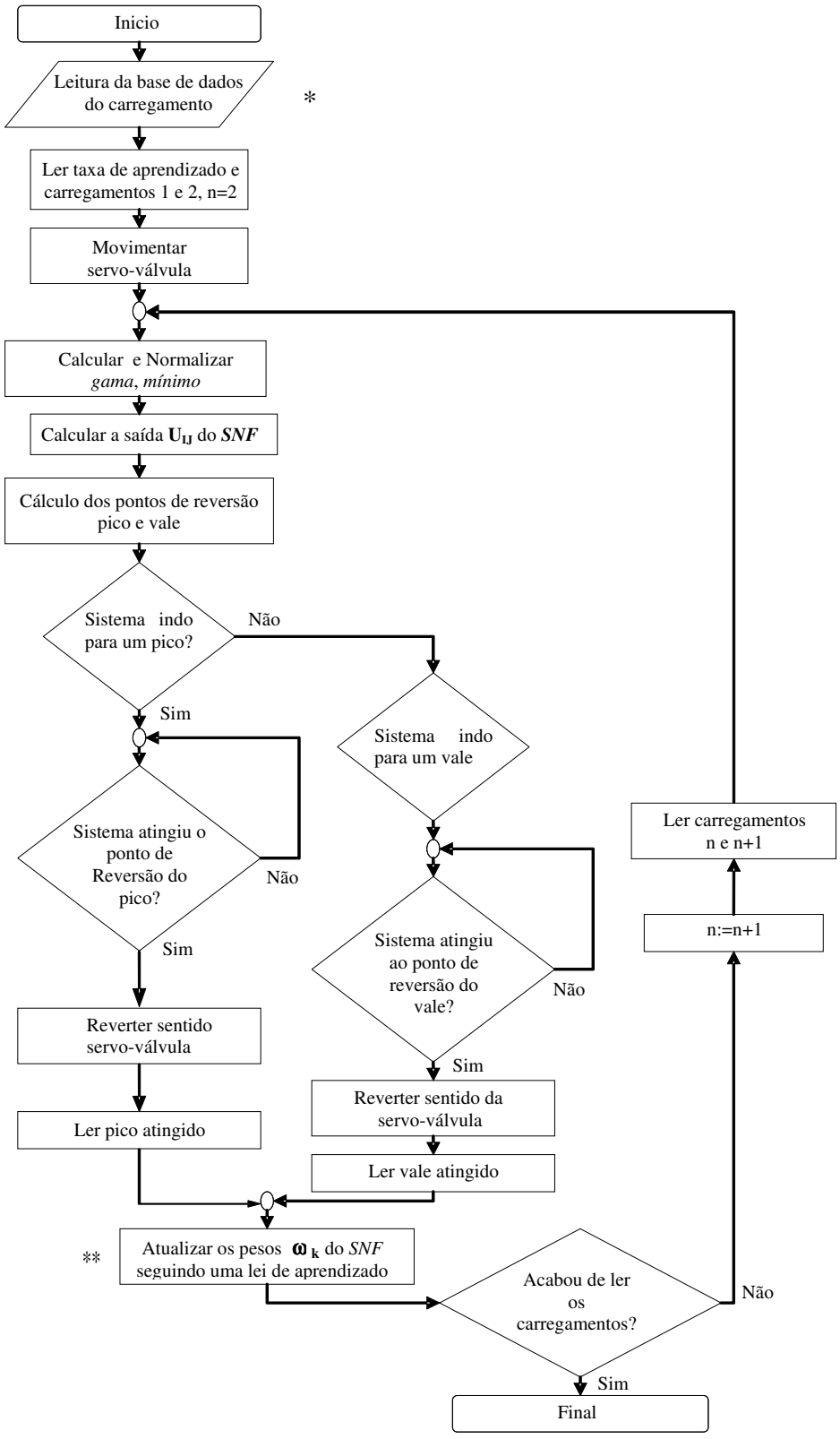


Figura 6.6. Algoritmo de controle por aprendizado Neuro-Fuzzy.

* Os carregamentos são picos e vales requeridos para o ensaio de fadiga

** Para atualizar os pesos ω_k é calculado o erro do pico e/ou vale e usada uma lei de aprendizado.

6.4.4.Resultado das simulações do controle por aprendizado *SNF*

As simulações do controle por aprendizado Neuro-Fuzzy proposto foram feitos no software de Matlab©. Para as simulações escolheram-se carregamentos de amplitude constantes e variáveis, obtendo melhores resultados que os obtidos pelas leis de aprendizado, vide Figuras 6.7 e 6.8.

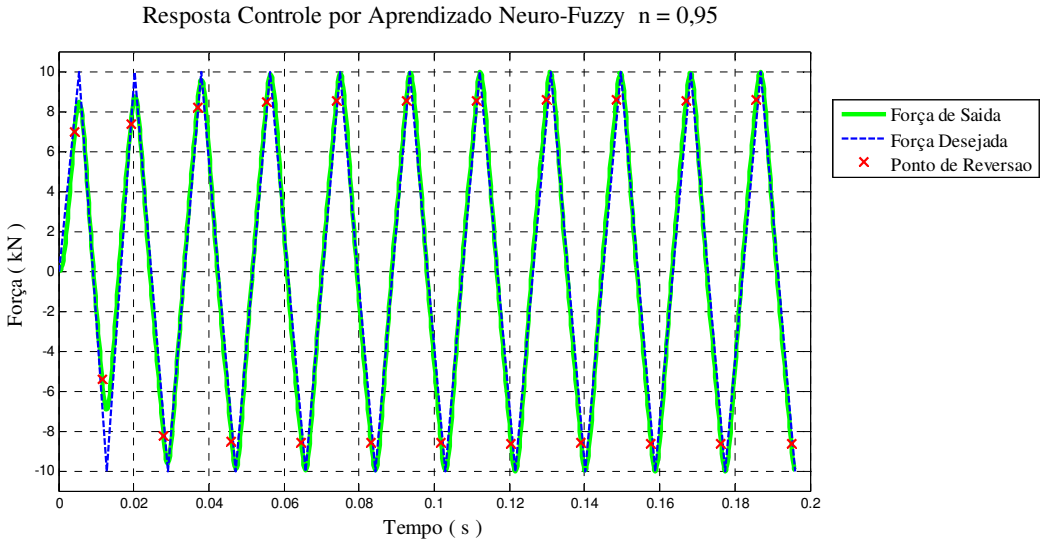


Figura 6.7. Resposta do controle por aprendizado Neuro-Fuzzy para um carregamento de amplitude constante de $\pm$ 10 kN.

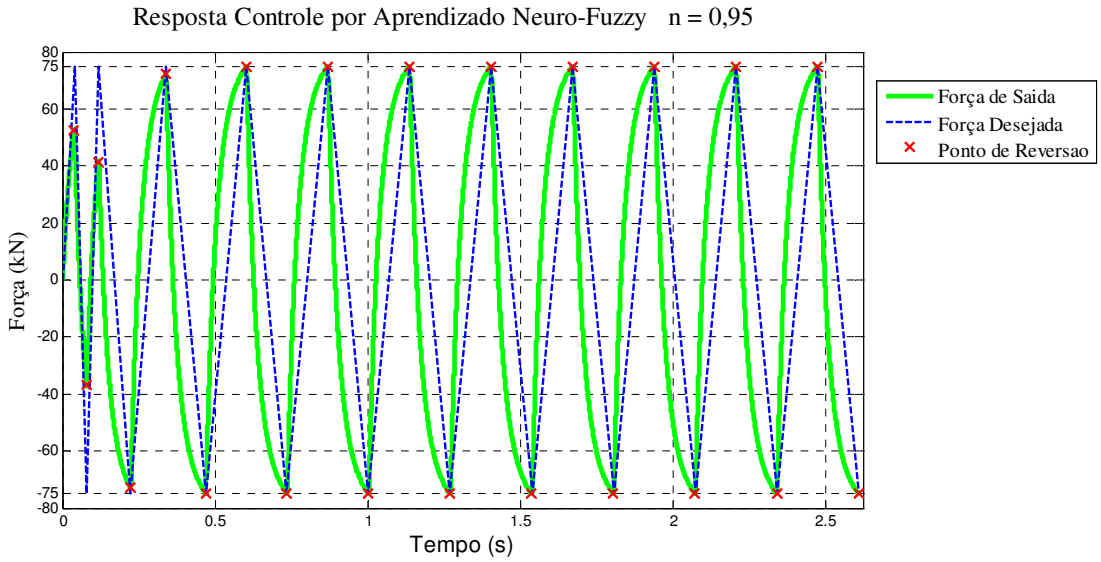


Figura 6.8. Resposta do controle por aprendizado Neuro-Fuzzy para um carregamento de amplitude constante de $\pm$ 75 kN.

Nas Figuras 6.7 e 6.8, mostram-se as simulações para carregamentos de

amplitude constante de ± 10 kN e ± 75 kN, respectivamente, com uma taxa de aprendizado $\eta = 0.95$. Os pontos de reversão de pico/vale são modificados à medida que são apresentados novos carregamentos, até convergir em um valor ótimo de reversão. Assim, no futuro para carregamentos da mesma amplitude, o controlador pode responder satisfatoriamente sem a necessidade de reaprendizado.

Na Figura 6.9 é apresentado o desempenho do controle por aprendizado Neuro-Fuzzy para carregamentos de amplitude variável. O algoritmo de aprendizado atualiza os pesos da estrutura Neuro-Fuzzy para cada combinação de *mínimo* e *gama*. Assim, os valores dos pontos de reversão são atualizados após cada ciclo até alcançar pontos de reversão ótimos, como mostrado na figura.

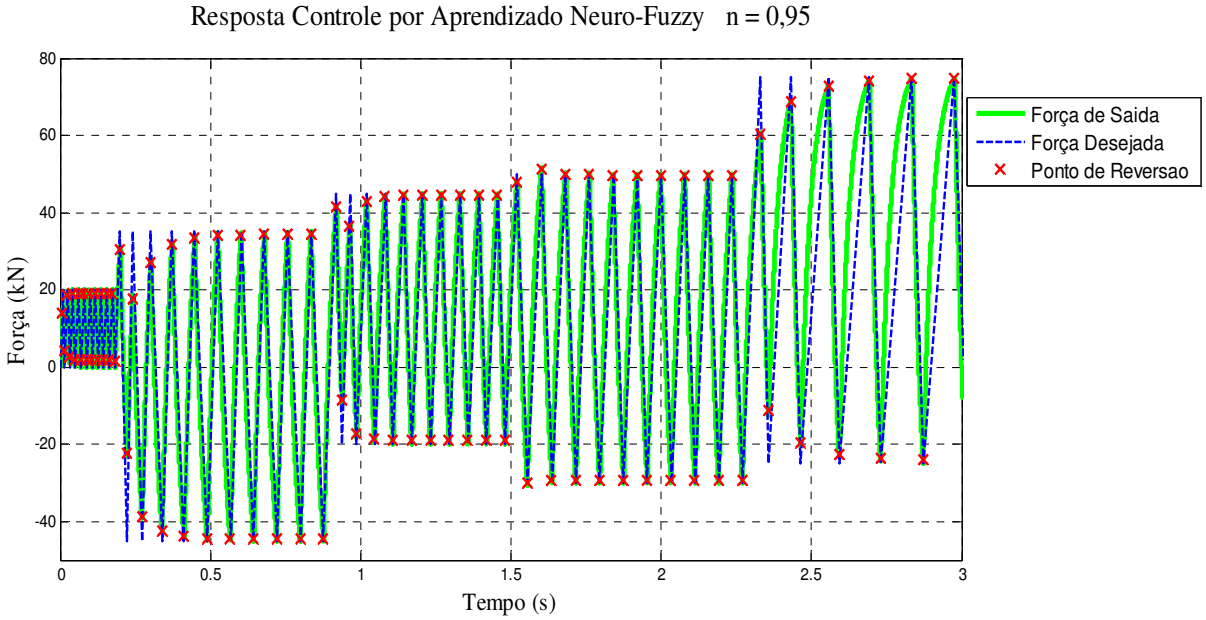


Figura 6.9. Resposta do controle por aprendizado Neuro-Fuzzy para diferentes amplitudes de carregamento.

Toda a informação obtida ao longo da operação do controlador é armazenada nos pesos da estrutura do sistema Neuro-Fuzzy. Portanto, no futuro o sistema não tem a necessidade de reaprendizado para carregamentos que já tiverem sido apresentados anteriormente.

Na figura 6.7-6.8 e 6.9 mostra-se que a força de saída não ultrapassa à força desejada, isto devido á que nos ensaios de fadiga o overshoot (ultrapassa a força

desejada) é indesejado. Entretanto, o undershoot (não ultrapassa a força desejada) é tolerável.

A Figura 6.10 apresenta a resposta do controle por aprendizado Neuro-Fuzzy para um carregamento que já tinham sido apresentados 5 vezes anteriormente. Pode-se notar que o sistema apresenta um ótimo desempenho desde o primeiro ciclo.

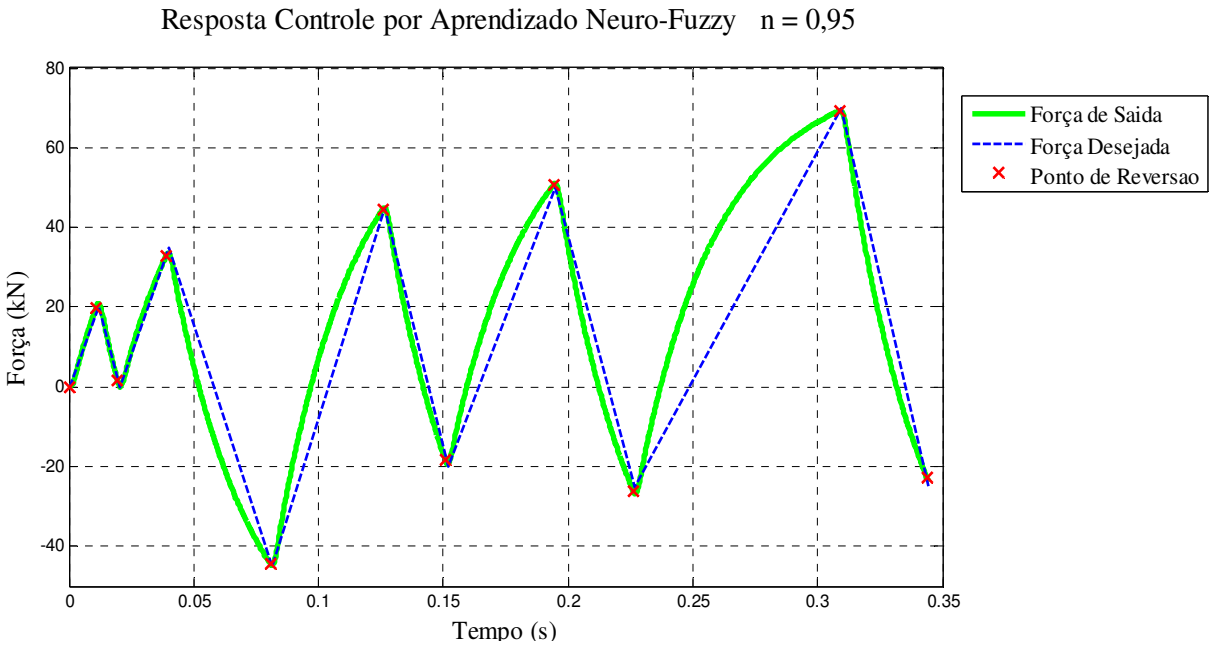


Figura 6.10. Resposta do controle por aprendizado Neuro-Fuzzy para carregamentos já apresentados anteriormente.

Na Figura 6.11, as curvas representam a convergência do erro normalizado ao longo do processo de aprendizagem. A linha "vermelha" representa a convergência do erro para o controle por aprendizado de Alva, enquanto a linha "azul" representa a convergência do aprendizado acelerado, e a linha "verde" do aprendizado Neuro-Fuzzy. Pode-se notar que o aprendizado Neuro-Fuzzy tem rápida convergência, com uma maior velocidade de aprendizado em relação aos outros dois modelos de controle.

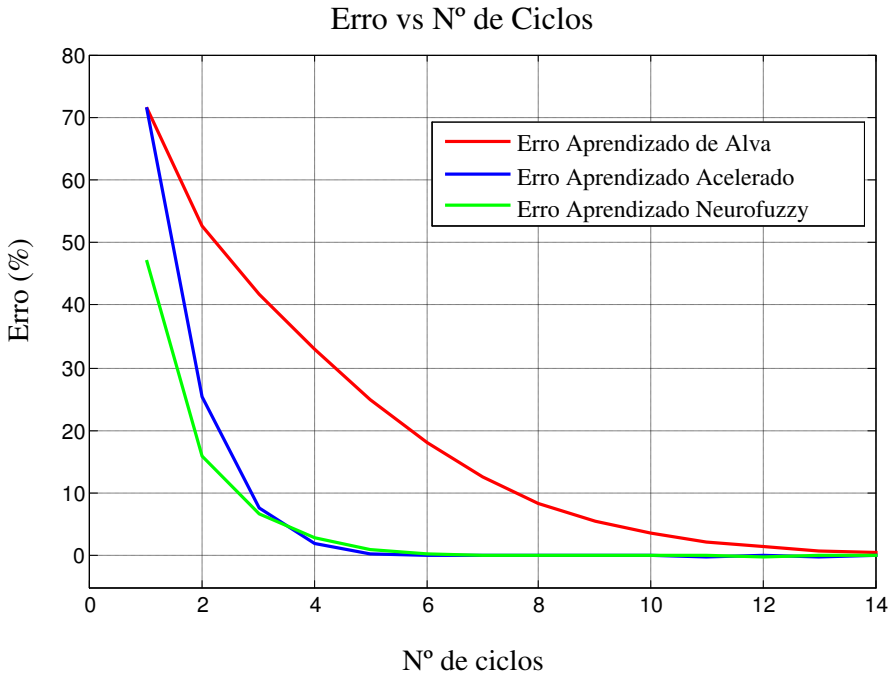


Figura 6.11. Desempenho dos Modelos de controle por aprendizado para carregamento constante ± 20 kN.

Todas as simulações apresentadas nas figuras anteriores foram executadas para uma taxa de aprendizado $\eta = 0.95$. Este parâmetro influi no processo de aprendizagem, fazendo com que o aprendizado seja mais rápido ou lento. Na figura 6.12 é apresentada a convergência do erro ao atingir picos ou vales para os três modelos de controle por aprendizado para $\eta = 0.1$.

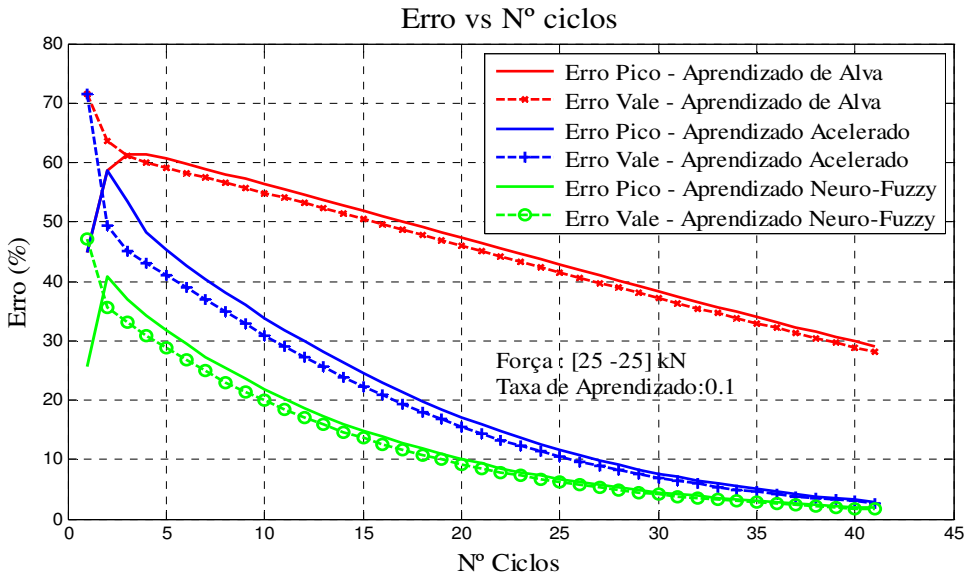


Figura 6.12. Desempenho do controle por aprendizado para $\eta = 0.1$ e carregamento de amplitude constante de $\pm$ 25 kN.

Pode-se observar que o erro do aprendizado Neuro-Fuzzy (linha verde) tem uma convergência mais rápida em relação aos outros tipos de controle. No entanto, o valor $\eta = 0.1$ faz com que ela tenha um aprendizado lento, e convergindo após 80 ciclos.

À medida que a taxa de aprendizado aumenta, a velocidade de convergência do erro aumenta, logo a velocidade de aprendizagem do controle é maior.

Na Figura 6.13 é apresentada a convergência do erro pico/vale dos três modelos de controle por aprendizado, para uma taxa de aprendizado $\eta = 0.95$. Pode-se observar que o erro do controle por aprendizado Neuro-Fuzzy (linha verde) converge para zero após cerca de 8 ciclos. O controle por aprendizado acelerado converge em 10 ciclos e o controle por aprendizado de Alva após 30 ciclos.

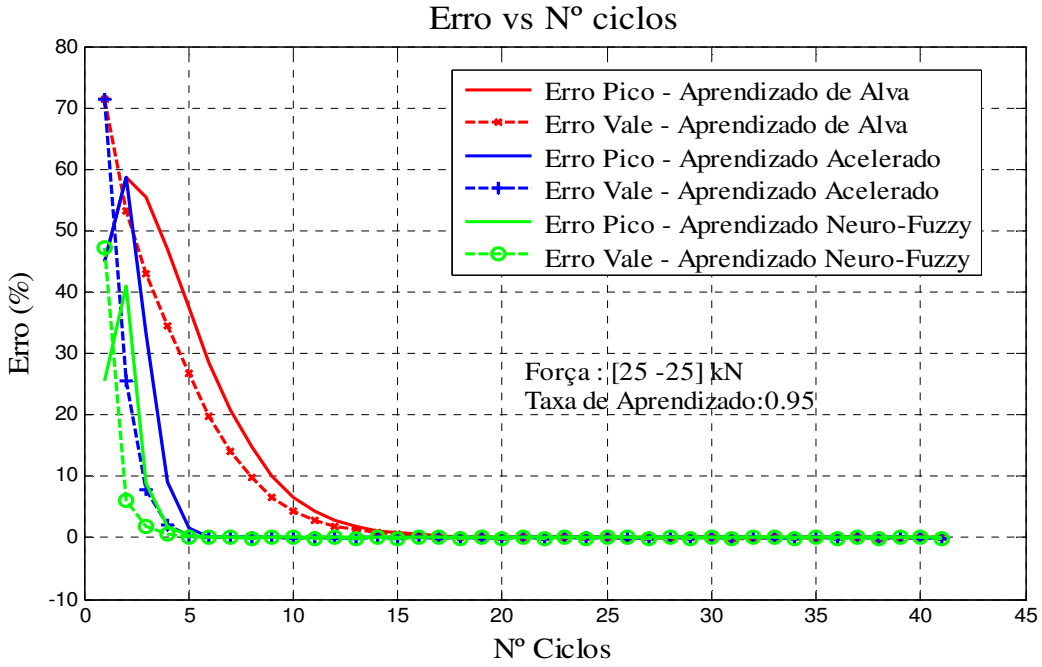


Figura 6.13. Desempenho do controle por aprendizado para $\eta = 0.95$ e carregamento de amplitude constante de $\pm$ 25 kN.

O aumento da taxa de aprendizado permite que a velocidade de aprendizagem seja maior. No entanto, uma taxa de aprendizagem muito alta apresenta oscilações no controle e o problema de *overshoot*, que é prejudicial em ensaios de fadiga por introduzir efeitos de sobrecarga.

A Figura 6.14 apresenta o comportamento do erro ao longo do processo de aprendizagem para $\eta = 1.5$. O controle por aprendizado Neuro-Fuzzy (linha verde) apresenta uma rápida convergência, mas apresenta o problema de oscilações, indesejado neste tipo de aplicação, além de um número de ciclos para convergência é maior que no caso $\eta = 0.95$.

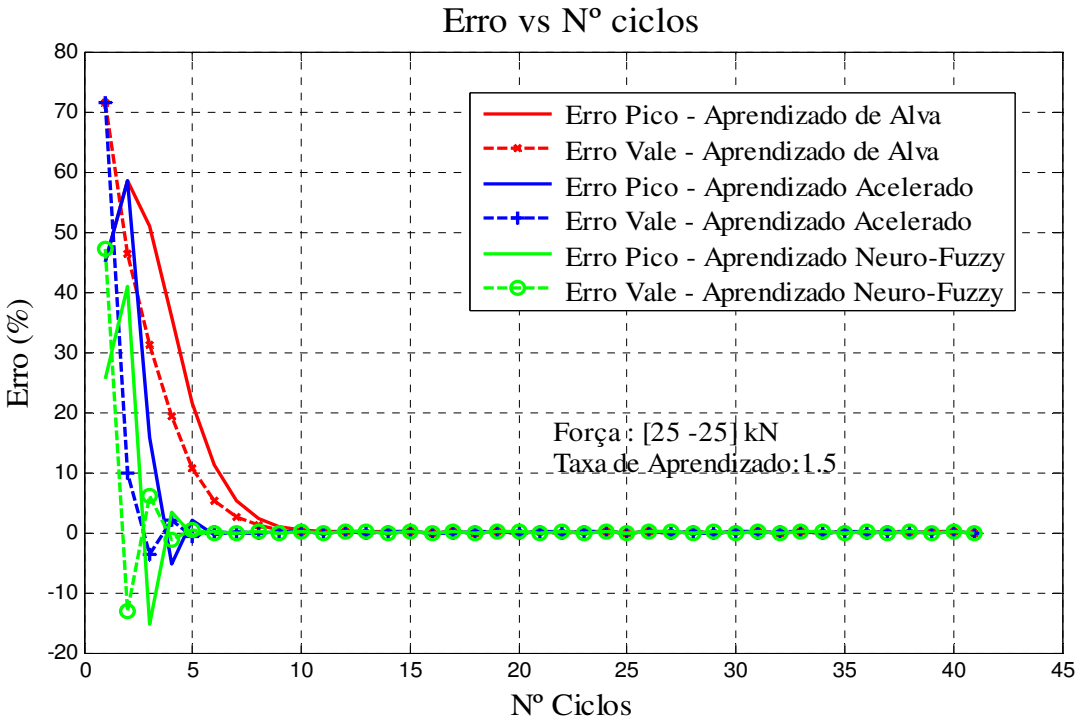


Figura 6.14. Desempenho do controle por aprendizado para $\eta = 1.5$ e carregamento constante de $\pm$ 25 kN.

A figura 6.15 apresenta o número de ciclos até que o controle consiga atingir um erro menor que um valor admissível, neste caso definido em 2% (98% de exatidão), em função da taxa de aprendizado, para um carregamento de $\pm$ 25 kN. O controle por aprendizado Neuro-Fuzzy (linha verde) tem uma aprendizagem muito mais rápida em relação ao controle por aprendizado desenvolvido por Alva. No entanto, tem um desempenho muito semelhante ao do aprendizado acelerado. Também se mostra que para taxas de aprendizado na faixa de [0.1,1] o número de ciclos até a convergência diminui à medida que aumenta a taxa de aprendizado. E para valores $\eta > 1$, o número de ciclos para convergência aumenta com a taxa de aprendizado.

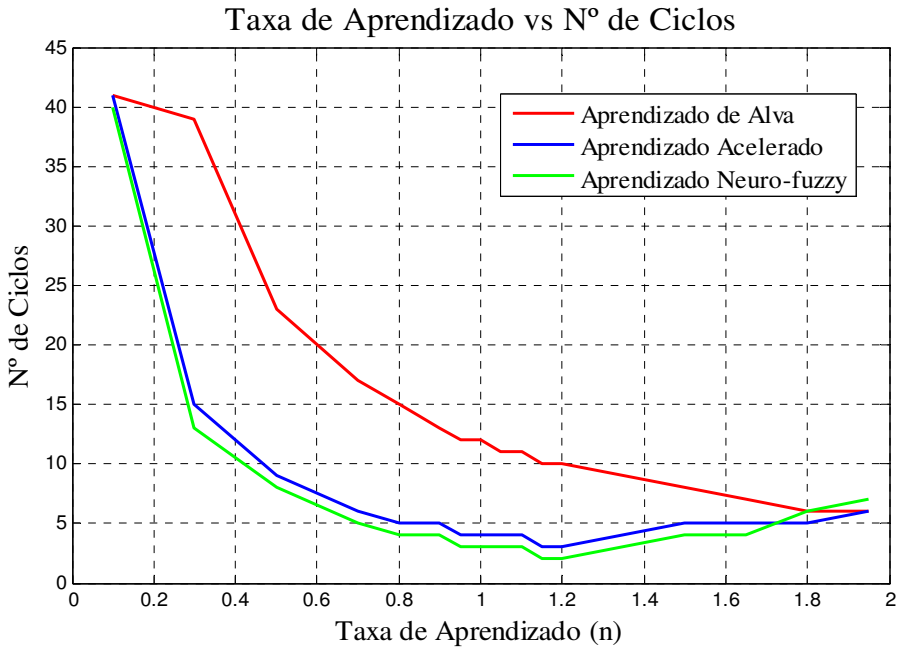


Figura 6.15. Número de ciclos de convergência em função da taxa de aprendizado, carregamento de $\pm$ 25 kN.

A Figura 6.16 apresenta o desempenho do erro em cada pico ou vale em função do número de ciclos para um carregamento de ± 80 kN e uma taxa de aprendizado $\eta = 0.1$. O comportamento é semelhante ao obtido para um carregamento de $\pm$ 25 kN.

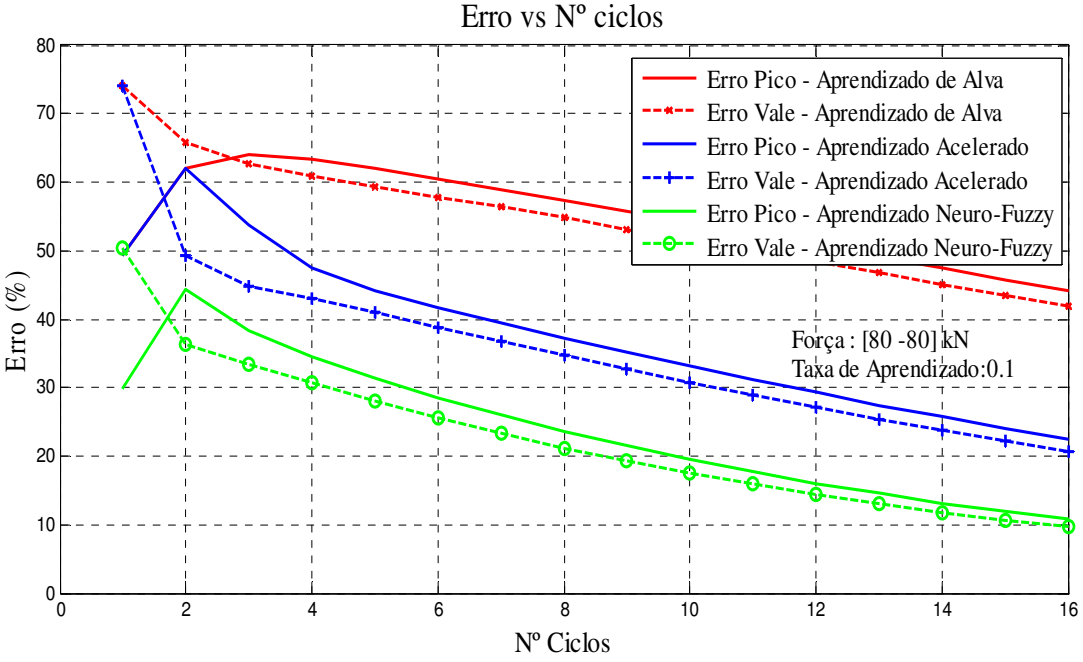


Figura 6.16. Desempenho do controle por aprendizado para $\eta = 0.1$ e carregamento de amplitude constante de $\pm$ 80 kN.

Na Figura 6.17 é apresentado o desempenho do erro para os mesmos níveis de carregamento e com uma taxa de aprendizado $\eta = 0.85$. Neste caso, para um carregamento maior, e para uma taxa de aprendizado menor em relação ao caso da Figura 6.13, o controle Neuro-Fuzzy tem um melhor desempenho, convergindo após apenas 6 ciclos.

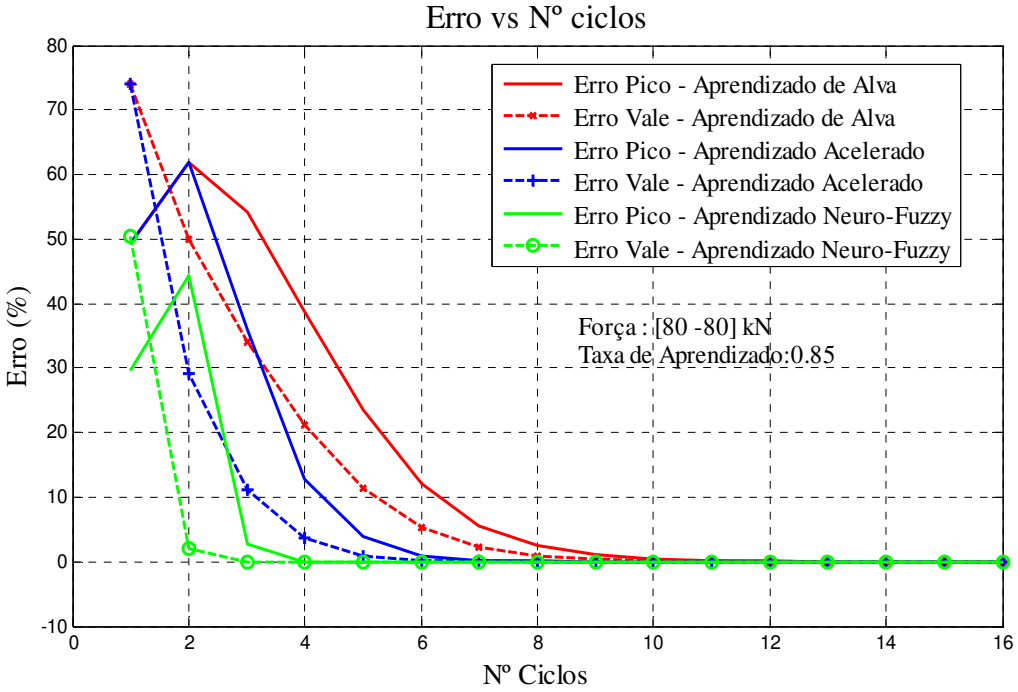


Figura 6.17. Desempenho do controle por aprendizado para $\eta = 0.85$ e carregamento de amplitude constante de $\pm$ 80 kN.

A Figura 6.18 mostra que o controle por aprendizado Neuro-Fuzzy apresenta oscilações para uma taxa de aprendizado de $\eta = 0.95$ para a amplitude de $\pm 80\ kN$, o que mostra que o desempenho do aprendizado também depende dos níveis de carregamento desejado.

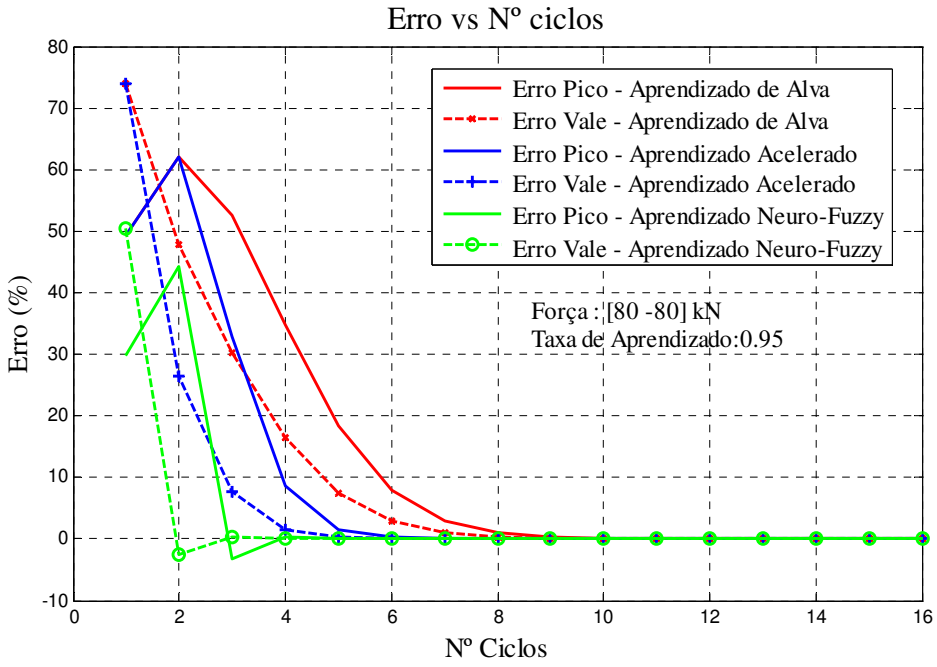


Figura 6.18. Desempenho do controle por aprendizado para $\eta = 0.95$ e carregamento de amplitude constante de $\pm$ 80 kN.

A figura 6.19 apresenta o número de ciclos até que o controle consiga atingir o erro de 2% (98% de exatidão), em função da taxa de aprendizado, para um carregamento de $\pm$ 80 kN.

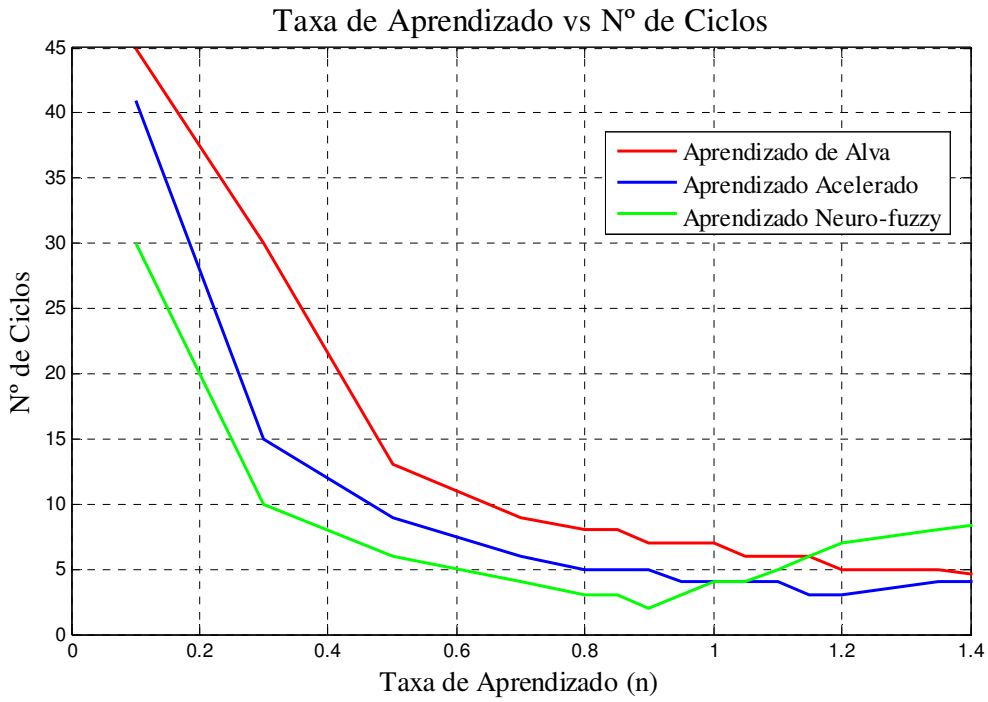


Figura 6.19. Número de ciclos de convergência em função da taxa de aprendizado, carregamento de $\pm$ 80 kN.

A tendência do desempenho é similar ao mostrado na Figura 6.15, com a diferença que no controle por aprendizado Neuro-Fuzzy (linha verde), para valores $\eta > 0.9$, o número de ciclos aumenta com a taxa de aprendizado.

Assim, o valor ótimo da taxa de aprendizado com o qual o erro de controle convergiu com o menor número de ciclos depende do nível de carregamento desejado. Isto é apresentado na Figura 6.20, onde se pode observar que o valor ótimo da taxa de aprendizado η para o sistema Neuro-Fuzzy para um carregamento de $\pm$ 25 kN(linha vermelha) é perto de 1.2, para o carregamento de $\pm$ 55 kN(linha azul) é 1.1, e para o carregamento $\pm$ 80 kN (linha verde) é 0.9.

Como a escolha de um valor da taxa de aprendizado maior que o ótimo apresenta oscilações e um maior número de ciclos para convergência, é conveniente escolher o menor η ótimo, que provavelmente estará associado à maior amplitude da história de carregamentos. Isto devido a que as oscilações apresentam casos de overshoot que são indesejados nos ensaios de fadiga.

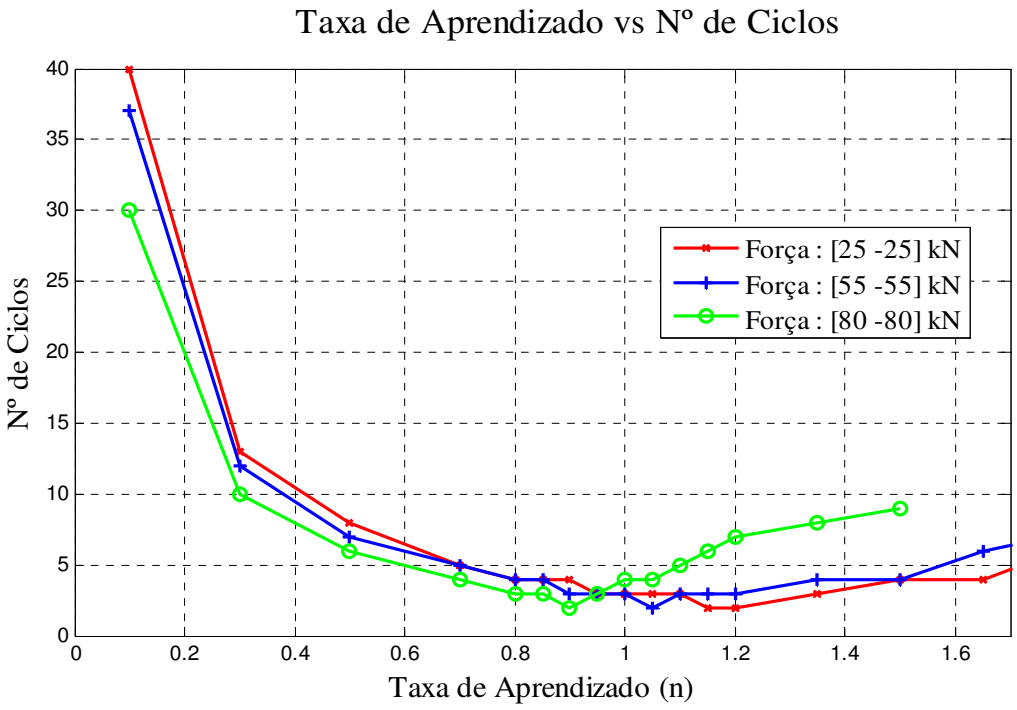


Figura 6.20. Número de ciclos de convergência em função da taxa de aprendizado, para diferentes carregamentos.

Uma sugestão ainda melhor é adotar η variável em uma mesma história, escolhendo seu valor em função da amplitude de cada evento do carregamento.

O controle por aprendizado Neuro-Fuzzy foi testado com uma história padrão de amplitude variável denominada SAE_GKN para uma taxa de aprendizado $\eta = 0.95$. Esta história é composta de 22 ciclos, formando um bloco de carregamento. Os erros para cada ciclo são mostrados na Figura 6.21. Nota-se que os erros se estabilizam a partir do terceiro bloco de carregamento.

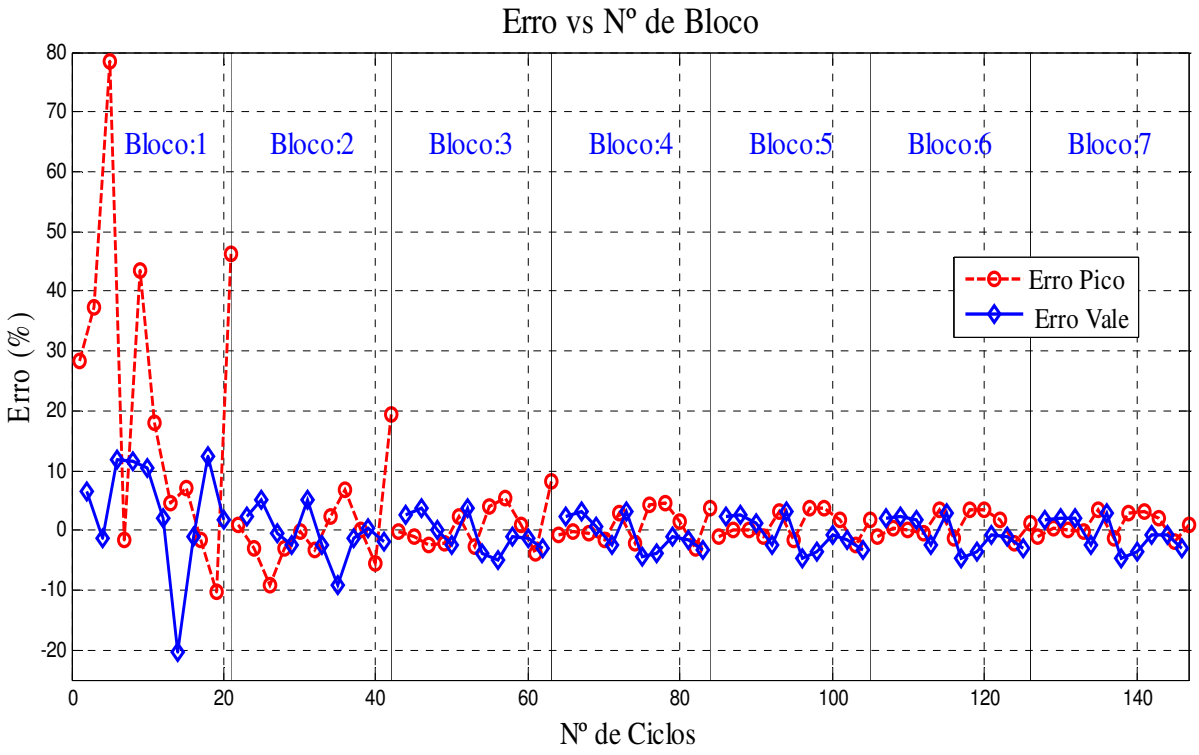


Figura 6.21. Convergência do erro a cada bloco para o carregamento SAE-GKN com picos máximos de 80 kN.

Uma sugestão para melhorar a convergência de um ensaio com carregamento variável é filtrar os carregamentos com amplitudes pequenas devido a que elas podem atrapalhar a convergência no processo de aprendizado.

No primeiro bloco, o erro obtido é relativamente grande, como mostrado na Figura 6.22.

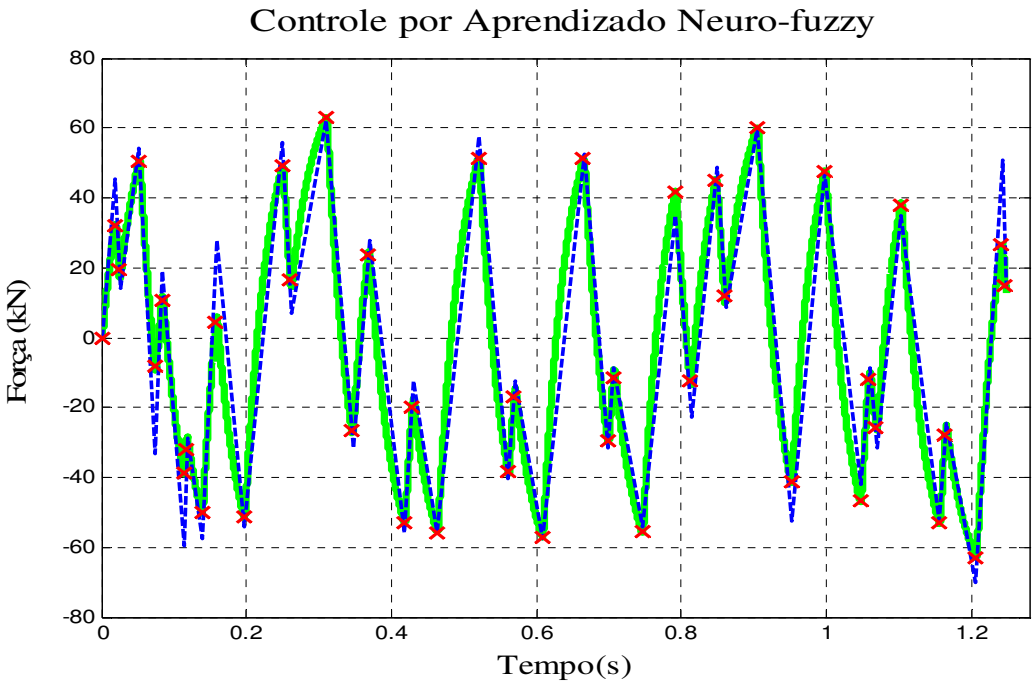


Figura 6.22. Resposta do controle Neuro-Fuzzy no primeiro bloco da história SAE-GKN.

A resposta do controle no quinto bloco é mostrada na Figura 6.23, apresentado erros muito menores que os do primeiro bloco.

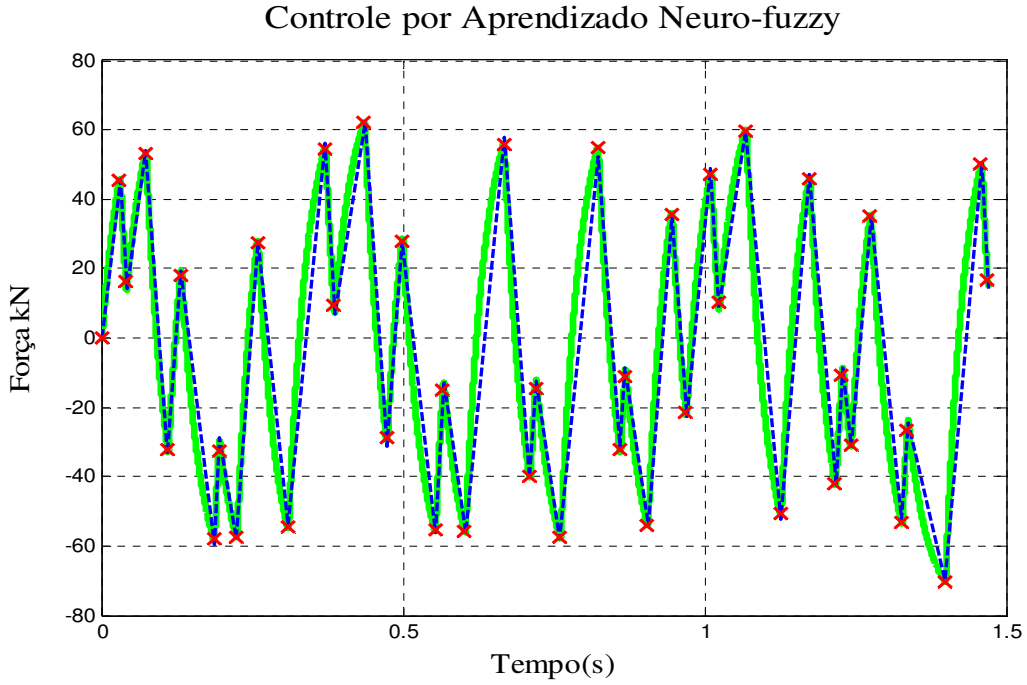


Figura 6.23. Resposta do controle Neuro-Fuzzy depois da convergência, no quinto bloco da história SAE-GKN.

No próximo capítulo, as técnicas de controle apresentadas são aplicadas ao sistema experimental.

7 .
Resultados Experimentais

7.1.Sistema Experimental

Os modelos de controle propostos nesta dissertação foram testados em uma máquina INSTRON modelo 8501 utilizada para ensaios de fadiga, do Laboratório de Fadiga da PUC-Rio. Ela é constituída de uma bomba hidráulica que fornece uma pressão de 190 bar, um atuador hidráulico cilindro-pistão com capacidade de 100 kN, e é comandada por uma servo-válvula MOOG modelo D562, que tem como entrada de controle um sinal de corrente de -40mA a 40mA. Além disso, a máquina dispõe de três sensores: um LVDT, que mede o deslocamento do atuador na faixa de [-60 mm, +60 mm], clip gages que medem a deformação do corpo de prova, e uma célula de carga de capacidade de 100 kN. Ela possui um controlador que chega a atingir frequências da ordem de 50 Hz para um corpo de prova de aço para um carregamento de amplitude 25 kN [4].

Figura 7.1. Máquina de Ensaios INSTRON 8501.

Para a implementação do controle por aprendizado no sistema experimental, é preciso utilizar um módulo computacional capaz de executar controle em tempo real. Isto é possível utilizando, e.g, o modulo *CompactRIO* da *National Instrument*, que juntamente com seus módulos de entrada/saída analógica, e módulos de excitação de extensômetros, pode atingir frequências de controle da ordem de kHz. Na Figura 7.2 é apresentado o esquema de conexões do sistema experimental como o controle por aprendizado proposto.

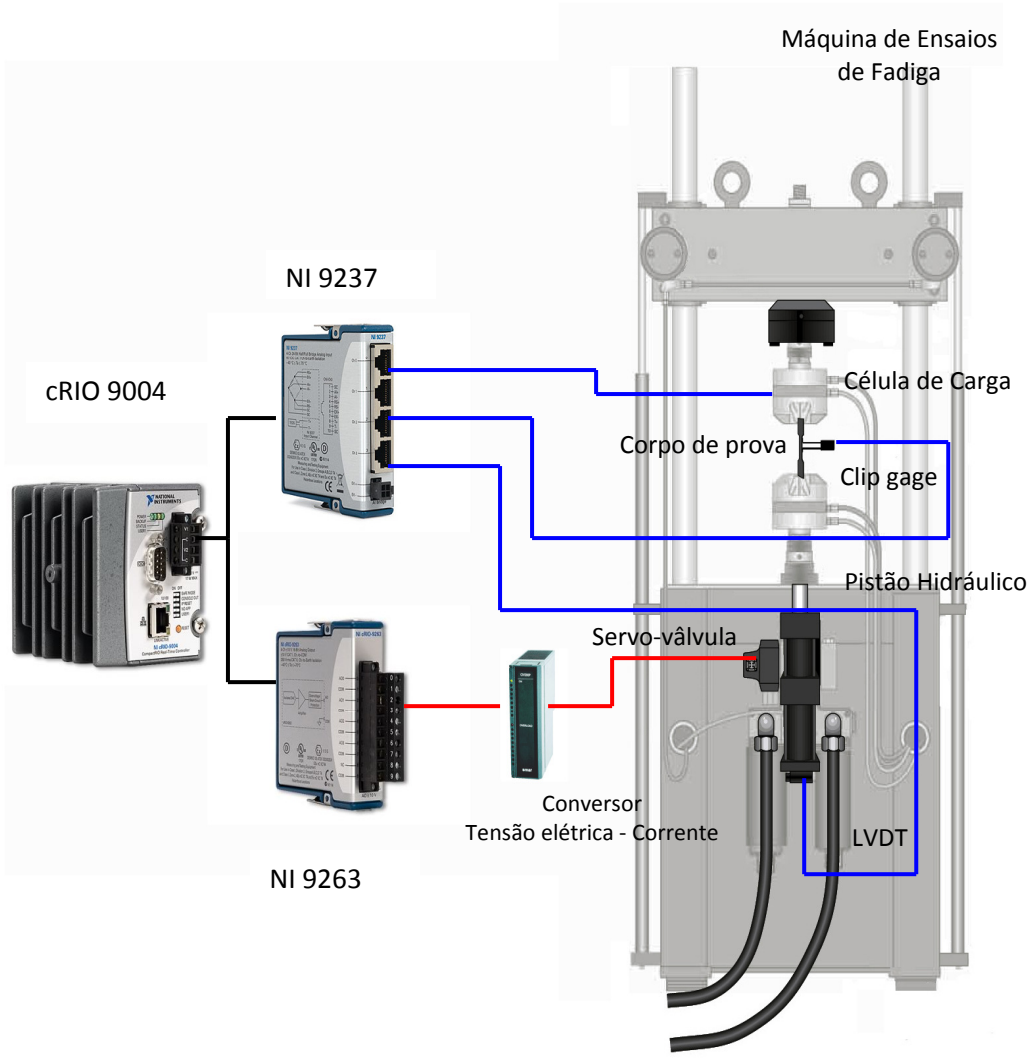


Figura 7.2. Conexões para o sistema de controle por aprendizado.

7.2. Modulo de Controle *CompactRIO*

O *CompactRIO* da *National Instruments* é um controlador programável de automação, que tem um sistema de controle reconfigurável e aquisição de dados projetado para aplicações que requerem alto desempenho e resposta em tempo real com alta confiabilidade. Ele combina um processador em tempo real integrado a um chip FPGA de alto desempenho e robustez, com módulos de entrada/saída intercambiáveis. O FPGA é conectado ao processador em tempo real via um bus PCI de alta velocidade, e cada módulo de entrada/saída é conectado diretamente ao FPGA. O CompactRIO usado no sistema de controle é o cRIO 9004, apresentado na Fig. 7.3.

Figura 7.3. Controlador *cRIO*-9004.

O *cRIO*-9004 tem incorporado um processador industrial classe Pentium de 195 MHz para executar aplicações determinísticas em tempo real desenvolvidas no software *LabVIEW* Real Time. O *cRIO* tem uma memória de 64 MB em *DRAM* e 512 MB de armazenamento CompactFlash não volátil, além de uma porta Ethernet para a programação pela rede. O *LabVIEW Real Time* tem funções internas para transferir dados entre o *FPGA* e o processador em tempo real dentro do sistema do *CompactRIO*.

O *FPGA* (*Field Programmable Gate Arrays*) é um chip de silício reprogramável, que utiliza blocos de lógica pre-construídos com entradas lógicas que não estão conectadas inicialmente, e que depois são configuradas e re-configuradas entre si para as diferentes aplicações que estejam sendo implementandas (vide Figura 7.4). Aplicações com algoritmos onde se precisa resposta em tempo real, sincronização, precisão, e execução de tarefas simultâneas

de forma paralela, são desenvolvidas no FPGA. O paralelismo é conseguido devido ao fato que o módulo LabVIEW FPGA executa sua lógica no hardware, tendo o programa a vantagem de processar as tarefas tais como aplicações de controle, leitura e gravação de saídas analógicas e/ou digitais, em tempo real e de forma determinística.

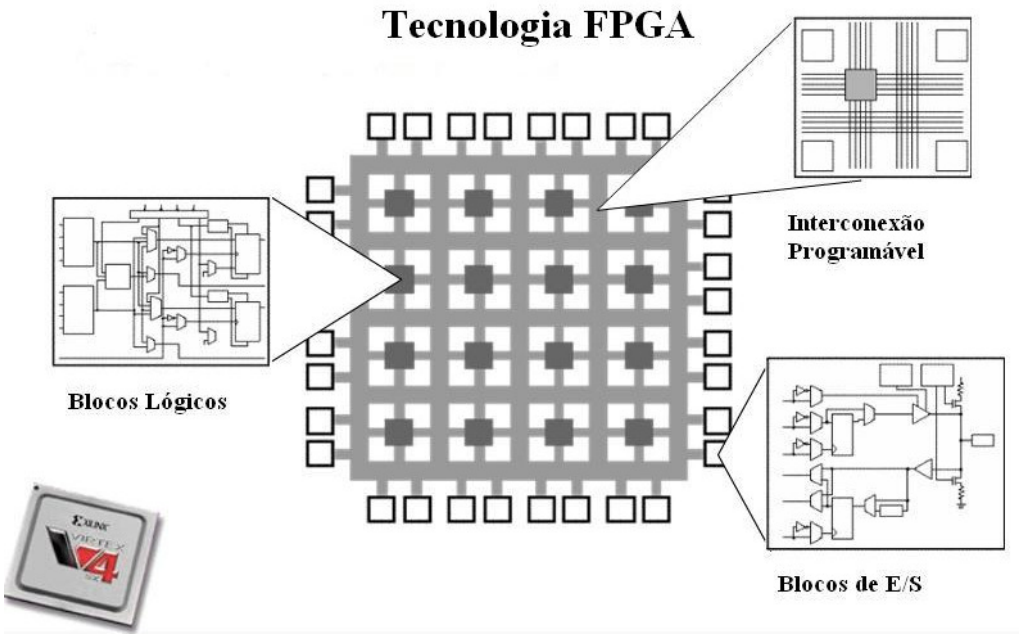
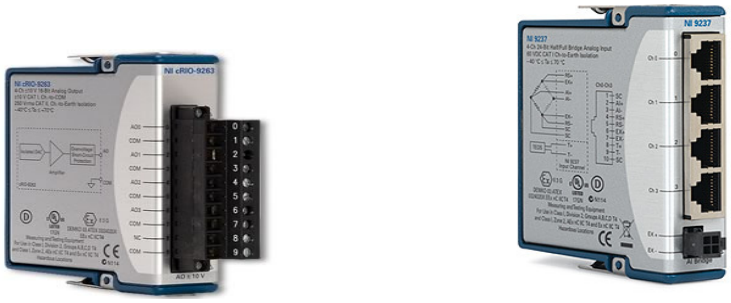


Figura 7.4. Componentes do Chip FPGA.

Para o controle da máquina servo-hidráulica, trabalhou-se com módulos de entradas e saídas analógicas e com um módulo de excitação de extensômetros.

O módulo *NI cRIO* 9263 apresentado na Figura 7.5 (a) é o modulo de saídas analógicas usado para gerar tensões elétricas entre -10 V e +10 V. Essas saídas analógicas são convertidas em saídas de corrente de -40 mA a +40 mA através de um conversor de tensão elétrica para corrente, o qual foi desenvolvido por Alva [5].

O *NI cRIO* 9237 apresentado na Figura 7.5 (b) é o módulo de excitador de extensômetros utilizado para excitar e medir o valor da força aplicada ao corpo de prova através da célula de carga, e também pode medir as deformações do corpo de prova através de um clip gage.



(a) (b)

Figura 7.5. (a) NI cRIO 9263 e (b) NI cRIO 9237.

7.3. Software desenvolvido em LabVIEW

Os modelos de controle por aprendizado da máquina servo-hidráulica para ensaios de fadiga foram desenvolvidos utilizando o software LabView. Este software utiliza três ambientes de programação: um computador conectado ao cRio, o Real Time do cRio e o FPGA do cRio. A parte do controle bang-bang, incluindo as leituras dos dados das entradas analógicas, e do excitador de *train gage*, foi feito no FPGA, para ter a certeza de uma resposta em tempo real, tendo como referência experimental que o FPGA pode trabalhar com frequências de até 55 kHz para dados analógicos, e para dados digitais frequências de até alguns MHz. A estrutura do sistema Neuro-Fuzzy, as tabelas de aprendizado (aprendizado acelerado) e os algoritmos de aprendizado, são feitos no Real Time. Finalmente, as configurações e apresentação de resultados são feitas no computador.

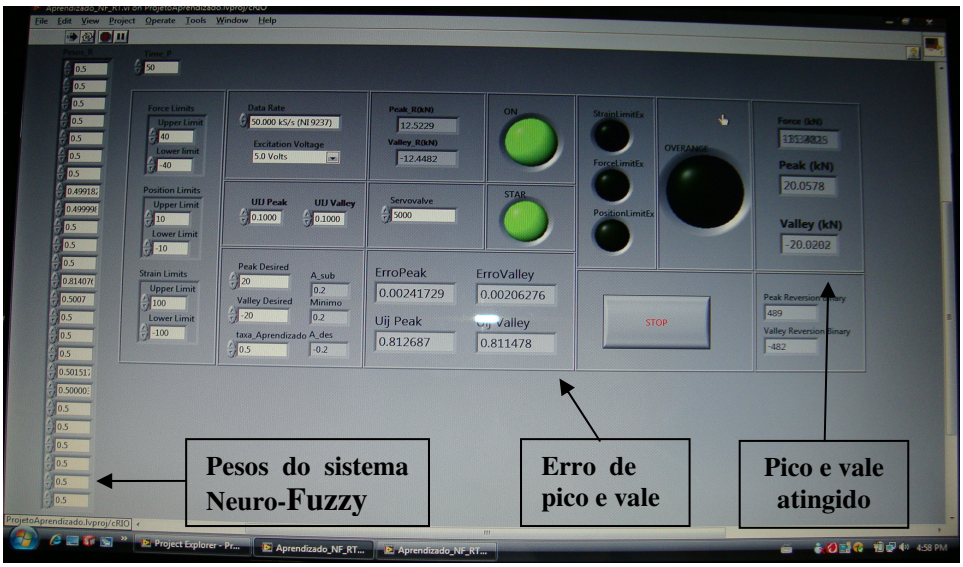


Figura 7.6. Tela de controle por aprendizado Neuro-Fuzzy.

7.4.Resultados Experimentais

Os ensaios são feitos para carregamentos de amplitudes entre 10 kN e 60 kN, todos eles para uma corrente de ± 20 mA ou ± 30 mA na servo-válvula. A servo-válvula é capaz de trabalhar até ± 40 mA, no entanto algumas limitações na frequência do controlador *CompactRio* geram problemas de overshoot para frequências muito altas, associadas ao ± 40 mA. A Figura 7.7 apresenta um ensaio de fadiga utilizando o controle por aprendizado Neuro-Fuzzy para um carregamento constante de 20 kN com carga média zero, em um corpo de prova εN feito de aço de 12 mm de diâmetro.

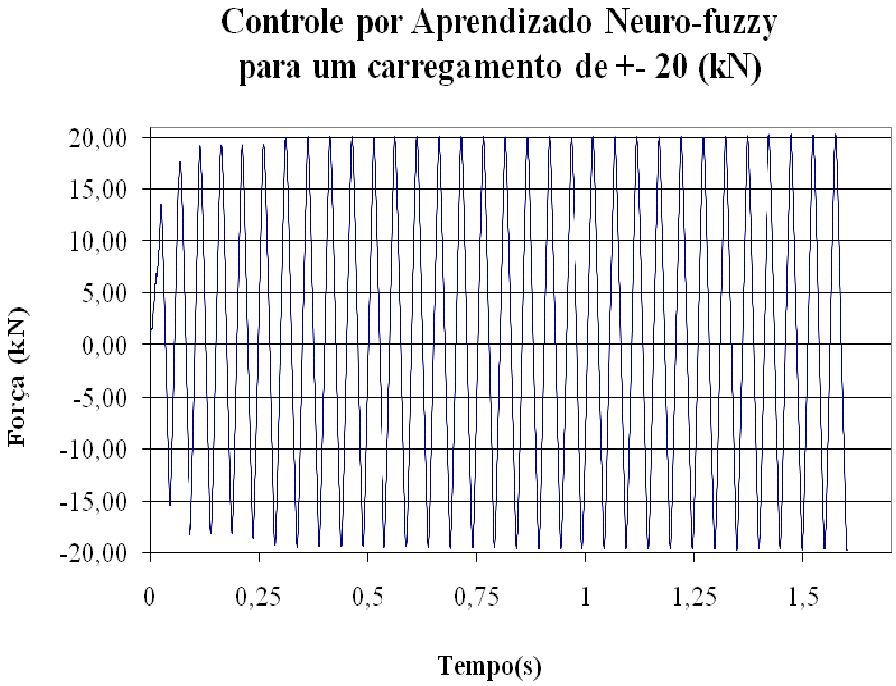


Figura 7.7. Resposta da máquina servo-hidráulica a um ensaio de fadiga sob amplitude constante com controle por aprendizado Neuro-Fuzzy.

O armazenamento das leituras dos sensores precisou ser feita externamente ao CompactRio, devido que a implementação desta funcionalidade atrapalha a frequência de controle do C-Rio. O sistema de medição externo registrou uma *overshoot* Maximo de 1% após o termino do aprendizado. Este *overshoot* é provavelmente causado pela precisão na medida da força e as limitações de frequência do C-Rio.

Na figura 7.8 é apresentado o erro normalizado em função do número de ciclos, para o ensaio de fadiga apresentado na figura acima, na qual os erros nos picos têm um aprendizado um pouco mais rápido que nos vales, convergindo após 7 ciclos.

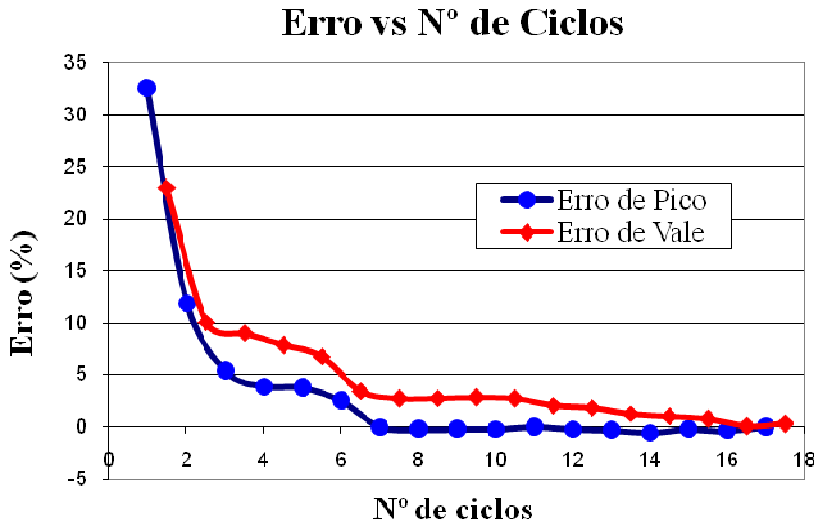


Figura 7.8 Convergência do erro em função do numero de ciclos.

Na versão atual do programa para os controles por aprendizado acelerado e Neuro-Fuzzy implementados no CompactRIO, notou-se duas limitações: a primeira relacionada com o controlador, o qual apresenta uma reduzida capacidade de memória em seu módulo de *FPGA*, obrigando desenvolver parte do controle no módulo *Real Time*, o qual apresenta uma baixa frequência de trabalho.

A outra limitação foi devido à hipótese que $0 < U_{IJ} < 1$. Em testes usando corrente de ± 40 mA na servo-válvula, observou-se que os valores ótimos de U_{IJ} poderiam adquirir valores negativos. Isto significa que a reversão em um pico precisaria ocorrer não apenas antes de atingir este pico ($U_{IJ} < 1$), mas antes mesmo de atingir ao vale anterior (i.e. $U_{IJ} < 0$). Isto é devido à pequena defasagem temporal entre o envio de sinal de atuação para a servo-válvula e sua movimentação efetiva. Este pequeno atraso passa a ser significativo em altas frequências resultando em $U_{IJ} < 0$, o que não é previsto nos modelos apresentado. Na Figura 7.9 se ilustram os pontos de reversão ótimo de subida e descida (circulo

de azul); à medida que a corrente é incrementada, o ponto de reversão do pico se antecipa do ponto 6 para o ponto 5, podendo até chegar ao ponto 4 (circulo de cor vermelho), associado a $U_{IJ} < 0$. De forma similar, o ponto de reversão de vale se antecipa do ponto 3 até eventualmente chegar ao ponto 1 para correntes de ± 40 mA.

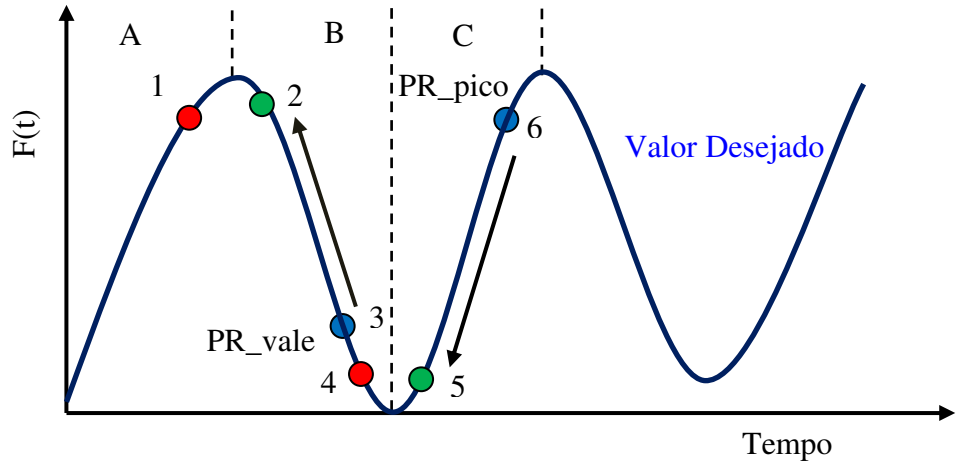


Figura 7.9. Antecipação dos pontos de reversão à medida que a frequência é aumentada.

Estes problemas limitaram a corrente de trabalho da servo-válvula a no máximo ± 30 mA para carregamentos maiores que 30 kN.

Mesmo com estas limitações, observou-se que o controle por aprendizado Neuro-Fuzzy com corrente de ± 30 mA (linha verde na Figura 7.10) para carregamentos maiores a 30 kN tem um melhor desempenho que o controlador INSTRON. Além disso, o controle por aprendizado Neuro-Fuzzy para uma corrente de ± 20 mA apresenta um melhor desempenho que o controlador INSTRON para baixas amplitudes, e somente após permitir o "overdrive" do controlador INSTRON , o qual utiliza correntes maiores que ± 40 mA, é que os resultados da INSTRON se mostraram melhores. Uma vez solucionadas as limitações da frequência do CompactRIO, espera-se poder trabalhar com correntes acima de ± 40 mA, para obter resultados ainda melhores que os do controle INSTRON com "*overdrive*". Entretanto a linha de preto representa o limite maximo da servo-válvula, a qual representa o maximo desempenho que a maquina poderia atingir.

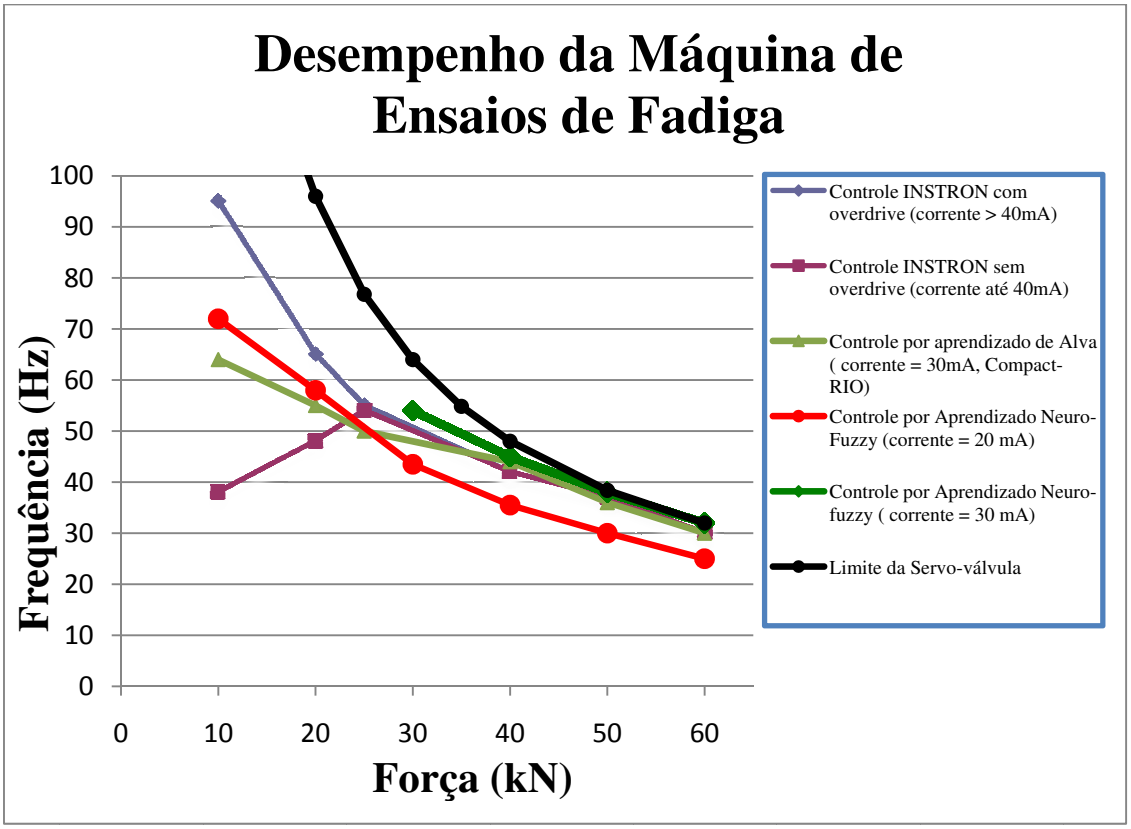


Figura 7.10. Comparação de desempenho usando diferentes controles na máquina INSTRON 8501.

8.
Conclusões

Neste trabalho, sistemas de controle por aprendizado acelerado e Neuro-Fuzzy foram desenvolvidos para acionar sistemas servo-hidráulicos. Estes modelos de controle propostos forem simulados e aplicados a uma máquina de ensaios de fadiga, implementando-se o controlador em um módulo de controle *CompactRIO* da *National Instruments*.

O primeiro modelo desenvolvido, o controle por aprendizado acelerado, é uma otimização do controle por aprendizado desenvolvido por Alva, com o propósito de melhorar a velocidade de aprendizagem. Este modelo inclui dentro da lei de controle uma taxa de aprendizado e um termo de momentum, que permitem acelerar o processo de aprendizagem.

O segundo modelo desenvolvido, o controle por aprendizado Neuro-Fuzzy, não necessita do uso de tabelas, pois todas as atualizações e informações do processo de aprendizado são armazenadas nos pesos do sistema Neuro-Fuzzy. Este modelo apresenta um melhor desempenho nas simulações tanto para carregamento de amplitude constante quanto variável. Da mesma forma, na parte experimental os resultados obtidos confirmam uma boa aplicabilidade no controle de sistemas servo-hidráulicos.

Os resultados mostraram que o controle proposto é capaz de gerar frequências mais altas que as do controlador original em uma máquina de ensaios INSTRON 8501, mesmo utilizando correntes mais baixas para o acionamento da servo-válvula.

Trabalhos futuros incluem a consideração explicita nos modelos de pontos de reversão associada a $U_{IJ} < 0$, o que permitiria aumentar ainda mais a frequência de teste, mesmo na presença de atrasos entre o comando e a movimentação da servo-válvula, e possibilitando, assim testes com corrente de ± 40 mA ou mais sem overshoots significativos.

Bibliografia

[1] Jelali, M, Kroll, A. “Hydraulic Servo - System: Modeling, Identification and Control”. New York: Springer, 2003.

[2] F. L. Lewis, J. Campos, R. Selmic. “Neuro-Fuzzy Control of industrial System with Actuator Nonlinearities”. Philadelphia, 2002.

[3] P. J. Costa Branco, J. A. Dente. “Design of an Electro-Hydraulic System Using Neuro-Fuzzy Techniques”. Mechatronics Laboratory. Department of Electrical and Computer Engineering. Instituto superior técnico, Lisbon. Portugal – 1998.pp 190-206.

[4] Somyot Kaitwanidvilai, M. Parnishkun. “Force Control in a Pneumatic System using Hibrid Adaptive Neuro-Fuzzy Model Reference Control”. School of Advanced Technologies, Asian Institute of Technology. Thailand – 2004.pp 23-41.

[5] Juan. G. C. Alva, “Controle por aprendizado de Sistemas Servo-hidráulicos de alta frequência”. Dissertação de mestrado, Departamento de Engenharia Mecânica, PUC - Rio. Rio de Janeiro, 2008, pp.64 – 89.

[6] Manual INSTRON. Model 8500 Plus. 1995. 200 pp.

[7] Shaw, I.S; Godoy M. “Controle e Modelagem Fuzzy”. 1ª. edição. São Paulo: Edgard Blucher: FAPESP, 1998. 165p.

[8] Manual Fuzzy Logic Toolbox do MATLAB, versão 7.0 . May, 2004.

[9] Tanscheit, R. “Fundamentos de Lógica Fuzzy e Controle Fuzzy”. http//www.ica.ele.puc-rio.br.

[10] Fernandes, R. T. “Supervisão de um Sistema Hibrido Eólico/Diesel usando Lógica Fuzzy”. Dissertação de mestrado, Universidade Federal de Mato Grosso do Sul. Campo Grande, 2005, 118p.

[11] Lewis, H. W. “The Foundations of Fuzzy Control”. New York: Plenum Press, 1997. 299p.

[12] Delgado. M. R. “Projeto Automatico de Sistemas Nebulosos: uma Abordagem CoEvolutiva”. UNICAMP. Tese de Doutorado. Campinas, SP,2002. 186p.

[13] Pedrycz, W., Gomide, F. “Na Introduction to Fuzzy Sets: Analysis and Design”. MIT Press. Cambridge, 1998. 456p.

[14] Simon Haykin “Neural Networks - A comprehensive Foundation”. Canada. Pearson Prentice Hall, 2001. pp 23.

[15] Braga. A. P., Carvalho. A. P. de L. F., Ludernir. T. B. “Fundamentos de Redes Neurais Artificiais”. Editora LTC, 2000 1º Edição.

[16] Z. L. Kovács, “Redes Neurais Artificiais”. Editora Acadêmica São Paulo, São Paulo, 1996.

[17] M. H. Hassoun, “Fundamentals of Artificial Neural Networks”. MIT Press, Massachusetts, 1995.

[18] Fayal Assis M. A, “Previsão de Vazão por Redes Neurais Artificiais e Transformada Wavelets”. Dissertação de mestrado, Departamento de Engenharia Mecânica, PUC - Rio. Rio de Janeiro, 2008.

[19] Vellasco. M. M. B. R. “Inteligência Computacional”, ICA: Núcleo de Pesquisa e IA. http//www.ica.ele.puc-rio.br.

[20] Beale R., Jackson T. “Neural Computing: an Introduction”. Bristol: Adam Hilger, 1990.

[21] Baron R. “Knowledge Extraction from Neural Network”. A Survey. (NeuroCOLT Technical Report), 1995.

[22] Mendes E., Carvalho A. “Target Recognition using Evolutionary Neural Networks”. Proceeding of V Brazilian Symposium on Neural Networks, 1998.

[23] Souza F. J. “Sistemas Neuro-Fuzzy”. ICA: Nucleo de Pesquiza em IA, http://www.ele.puc-rio.br/labs/ica/icahome.html.