ADEMIR NIED

TREINAMENTO DE REDES NEURAIS ARTIFICIAIS BASEADO EM SISTEMAS DE ESTRUTURA VARIÁVEL COM TAXA DE APRENDIZADO ADAPTATIVA

BELO HORIZONTE

2007

"TREINAMENTO DE REDES NEURAIS ARTIFICIAIS BASEADO EM SISTEMAS DE ESTRUTURA VARIÁVEL COM TAXA DE APRENDIZADO ADAPTATIVA"

Ademir Nied

Tese de Doutorado submetida à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como parte dos requisitos necessários à obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 09 de março de 2007.

Por:

Benjamim Rodrigues de Menezes, Dr.
DELT/UFMG - Orientador

Seleme Isaac Seleme Jr., Dr.
DELT/UFMG

Antônio de Pádua Braga, Ph.D.
DELT/UFMG

Reinaldo Martinez Palhares, Dr.
DELT/UFMG

Marcelo Carvalho Minhoto Teixeira, Dr.
DEE/UNESP-Ilha Solteira

João Onofre Pereira Pinto, Ph.D.
DEL/UFMS

Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

TREINAMENTO DE REDES NEURAIS ARTIFICIAIS BASEADO EM SISTEMAS DE ESTRUTURA VARIÁVEL COM TAXA DE APRENDIZADO ADAPTATIVA

por

ADEMIR NIED

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerias, como requisito parcial para a obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Benjamim Rodrigues de Menezes
Co-Orientador: Prof. Dr. Gustavo Guimarães Parma

Belo Horizonte, 09 de março de 2007.

Março 2007

Esta tese foi escrita usando LaTeX2e.

As figuras foram feitas em jfig3.

Os gráficos foram gerados em MatLab da Mathworks Inc.

Dedico esta tese à minha esposa Márcia,
e aos meus pais Heldo e Nelly.

Porque Deus amou ao mundo de tal maneira que deu o seu Filho unigênito, para que todo o que nele crê não pereça, mas tenha a vida eterna. João 3.16

Feliz aquele que transfere o que sabe e aprende o que ensina.
Cora Coralina

AGRADECIMENTOS

A Deus, pela vida e por todas as bençãos recebidas.

Aos meus pais, Heldo Nied e Ivia Nelly Nied, pelo exemplo de vida, pelo amor e cuidado que sempre me dispensaram.

À minha querida esposa Márcia, pelo companheirismo, incentivo, amor, carinho e, principalmente, pelas orações.

Aos professores Benjamim Rodrigues de Menezes e Gustavo Guimarães Parma, pela amizade e orientação neste trabalho de tese.

Ao professor Selênio Rocha Silva, pela amizade, incentivo, contribuições e acolhida quando da minha chegada na UFMG.

Ao professor Seleme Isaac Seleme Júnior, pela amizade e pela ajuda na co-orientação deste trabalho.

Aos senhores membros da banca examinadora, professores Antônio de Pádua Braga, Reinaldo Martinez Palhares, Marcelo Carvalho Minhoto Teixeira e João Onofre Pereira Pinto, pelas valiosas contribuições que deram para o aperfeiçoamento deste trabalho.

A todos os professores do DEE e DELT, em especial aos professores Walmir Matos Caminhas, Alessandro Fernandes Moreira, Braz de Jesus Cardoso Filho, Renato de Oliveira da Costa Lyra, Luis Antonio Aguirre, Paulo Fernando Seixas, Marcos Antônio Severo Mendes, Porfírio Cabaleiro Cortizo, José Carlos Rodrigues de Oliveira, Glássio Costa de Miranda e Ivan José da Silva Lopes, pela acolhida, pela amizade, incentivo e contribuições.

Ao professor Marcelo Azevedo Costa, pelas contribuições no desenvolvimento do trabalho.

A todos os amigos que me incentivaram nessa jornada, em especial àqueles que estiveram mais próximos: Júlio, Eduardo, Rodrigo, Clodoaldo, Finzi, Leandro, Stopa, Cássia, Beth, Ronan e Loran.

Aos bolsistas de Iniciação Científica, Marcelo e Daniel.

À Universidade Federal de Minas Gerais (UFMG) e ao PPGEE.

Aos colegas do Departamento de Engenharia Elétrica da Universidade do Estado de Santa Catarina (UDESC), e à própria UDESC pela oportunidade de realização deste trabalho, em especial aos professores Alcindo do Prado Junior e André Bittencourt Leal, e ao Sandro, pela ajuda na fase final do trabalho.

À CAPES pelo suporte financeiro através do projeto PROCAD.

Resumo

Neste trabalho são propostos novos algoritmos de treinamento de redes neurais artificias para a topologia de redes de múltiplas camadas (MLP - *multilayer perceptron*), baseados na teoria de controle de sistemas de estrutura variável, mais especificamente, controle por modos deslizantes. A característica fundamental dos algoritmos propostos é a obtenção de um ganho (taxa de aprendizado) adaptativo, determinado iterativamente, a cada passo de atualização dos pesos, dispensando a necessidade do uso de métodos heurísticos na determinação do ganho da rede.

Foram desenvolvidos dois algoritmos para treinamento em tempo real de redes MLP de duas camadas com a camada de saída linear, permitindo que a rede neural adapte continuamente seus parâmetros livres às variações do sinal de entrada. Os algoritmos propostos seguem a mesma metodologia para a obtenção do ganho adaptativo diferindo, principalmente, na definição da superfície de deslizamento e na expressão usada para atualização dos pesos da rede. Assim, a primeira proposta é mais generalista, possibilitando o uso de redes com múltiplas saídas, enquanto a segunda é limitada a apenas uma saída escalar. Por seu vez, a segunda proposta atualiza os pesos da rede usando uma lei que permite a estabilidade assintótica de acordo com a teoria de estabilidade de Lyapunov, para um conjunto de pesos que corresponde ao mínimo global.

Os algoritmos propostos foram validados na aproximação de uma função periódica e no acionamento elétrico de um motor de indução (MI). Nesta última aplicação, a rede foi usada como neurocontrolador e como observador neural do fluxo de estator do MI. Estas aplicações necessitam que o treinamento da rede seja feito em tempo real, impondo um contínuo ajuste dos pesos da rede às exigências do sistema no qual a rede neural está inserida. Pode-se, portanto, distinguir duas características interessantes nos algoritmos propostos: facilidade de uso, sem a necessidade da escolha, pelo projetista, de um ganho para o treinamento da RNA e, um comportamento adaptativo, sem a necessidade de qualquer informação do modelo matemático no qual o rede neural está inserida.

Abstract

This work presents new algorithms for training multilayer perceptron artificial neural networks based on stability properties of sliding mode variable structure systems. The main feature of the proposed algorithms is the adaptability of the gain (learning rate), which is obtained from each update step of the network weights, without the use of heuristics methods to obtain this gain.

Two algorithms for continuous time learning multilayer perceptron artificial neural networks with two layer and with linear output layer are developed, allowing the neural network continuously to adapt the network parameters following the input signal variation. The proposed algorithms pursue the same methodology to obtain the adaptive gain. The differences between them are related with the sliding mode definition and the network weight update rule. In such a manner, the first algorithm is associated with multiple output networks, and the second is used only with the single output networks. In its turn, the second algorithm update the network weights using one expression that guarantee the asymptotical stability around the global minimum weight according to the Lyapunov stability theory.

In order to verify the performance of the proposed algorithms, both algorithms were applied to periodic function approximation and induction motor drive. In this last application, the neural network was used as neurocontroller and as induction motor stator flux neural observer. These applications need that neural training has to be made in continuous time, imposing a continuous network weight update according to the overall system requirements. Therefore, the algorithms present two interesting features: easy to use, without the necessity to choose the learning rate parameter by designer; and, adaptive behaviour, without requiring any information about mathematical model of the overall system.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ADALINE	*adaptive linear neuron*
ARC	aplicação e retirada de carga
BP	retropopagação (*backpropagation*)
CMD	controle por modos deslizantes
DSP	processador digital de sinais (*digital signal processor*)
EKF	filtro de Kalman estendido (*extended Kalman filter*)
FIR	resposta a impulso de duração finita (*finite-duration impulse response*)
LMS	mínimo quadrado médio (*least-mean-square*)
MCP	McCulloch and Pitts (modelo de rede neural proposto em 1943)
MI	motor de indução
MIMO	múltiplas entradas múltiplas saídas
MLP	redes neurais de múltiplas camadas (*multilayer perceptron*)
PRV	partida e reversão de velocidade
RNA	redes neurais artificiais
SEV	sistemas de estrutura variável
TLFN	rede alimentada adiante atrasada no tempo (*time lagged feedforward network*)
UFO	controlador universal orientado pelo campo (*universal field oriented controller*)
UFOV	controlador universal por tensão orientado pelo campo (*universal field oriented voltage*)
UFOVS	controlador universal por tensão orientado pelo campo de estator (*universal field oriented voltage stator*)

Lista de Símbolos

$\mathbf{A^T}(\mathbf{x^T})$	a transposta de uma matriz $\mathbf{A}$ (um vetor $\mathbf{x}$)
A_j	valor da derivada da função de ativação do neurônio j
$\mathbf{B}$	matriz de ganhos sistema MIMO não-linear não-autônomo
B_v	constante positiva que limita v
$\mathbf{b}$	vetor de polarização da RNA
C	constante positiva
$\mathcal{C}$	conjunto de todos os neurônios da camada de saída da RNA
c	coeficiente da variável η
c_1, c_2, c_3	coeficientes de uma função quadrática: $\{c_1, c_2, c_3\} \in \mathbb{R}$
$\mathbf{D}$	matriz diagonal positiva definida sistema MIMO não-linear não-autônomo
$\mathbf{d}$	vetor de saída desejada da RNA
d	comprimento da janela de tempo
$E(n)$	soma instantânea dos erros quadráticos ou energia do erro na iteração n
E_{med}	energia média do erro médio
$e(\mathbf{w}, t)$	soma do erro quadrático de saída da RNA no instante t
$\mathbf{e}$	vetor do erro de saída da RNA
$f(\cdot)$	função de mapeamento entre dois conjuntos
$f'(\cdot)$	derivada primeira de um valor real da função f
$\dot{f}$	derivada primeira de f com respeito ao tempo
$\mathbf{G}$	matriz de ganhos sup. de desl. sistema MIMO não-linear não-autônomo
i, j, k	índices referentes a diferentes neurônios da rede MLP
J	momento de inércia do MI
L_s	indutância de dispersão de estator
L_r	indutância de dispersão de rotor
$l = 0, 1, ..., L$	número de camadas da rede MLP sendo L a profundidade da rede
m_l	tamanho (número de neurônios) da camada l da rede MLP
M	indutância mútua
N	total de padrões (exemplos) contidos no conjunto de treinamento da RNA
n	tempo discreto
p	pares de pólos do MI
R_s	resistência de estator

R_r	resistência de rotor
$\mathbf{S}(\cdot)$, s	superfície de deslizamento
$sign(\cdot)$	a função sinal
T	período de amostragem
T_e	torque eletromagnético
T_c	torque de carga
t, τ	tempo contínuo
t_h	tempo de alcance (*hitting time*)
$\mathbf{U}$	matriz das entradas sistema MIMO não-linear não-autônomo
$\mathbf{V}(\cdot)$, $V(t)$	função candidata de Lyapunov
$v_j(n)$	sinal aplicado à função de ativação associada ao neurônio j na iteração n
$\mathbf{v_s}$, $\mathbf{i_s}$, $\lambda_\mathbf{s}$	vetores tensão, corrente e fluxo de estator, respectivamente
$\mathbf{v_r}$, $\mathbf{i_r}$, $\lambda_\mathbf{r}$	vetores tensão, corrente e fluxo de rotor, respectivamente
v_{sd}, i_{sd}, λ_{sd}	tensão, corrente e fluxo de estator de eixo d, respectivamente
v_{sq}, i_{sq}, λ_{sq}	tensão, corrente e fluxo de estator de eixo q, respectivamente
X	variável auxiliar da superfície de deslizamento
$\mathbf{X}$	matriz de variáveis de estado sistema MIMO não-linear não-autônomo
$\mathbf{x}$	vetor de entrada da RNA
$\|\|\mathbf{x}\|\|$	norma euclidiana (comprimento) do vetor $\mathbf{x}$
$\|x\|$	valor absoluto (magnitude) de um escalar x
$\mathbf{y}$	vetor de saída da RNA
$\mathbf{w}$	vetor de peso
$\mathbf{W}$	matriz de peso
$\mathbf{w}^*$	vetor de peso ótimo
w_a	velocidade elétrica arbitrária
w_e	velocidade elétrica síncrona
w_r	velocidade elétrica de rotor
Δw	pequena variação aplicada ao peso w
δ_j	gradiente local do neurônio j da rede MLP
ε	número real maior que zero
η	parâmetro da taxa de aprendizado
ξ	precisão desejada para aproximação de uma função através de série de Taylor
ξ_1, ξ_2	número real maior que um
$\varphi_j(\cdot)$	função de ativação não-linear do neurônio j
∇	função gradiente

Outros símbolos

$\diamond$	fim de teorema
$\square$	fim de prova

Capítulo 1

Introdução

O presente trabalho traz contribuições à teoria de treinamento em tempo real de redes neurais artificiais (RNA) considerando a topologia de múltiplas camadas (MLP). Entende-se por treinamento em tempo real, aquele no qual o processo de aprendizagem é realizado enquanto o processamento de sinal está sendo executado pelo sistema, ou seja, a rede neural adapta continuamente seus parâmetros livres às variações do sinal incidente em tempo real (Haykin, 2001).

Este capítulo faz uma introdução ao tema da tese apresentando, inicialmente, na Seção 1.1, os trabalhos relacionados ao treinamento de RNA, mais especificamente da topologia MLP, bem como o objeto de estudo desta tese; na Seção 1.2 são enumeradas as principais contribuições deste trabalho e, por fim, na Seção 1.3 apresenta-se a forma com que este documento está organizado.

1.1 Trabalhos Relacionados e Objeto de Estudo

Uma Rede Neural Artificial é um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural de armazenar conhecimento experimental e torná-lo disponível para uso (Haykin, 2001). Estas unidades (também chamadas de neurônios) são dispositivos não-lineares e adaptáveis, embora muito simples em termos de poder computacional e de memória. Porém, quando interligadas, apresentam um enorme potencial para mapeamentos não-lineares. O algoritmo de aprendizagem é o procedimento utilizado para realizar o processo de aprendizagem, cuja função é modificar os pesos sinápticos da rede de uma forma ordenada para alcançar um objetivo de projeto desejado (Haykin, 2001).

Embora inicialmente utilizadas apenas em problemas de reconhecimento de padrões e processamento de sinais e imagens, atualmente, as RNA são utilizadas para a solução de vários tipos de problemas em várias áreas do conhecimento humano.

Uma característica importante das RNA é a sua capacidade de generalização, ou seja, a capacidade da rede de apresentar respostas em relação a padrões desconhecidos ou que não

foram apresentados na etapa de treinamento. Dentre os fatores que exercem influência na capacidade de generalização das RNA, cita-se: a topologia da rede e o tipo de algoritmo utilizado para treinamento da rede (Costa, 2002).

A topologia da rede diz respeito ao número de entradas, saídas, número de camadas, número de neurônios por camada e função de ativação. A partir do trabalho de Cybenko (Cybenko, 1989), redes com a topologia MLP tiveram seu uso difundido por possuírem a característica de aproximador universal de funções contínuas. Basicamente, uma rede MLP é subdividida nas seguintes camadas: camada de entrada, camada(s) intermediária(s) ou escondida(s) e camada de saída. O funcionamento de uma rede MLP é síncrono, ou seja, dado um vetor de entrada, este é propagado para a saída multiplicando-se pelos pesos de cada camada, aplicando-se a função de ativação (o modelo de cada neurônio da rede inclui uma função de ativação não-linear, sendo a não-linearidade diferenciável em qualquer ponto) e propagando-se este valor para a camada seguinte até que a camada de saída seja atingida.

Questões como flexibilidade da rede para evitar soluções polarizadas (*underfitting*) e, em contrapartida, a limitação da complexidade da topologia da rede, evitando assim a variabilidade de soluções (*overfitting*), são aspectos inerentes à definição da melhor topologia para uma rede MLP. Esse equilíbrio entre a polarização e a variância é conhecido na literatura como "o dilema entre a polarização e a variância" (German et al., 1992).

Diversos algoritmos que buscam aprimorar a capacidade de generalização de redes MLP são propostos na literatura (Reed, 1993). Alguns algoritmos utilizam técnicas de construção, alterando a topologia da rede. Ou seja, a partir de uma rede super-dimensionada já treinada, aplicam-se métodos de *pruning* (ou poda) com o objetivo de determinar a melhor topologia considerando o melhor equilíbrio entre polarização e variância. Outros métodos, utilizam técnicas de restrição dos valores dos pesos de redes MLP sem alterar a topologia original (Teixeira, 2001), (Costa, 2002). Entretanto, nem sempre é possível medir a complexidade de um problema, o que torna a escolha da topologia da rede um processo empírico.

Em relação ao tipo de algoritmo utilizado para treinamento de redes MLP, a formulação do algoritmo de retropropagação (BP - *backpropagation*) (Rumelhart et al., 1986) possibilitou o treinamento de redes alimentadas adiante com múltiplas camadas (MLP). O algoritmo BP é baseado na regra de aprendizagem por correção de erro e pode ser visto como uma generalização do algoritmo do mínimo quadrado médio (LMS) (Widrow e Hoff, 1960), também conhecido como regra delta.

No entanto, por apresentar uma convergência lenta, dependente das condições iniciais, e poder parar o processo de treinamento em regiões de mínimos locais onde os gradientes são nulos, outros métodos de treinamento surgiram visando corrigir ou minimizar estas deficiências, tais como: Momentum (Rumelhart et al., 1986), QuickProp (Fahlman, 1988), RProp (Riedmiller e Braun, 1993), ajuste da taxa de aprendizado (Silva e Almeida, 1990), (Tollenaere, 1990), o algoritmo do gradiente conjugado (Brent, 1991), o algoritmo de Levenberg-Marquardt (Hagan e Menhaj, 1994), (Parisi et al., 1996), o algoritmo de aprendizado rápido baseado no gradiente descendente no espaço dos neurônios (Zhou e Si, 1998),

o algoritmo de aprendizado em tempo real de redes neurais com taxa de convergência exponencial (Zhao, 1996), e recentemente, uma generalização do algoritmo BP, mostrando que os algoritmos mais comuns baseados no algoritmo BP são casos especiais do algoritmo desenvolvido (Yu et al., 2002).

Porém, a despeito dos métodos citados anteriormente acelerarem a convergência da rede, não podem evitar regiões de mínimos locais (Yu et al., 2002), ou seja, regiões onde os gradientes são nulos devido a derivada da função de ativação apresentar um valor nulo ou próximo de zero, mesmo que a diferença entre a saída desejada e a saída real do neurônio seja diferente de zero.

Além dos problemas citados anteriormente, verifica-se também que a estratégia de aprendizado dos algoritmos de treinamento baseados no príncípio da retropropagação não é protegida contra distúrbios externos associados aos sinais de excitação (Efe e Kaynak, 2000), (Efe e Kaynak, 2001).

O ótimo desempenho do controle de sistemas de estrutura variável (SEV) (Itkis, 1976) em lidar com incertezas e imprecisões, tem motivado o uso do controle por modos deslizantes (CMD) (Utkin, 1978) no treinamento de RNA (Parma et al., 1998a), (Parma, 2000). Esta abordagem foi escolhida por três motivos: por ser uma teoria bem consolidada; por permitir o ajuste dos parâmetros (pesos) da rede; e, por possibilitar um estudo analítico dos ganhos envolvidos no treinamento. Dessa forma, o problema de treinamento de redes MLP é tratado e solucionado como um problema de controle, herdando características de robustez e convergência inerentes a sistemas que utilizam CMD.

Os resultados apresentados em (Efe e Kaynak, 2000), (Efe et al., 2000) mostraram que as propriedades de convergência das estratégias de treinamento de RNA baseadas no gradiente, amplamente usadas em RNA, podem ser melhoradas usando o CMD. Contudo, o método apresentado usa indiretamente a teoria de SEV. Alguns estudos usando diretamente a estratégia do CMD são também encontrados na literatura. Sira-Ramirez e Colina-Morles em seu artigo (Sira-Ramirez e Colina-Morles, 1995) propôem um algoritmo onde o ajuste dos pesos de um modelo ADALINE (Widrow e Hoff, 1960) é controlado por uma superfície para o erro instantâneo. Este método foi então estendido em (Yu et al., 1998) pela introdução de um ganho adaptativo para a lei de correção dos pesos em função da superfície de deslizamento definida. Em (Topalov et al., 2003), (Topalov e Kaynak, 2003) a estratégia de modos deslizantes para o aprendizado de redes analógicas ADALINE, proposto por (Sira-Ramirez e Colina-Morles, 1995), foi estendido para uma classe mais geral de redes multicamadas (do tipo MLP) com uma saída escalar.

O primeiro algoritmo para treinamento em tempo real de redes MLP usando CMD foi proposto por (Parma et al., 1998a). O algoritmo, além de propiciar um rápido treinamento da rede, usa a teoria de CMD para guiar o aprendizado da rede neural como um sistema a ser controlado. Este algoritmo difere daqueles apresentados em (Sira-Ramirez e Colina-Morles, 1995), (Yu et al., 1998) e (Topalov et al., 2003), principalmente, por usar superfícies de deslizamento separadas para cada camada da rede MLP. O uso do CMD para treinamento

de redes MLP possibilitou o desenvolvimento de quatro algoritmos, demonstrando a versatilidade da metodologia proposta: dois para treinamento em tempo real (Parma et al., 1998a), (Parma et al., 1998b), e dois para treinamento *off-line* [1] (Parma et al., 1999a), (Parma et al., 1999b). Uma ampla revisão sobre SEV e CMD pode ser vista em (Hung et al., 1993) e, uma recente revisão acerca da fusão de metodologias de inteligência computacional e CMD pode ser encontrada em (Kaynak et al., 2001).

Costa (Costa, 2002) propôs que o ajuste de pesos de uma rede MLP fosse controlado por uma função multi-objetivo, ou seja, o algoritmo proposto faz uso de duas superfícies de deslizamento, uma definida para o erro de treinamento e a outra para a norma do vetor de pesos. Este algoritmo (e suas variações) não foi desenvolvido para treinamento em tempo real de redes MLP.

Apesar da metodologia utilizada por Parma permitir determinar os limites de parâmetros envolvidos no treinamento de redes MLP, a sua complexidade ainda torna necessário o uso de métodos heurísticos na determinação do ganho mais adequado a ser utilizado, de forma a garantir o melhor desempenho da rede para um determinado treinamento.

Neste trabalho são desenvolvidos dois algoritmos para treinamento em tempo real de redes MLP baseados na teoria do CMD. A característica principal destes algoritmos é a obtenção de um ganho adaptativo, determinado a partir de expressões analíticas que definem seu limite mínimo e máximo, considerando uma única superfície de deslizamento para uma rede MLP. Este ganho é obtido iterativamente, a cada passo de atualização dos pesos, dispensando a necessidade do uso de métodos heurísticos na determinação do ganho final da rede. As diferenças entre os algoritmos propostos consistem, principalmente, na definição da superfície de deslizamento e na lei de correção dos pesos utilizada.

Para avaliação dos algoritmos propostos foram realizadas simulações considerando duas aplicações distintas: aproximação de função e no acionamento elétrico de um motor de indução (MI). A topologia das redes MLP utilizadas foi definida em função da melhor resposta possível com o menor número de neurônios na camada escondida, sem comprometer a capacidade de generalização da rede. As redes usadas nas simulações realizadas possuem apenas uma camada escondida, diferindo no número de neurônios desta camada e no número de entradas e saídas da rede, as quais foram escolhidas de acordo com a aplicação definida para rede MLP.

1.2 Resumo das Contribuições

As principais contribuições desta tese são as seguintes:

- Desenvolvimento de dois algoritmos para treinamento em tempo real de redes MLP e que utilizam a teoria de CMD para a determinação do ganho adaptativo da rede.

[1] Conforme definido em (Parma, 2000), o treinamento *off-line* é aquele realizado fora da operação em tempo real da rede neural e no qual tem-se o conhecimento de todo o conjunto de treinamento ou da resposta esperada da rede.

- Aplicação dos algoritmos desenvolvidos no acionamento elétrico de MI, demonstrando a generalidade e a versatilidade dos algoritmos propostos.
- Participação no desenvolvimento de uma bancada experimental utilizando recursos de hardware comerciais e que permite a implementação e avaliação de estratégias de acionamento elétrico de MI.

Estas contribuições foram apresentadas à comunidade científica por intermédio das seguintes publicações:

- Congressos nacionais e internacionais: (Nied et al., 2003a), (Nied et al., 2003b), (Justino et al., 2003), (Nied et al., 2004), (Justino et al., 2004a), (Justino et al., 2004b), (Justino et al., 2004c), (Nied et al., 2005a), (Nied et al., 2005b).
- Periódicos: (Nied et al., 2007).

1.3 Organização do Documento

O restante deste documento está organizado conforme segue:

O Capítulo 2 faz uma revisão dos principais conceitos e algoritmos de treinamento de RNA, mais especificamente de redes MLP, enfocando também a questão do treinamento em tempo real. São também revistos os conceitos de SEV e CMD. O objetivo deste capítulo é possibilitar ao leitor uma rápida revisão sobre o treinamento de redes MLP fornecendo assim, os subsídios mínimos para o entendimento dos algoritmos propostos nesta tese.

No Capítulo 3 são desenvolvidos dois algoritmos para treinamento em tempo real de redes MLP de duas camadas com a camada de saída linear, os quais possibilitam a determinação de um ganho adaptativo, obtido iterativamente, a cada passo de atualização dos pesos da rede. Os algoritmos propostos têm duas diferenças principais: a definição da superfície de deslizamento e a expressão usada para atualização dos pesos da rede. Como consequência disso, a primeira proposta é mais generalista, possibilitando o uso de redes com múltiplas saídas, enquanto a segunda proposta é limitada a apenas uma saída escalar. Por sua vez, a segunda proposta atualiza os pesos da rede usando uma lei que permite a estabilidade assintótica de acordo com a teoria de estabilidade de Lyapunov, para um conjunto de pesos que corresponde ao mínimo global.

No Capítulo 4 são apresentados os resultados de simulação dos algoritmos propostos para treinamento em tempo real de redes MLP. Os resultados apresentados consideram o uso dos algoritmos propostos em duas aplicações: na aproximação de uma função periódica e no acionamento elétrico de um MI. São também apresentados alguns resultados experimentais usando a rede neural como observador de fluxo de estator do MI e, no final do capítulo, é feita uma descrição da bancada experimental desenvolvida para a implementação e avaliação de estratégias de acionamento elétrico de MI.

Finalmente, no Capítulo 5, são feitas as principais conclusões e apresentadas as propostas de continuidade desta tese.

Capítulo 2

Treinamento de Redes MLP

Neste capítulo é apresentado um breve histórico de RNA. Na Seção 2.1 é dado o enfoque para o treinamento de redes MLP, onde a questão do treinamento em tempo real também é abordada. Visando fornecer subsídios para o entendimento dos algoritmos que serão propostos no próximo capítulo, na Seção 2.2 a teoria de SEV e CMD é revisada. Com esse mesmo objetivo, na Seção 2.3, são também apresentados: o modelo ADALINE e o algoritmo LMS ou regra delta (Widrow e Hoff, 1960), que serve de base para o entendimento do algoritmo de retropopagação; o algoritmo BP (Rumelhart et al., 1986) para uma topologia MLP de duas camadas; o algoritmo usando a estratégia de modos deslizantes para treinamento em tempo real de redes MLP com uma saída escalar (Topalov e Kaynak, 2003); e, por último, o algoritmo de modos deslizantes para treinamento em tempo real de redes MLP (Parma et al., 1998a). Por fim, na Seção 2.4, são apresentadas as conclusões do capítulo.

2.1 Introdução

O surgimento das RNA deu-se com o modelo matemático do neurônio biológico proposto por Warren McCulloch e Walter Pitts em 1943 (McCulloch e Pitts, 1943). O modelo, denominado neurônio MCP (McCulloch-Pitts), é descrito por um conjunto de n entradas ao qual cada entrada é multiplicada por um determinado peso e, em seguida, os resultados são somados e comparados a um limiar.

Em 1949, Donald Hebb mostrou como a plasticidade da aprendizagem de redes neurais é conseguida através da variação dos pesos de entrada dos neurônios (Hebb, 1949). Esta teoria deu origem a chamada "Regra de Hebb", utilizada em vários algoritmos de treinamento de RNA (Braga et al., 2000).

Em 1958, Frank Rosenblatt propôs uma topologia de rede denominada perceptron, constituída por neurônios MCP e arranjada em forma de rede composta de duas camadas (Rosenblatt, 1958). A primeira camada (camada de entrada) era constituída por pesos definidos aleatoriamente e constantes durante o treinamento, ficando o aprendizado restrito ao

ajuste dos pesos da segunda camada (camada de saída) [1] . Este tipo de perceptron comporta-se como um classificador de padrões, sendo somente capaz de classificar padrões que sejam linearmente separáveis.

A rede proposta por Rosenblatt possibilitou um aumento de trabalhos na área de RNA até 1969. Neste mesmo ano, o trabalho publicado por Minsky e Papert (Minsky e Papert, 1969) mostrando deficiências e limitações do nodo MCP provocou um desinteresse na comunidade científica pela continuação dos estudos sobre RNA. O trabalho de Minsky e Papert chamou a atenção para o fato de que o perceptron não era capaz de executar algumas tarefas, tais como detectar paridade, conectividade e simetria, as quais são exemplos de "problemas difíceis de aprender" e que formam uma classe grande de funções que não podem ser desprezadas. Afirmavam também que não havia razão para supor que qualquer uma das limitações do perceptron proposto por (Rosenblatt, 1958) (sem camadas intermediárias) poderia ser superada na versão de múltiplas camadas.

Somente a partir de 1982, com a publicação do trabalho de Hopfield (Hopfield, 1982) e com a formulação do algoritmo BP (Rumelhart et al., 1986) possibilitando o treinamento de redes alimentadas adiante (*feedforward*) com múltiplas camadas - comumente chamadas *multilayer perceptrons* - foi novamente despertado o interesse pelos estudos sobre RNA. O algoritmo BP é baseado na regra de aprendizagem por correção de erro e pode ser visto como uma generalização do algoritmo LMS (Widrow e Hoff, 1960), ou regra delta. O algoritmo LMS é simples de implementar e serve de base da filtragem adaptativa *linear*. Linear no sentido de que o neurônio opera no seu modo linear.

As redes MLP representam uma generalização do perceptron de camada única proposto por Rosenblatt e podem tratar com dados que não são linearmente separáveis, ou seja, apresentam um poder computacional muito maior do que aquele demonstrado pelas redes sem camada intermediária. Em seu trabalho publicado em 1989 (Cybenko, 1989), Cybenko prova a universalidade na aproximação de funções contínuas por RNA alimentadas adiante com uma camada escondida, utilizando função de ativação sigmoidal, e com uma camada de saída utilizando a saída linear dos neurônios. Em 1995 (Bishop, 1995), Bishop explora a relação apenas linear do número de parâmetros a serem ajustados na RNA em função do número de variáveis da função que se deseja aproximar. Os trabalhos de Cybenko e Bishop mostraram a viabilidade do uso de RNA como aproximadores universais de funções.

Uma rede MLP é subdividida em camadas: camada de entrada, camada(s) intermediária(s) ou escondida(s) e camada de saída. A Figura 2.1 mostra o grafo arquitetural de uma rede MLP com duas camadas escondidas e uma camada de saída.

Os nós de fonte da camada de entrada da rede fornecem os respectivos elementos do padrão de ativação (vetor de entrada), que constituem os sinais de entrada aplicados aos neurônios (nós computacionais) na segunda camada (ou primeira camada escondida). Os sinais de saída da segunda camada são utilizados como entradas para a terceira camada, e assim por

[1] Esta rede pode ser chamada de rede de camada única, sendo que a designação "camada única" se refere à camada de saída dos neurônios. A camada de entrada de neurônios de fonte não é contada porque os pesos desta camada são mantidos constantes durante o treinamento da rede.

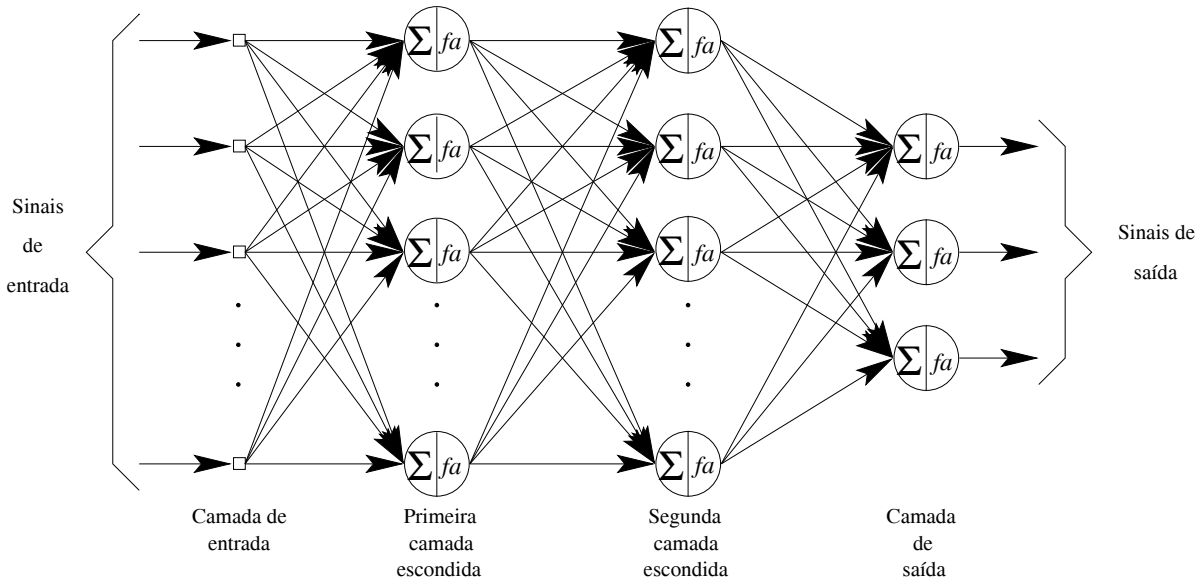


Figura 2.1: Grafo arquitetural de uma rede MLP com duas camadas escondidas.

diante para o resto da rede. Tipicamente, os neurônios em cada camada da rede têm como suas entradas apenas os sinais de saída da camada precedente, constituindo a característica *feedforward* desta rede neural. O conjunto de sinais de saída da camada de saída (final) constitui a resposta global da rede para o padrão de ativação fornecido pelos nós de fonte da camada de entrada (Haykin, 2001). Pode-se dizer também que esta rede é estritamente do tipo alimentada adiante (*feedforward*) ou acíclica.

Um perceptron de múltiplas camadas tem três características distintas (Haykin, 2001):

1. O modelo de cada neurônio (ver Apêndice A) da rede inclui uma *função de ativação não-linear*, sendo a não-linearidade diferenciável em qualquer ponto. Uma forma normalmente utilizada de não-linearidade que satisfaz esta exigência é uma *não-linearidade sigmóide* [2] definida pela *função logística*:

$$y_k = \frac{1}{1 + \exp(-v_k)} \tag{2.1}$$

   onde v_k é a soma ponderada de todas as entradas sinápticas acrescidas do *bias* do neurônio k, e y_k é a saída do neurônio. A existência de não-linearidades é importante pois, do contrário, a relação entrada-saída da rede poderia ser reduzida àquela de um perceptron de camada única (Cybenko, 1989). A escolha da função logística tem motivação biológica, pois procura levar em conta a fase refratária de neurônios reais.

2. A rede contém uma ou mais camadas de *neurônios escondidos*, que não são parte da entrada ou da saída da rede. São estes neurônios escondidos que capacitam a rede a

[2] As funções sigmóides são chamadas assim porque seus gráficos apresentam a forma de "s".

aprender tarefas complexas extraindo progressivamente as características mais significativas dos padrões (vetores) de entrada.

3. A rede exibe um alto grau de *conectividade*, determinado pelas sinapses (ou elos de conexões) da rede. Uma modificação na conectividade da rede é obtida por uma mudança no número de sinapses ou de seus pesos.

A aprendizagem de redes MLP por retropopagação (algoritmo BP) consiste de duas etapas através das diferentes camadas da rede: a *propagação* e a *retropropagação*. Na etapa de *propagação*, um padrão de ativação é aplicado aos nós da camada de entrada da rede e seu efeito se propaga através da rede, camada por camada. Na última camada, um conjunto de saídas é produzido como resposta real da rede. Deve-se salientar que, durante a etapa de propagação, os pesos sinápticos da rede são todos fixos. Durante a etapa de *retropopagação*, os pesos sinápticos são todos ajustados de acordo com uma regra de correção de erro (treinamento supervisionado), i.e., a resposta real da rede é subtraída de uma resposta desejada para produzir um sinal de erro. Este sinal de erro é então propagado para trás através da rede, contra a direção das conexões sinápticas, sendo os pesos sinápticos ajustados para fazer com que a resposta real da rede se aproxime da resposta desejada, num sentido estatístico.

O uso do algoritmo BP implica na determinação de um ganho ou taxa de aprendizado para o treinamento da rede. A escolha de um ganho *fixo* para este treinamento deve ser feita com cuidado. Se este ganho for grande, o aprendizado pode ocorrer rapidamente, mas ele pode também se tornar instável. De outro modo, se o ganho for suficientemente pequeno, pode-se garantir estabilidade no treinamento, mas existe o risco de haver um tempo de treinamento muito longo. A escolha deste ganho pode se tornar ainda mais difícil se se considerar inicializações diferentes para os pesos da rede e diferentes topologias de redes MLP. Ou seja, um bom ganho para o treinamento de uma determinada rede não é necessariamente bom para o treinamento de outra. Outra importante característica do algoritmo BP é a possibilidade dele parar o processo de treinamento da rede em regiões de mínimos locais, onde os gradientes são nulos ou próximos de zero. Estas características são devidas ao método do gradiente descendente e da regra da cadeia (Widrow e Lehr, 1990), que servem de base para a elaboração do algoritmo BP. Para questões de convergência do algoritmo BP ver (Kuan e Hornik, 1991).

Visando uma formalização dos conceitos de mínimo local e mínimo global, passa-se a definí-los a seguir. Diz-se que um vetor $\mathbf{w}^*$ é um *mínimo local* de uma função de entrada-saída f se ele não for pior que seus *vizinhos*, isto é, se existir um ε tal que (Bertsekas, 1995)

$$f(\mathbf{w}^*) \leq f(\mathbf{w}) \text{ para todo } \mathbf{w} \text{ com } \|\mathbf{w} - \mathbf{w}^*\| < \varepsilon. \tag{2.2}$$

Diz-se que um vetor $\mathbf{w}^*$ é um *mínimo global* da função f se ele não for pior que *todos* os outros vetores; isto é,

$$f(\mathbf{w}^*) \leq f(\mathbf{w}) \text{ para todo } \mathbf{w} \in \mathbb{R}^{\mathbf{n}} \tag{2.3}$$

onde $\mathbf{n}$ é a dimensão de $\mathbf{w}$ (ou para o domínio definido).

Assim, em virtude do algoritmo BP apresentar uma convergência lenta, dependente das condições iniciais, e poder parar o processo de treinamento em regiões de mínimos locais, foram desenvolvidas variações deste algoritmo com o objetivo de corrigir ou minimizar as deficiências citadas anteriormente.

Dentre os algoritmos encontrados na literatura pode-se citar: Momentum (Rumelhart et al., 1986), QuickProp (Fahlman, 1988), RProp (Riedmiller e Braun, 1993) e suas variações. Alguns algoritmos implementam técnicas de ajuste da taxa de aprendizado. Em (Silva e Almeida, 1990) é utilizada uma taxa de aprendizado para cada conexão, sendo que esta taxa é adaptada em função do sinal do gradiente do erro no instante atual e no instante anterior. Em (Tollenaere, 1990) é proposto um algoritmo semelhante ao apresentado por Silva e Almeida (1990) porém, não são feitas as atualizações nos pesos que causam as alterações no sinal do gradiente. Os dois últimos algoritmos citados, apesar de não mais dependerem das condições inicias como acontece com o algoritmo BP, necessitam da determinação de três parâmetros ao invés de um como acontece no algoritmo BP, tornando o uso destes algoritmos bastante trabalhoso, pois não existe uma metodologia para a escolha dos parâmetros, podendo ocorrer instabilidade no treinamento se for feita uma escolha inadequada destes parâmetros.

Posteriormente, foram utilizadas técnicas avançadas de otimização para implementar o ajuste dos pesos. Dentre os algoritmos que utilizam estas técnicas, cita-se o algoritmo do gradiente conjugado (Brent, 1991) e o algoritmo de Levenberg-Marquardt (Marquardt, 1963), (Hagan e Menhaj, 1994), que é o mais conhecido dentre os algoritmos que usam o método de Newton [3] . Estes algoritmos têm um ganho significativo em termos de número de iterações se comparado ao algoritmo BP mas, em contrapartida, apresentam um custo computacional [4] mais elevado. Em virtude da complexidade dos cálculos envolvidos, um aumento na dimensão da rede MLP ou do número de padrões de treinamento pode causar a perda da eficiência do algoritmo. Na tentativa de diminuir o custo computacional e melhorar as propriedades de convergência do algoritmo de Levenberg-Marquardt, algumas alternativas foram propostas, das quais pode-se citar o algoritmo de aprendizado rápido baseado na deficiência em posto da matriz jacobiana do perceptron de múltiplas camadas (Zhou e Si, 1998).

Outros algoritmos de otimização camada por camada foram propostos onde cada camada da rede MLP é decomposta em uma parte linear e uma parte não-linear (Parisi et al., 1996), (Yam e Chow, 1997), (Oh e Lee, 1999). A parte linear é resolvida via formulação do problema por mínimos quadrados. Embora estes algoritmos apresentem convergência mais rápida com menos complexidade computacional do que os algoritmos que usam o gradiente conjugado ou o método de Newton (ou quase-Newton), eles têm que lidar com a possibilidade da camada escondida não ser linearmente separável em relação a uma especificação, tornando impossível

[3] Para superar algumas dificuldades no uso do método de Newton, como por exemplo, o cálculo da matriz Hessiana inversa $\mathbf{H}^{-1}(n)$, pode-se usar um método *quase-Newton*, que requer apenas uma estimativa do vetor gradiente. Esta modificação do método de Newton mantém uma estimativa definida positiva da matriz inversa $\mathbf{H}^{-1}(n)$ diretamente, sem inversão matricial. Entretanto, ainda se tem uma complexidade computacional que é $O(W^2)$, onde W é o tamanho do vetor peso $\mathbf{w}$.

[4] Entende-se custo computacional como o número de operações computacionais e requisitos de memória necessários para se obter um resultado desejado na solução de um problema usando um determinado algoritmo.

a redução dos erros de mínimo quadrado nas camadas de saída e escondida(s).

Uma outra classe de algoritmos de aprendizado rápido para treinamento de redes MLP é baseada na técnica de filtro de Kalman extendido (EKF) (Iiguni et al., 1992). Estes algoritmos aumentam a taxa de convergência do treinamento da rede consideravelmente e exibem um bom desempenho, porém, a estabilidade numérica não é garantida. Isto pode degradar a convergência do aprendizado, aumentar o tempo de treinamento e, geralmente, pode fazer com que implementações em tempo real destes algoritmos sejam questionáveis.

Adicionalmente, pode-se dizer que os algoritmos de otimização não são aplicáveis para os casos em tempo real, nos quais novos exemplos de entradas e saídas são adicionados continuamente ao conjunto de treinamento, uma vez que eles somente podem iniciar o processo de otimização quando todas as amostras estão disponíveis. Em relação ao treinamento em tempo real, pode-se citar o algoritmo de aprendizado em tempo real de redes neurais com taxa de convergência exponencial (Zhao, 1996). Este algoritmo usa uma técnica para busca do mínimo de uma função de custo temporal, onde o gradiente da função se aproxima de zero exponencialmente, fazendo com os pesos da rede sejam rapidamente atualizados para o mínimo da função objetivo temporal, continuando assim durante todo o processo de aprendizado.

Mais recentemente, foi proposta uma generalização do algoritmo BP para redes alimentadas adiante, o qual unifica as variações do algoritmo BP (Yu et al., 2002). Uma função de Lyapunov é usada para uma análise rigorosa da convergência dos pesos, sendo mostrado que é inerente aos algoritmos de treinamento baseados no príncípio da retropropagação (algoritmos derivados do BP) ficarem presos em um mínimo local durante o treinamento, ou seja, apesar dos métodos citados anteriormente acelerarem a convergência da rede, não podem evitar regiões de mínimos locais. A convergência dos pesos para um mínimo global somente é possível se os pesos iniciais estiverem próximos do mínimo global, ou se a distribuição geométrica dos pesos permitir a eles a convergência para o mínimo global.

Além disso, pode-se constatar que, os algoritmos que usam a informação do gradiente são bastante sensíveis à presença de distúrbios externos associados aos sinais de alimentação da rede, os quais podem excitar dinâmicas internas indesejáveis nestes algoritmos (Efe e Kaynak, 2000), (Efe e Kaynak, 2001). Devido a multidimensionalidade do problema de treinamento de RNA, uma análise mais detalhada visando distinguir a informação útil do distúrbio relacionado aos sinais de excitação é mais uma dificuldade a ser apontada.

Portanto, diante das questões mencionadas anteriormente, tornar o algoritmo de treinamento mais robusto é uma necessidade inevitável. Assim, uma estratégia de treinamento baseada na teoria de sistemas de estrutura variável (SEV) (Itkis, 1976) usando controle por modos deslizantes (CMD) (Utkin, 1978), a qual é bem desenvolvida para controle de sistemas não-lineares incertos, parece ser uma boa condidata para eliminar os efeitos adversos presentes nos sinais de excitação. A idéia subjacente tem sido a de explorar as propriedades de invariância introduzidas pela teoria de SEV junto com a flexibilidade paramétrica das arquiteturas de RNA, permitindo assim um ajuste dos pesos da rede e possibilitando um estudo analítico dos ganhos envolvidos no treinamento. A propriedade mais significativa do sistema usando

CMD é sua robustez. De uma maneira geral, pode-se dizer que quando o sistema está em um modo deslizante, ele é insensível às variações paramétricas ou distúrbios externos. Na Seção 2.2 é feita uma revisão de SEV e CMD.

Os resultados apresentados em (Efe e Kaynak, 2000), (Efe et al., 2000) mostraram que as propriedades de convergência das estratégias de treinamento de RNA baseadas no gradiente podem ser melhoradas usando o CMD. Contudo, o método apresentado usa indiretamente a teoria de SEV. Outros estudos usando diretamente a estratégia do CMD são também encontrados na literatura. Usando idéias de controle por modos quase-deslizantes [5] , (Sira-Ramirez e Zak, 1991) propõe uma modificação da regra Delta, por meio da qual uma estratégia chaveada de adaptação dos pesos consegue impor uma dinâmica linear de tempo discreto e estável assintoticamente para o aprendizado do erro entre a saída atual da rede e o valor desejado. Este algortimo foi proposto para treinamento de perceptrons simples e de múltiplas camadas com função de ativação não-linear descontínua, e considerava que o mesmo vetor de entrada era apresentado em sucessivas iterações.

Em seguida, Sira-Ramirez e Colina-Morles em seu artigo (Sira-Ramirez e Colina-Morles, 1995) propoêm um algoritmo que usa CMD em tempo contínuo para o ajuste robusto dos pesos de um modelo analógico ADALINE (Widrow e Hoff, 1960) com uma saída escalar. A correção do erro de saída do modelo é feita por uma superfície para o erro instantâneo e uma lei de correção dos pesos é proposta. Esta lei induz o estado do sistema, em tempo finito, a um regime de deslizamento, o qual mantém, de forma robusta, a condição de erro zero. O algoritmo proposto considera vetores de entrada não-constantes, i.e., com dependência temporal, sendo por isso adequado para treinamento em tempo real. Este método foi então estendido em (Yu et al., 1998) pela introdução de um ganho adaptativo para a lei de correção dos pesos em função da superfície de deslizamento definida.

O primeiro algoritmo para treinamento em tempo real de redes MLP usando CMD foi proposto por (Parma et al., 1998a). O algoritmo, além de propiciar um rápido treinamento da rede, usa a teoria de CMD para guiar o aprendizado da rede neural como um sistema a ser controlado. Este algoritmo tem como características principais: é genérico, podendo ser aplicado a qualquer configuração de redes MLP; define superfícies de deslizamento distintas para a camada de saída e para a(s) camada(s) escondida(s); possibilita a determinação dos limites para os parâmetros envolvidos no treinamento da rede. A partir da metodologia proposta, foram desenvolvidos dois algoritmos para o treinamento em tempo real e dois para treinamento *off-line* de redes MLP (Parma, 2000).

Costa (Costa, 2002) propôs um algoritmo multi-objetivo que utiliza CMD para o treinamento de redes MLP. O algoritmo é capaz de controlar a trajetória da rede em um plano de estados definido por duas funções objetivo: o erro de treinamento e a norma do vetor de

[5] No desenvolvimento da teoria de SEV e CMD em tempo contínuo, supõe-se que o sistema a ser controlado será chaveado com frequência infinita, possibilitando o estado do sistema deslizar sobre a superfície de deslizamento. Na implementação em tempo discreto, deve-se levar em conta que o sistema será chaveado com uma frequência máxima limitada pela frequência de amostragem. Desta forma, o regime de deslizamento não será ideal, sendo chamado de regime *quase-deslizante*, onde o estado do sistema estará numa vizinhança da superfície de deslizamento.

pesos. Este algoritmo (e suas variações) não foi desenvolvido para treinamento em tempo real de redes MLP.

Em (Topalov et al., 2003) e (Topalov e Kaynak, 2003), a estratégia de modos deslizantes para o aprendizado de redes analógicas ADALINE, proposto por (Sira-Ramirez e Colina-Morles, 1995), foi estendida para uma classe mais geral de redes com múltiplas camadas (do tipo MLP) com uma saída escalar. Com isso, as limitações existentes nas propostas de (Sira-Ramirez e Zak, 1991) e (Sira-Ramirez e Colina-Morles, 1995) devido ao uso do perceptron ou do modelo ADALINE, são em muito diminuídas. A estratégia proposta tem as seguintes características principais: foi desenvolvida para treinamento em tempo real de redes MLP com uma saída linear (é usada apenas a saída linear do neurônio de saída); considera a rede MLP com uma única camada escondida; é definida apenas uma superfície de deslizamento para o erro instantâneo de saída da rede. Segundo Topalov et al. (2003), a limitação de apenas uma saída escalar para a rede MLP não deve ser considerada tão restritiva em relação a aplicabilidade da proposta, uma vez que é possível se ter duas ou mais estruturas de redes MLP compartilhando as mesmas entradas.

Assim, independentemente do algoritmo utilizado para treinamento em tempo real de redes neurais, tanto a dimensão *espaço* como a dimensão *tempo* são fundamentais no processo de aprendizagem. Quando uma rede opera em um ambiente *estacionário* (i.e., um ambiente cujas características estatísticas não mudam com o tempo), as estatísticas essenciais deste ambiente podem ser, pelo menos em teoria, aprendidas pela rede através de uma aprendizagem supervisionada. Partindo-se dessa premissa, pode-se então calcular os pesos sinápticos da rede submetendo-a a uma sessão de treinamento usando um conjunto de dados que é representativo do ambiente. Após o término do treinamento, os pesos sinápticos da rede capturariam a estrutura estatística subjacente do ambiente, o que justificaria o "congelamento" de seus valores após isto. Desta forma, o sistema de aprendizagem se baseia em uma memória de longo prazo para recordar e explorar experiências passadas.

Por sua vez, se uma rede opera em um ambiente *não-estacionário*, os parâmetros estatísticos dos sinais gerados por este ambiente variam com o tempo. Neste caso, os métodos tradicionais de aprendizagem supervisionada podem se mostrar inadequados, pois a rede não está equipada com os meios necessários para seguir as variações estatísticas do ambiente no qual opera (Haykin, 2001). Para superar esta dificuldade, é desejável que uma rede neural possa *adaptar* continuamente seus parâmetros livres às variações do sinal incidente em tempo real, ou seja, possa ter uma aprendizagem contínua ou em tempo real.

Uma forma de abordar a questão da aprendizagem contínua ou em tempo real é considerando que as características estatísticas de um processo não-estacionário normalmente variam de forma suficientemente lenta para que o processo possa ser considerado *pseudo-estacionário* em uma janela de tempo com duração suficientemente curta. A formulação desta abordagem pode ser feita da seguinte maneira (Zhao, 1996): considere que as m_0 entradas e as m_L saídas desejadas de uma rede estritamente do tipo alimentada adiante são $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_{m_0}(t)]^T$ e $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_{m_L}(t)]^T$, respectivamente, onde

m_l representa o tamanho (número de neurônios) da camada l do perceptron de múltiplas camadas; $l = 0, 1, ..., L$, sendo L a profundidade da rede.

A soma do erro quadrático de saída no instante de tempo t é dada por

$$e(\mathbf{w}, t) = \frac{1}{2} \sum_{k=1}^{m_L} [y_k(t) - \varphi_k(\mathbf{w}, \mathbf{x})]^2 \tag{2.4}$$

onde $\varphi_k(\mathbf{w}, \mathbf{x})$ é a saída atual da k-ésima unidade de saída da rede e $\mathbf{w}$ é o vetor de todos os pesos da rede. O aprendizado é realizado através da atualização dos pesos com o objetivo de minimizar a função custo:

$$f(\mathbf{w}, t) = \frac{1}{t} \int_0^t e(\mathbf{w}, \tau) d\tau. \tag{2.5}$$

Para o aprendizado em tempo real, a função custo deveria ser formulada como

$$f(\mathbf{w}, t) = \frac{1}{d} \int_{t-d}^t e(\mathbf{w}, \tau) d\tau. \tag{2.6}$$

O processo de aprendizado em tempo real é, na realidade, um processo de atualização dos pesos da rede para aprender as relações entrada-saída durante $[t-d, t]$, onde d é o comprimento da janela de tempo. Contudo, para um conjunto de dados discretos, desde que a avaliação dos erros pode ser feita somente em "momentos discretos", pode-se reescrever (2.6) como

$$f(\mathbf{w}, n) = \lim_{d \to 0} \frac{1}{d} \int_{n-d}^n e(\mathbf{w}, \tau) d\tau = e(\mathbf{w}, n) \tag{2.7}$$

onde n representa o tempo discreto.

Assim, através desta abordagem pode-se incorporar a estrutura temporal no projeto de uma rede neural fazendo com que ela sofra *treinamento continuado* com *exemplos ordenados no tempo*. De acordo com esta abordagem dinâmica, uma rede neural é vista como um *filtro adaptativo não-linear*, representando uma generalização dos filtros adaptativos lineares (Haykin, 2001). Entretanto, para que esta abordagem possa ser realizável, os recursos disponíveis devem ser suficientemente rápidos para completar os cálculos necessários durante um período de amostragem, permitindo que o filtro acompanhe as variações na entrada.

Uma outra forma de abordar a questão da aprendizagem contínua ou em tempo real é utilizar uma rede neural cuja estrutura seja adequada para este tipo de aprendizagem, i.e., incorporando memória de curto prazo na estrutura da rede neural através de *atrasos de tempo*, que podem ser implementados a nível sináptico dentro da rede ou na camada de entrada da rede. Verifica-se assim, uma clara separação de responsabilidades: a rede estática é responsável pela não-linearidade, e a memória é responsável pelo tempo.

A *rede alimentada adiante atrasada no tempo* (TLFN - *time lagged feedforward network*) *focada* é um exemplo da incorporação de atrasos de tempo na camada de entrada da rede, conforme mostra a Figura 2.2 (Haykin, 2001). Ela é uma combinação de elementos de atraso

unitário e pesos sinápticos associados e pode ser vista como um *filtro de resposta a impulso de duração finita* (FIR - *finite-duration impulse response*) de ordem p, conforme mostrado na Figura 2.3 (Haykin, 2001).

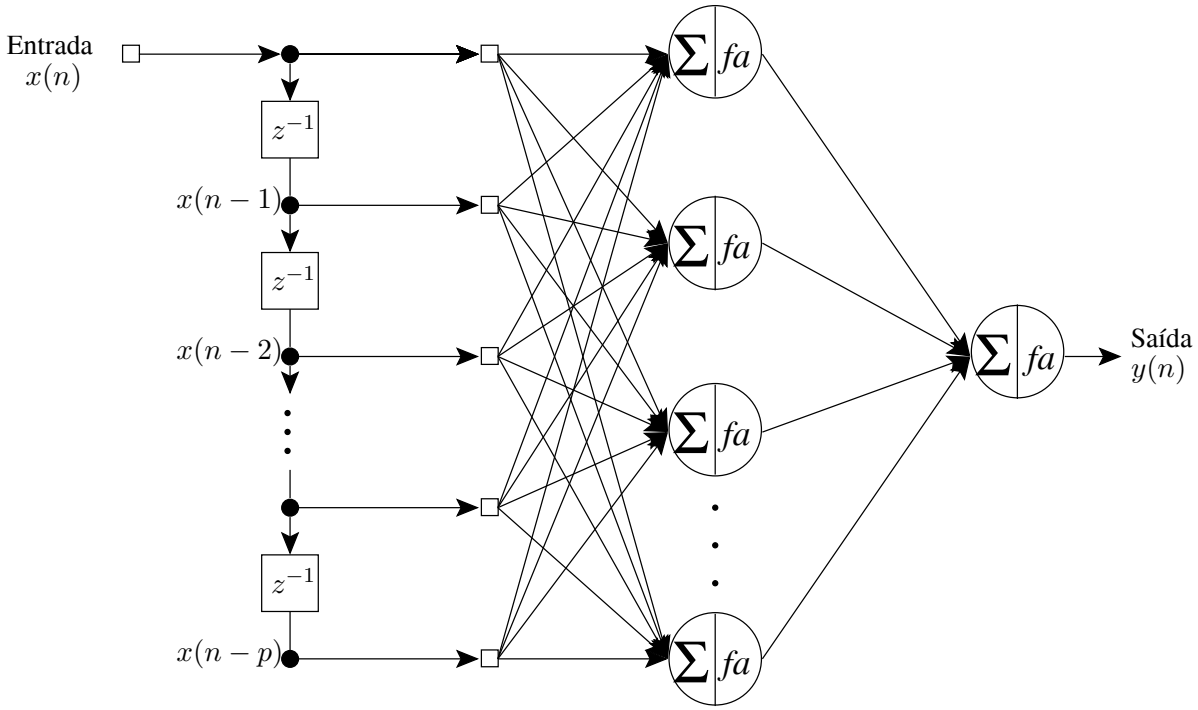


Figura 2.2: Rede TLFN focada.

A rede TLFN focada é treinada utilizando-se o algoritmo BP (Rumelhart et al., 1986) e é adequada apenas para uso em ambientes estacionários. Para superar esta limitação pode-se utilizar uma *rede alimentada adiante atrasada no tempo* (TFLN - *time lagged feedforward network*) *distribuída*, distribuindo através da rede a influência implícita do tempo. A construção desta rede é baseada no filtro neural de múltiplas entradas da Figura 2.4 como modelo espaço-temporal de um neurônio (Haykin, 2001). Esta rede pode ser convenientemente treinada utilizando o algoritmo BP *temporal* (Wan, 1990a), (Wan, 1990b).

Uma terceira forma de incorporar o tempo na operação de uma rede neural de uma maneira implícita é através do uso de *realimentação*. Redes recorrentes são aquelas que possuem um ou mais laços de realimentação que proporcionam comportamento dinâmico. Esta realimentação pode ser *local* ao nível de um neurônio dentro da rede, ou *global* abrangendo toda a rede. Há muitas variações de arquiteturas de redes recorrentes porém, todas elas compartilham as seguintes características comuns: incorporam um perceptron de múltiplas camadas *estático* ou partes dele; compartilham a capacidade de mapeamento não-linear do perceptron de múltiplas camadas. Pode-se treinar uma rede recorrente usando o algoritmo de retropropagação através do tempo (Werbos, 1990), que opera com a premissa de que a operação temporal de uma rede recorrente pode ser desdobrada em um perceptron de múltiplas camadas, permitindo assim a

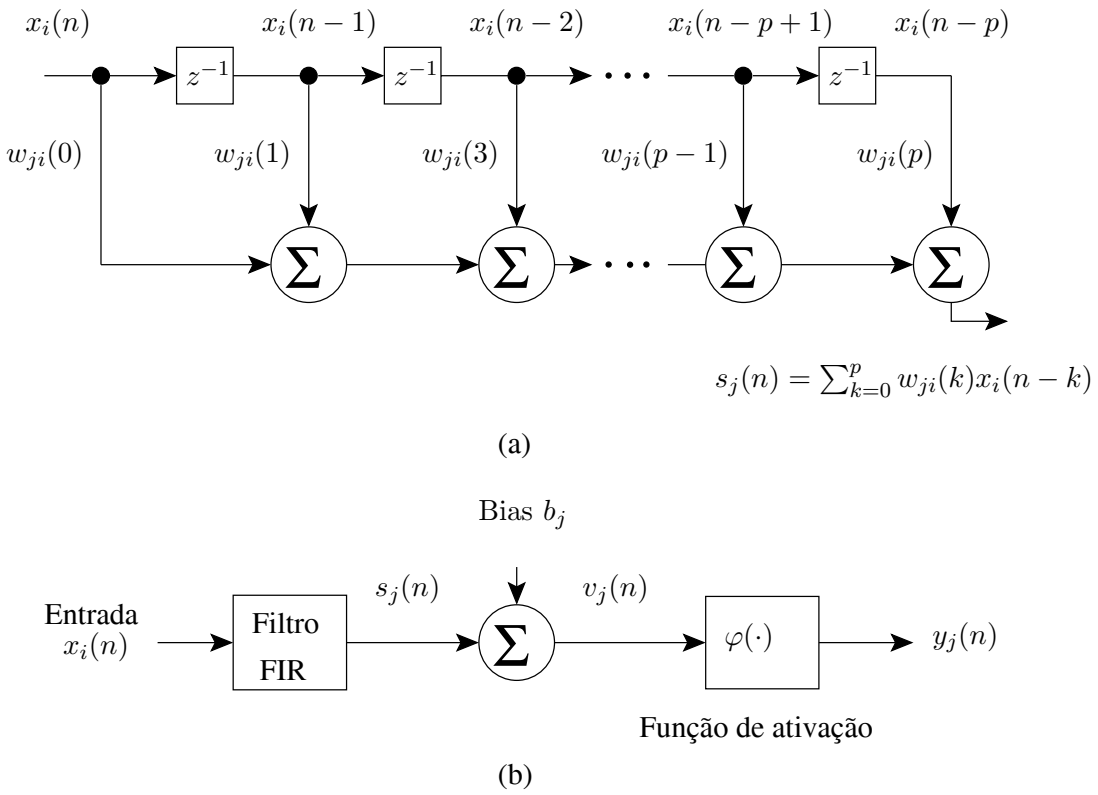


Figura 2.3: (a) Filtro FIR; (b) Interpretação do filtro neural como um filtro FIR não-linear.

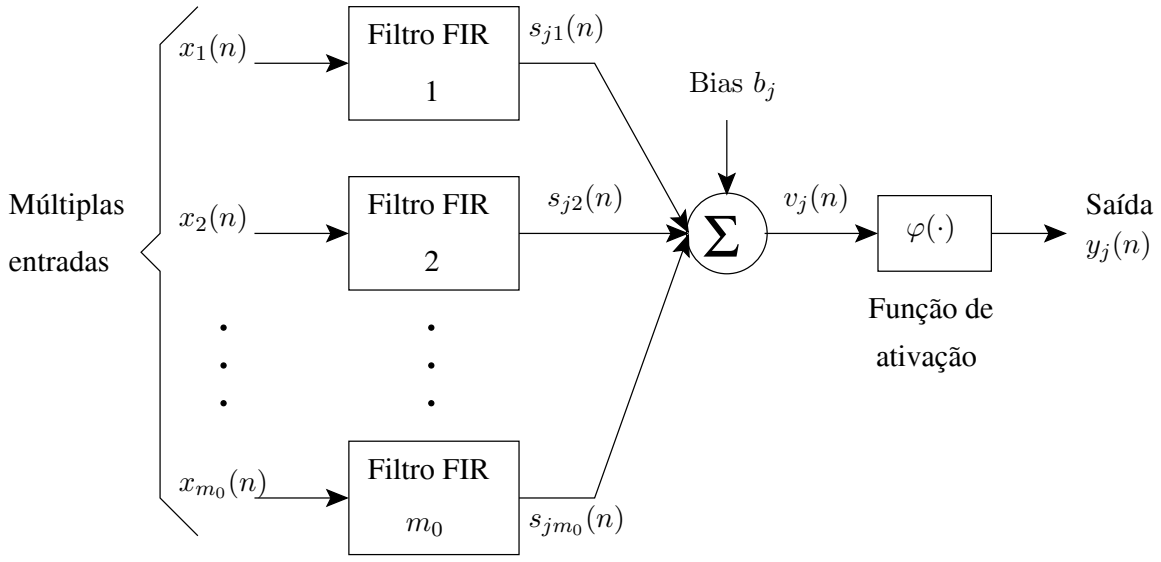


Figura 2.4: Filtro neural de múltiplas entradas.

aplicação do algoritmo de retropropagação normal. Uma outra possibilidade de treinar uma rede recorrente é através do algoritmo de aprendizagem recorrente em tempo real (Williams e Peng, 1989), que deriva seu nome do fato de serem realizados ajustes em tempo real dos pesos sinápticos de uma rede recorrente totalmente conectada.

Verifica-se, portanto, que o treinamento em tempo real tem, dentre outras, as seguintes

características (Parma, 2000):

- Não possui, a *priori*, os vetores de dados com os quais a rede será treinada.
- Os vetores de treinamento não são, na sua maioria, armazenados.
- O fator tempo tem que ser considerado como elemento sequenciador dos vetores de treinamento.

Portanto, a partir destas características, é muito importante que o algoritmo utilizado para o treinamento em tempo real seja eficiente, possibilitando um treinamento rápido e com baixo custo computacional.

A necessidade do processamento temporal aparece em diversas aplicações das quais pode-se citar (Haykin, 2001):

- Previsão e modelagem de séries temporais (Haykin, 1996);
- Cancelamento de ruído (Widrow e Steams, 1985), (Haykin, 1996);
- Equalização adaptativa de um canal de comunicação desconhecido (Proakis, 1989), (Haykin, 1996);
- Controle adaptativo (Narendra e Parthasarathy, 1990);
- Identificação de sistemas (Ljung, 1987);

Na próxima seção é feita uma breve revisão de teoria de SEV e CMD, apresentando os conceitos mínimos necessários para o entendimento dos algoritmos que serão mostrados na Seção 2.3 e no Capítulo 3.

2.2 Revisão de Sistemas de Estrutura Variável

A teoria de sistemas de estrutura variável foi primeiramente proposta por (Emelyanov, 1959). Porém, devido às dificuldades de implementação, somente a partir de 1970 a proposta recebeu a devida atenção. A principal característica de um SEV é que o sinal de realimentação é descontínuo, chaveando entre uma ou mais superfícies no espaço de estados. Quando o estado do sistema cruza uma dessas superfícies de chaveamento, a estrutura do sistema realimentado é alterada. Sob certas condições, os movimentos numa vizinhança de uma superfície podem ser direcionados para a superfície e em consequência, um movimento de deslizamento em um subespaço pré-definido do espaço de estados é estabelecido, no qual o estado do sistema repetidamente cruza a superfície de chaveamento (Utkin, 1992). Este movimento de deslizamento ou chaveamento, conhecido na literatura como modos deslizantes (Utkin, 1978), tem propriedades de invariância úteis diante de incertezas no modelo da planta, que o torna um bom candidato para o controle de sistemas não-lineares incertos.

O controlador a estrutura variável em regime de modos deslizantes (CMD), força a trajetória dos estados do sistema para um lugar no espaço de estados, cuja dinâmica é escolhida pelo projetista, e onde o sistema, de uma maneira geral, seja insensível a variações paramétricas ou distúrbios externos. O CMD é usado em várias áreas de aplicação (Young et al., 1999) e deve esta popularidade ao seu bom desempenho no controle de sistemas não-lineares, à sua aplicabilidade em sistemas com várias entradas e saídas e, na existência de critérios de projeto bem definidos para sistemas de tempo contínuo.

Em aplicações práticas, o CMD sofre de algumas desvantagens, tais como: impossibilidade de chavear instantaneamente o controle de um valor para outro, devido às não-linearidades de um sistema real. Por causa disto, o CMD nem sempre conseguirá manter a trajetória de estados do sistema deslizando sobre a superfície de deslizamento, originando o que se conhece como *chattering*, i.e., um chaveamento de alta frequência em torno da superfície de deslizamento. Uma outra desvantagem está relacionada ao fato do CMD ser extremamente vulnerável a ruídos de medição, uma vez que a entrada depende do sinal de uma variável medida que é muito próxima de zero (Bartoszewicz, 1998b). Outras duas desvantagens estão relacionadas ao uso de sinais de controle desnecessariamente grandes para superar incertezas paramétricas e, a existência de dificuldades apreciáveis no cálculo do que se conhece como controle equivalente (Kaynak et al., 2001).

Para aliviar as dificuldades citadas anteriormente, muitas modificações têm sido propostas para a lei de controle de modos deslizantes original. A mais popular, no entanto, é aquela que utiliza uma camada limite em torno das superfícies de chaveamento. Dentro da camada limite, o controle deixa de ser descontínuo e passa a ser um controle contínuo de alto ganho. Outras questões relativas a SEV e CMD podem ser vistas em (Hung et al., 1993).

A seguir, é apresentada uma formulação básica para o projeto de um CMD. Intuitivamente, um SEV com um CMD é baseado no argumento de que o controle de sistemas de primeira ordem é mais fácil, mesmo quando eles são não-lineares ou incertos, do que o controle de sistemas de ordem maior (Kaynak et al., 2001).

A. Descrição da Dinâmica de uma Planta Geral sob Controle

Considere um sistema não-linear não-autônomo de múltiplas entradas múltiplas saídas (MIMO) da forma

$$x_i^{(k_i)} = f_i(\mathbf{X}) + \sum_{j=1}^{m} b_{ij} u_j \tag{2.8}$$

onde $x_i^{(k_i)}$ indica a k-ésima derivada de x_i e

$$\mathbf{X} = [x_1 \;\; \dot{x}_1 \;\; \ldots \;\; x_1^{k_{1-1}} \;\; \ldots x_m \;\; \dot{x}_m \;\; \ldots \;\; x_m^{k_{m-1}}]^T. \tag{2.9}$$

Definindo

$$\mathbf{U} = [u_1 \;\; \ldots u_m]^T \tag{2.10}$$

e assumindo que $\mathbf{X}$ é $(n \times 1)$, a equação do sistema torna-se

$$\dot{\mathbf{X}}(t) = \mathbf{F}(\mathbf{X}) + \mathbf{B}\mathbf{U}(t) \tag{2.11}$$

onde $\mathbf{B}$ é a matriz $(n \times m)$ de ganhos da entrada. Tais sistemas são chamados sistemas quadrados pois eles têm tantos controles de entradas quanto as saídas x_i a serem controladas.

B. Determinação da Superfície de Deslizamento

Para o sistema dado em (2.11), a superfície de deslizamento $\mathbf{S}\,(m \times 1)$ é selecionada geralmente como

$$\mathbf{S}(\mathbf{X}, t) = \mathbf{G}(\mathbf{X}^d(t) - \mathbf{X}(t)) = \mathbf{\Phi}(t) - \mathbf{S}_a(\mathbf{X}) \tag{2.12}$$

onde

$$\mathbf{\Phi}(t) = \mathbf{G}\mathbf{X}^d(t) \;\; e \;\; \mathbf{S}_a(\mathbf{X}) = \mathbf{G}\mathbf{X}(t) \tag{2.13}$$

são as partes da função de deslizamento dependentes do tempo e do estado, respectivamente.

Em (2.12), $\mathbf{X}^d$ representa o vetor de estado desejado (referência) e $\mathbf{G}$ é a matriz $(m \times n)$ de ganhos da superfície de deslizamento. Geralmente, a matriz $\mathbf{G}$ é selecionada de modo que a função da superfície de deslizamento torna-se

$$s_i(t) = \left(\frac{de_i(t)}{dt} + \lambda_i e_i(t)\right)^{k_i - 1} \tag{2.14}$$

onde e_i é o erro para $x_i (e_i = x_i^d - x_i)$ e os λ_i's são selecionados como constantes positivas. Portanto, e_i vai para zero quando s_i torna-se zero. O CMD força os estados do sistema para a superfície de deslizamento. Uma vez que os estados estejam na superfície de deslizamento, os erros do sistema convergem para zero com uma dinâmica ditada pela matriz $\mathbf{G}$. Assim, para condições iniciais diferentes de zero, pode-se distinguir duas fases para o CMD: a fase de *alcance*, que compreende o instante inicial até o momento em que o vetor erro toca a superfície de deslizamento; a fase de *deslizamento*, quando então (2.14) torna-se zero, forçando o vetor erro a se mover para a origem.

C. Projeto do CMD

Existem alguns métodos a disposição do projetista para o projeto do CMD. O método descrito a seguir, é baseado na seleção de uma função de Lyapunov. O controle deve ser escolhido de modo que a função condidata de Lyapunov satisfaça o critério de estabilidade de Lyapunov.

A função de Lyapunov é selecionada como

$$\mathbf{V}(\mathbf{S}) = \frac{\mathbf{S}^T\mathbf{S}}{2}. \tag{2.15}$$

Deve ser notado que esta função deve ser positiva definida ($\mathbf{V}(\mathbf{S}=0)=0$ e $\mathbf{V}(\mathbf{S})>0\ \forall\ \mathbf{S}\neq 0$).

É requerido também que a derivada da função de Lyapunov seja definida negativa. Isto pode ser garantido se a expressão a seguir for verificada

$$\frac{d\mathbf{V}(\mathbf{S})}{dt} = -\mathbf{S}^T\mathbf{D}\ sign(\mathbf{S}) \tag{2.16}$$

onde, $\mathbf{D}$ é a $(m \times m)$ matriz diagonal positiva definida de ganhos e $sign(\mathbf{S})$ denota a função sinal, aplicada a cada elemento de $\mathbf{S}$, i.e.,

$$sign\mathbf{S} = [sign(s_1)\ \ \ldots\ \ sign(s_m)]^T \tag{2.17}$$

e $sign(s_i)$ é definido como

$$sign(s_i) = \begin{cases} +1, & s_i\ >\ 0 \\ 0, & s_i\ =\ 0 \\ -1, & s_i\ <\ 0. \end{cases} \tag{2.18}$$

Tomando-se a derivada de (2.15) e igualando a (2.16) obtém-se a seguinte equação:

$$\mathbf{S}^T\frac{d\mathbf{S}}{dt} = -\mathbf{S}^T\mathbf{D}\ sign(\mathbf{S}). \tag{2.19}$$

Tomando a derivada temporal de (2.12) e usando a equação da planta

$$\frac{d\mathbf{S}}{dt} = \frac{d\mathbf{\Phi}}{dt} - \frac{\partial\mathbf{S_a}}{\partial\mathbf{X}}\frac{d\mathbf{X}}{dt} = \frac{d\mathbf{\Phi}}{dt} - \mathbf{G}(\mathbf{F}(\mathbf{X}) + \mathbf{B}\mathbf{U}) \tag{2.20}$$

é obtido. Substituindo (2.20) em (2.19), o controle do sinal de entrada pode ser obtido como

$$\mathbf{U}(t) = \mathbf{U}_{eq}(t) + \mathbf{U}_c(t) \tag{2.21}$$

onde $\mathbf{U}_{eq}(t)$ é o controle equivalente e é escrito como

$$\mathbf{U}_{eq}(t) = -(\mathbf{G}\mathbf{B})^{-1}\left(\mathbf{G}\mathbf{F}(\mathbf{X}) - \frac{d\mathbf{\Phi}(t)}{dt}\right). \tag{2.22}$$

e $\mathbf{U}_c(t)$ é o termo corretivo do controle e é escrito como

$$\mathbf{U}_c(t) = (\mathbf{G}\mathbf{B})^{-1}\mathbf{D}\ sign(\mathbf{S}) = \mathbf{K}\ sign(\mathbf{S}). \tag{2.23}$$

A despeito das dificuldades práticas da implementação de um esquema de CMD, o sinal de controle em (2.21) é aplicável se uma representação nominal do sistema sob controle está disponível.

2.3 Algoritmos de Treinamento de Redes MLP

Nesta seção são apresentados conceitos básicos sobre filtragem adaptativa e alguns algoritmos para treinamento de redes MLP, os quais foram selecionados com o objetivo de fornecer subsídios para um melhor entendimento dos algoritmos que serão propostos no Capítulo 3. Inicialmente, são apresentadas a notação adotada e algumas definições que servem de base para a derivação dos algoritmos utilizados no decorrer do texto da tese.

Notação e Definições

- O tempo contínuo é representado na variável independente t, e o tempo discreto é representado por n.
- Na iteração (passo de tempo) n, o n-ésimo padrão de treinamento (exemplo) é apresentado à rede.
- Os índices i, j e k se referem a neurônios diferentes na rede; com os sinais se propagando através da rede da esquerda para a direita, o neurônio j se encontra em uma camada à direita do neurônio i, e o neurônio k se encontra em uma camada à direita do neurônio j, quando o neurônio j é uma camada oculta.
- O i-ésimo elemento do vetor de entrada é representado por $x_i(n)$.
- O k-ésimo elemento do vetor de saída global é representado por $y_k(n)$.
- O símbolo $y_j(n)$ se refere ao sinal funcional que aparece na saída do neurônio j, na iteração n.
- O símbolo $d_j(n)$ se refere à resposta desejada para o neurônio j e é usado para calcular $e_j(n)$.
- O símbolo $e_j(n)$ se refere ao sinal de erro na saída do neurônio j, para a iteração n.
- O símbolo $E(n)$ se refere à soma instantânea dos erros quadráticos ou energia do erro na iteração n. A média de $E(n)$ sobre todos os valores de n (i.e., o conjunto inteiro de treinamento) produz a energia média do erro médio E_{med}.
- O símbolo $w_{ji}(n)$ representa o peso sináptico conectando a saída do neurônio i à entrada do neurônio j, na iteração n. A correção aplicada a este peso na iteração n é representada por $\Delta w_{ji}(n)$.
- O campo local induzido (i.e., a soma ponderada de todas as entradas sinápticas acrescida do bias) do neurônio j na iteração n é representado por $v_j(n)$; constitui o sinal aplicado à função de ativação associada com o neurônio j.
- A função de ativação, que descreve a relação funcional de entrada-saída da não-linearidade associada ao neurônio j, é representada por $\varphi_j(\cdot)$.

- O bias aplicado ao neurônio j é representado por b_j; o seu efeito é representado por uma sinapse de peso $w_{j0} = b_j$ conectada a uma entrada fixa igual a $+1$.

- O parâmetro de aprendizagem é representado por η.

- O símbolo m_l representa o tamanho (i.e., o número de neurônios) da camada l do perceptron de múltiplas camadas; $l = 0, 1, \ldots, L$, onde L é a "profundidade" da rede. Assim, m_0 representa o tamanho da camada de entrada, m_1 representa o tamanho da primeira camada oculta e m_L representa o tamanho da camada de saída.

- Para redes com uma ou mais camadas escondidas (i.e., $l \geq 2$), deve-se acrescentar um índice a mais nas variáveis pesos sinápticos, campo local induzido e saídas, a fim de identificar a que camada se referem. Por exemplo, para uma rede com uma camada oculta ($l = 2$) tem-se: $w1_{11}, v1_1, y1_1$ são variáveis da camada oculta e $w2_{11}, v2_1, y2_1$ são variáveis da camada de saída.

2.3.1 ADALINE e a Regra Delta

O perceptron é a forma mais simples de uma rede neural usada para a classificação de padrões ditos linearmente separáveis, i.e., padrões que se encontram em lados opostos de um hiperplano. Ele consiste, basicamente, de um único neurônio com pesos sinápticos ajustáveis e bias, e um limitador abrupto na saída. O algoritmo usado para ajustar os pesos sinápticos deste neurônio foi primeiro proposto por (Rosenblatt, 1958) para o seu modelo cerebral do perceptron.

O modelo ADALINE do neurônio, proposto por (Widrow e Hoff, 1960), apesar de ser topologicamente semelhante ao perceptron, ajusta os pesos sinápticos utilizando o chamado algoritmo LMS, também conhecido como regra delta. Este algoritmo usa o gradiente descendente da função de custo da saída linear do neurônio, ou seja, antes da aplicação da função de ativação não-linear.

O neurônio único também forma a base de um *filtro adaptativo linear*, linear no sentido de que o neurônio opera no seu modo linear, utilizando para treinamento o algoritmo LMS, o qual é simples de implementar porém, muito efetivo em relação à sua aplicação. A Figura 2.5 mostra o grafo de fluxo de sinal do modelo ADALINE e do filtro adaptativo linear.

Desta forma, o modelo neuronal opera sob a influência de um algoritmo que *controla* os ajustes necessários dos pesos sinápticos do neurônio, consistindo dos seguintes passos:

- O algoritmo inicia com uma configuração arbitrária para os pesos sinápticos do neurônio.

- Os ajustes dos pesos sinápticos, em resposta a variações estatísticas do comportamento do sistema, são feitos de uma forma contínua (i.e., o tempo é incorporado na constituição do algoritmo) e consistem de dois processos: processo de *filtragem*, envolvendo a computação do sinal de saída $j(n)$ e do sinal de erro $e(n)$; e o processo *adaptativo*, que envolve o ajuste dos pesos de acordo com o sinal de erro calculado no instante n. Estes

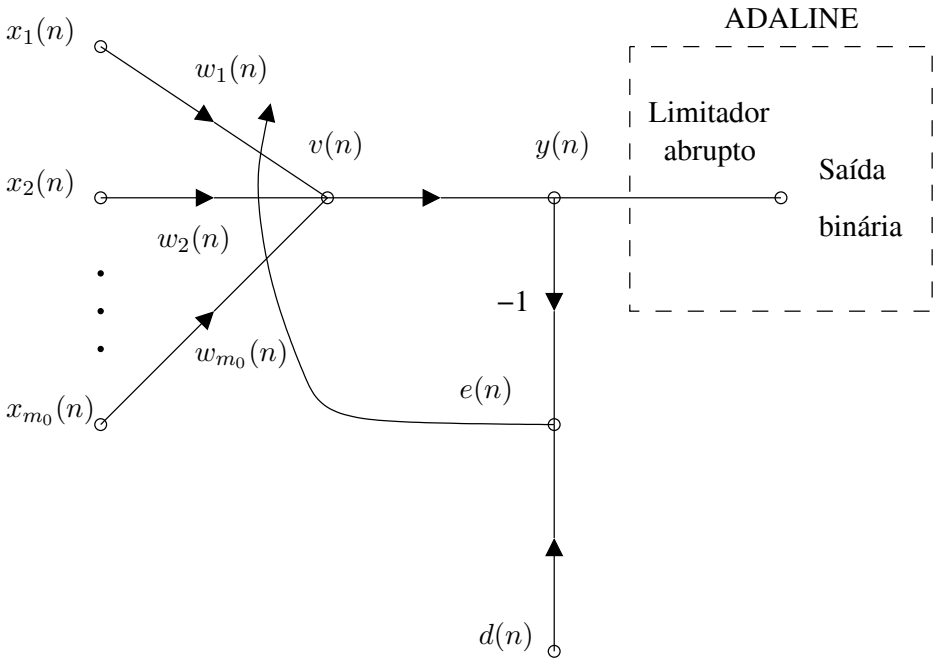


Figura 2.5: Grafo de fluxo de sinal do modelo ADALINE (saída binária) e do filtro adaptativo linear ($y(n)$).

dois processos atuando juntos contituem um *laço de realimentação* que age em torno do neurônio.

- Os cálculos dos ajustes dos pesos sinápticos são completados dentro de um intervalo de tempo que é igual a um período de amostragem.

A maneira pela qual o sinal de erro $e(n)$ é usado para controlar os ajustes dos pesos sinápticos do neurônio é determinado pela função de custo utilizada para derivar o algoritmo de filtragem adaptativa de interesse.

O algoritmo LMS (ou regra delta), por exemplo, é baseado na utilização de *valores instantâneos* para a função custo, ou seja,

$$E(\mathbf{w}) = \frac{1}{2}\, e^2(n). \tag{2.24}$$

Diferenciando $E(\mathbf{w})$ em relação ao vetor de peso $\mathbf{w}$, obtém-se

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = e(n)\frac{\partial e(n)}{\partial \mathbf{w}}. \tag{2.25}$$

O algoritmo LMS opera com um neurônio linear de forma que se pode expressar o sinal de erro como

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n). \tag{2.26}$$

Com isso,

$$\frac{\partial e(n)}{\partial \mathbf{w}(n)} = -\mathbf{x}(n) \tag{2.27}$$

e

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}(n)} = -\mathbf{x}(n)e(n). \tag{2.28}$$

Utilizando este último resultado para o vetor do gradiente, pode-se escrever

$$\nabla E(\mathbf{w}) = \mathbf{g}(n) = -\mathbf{x}(n)e(n). \tag{2.29}$$

Finalmente, usando (2.29) para o vetor gradiente do método de descida mais íngreme [6], pode-se formular o algoritmo LMS como segue:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta\mathbf{g}(n) \tag{2.30}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\mathbf{x}(n)e(n). \tag{2.31}$$

O laço de realimentação em torno do vetor de peso $\mathbf{w}(n)$ no algoritmo LMS se comporta como um *filtro passa-baixa*, deixando passar as componentes de baixa frequência do sinal de erro e atenuando suas componentes de alta frequência (Haykin, 1996). Deve-se ressaltar ainda que a expressão (2.31) produz uma estimativa do vetor de peso que resultaria da utilização do método da descida mais íngreme, ou seja, o vetor de peso $\mathbf{w}(n)$ traça uma trajetória aleatória ao invés de uma trajetória bem definida no espaço de estados (Haykin, 2001). Por esta razão, o algoritmo LMS é algumas vezes denominado "algoritmo do gradiente estocástico".

2.3.2 Algoritmo BP

O algortimo BP foi formulado por (Rumelhart et al., 1986) e é descrito a seguir conforme apresentado em (Haykin, 2001). O sinal de erro na saída do neurônio j, na iteração n, sendo o neurônio j um neurônio de saída da rede, é definido por

$$e_j(n) = d_j(n) - y_j(n). \tag{2.32}$$

Definindo, conforme anteriormente, o valor instantâneo da energia do erro para o neurônio j como $\frac{1}{2}e_j^2(n)$, correspondentemente, o valor instantâneo $E(n)$ da energia total do erro é obtido somando-se os termos $\frac{1}{2}e_j^2(n)$ de *todos os neurônios da camada de saída*, ou seja,

$$E(n) = \frac{1}{2}\sum_{j \in \mathcal{C}} e_j^2(n) \tag{2.33}$$

[6] No método de descida mais íngreme, os ajustes sucessivos aplicados ao vetor de peso $\mathbf{w}$ são na direção da descida mais íngreme, i.e., na direção oposta ao vetor do gradiente $\nabla E(\mathbf{w})$.

onde o conjunto $\mathcal{C}$ inclui todos os neurônios da camada de saída da rede. Considere que N represente o número total de padrões (exemplos) contidos no conjunto de treinamento. A *energia média do erro quadrado* é obtida somando-se os $E(n)$ para todos os n e então normalizando em relação ao tamanho do conjunto N, ou seja,

$$E_{med} = \frac{1}{N} \sum_{n=1}^{N} E(n). \tag{2.34}$$

Para um dado conjunto de treinamento, E_{med} representa a *função de custo* como uma medida do desempenho de aprendizagem, ou seja, o objetivo do processo de aprendizagem é ajustar os parâmetros livres da rede para minimizar E_{med}.

Antes da continuação da apresentação do algoritmo, é conveniente que sejam feitos alguns comentários a respeito dos modos de treinamento sequencial e por lote. Para um dado conjunto de treinamento, a aprendizagem por retropropagação pode proceder de uma entre duas formas básicas: no modo *sequencial* (também conhecido como *padrão* ou *estocástico* [7]), a partir de uma época consistindo de N exemplos (vetores) de treinamento arranjados na ordem $(\mathbf{x}(1), \mathbf{d}(1)), \ldots, (\mathbf{x}(N), \mathbf{d}(N))$, a atualização dos pesos é realizada após a apresentação de cada exemplo de treinamento; no modo *por lote*, o ajuste dos pesos é realizado após a apresentação de todos os exemplos de treinamento que constituem uma época. Questões relativas às vantagens e desvantagens de cada modo fogem ao escopo desta tese porém, pode-se dizer que o modo sequencial é muito popular para uso na aprendizagem por retropopagação por duas razões: o algoritmo é simples de implementar e, ele fornece soluções efetivas a problemas complexos.

O algortimo BP utiliza uma aproximação similar *em raciocínio* àquela usada na derivação do algoritmo LMS para minimizar E_{med}, ou seja, os pesos são atualizados de padrão em padrão até formar uma época. A média aritmética destas alterações individuais de peso sobre o conjunto de treinamento é, na realidade, uma estimativa da alteração real que resultaria da modificação dos pesos baseada na minimização da função de custo E_{med} sobre o conjunto de treinamento inteiro. Questões relativas à qualidade desta estimativa podem ser encontradas, por exemplo, em (Haykin, 2001).

Considere então a Figura 2.6, que representa o neurônio j recebendo um conjunto de sinais funcionais produzidos por uma camada de neurônios à sua esquerda.

O campo local induzido $v_j(n)$ produzido na entrada da função de ativação associada ao neurônio j é

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{2.35}$$

onde m é o número total de entradas (incluindo o bias) aplicadas ao neurônio j. Assim, o

[7] O modo sequencial também pode ser chamado de *on-line*, porém, para evitar confusão com a denominação do treinamento em tempo real, ele foi omitido.

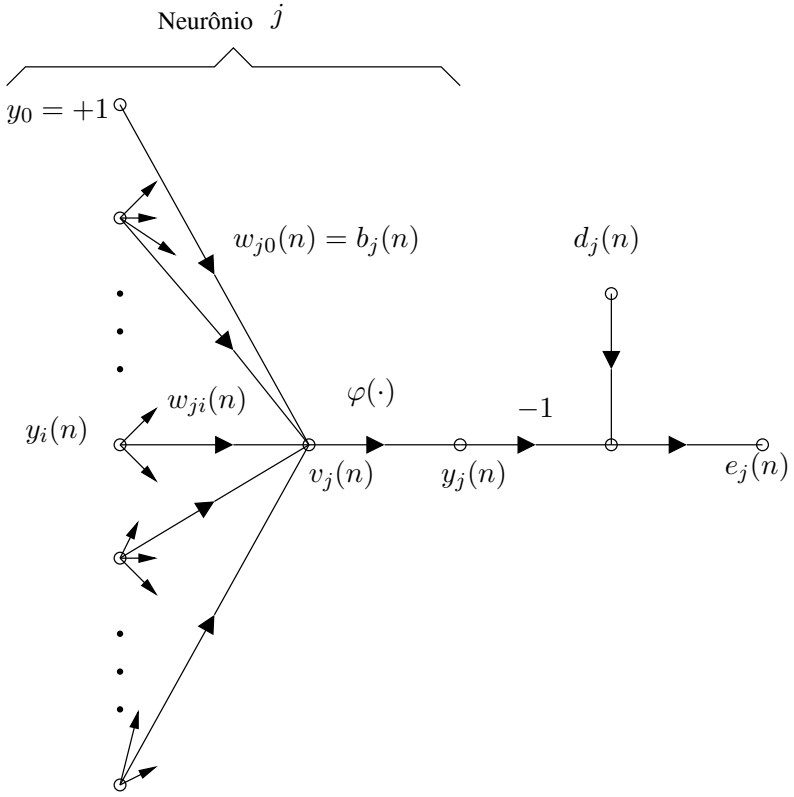


Figura 2.6: Grafo de fluxo de sinal do neurônio de saída j.

sinal funcional $y_j(n)$ que aparece na saída do neurônio j na iteração n é

$$y_j(n) = \varphi_j(v_j(n)). \tag{2.36}$$

De forma similar ao algoritmo LMS, o algoritmo BP aplica uma correção $\Delta w_{ji}(n)$ ao peso sináptico $w_{ji}(n)$, que é proporcional à derivada parcial $\frac{\partial E(n)}{\partial w_{ji}(n)}$. De acordo com a regra da cadeia, pode-se expressar este gradiente como:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_i(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{2.37}$$

A derivada parcial $\frac{\partial E(n)}{\partial w_{ji}(n)}$ determina a direção de busca no espaço de pesos para o peso sináptico w_{ji}.

Diferenciando ambos os lados de (2.33) em relação a $e_j(n)$ obtém-se

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n). \tag{2.38}$$

Diferenciando ambos os lados de (2.32) em relação a $y_j(n)$, obtém-se

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1. \tag{2.39}$$

A seguir, diferenciando (2.36) em relação a $v_j(n)$, obtém-se

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \dot{\varphi}_j(v_j(n)), \tag{2.40}$$

mostrando que a função de ativação é diferenciada em relação ao argumento. Finalmente, diferenciar (2.35) em relação a $w_{ji}(n)$ produz

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n). \tag{2.41}$$

O uso das Eqs. (2.38) a (2.41) em (2.37) produz

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\dot{\varphi}_j(v_j(n))y_i(n). \tag{2.42}$$

A correção $\Delta w_{ji}(n)$ aplicada a $w_{ji}(n)$ é definida pela *regra delta*:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{2.43}$$

onde η é o parâmetro da taxa de aprendizagem do algoritmo de retropropagação. O uso do sinal negativo indica a descida do gradiente no espaço de pesos. Correspondentemente, o uso de (2.42) em (2.43) produz

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{2.44}$$

onde o gradiente local $\delta_j(n)$ é definido por

$$\begin{aligned} \delta_j(n) &= -\frac{\partial E(n)}{\partial v_j(n)} \\ &= -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_i(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= e_j(n)\dot{\varphi}_j(v_j(n)), \end{aligned} \tag{2.45}$$

apontando para as modificações necessárias nos pesos sinápticos.

Pode-se notar a partir de (2.44) e (2.45), que um fator importante envolvido no cálculo do ajuste do peso $\Delta w_{ji}(n)$ é o sinal de erro $e_j(n)$ na saída do neurônio j. Neste contexto, pode-se identificar dois casos:

Caso 1 - O neurônio j é um neurônio de saída

Neste caso, pode-se utilizar (2.32) para calcular o sinal de erro $e_j(n)$ associado com este neurônio. Tendo-se determinado $e_j(n)$, pode-se então calcular diretamente o gradiente local $\delta_j(n)$ usando (2.45).

Caso 2 - O neurônio j é um neurônio oculto

Quando o neurônio j está localizado em uma camada oculta da rede, não existe uma resposta desejada especificada para aquele neurônio. Consequentemente, o sinal de erro para um neurônio oculto deve ser determinado recursivamente, em termos dos sinais de erro de todos os neurônios aos quais o neurônio oculto está diretamente conectado.

Considere a situação mostrada na Figura 2.7, que representa o neurônio j como um nó oculto da rede.

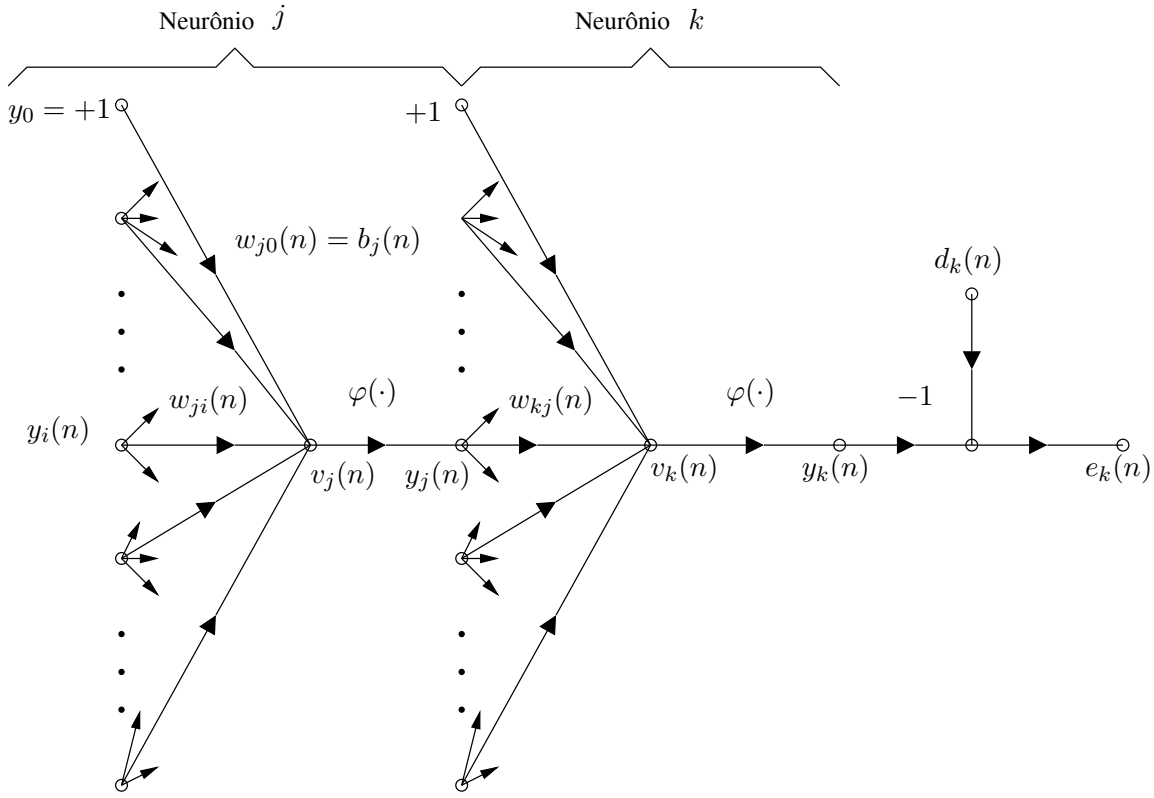


Figura 2.7: Grafo de fluxo de sinal do neurônio de saída k conectado ao neurônio oculto j.

De acordo com (2.45), pode-se redefinir o gradiente local $\delta_j(n)$ para o neurônio oculto j como

$$\begin{aligned} \delta_j(n) &= -\frac{\partial E(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= -\frac{\partial E(n)}{\partial y_j(n)}\dot{\varphi}_j(v_j(n)), \end{aligned} \tag{2.46}$$

onde, para o segundo termo do lado direito de (2.46) foi usada (2.40). Para calcular a derivada parcial $\frac{\partial E(n)}{\partial y_j(n)}$, procede-se da seguinte maneira. Da Figura 2.7 vê-se que

$$E(n) = \frac{1}{2} \sum_{k \in \mathcal{C}} e_k^2(n), \tag{2.47}$$

que é (2.33) com o índice k no lugar do índice j, pois agora o neurônio k é o neurônio de saída. Diferenciando (2.47) em relação ao sinal funcional $y_j(n)$, obtém-se

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)}. \tag{2.48}$$

A seguir, utiliza-se a regra da cadeia para a derivada parcial $\frac{\partial e_k(n)}{\partial y_j(n)}$ reescrevendo então (2.48) na forma equivalente

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}. \tag{2.49}$$

Entretanto, da Figura 2.7 nota-se que

$$\begin{aligned} e_k(n) &= d_k(n) - y_k(n) \\ &= d_k(n) - \varphi_k(v_k(n)). \end{aligned} \tag{2.50}$$

Assim,

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\dot{\varphi}_k(v_k(n)). \tag{2.51}$$

Também pode-se notar da Figura 2.7 que para o neurônio k o campo local induzido é

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n), \tag{2.52}$$

onde m é o número total de entradas (incluindo o bias) aplicadas ao neurônio k. Diferenciar (2.52) em relação a $y_j(n)$ produz

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n). \tag{2.53}$$

Utilizando as Eqs. (2.51) e (2.53) em (2.49), obtém-se a derivada parcial desejada

$$\begin{aligned} \frac{\partial E(n)}{\partial y_j(n)} &= -\sum_k e_k(n) \dot{\varphi}_k(v_k(n)) w_{kj}(n) \\ &= -\sum_k \delta_k(n) w_{kj}(n) \end{aligned} \tag{2.54}$$

onde, na segunda linha foi utilizada a definição do gradiente local $\delta_k(n)$ dada em (2.45), com

o índice j substituído por k.

Finalmente, utilizando (2.54) em (2.46), obtém-se a *fórmula de retropropagação* para o gradiente local $\delta_j(n)$ como descrito:

$$\delta_j(n) = \dot{\varphi}_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n). \tag{2.55}$$

O fator $\dot{\varphi}_j(v_j(n))$ envolvido na cálculo do gradiente local $\delta_j(n)$ depende unicamente da função de ativação associada ao neurônio oculto j. O fator restante envolvido no cálculo, i.e., o somatório sobre k, depende de dois termos: o primeiro, $\delta_k(n)$, requer conhecimento dos sinais de erro $e_k(n)$, para todos os neurônios que se encontram na camada imediatamente à direita do neurônio oculto j e que estão diretamente conectados ao neurônio j; o segundo, $w_{kj}(n)$, consiste dos pesos sinápticos associados com estas conexões.

Para concluir a descrição do algoritmo BP, são necessários ainda dois comentários: o primeiro está relacionado com a função de ativação. O cálculo do δ para cada neurônio do perceptron de múltiplas camadas requer o conhecimento da derivada da função de ativação $\varphi_j(\cdot)$ associada àquele neurônio. Para esta derivação existir, é necessário que a função $\varphi_j(\cdot)$ seja contínua. Ou seja, a *diferenciabilidade* é a única exigência que a função de ativação deve satisfazer. Uma função de ativação não-linear, continuamente diferenciável, normalmente utilizada pelas redes MLP é a função sigmóide (ver Apêndice A).

O segundo comentário diz respeito a taxa de aprendizagem η. O algortimo BP fornece uma "aproximação" para a trajetória no espaço dos pesos calculada pelo método da descida mais íngreme. Assim, um pequeno η causa pequenas variações nos pesos sinápticos implicando numa taxa de aprendizagem lenta. Para um η maior, pode-se acelarar a taxa de aprendizagem porém, isto pode tornar a rede instável. Um método simples de melhorar esta dicotomia é modificar a regra delta em (2.44) incluindo um termo de momento (Rumelhart et al., 1986):

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n), \tag{2.56}$$

onde α é usualmente um número positivo chamada de *constante de momento*. A Eq. (2.56) é chamada de *regra delta generalizada*, porque inclui a regra delta como um caso especial (i.e., para $\alpha = 0$).

Recentemente, (Behera et al., 2006) propuseram um algoritmo para treinamento de redes MLP com taxa de aprendizado adaptativa que garante convergência global. Esta taxa de aprendizado adaptativa é formalmente obtida utilizando uma função de Lyapunov. Os resultados de simulação apresentados no artigo mostraram que o algoritmo proposto é mais rápido do que o algoritmo BP convencional.

2.3.3 Algoritmo de Modos Deslizantes para Redes com uma Saída Escalar

Conforme mencionado na Seção 2.2, existem algumas dificuldades de ordem prática para a implementação do CMD. Uma das propostas usadas para reduzir estas dificuldades consiste na fusão de metodologias de inteligência computacional [8] com técnicas de CMD. Na situação inversa, o uso do CMD em sistemas baseados em inteligência computacional permite uma análise rigorosa de projeto e estabilidade destes sistemas.

Os primeiros trabalhos usando CMD para treinamento de um neurônio foram apresentados por (Sira-Ramirez e Zak, 1991) e (Sira-Ramirez e Colina-Morles, 1995), gerando resultados teóricos e de simulação bastante interessantes. Esta estratégia de CMD para aprendizado de um modelo ADALINE foi então estendida para uma classe mais geral de redes MLP contendo uma saída escalar (Topalov et al., 2003), (Topalov e Kaynak, 2003). Os resultados teóricos desta última proposta foram então validados experimentalmente em (Giordano et al., 2004) e (Cascella et al., 2005). Nestes trabalhos, o CMD foi utilizado para treinar redes do tipo MLP, as quais foram usadas para identificação em tempo real de manipuladores robóticos e no controle adaptativo de motores elétricos. A proposta apresentada em (Topalov et al., 2003) é mostrada a seguir.

A. Definições e Suposições Iniciais

Considere uma rede MLP de duas camadas. As seguintes definições serão usadas para a derivação do algortimo de adaptação dos pesos sinápticos em tempo real:

$\mathbf{x}(t) = [x_1(t),\ \ldots\ , x_i(t),\ \ldots\ , x_{m_0}(t)]^T$ - vetor dos sinais de entrada aumentado pelo termo de bias.

$\mathbf{y1}(t) = [y1_1(t),\ \ldots\ , y1_j(t),\ \ldots\ , y1_{m_1}(t)]^T$ - vetor dos sinais de saída dos neurônios da camada escondida.

$y(t)$ - sinal escalar representando a saída (única) da rede. O índice $l = 2$ foi suprimido visando uma melhor legibilidade do algoritmo, tendo em vista a existência de apenas um neurônio de saída.

$\mathbf{W1}(t)$ - matriz $(m_1 \times m_0)$ das conexões dos pesos entre os neurônios da entrada e da camada escondida, onde cada elemento da matriz $w1_{ji}(t)$ significa o peso da conexão do neurônio j para sua entrada i. Esta matriz é considerada aumentada por incluir os componentes do peso de bias para os neurônios da camada escondida.

$\mathbf{w2}(t)$ - vetor $(1 \times (m_1 + 1))$ das conexões dos pesos entre os neurônios da camada escondida e o neurônio de saída. Este vetor é considerado aumentado por incluir o componente do peso de bias para o neurônio de saída.

$\mathbf{v1}(t)$ - vetor dos sinais de saída dos neurônios da camada escondida antes da aplicação da função de ativação.

$\varphi(\cdot)$ - função de ativação não-linear, diferenciável e monotonicamente crescente dos neurônios da camada escondida da rede (por exemplo, função logística ou função tangente hiperbólica).

[8] Esta terminologia abriga técnicas definidas pelo Prof. L. A. Zadeh como *soft-computing*, pois são orientadas para o projeto e análise de sistemas inteligentes, sendo as RNA uma das técnicas empregadas para este fim.

Para o *neurônio da camada de saída é considerada apenas a sua saída linear.*

Uma suposição é feita de que o vetor de entrada $\mathbf{x}(t) = [x_1(t),\ \ldots\ , x_i(t),\ \ldots\ , x_{m_0}(t)]^T$ e sua derivada $\dot{\mathbf{x}}(t) = [\dot{x}_1(t),\ \ldots\ , \dot{x}_i(t),\ \ldots\ , \dot{x}_{m_0}(t)]^T$ são limitados, i.e.

$$\|\mathbf{x}(t)\| = \sqrt{x_1^2(t),\ \ldots\ , x_i^2(t),\ \ldots\ , x_{m_0}^2(t)}\ \leq B_x,\ \forall t \tag{2.57}$$

e

$$\|\dot{\mathbf{x}}(t)\| = \sqrt{\dot{x}_1^2(t),\ \ldots\ , \dot{x}_i^2(t),\ \ldots\ , \dot{x}_{m_0}^2(t)}\ \leq B_{\dot{x}},\ \forall t \tag{2.58}$$

onde B_x e $B_{\dot{x}}$ são constantes positivas. Supõe-se também que, devido à restrições físicas, a magnitude de todos os vetores linha $\mathbf{w1}_j(t)$ que constituem a matriz $\mathbf{W1}(t)$ e os elementos do vetor $\mathbf{w2}(t)$ são todos limitados em cada instante de tempo t por meio de

$$\|\mathbf{w1}_j(t)\| = \sqrt{w1_{j1}^2(t),\ \ldots\ , w1_{ji}^2(t),\ \ldots\ , w1_{jm_0}^2(t)}\ \leq B_{w1},\ \forall t \tag{2.59}$$

$$\mid w2_j(t) \mid \leq B_{w2},\ \forall t, \tag{2.60}$$

onde B_{w1} e B_{w2} são constantes conhecidas.

O sinal de saída $y1_j(t)$ do j-ésimo neurônio da camada escondida e o sinal de saída da rede $y(t)$ são definidos como:

$$y1_j(t) = \varphi\left(\sum_{i=1}^{m_0} w1_{ji}(t)x_i(t)\right) \tag{2.61}$$

e

$$y(t) = \sum_{j=1}^{m_1+1} w2_j(t)y1_j(t), \tag{2.62}$$

sendo $0 < A_j(t) = \dot{\varphi}\left(\sum_{i=1}^{m_0} w1_{ji}(t)x_i(t)\right) \leq B_A\ \forall i, j$ a derivada da função de ativação $\varphi(\cdot)$ dos neurônios e B_A corresponde a seu valor máximo, $x_{m_0}(t) = 1$, $y1_{m_1+1}(t) = 1$ e, $\dot{y1}_{m_1+1}(t) = 0$.

O sinal escalar $d(t)$ representa a saída desejada para a rede neural. É assumido que $d(t)$ e $\dot{d}(t)$ são sinais limitados, i.e.

$$\mid d(t) \mid \leq B_d,\ \mid \dot{d}(t) \mid \leq B_{\dot{d}}\ \forall t, \tag{2.63}$$

onde B_d e $B_{\dot{d}}$ são constantes positivas.

Define-se o erro de aprendizado $e(t)$ como uma quantidade escalar obtida a partir de

$$e(t) = y(t) - d(t). \tag{2.64}$$

Usando a abordagem do CMD, define-se o valor zero do erro de aprendizagem $e(t)$ como

uma superfície de deslizamento, i.e.

$$s(e(t)) = e(t) = y(t) - d(t) = 0, \tag{2.65}$$

a qual é a condição que garante que a saída $y(t)$ da rede neural coincide com o sinal de saída desejado $d(t)$ para todo $t > t_h$, onde t_h é o tempo de alcance (*hitting time*) para $e = 0$.

B. O Algoritmo de Treinamento em Tempo Real baseado em CMD

Definição 2.1 *Um movimento de deslizamento irá ocorrer em uma superfície de deslizamento* $s(e(t)) = e(t) = y(t) - d(t) = 0$*, após o tempo* t_h*, se a condição* $s(t)\dot{s}(t) = e(t)\dot{e}(t) < 0$ *é verdadeira para todo* t *em um subintervalo não-trivial semi-aberto de tempo da forma* $[t, t_h) \subset (-\infty, t_h)$*.*

O algoritmo de aprendizado para os pesos da rede neural $\mathbf{W}1(t)$ e $\mathbf{w}2(t)$ deve ser obtido de tal forma que a condição de modos deslizantes definida anteriormente seja imposta. Denotando como $sign(e(t))$ a função sinal do erro definida como:

$$sign(e(t)) = \begin{cases} +1, & e(t) > 0 \\ 0, & e(t) = 0 \\ -1, & e(t) < 0, \end{cases} \tag{2.66}$$

para possibilitar que $s = 0$ seja alcançado, usa-se o seguinte teorema:

Teorema 2.1 *Se o algoritmo de aprendizado para os pesos* $\mathbf{W}1(t)$ *e* $\mathbf{w}2(t)$ *é escolhido, respectivamente, como*

$$\dot{w}1_{ji}(t) = -\left(\frac{w2_j(t)x_i(t)}{\mathbf{x}^T(t)\mathbf{x}(t)}\right)\eta\ sign(e(t)) \tag{2.67}$$

$$\dot{w}2_j(t) = -\left(\frac{y1_j(t)}{\mathbf{y}1^T(t)\mathbf{y}1(t)}\right)\eta\ sign(e(t)) \tag{2.68}$$

com η *sendo uma constante positiva que satisfaça a seguinte inequação*

$$\eta > m_1 B_A B_{w1} B_{\dot{x}} B_{w2} + B_{\dot{d}} \tag{2.69}$$

então, para qualquer condição inicial arbitrária $e(0)$*, o erro de aprendizado* $e(t)$ *irá convergir para zero durante um tempo finito* t_h *que pode ser estimado como*

$$t_h \leq \frac{|e(0)|}{\eta - m_1 B_A B_{w1} B_{\dot{x}} B_{w2} - B_{\dot{d}}} \tag{2.70}$$

e um movimento de deslizamento irá ser mantido em $e = 0$ *para todo* $t > t_h$*.* ◇

Prova: Ver (Topalov et al., 2003).

Para o caso do vetor $\dot{\mathbf{x}}(t)$ ser mensurável, uma estratégia de CMD mais relaxada do que aquela apresentada em (2.67) e (2.68) pode ser obtida. Com isso, menores ganhos η são requeridos para se obter um movimento de deslizamento correspondente. Para maiores detalhes ver (Topalov e Kaynak, 2003).

Nas duas propostas desenvolvidas e apresentadas em (Topalov et al., 2003) e (Topalov e Kaynak, 2003), o ganho da rede é obtido de maneira heurística, e é fixo durante todo o treinamento.

2.3.4 Algoritmo de Modos Deslizantes para Redes com Múltiplas Saídas

O primeiro algoritmo para treinamento em tempo real de redes MLP com múltiplas saídas usando CMD foi proposto por (Parma et al., 1998a). O algoritmo, além de poder ser aplicado a qualquer configuração de redes MLP, tem como principal característica a definição de superfícies de deslizamento distintas para a camada de saída e para a(s) camada(s) escondida(s) da rede. Além disso, o uso da teoria do CMD possibilita a determinação dos limites teóricos para os parâmetros das superfícies definidas para cada camada. A partir da metodologia proposta, foram desenvolvidos quatro algoritmos, sendo dois deles para o treinamento em tempo real de redes MLP (Parma, 2000). A primeira proposta foi originalmente apresentada em tempo contínuo (Parma et al., 1998a), (Parma, 2000), mas aqui é descrita na versão em tempo discreto, conforme apresentada em (Justino, 2004). A segunda proposta é descrita em tempo discreto como apresentada em (Parma, 2000).

A. Definições e Suposições Iniciais

Considere uma rede MLP de duas camadas. As seguintes definições serão usadas para a obtenção dos algoritmos de adaptação dos pesos sinápticos em tempo real:

$\mathbf{x}(n) = [x_1(n),\ \ldots\ , x_i(n),\ \ldots\ , x_{m_0}(n)]^T$ - vetor dos sinais de entrada aumentado pelo termo de bias.

$\mathbf{y1}(n) = [y1_1(n),\ \ldots\ , y1_j(n),\ \ldots\ , y1_{m_1}(n)]^T$ - vetor dos sinais de saída dos neurônios da camada escondida.

$\mathbf{y2}(n) = [y2_1(n),\ \ldots\ , y2_k(n),\ \ldots\ , y2_{m_2}(n)]^T$ - vetor dos sinais de saída dos neurônios da camada de saída.

$\mathbf{W1}(n)$ - matriz $(m_1 \times m_0)$ das conexões dos pesos entre os neurônios da entrada e da camada escondida, onde cada elemento da matriz $w1_{ji}(n)$ significa o peso da conexão do neurônio j para sua entrada i. Esta matriz é considerada aumentada por incluir os componentes do peso de bias para os neurônios da camada escondida.

$\mathbf{W2}(n)$ - matriz $(m_2 \times (m_1+1))$ das conexões dos pesos entre os neurônios da camada escondida e da saída, onde cada elemento da matriz $w2_{kj}(n)$ significa o peso da conexão do neurônio k para sua entrada j. Esta matriz é considerada aumentada por incluir os componentes do peso de bias para os neurônios da camada de saída.

$\mathbf{v1}(n)$ - vetor dos sinais de saída dos neurônios da camada escondida antes da aplicação da função de ativação.

$\mathbf{v2}(n)$ - vetor dos sinais de saída dos neurônios da camada de saída antes da aplicação da função de ativação.

$\varphi(\cdot)$ - função de ativação não-linear, diferenciável e monotonicamente crescente dos neurônios da camada escondida e de saída da rede (por exemplo, função logística ou função tangente hiperbólica).

Supõe-se que o vetor de entrada $\mathbf{x}(n)$ seja limitado, i.e.,

$$\|\mathbf{x}(n)\| = \sqrt{x_1^2(n),\ \ldots\ , x_i^2(n),\ \ldots\ , x_{m_0}^2(n)}\ \leq B_x,\ \forall n. \tag{2.71}$$

Supõe-se também que, a magnitude de todos os vetores linha $\mathbf{w1}_j(n)$ que constituem a matriz $\mathbf{W1}(n)$ e todos os vetores linha $\mathbf{w2}_k(n)$ que constituem a matriz $\mathbf{W2}(n)$ é intrinsicamente limitada em cada instante de tempo n devido ao uso de funções de ativação não-lineares do tipo sigmóide, ou seja,

$$\|\mathbf{w1}_j(n)\| = \sqrt{w1_{j1}^2(n),\ \ldots\ , w1_{ji}^2(n),\ \ldots\ , w1_{jm_0}^2(n)}\ \leq B_{w1},\ \forall n \tag{2.72}$$

e

$$\|\mathbf{w2}_k(n)\| = \sqrt{w2_{k1}^2(n),\ \ldots\ , w2_{kj}^2(n),\ \ldots\ , w2_{k(m_1+1)}^2(n)}\ \leq B_{w2},\ \forall n, \tag{2.73}$$

onde B_{w1} e B_{w2} são constantes conhecidas.

O sinal de saída $y1_j(n)$ do j-ésimo neurônio da camada escondida e o sinal de saída $y2_k(n)$ do k-ésimo neurônio da camada de saída da rede são definidos como:

$$y1_j(n) = \varphi\left(\sum_{i=1}^{m_0} w1_{ji}(n)x_i(n)\right) \tag{2.74}$$

e

$$y2_k(n) = \varphi\left(\sum_{j=1}^{m_1+1} w2_{kj}(n)y1_j(n)\right), \tag{2.75}$$

sendo

$$0 < A1_j(n) = \dot{\varphi}\left(\sum_{i=1}^{m_0} w1_{ji}(n)x_i(n)\right) \leq B_{A1}\ \forall i, j \tag{2.76}$$

e

$$0 < A2_k(n) = \dot{\varphi}\left(\sum_{j=1}^{m_1+1} w2_{kj}(n)y1_j(n)\right) \leq B_{A2}\ \forall j, k \tag{2.77}$$

as derivadas das funções de ativação $\varphi(\cdot)$ dos neurônios das camadas escondida e de saída, e B_{A1}, B_{A2} correspondem a seus valores máximos, respectivamente.

O vetor $\mathbf{d}(n)$ representa os valores desejados para a saída da rede, i.e.,

$$\|\mathbf{d}(n)\| = \sqrt{d_1^2(n),\ \ldots\ , d_k^2(n),\ \ldots\ , d_{m_2}^2(n)}\ \leq B_d,\ \forall n, \tag{2.78}$$

onde B_d é uma constante positiva.

Define-se o erro de aprendizado $e_k(n)$ do k-ésimo neurônio de saída como uma quantidade obtida a partir de

$$e_k(n) = d_k(n) - y2_k(n). \tag{2.79}$$

B. Primeira Proposta

As superfícies de deslizamento são definidas como função dos erros e respectivas derivadas das camadas de saída e escondida, sendo $C1,\ C2$ constantes positivas e T o período de amostragem:

- Camada de saída:

$$S2_{kj}(n) = C2X21_{kj}(n) + X22_{kj}(n) \tag{2.80}$$

onde,

$$X21_{kj}(n) = e_k(n), \tag{2.81}$$

$$X22_{kj}(n) = \frac{X21_{kj}(n) - X21_{kj}(n-1)}{T}. \tag{2.82}$$

- Camada escondida:

$$S1_{ji}(n) = C1X11_{ji}(n) + X12_{ji}(n) \tag{2.83}$$

onde,

$$X11_{ji}(n) = E(n) = \frac{1}{2}\sum_{k=1}^{m_2} e_k^2(n), \tag{2.84}$$

$$X12_{ji}(n) = \frac{X11_{ji}(n) - X11_{ji}(n-1)}{T}. \tag{2.85}$$

A convergência do estado da rede para as superfícies de deslizamento definidas em (2.80) e (2.83) pode ser analizada considerando o seguinte teorema.

Teorema 2.2 *Seja a superfície definida por* $S(n) = -CX(n)sign(S(n)) + Y(n)$*, tal que em regime de deslizamento* $S(n) \triangleq 0$*.* $S(n) : \Re^2 \rightarrow \Re,\ \{X(n), Y(n), C\} \in \Re$ *tal que* $X(n) > 0$*,* $C > 0$ *e* $sign(S(n)) = \begin{cases} +1 & S(n) \geq 0 \\ -1 & S(n) < 0 \end{cases}$*. Então, um ponto representativo do sistema, no espaço de estados bidimensional, irá convergir à superfície de deslizamento se:*

$$C < min\left\{\frac{|\ Y(n)\ |}{X(n)}, \frac{|\ Y(n-1)\ | - |\ Y(n)\ |}{X(n-1) - X(n)}\right\}. \tag{2.86}$$

$\diamond$

Prova: Ver (Parma, 2000) ou (Justino, 2004).

Assim, a partir do Teorema 2.2, as seguintes regras de atualização dos pesos são definidas:

- Camada de saída:

$$\Delta w2_{kj}(n) = \frac{\eta 2 \mid X21_{kj}(n) \mid sign(S2_{kj}(n))y1_j(n)}{\dot{\varphi}(\sum_{j=1}^{m_1+1} w2_{kj}(n)y1_j(n))}. \tag{2.87}$$

- Camada escondida:

$$\Delta w1_{ji}(n) = \frac{\eta 1 \mid X11_{ji}(n) \mid sign(S1_{ji}(n))x_i(n)}{\dot{\varphi}(\sum_{i=1}^{m_0} w1_{ji}(n)x_i(n)) \sum_{k=1}^{m_2} \left[e_k(n)\dot{\varphi}(\sum_{j=1}^{m_1+1} w2_{kj}(n)y1_j(n))w2_{kj}\right]}, \tag{2.88}$$

  sendo $\eta 2$ e $\eta 1$ as taxas de aprendizado do treinamento para as camadas de saída e escondida, respectivamente.

Para evitar instabilidade nas Equações (2.87) e (2.88), dois diferentes procedimentos podem ser usados: no primeiro, a atualização de pesos é interrompida se o denominador das equações é menor do que um valor escolhido e, num segundo caso, é adicionado um pequeno valor ao denominador das equações, evitando que eles se tornem zero.

B. Segunda Proposta

Utilizando as Equações (2.80) e (2.83), os termos $X21, X22, X11, X12$ das superfícies de deslizamento das camadas de saída e escondida são definidos da seguinte maneira:

- Camada de saída:

$$X21_{kj}(n) = \frac{\partial E(n)}{\partial w2_{kj}(n)}, \tag{2.89}$$

$$X22_{kj}(n) = \frac{X21_{kj}(n) - X21_{kj}(n-1)}{T}. \tag{2.90}$$

- Camada escondida:

$$X11_{ji}(n) = \frac{\partial E(n)}{\partial w1_{ji}(n)}, \tag{2.91}$$

$$X12_{ji}(n) = \frac{X11_{ji}(n) - X11_{ji}(n-1)}{T}. \tag{2.92}$$

Utilizando o Teorema 2.2, as seguintes regras de atualização dos pesos são definidas:

- Camada de saída:

$$\Delta w2_{kj}(n) = -\eta 2 \mid X21_{kj}(n) \mid sign(S2_{kj}(n)). \tag{2.93}$$

- Camada escondida:

$$\Delta w1_{ji}(n) = -\eta 1 \mid X11_{ji}(n) \mid sign(S1_{ji}(n)). \tag{2.94}$$

Como pode ser observado em (2.93) e (2.94), não existe nenhum termo no denominador, evitando-se, desta forma, os problemas numéricos encontrados na primeira proposta.

2.4 Conclusão

Este capítulo apresentou um breve histórico de redes neurais, enfocando o treinamento de redes MLP. Foram mostrados também alguns algoritmos selecionados com o objetivo de fornecer subsídios para o entendimento dos algoritmos que serão propostos no próximo capítulo.

Pode-se distinguir dois aspectos responsáveis pelas deficiências no conhecimento atual sobre o comportamento de uma rede MLP: primeiro, a presença de uma forma distribuída de não-linearidade e alta conectividade tornam difícil uma análise teórica da rede; segundo, a utilização de neurônios ocultos torna o processo de aprendizagem mais difícil de ser visualizado, pois este deve decidir quais características do padrão de entrada devem ser representadas pelo neurônios ocultos (Haykin, 2001).

Visando diminuir as deficiências citadas anteriormente, no próximo capítulo são apresentados dois algoritmos para treinamento em tempo real de redes MLP que determinam o ganho da rede de forma adaptativa, diferentemente do que ocorre nos algoritmos apresentados neste capítulo, onde é necessário o uso de métodos heurísticos na determinação do ganho (fixo) a ser utilizado para o treinamento da rede neural.

Capítulo 3

Algoritmos Propostos

Neste capítulo são apresentados os algoritmos propostos para treinamento em tempo real de redes MLP e que têm como principal característica a obtenção de um ganho adaptativo, determinado iterativamente, a cada passo de atualização dos pesos da rede. Estes algoritmos foram desenvolvidos em duas versões: na Seção 3.1 é apresentado o algoritmo com ganho adaptativo para redes com múltiplas saídas, enquanto que na Seção 3.2 é apresentada a versão para redes com uma saída escalar. A ordem de apresentação dos algoritmos preserva a sequência original de desenvolvimento, a despeito da aparente inversão na ordem de apresentação dos mesmos em relação a complexidade das redes consideradas.

Outro aspecto a ser comentado diz respeito à abordagem usada para desenvolver os algoritmos. Os algoritmos propostos foram desenvolvidos no domínio do tempo discreto.

Finalmente, a notação adotada para apresentação dos algoritmos será a mesma daquela apresentada na Seção 2.3. As considerações finais do capítulo são feitas na Seção 3.3.

3.1 Algoritmo com Ganho Adaptativo para Redes com Múltiplas Saídas

Nesta seção é apresentado o algoritmo com ganho adaptativo para treinamento em tempo real de redes MLP com múltiplas saídas que opera em modos quase-deslizantes. O termo "regime quase-deslizante" foi introduzido por (Miloslavjevic, 1985) para expressar o fato de que a extensão para o caso de tempo discreto das condições usuais de tempo contínuo para a existência de um regime de deslizamento, não necessariamente garante movimento de chaveamento (*chattering*) próximo da superfície de deslizamento nos mesmos moldes que se verifica em sistemas de tempo contínuo. Além disso, em (Sarpturk et al., 1987), foi demonstrado que a condição proposta por Miloslavjevic (1985) para a existência de um regime de quase-deslizamento poderia levar o sistema a se tornar instável. A seguir, é especificado como são entendidos nesta tese o regime quase-deslizante e a condição de alcance para a superfície de quase-deslizamento.

Definição 3.1 *Define-se um regime quase-deslizante em uma vizinhança ε de uma superfície de deslizamento $s(n) = 0$ a um movimento do sistema tal que*

$$|s(n)| \leq \varepsilon \tag{3.1}$$

onde a constante positiva ε é chamada de largura da banda de modos quase-deslizantes (Bartoszewicz, 1998a).

Esta definição difere daquela proposta por (Gao et al., 1995) pois não requer que os estados do sistema cruzem a superfície de deslizamento $s(n) = 0$ em cada passo sucessivo de controle.

A convergência do estado do sistema à superfície de quase-deslizamento pode ser analisada considerando-se a convergência da série

$$\sum_{n=1}^{\infty} s(n). \tag{3.2}$$

Uma vez garantida a convergência desta série, pode-se garantir que o estado do sistema irá convergir, ao menos assintoticamente, para a superfície de deslizamento $s(n) = 0$.

Considere-se o princípio de convergência de Cauchy (Kreyszig, 1993): A série $s_1 + s_2 + \cdots + s_n$ converge se e somente se, para um dado valor $\varepsilon \in \Re^+$, puder ser encontrado um valor N tal que $\mid s_{n+1} + s_{n+2} + \cdots + s_{n+p} \mid < \varepsilon$ para todo $n > N$ e $p = 1, 2, \cdots$. Uma série será absolutamente convergente se:

$$\sum_{n=1}^{\infty} |s(n)| \tag{3.3}$$

for convergente. Para o estudo da convergência da série dada por (3.3) é usado o teste da razão (Butkov, 1978). Assim, tem-se:

$$\left| \frac{s(n+1)}{s(n)} \right| \leq Q < 1. \tag{3.4}$$

Definição 3.2 *Diz-que que o estado do sistema converge para um regime quase-deslizante na vizinhança ε de uma superfície de deslizamento $s(n) = 0$ se a seguinte condição é satisfeita:*

$$|s(n+1)| < |s(n)|. \tag{3.5}$$

Nota: A partir da Definição 3.2, cruzar o plano $s(n) = 0$ é permitido, mas não requerido.

Teorema 3.1 *Seja $s(n) : \Re^2 \rightarrow \Re$, a superfície de deslizamento definida por $s(n) = CX1(n) + X2(n)$, onde $\{C, X1(n)\} \in \Re^+$ e $X2(n) \in \Re$. Se $X1(n) = E(n)$, sendo $E(n) = \frac{1}{2}\sum_{k=1}^{m_L} e_k^2(n)$ definido como o valor instantâneo da energia total do erro de todos os neurônios da camada de saída de uma rede MLP, onde $e_k(n) = d_k(n) - y_k(n)$ é o sinal de erro entre o valor desejado*

e o valor atual na saída do neurônio k de saída da rede na iteração n, m_L é o número de neurônios da camada de saída da rede, e $X2(n) = \frac{X1(n)-X1(n-1)}{T}$ é definido como a variação de $X1(n)$ em um período de amostragem T, então, para que o estado atual de $s(n)$ convirja para uma vizinhança ε de $s(n) = 0$, é necessário e suficiente que a rede satisfaça as seguintes condições:

$$sign(s(n))\,[C(X1(n+1) - X1(n)) + X2(n+1) - X2(n)] < 0 \tag{3.6}$$

$$sign(s(n))\,[C(X1(n+1) + X1(n)) + X2(n+1) + X2(n)] > 0, \tag{3.7}$$

sendo $sign(s(n)) = \begin{cases} +1, & s(n) \geq 0 \\ -1, & s(n) < 0 \end{cases}$ *a função sinal de* $s(n)$. ◇

Prova: Definindo-se o valor absoluto da superfície de deslizamento como segue

$$|s(n)| = sign(s(n))s(n), \tag{3.8}$$

a partir de (3.5) tem-se

$$|s(n+1)| < |s(n)| \Rightarrow sign(s(n+1))s(n+1) < sign(s(n))s(n).$$

Como $sign(s(n))sign(s(n)) = 1$, obtém-se

$$sign(s(n))[sign(s(n))sign(s(n+1))s(n+1) - s(n)] < 0.$$

Se $sign(s(n+1)) = sign(s(n))$, então $sign(s(n))[s(n+1) - s(n)] < 0$. Substituindo-se a definição de $s(n)$ como dada no Teorema 3.1 tem-se

$$sign(s(n))\,[CX1(n+1) + X2(n+1) - (CX1(n) + X2(n))] < 0 \Rightarrow (3.6).$$

Se $sign(s(n+1)) = -sign(s(n))$, então $sign(s(n))[-s(n+1) - s(n)] < 0$. Substituindo-se a definição de $s(n)$ como dada no Teorema 3.1 tem-se

$$sign(s(n))\,[CX1(n+1) + X2(n+1) + CX1(n) + X2(n)] > 0 \Rightarrow (3.7).$$

Para a prova de que as condições do Teorema 3.1 são suficientes, duas situações devem ser estabelecidas:

- A superfície de deslizamento não é atravessada durante a convergência. Nesta situação tem-se

  $$sign(s(n+1)) = sign(s(n)).$$

  Considerando $s(n) = CX1(n) + X2(n)$ e $s(n+1) = CX1(n+1) + X2(n+1)$, pode-se

escrever (3.6) como

$$sign(s(n))[s(n+1) - s(n)] < 0 \Rightarrow sign(s(n+1))s(n+1) < sign(s(n))s(n),$$

e usando (3.8) obtém-se $|s(n+1)| < |s(n)|$. A validade de (3.7) para esta situação é trivial, i.e.:

$$sign(s(n))[s(n+1) + s(n)] = |s(n+1)| + |s(n)| \Rightarrow (3.7).$$

- A superfície de deslizamento é atravessada durante a convergência. Assim, nesta situação tem-se

  $$sign(s(n+1)) = -sign(s(n)).$$

  Considerando, novamente, $s(n) = CX1(n)+X2(n)$ e $s(n+1) = CX1(n+1)+X2(n+1)$, pode-se escrever (3.7) como

  $$sign(s(n))[s(n+1) + s(n)] > 0 \Rightarrow sign(s(n+1))s(n+1) < sign(s(n))s(n),$$

  e usando (3.8) obtém-se $|s(n+1)| < |s(n)|$. A validade de (3.6) para esta situação é trivial, i.e.:

  $$sign(s(n))[s(n+1) - s(n)] = -|s(n+1)| - |s(n)| \Rightarrow (3.6).$$

□

A partir do Teorema 3.1, verifica-se que (3.6) é responsável pela existência de um regime quase-deslizante em torno de $s(n) = 0$, enquanto (3.7) garante a convergência das trajetórias do estado da rede para uma vizinhança da superfície de deslizamento $s(n) = 0$. Observa-se também, que o termo referente ao sinal da superfície de deslizamento $sign(s(n))$ determina os limites externos e internos do intervalo de convergência em relação às seguintes expressões:

$$C(X1(n+1) - X1(n)) + X2(n+1) - X2(n) \tag{3.9}$$

$$C(X1(n+1) + X1(n)) + X2(n+1) + X2(n). \tag{3.10}$$

Para o estudo da convergência da superfície de deslizamento $s(n) = CX1(n) + X2(n)$ é necessária a decomposição de (3.9) e (3.10) em relação a um ganho η, de modo a se obter um conjunto de equações para estas variáveis e, a partir das condições definidas pelo Teorema 3.1, determinar um intervalo em $\Re$ em função do ganho η, capaz de garantir a convergência do método proposto.

Teorema 3.2 *Seja $s(n) : \Re^2 \rightarrow \Re$, a superfície de deslizamento definida por $s(n) = CX1(n) + X2(n)$, onde $\{C, X1(n)\} \in \Re^+$ e $X2(n) \in \Re$. Se $X1(n)$, $X2(n)$ e T são definidos como no Teorema 3.1, então, para que o estado atual de $s(n)$ convirja para uma vizinhança ε de $s(n) = 0$, é necessário e suficiente que a rede satisfaça as seguintes condições:*

$$sign(s(n)) \left[c_1\eta^2 + c_2\eta - s(n) + CX1(n)\right] < 0 \tag{3.11}$$

$$sign(s(n)) \left[c_1\eta^2 + c_2\eta + s(n) + CX1(n)\right] > 0, \tag{3.12}$$

onde $\{c_1, c_2\} \in \Re$. Se as seguintes restrições são respeitadas:

$$c_1 > 0 \tag{3.13}$$

$$c_2 < 0 \tag{3.14}$$

$$\Delta = c_2^2 - 4c_1c_3 > 0, \tag{3.15}$$

sendo $c_3 = \begin{cases} -s(n) + CX1(n), & \textit{(na condição (3.11))} \\ s(n) + CX1(n), & \textit{(na condição (3.12))} \end{cases}$ então, a existência de um intervalo para o ganho η que satisfaça ambas as condições de convergência é garantida. $\diamond$

Prova: Considere, inicialmente, que:

$$X1(n) = \frac{1}{2}\sum_{k=1}^{m_L}(d_k(n) - y_k(n))^2 = \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2(n) - 2d_k(n)y_k(n) + y_k^2(n)), \tag{3.16}$$

$$X1(n+1) = \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2(n+1) - 2d_k(n+1)y_k(n+1) + y_k^2(n+1)), \tag{3.17}$$

$$X1(n-1) = \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2(n-1) - 2d_k(n-1)y_k(n-1) + y_k^2(n-1)), \tag{3.18}$$

$$X2(n+1) = \frac{X1(n+1) - X1(n)}{T}. \tag{3.19}$$

A partir de (3.16), (3.17), (3.18), (3.19) e da definição de $X2(n)$ dada no Teorema 3.1, pode-se expandir os termos de (3.9) considerando que $d_k(n-1) = d_k(n) = d_k(n+1) = d_k$. Assim, tem-se:

$$\begin{aligned} &C(X1(n+1) - X1(n)) + X2(n+1) - X2(n) = \\ &C(X1(n+1) - X1(n)) + \left(\frac{X1(n+1) - X1(n)}{T}\right) - \left(\frac{X1(n) - X1(n-1)}{T}\right) \\ &= \frac{1}{T}\left[(TC+1)X1(n+1) - (TC+2)X1(n) + X1(n-1)\right] \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{T}\left[(TC+1)\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_k y_k(n+1) + y_k^2(n+1))\right. \\
&\quad \left. -(TC+2)\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_k y_k(n) + y_k^2(n)) + \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_k y_k(n-1) + y_k^2(n-1))\right] \\
&= \frac{1}{T}\frac{1}{2}\sum_{k=1}^{m_L}\left[TC(-2d_k y_k(n+1) + y_k^2(n+1) + 2d_k y_k(n) - y_k^2(n)) - 2d_k y_k(n+1)\right. \\
&\qquad \left. +y_k^2(n+1) + 4d_k y_k(n) - 2y_k^2(n) - 2d_k y_k(n-1) + y_k^2(n-1)\right]. \qquad (3.20)
\end{aligned}
$$

Igualmente, pode-se expandir (3.10) usando as mesmas considerações feitas para expandir (3.9). Logo:

$$
\begin{aligned}
&C(X1(n+1) + X1(n)) + X2(n+1) + X2(n) = \\
&C(X1(n+1) + X1(n)) + \left(\frac{X1(n+1) - X1(n)}{T}\right) + \left(\frac{X1(n) - X1(n-1)}{T}\right) \\
&= \frac{1}{T}\left[(TC+1)X1(n+1) + TCX1(n) - X1(n-1)\right] \\
&= \frac{1}{T}\left[(TC+1)\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_k y_k(n+1) + y_k^2(n+1))\right. \\
&\quad \left. +TC\frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_k y_k(n) + y_k^2(n)) - \frac{1}{2}\sum_{k=1}^{m_L}(d_k^2 - 2d_k y_k(n-1) + y_k^2(n-1))\right] \\
&= \frac{1}{T}\frac{1}{2}\sum_{k=1}^{m_L}\left[TC(d_k^2 - 2d_k y_k(n+1) + y_k^2(n+1) + d_k^2 - 2d_k y_k(n) + y_k^2(n))\right. \\
&\qquad \left. -2d_k y_k(n+1) + y_k^2(n+1) - 2d_k y_k(n-1) - y_k^2(n-1)\right]. \qquad (3.21)
\end{aligned}
$$

A partir de (3.20) e (3.21) pode-se identificar o termo $y_k(n+1)$ como sendo a variável de interesse através da qual se deseja obter o ganho η. Então, fazendo

$$y_k(n+1) = y_k(n) + c\eta, \qquad (3.22)$$

$$y_k^2(n+1) = y_k^2(n) + 2y_k(n)c\eta + (c\eta)^2, \qquad (3.23)$$

substituindo (3.22) e (3.23) em (3.20) e (3.21) e, considerando $e_k(n) = d_k - y_k(n)$, obtém-se:

$$
\begin{aligned}
&\frac{1}{T}\frac{1}{2}\sum_{k=1}^{m_L}\left[(TC+1)c^2\eta^2 - 2(TC+1)ce_k(n)\eta\right. \\
&\qquad \left. +2d_k y_k(n) - y_k^2(n) - 2d_k y_k(n-1) + y_k^2(n-1)\right] \qquad (3.24)
\end{aligned}
$$

e

$$\frac{1}{T}\frac{1}{2}\sum_{k=1}^{m_L}\big[(TC+1)c^2\eta^2 - 2(TC+1)ce_k(n)\eta \\ +2TC(d_k - y_k(n))^2 - 2d_k y_k(n) + y_k^2(n) + 2d_k y_k(n-1) - y_k^2(n-1)\big] \quad (3.25)$$

Finalmente, levando em conta o resultado de $X1(n) - X1(n-1)$, obtém-se as condições (3.11) e (3.12) definidas no Teorema 3.2, com os respectivos coeficientes dados por:

$$\begin{aligned} c_1 &= \frac{1}{2}\left(C + \frac{1}{T}\right)c^2 \\ c_2 &= -\left(C + \frac{1}{T}\right)\sum_{k=1}^{m_L} ce_k(n) \\ c_3 &= \begin{cases} -s(n) + CX1(n), & \text{(na condição (3.11))} \\ s(n) + CX1(n), & \text{(na condição (3.12)).} \end{cases} \end{aligned} \quad (3.26)$$

Para analisar os intervalos de convergência limitados pelas condições (3.11) e (3.12) é necessário determinar os limites destes intervalos. Verifica-se, facilmente, que os intervalos de convergência são obtidos a partir de uma parábola, sendo a concavidade desta parábola determinada pelo valor de c_1 (neste caso, concavidade positiva, pois $c_1 > 0$).

A forma geral para a equação de segundo grau relacionada às condições de convergência pode ser escrita como:

$$c_1\eta^2 + c_2\eta + c_3 \quad (3.27)$$

onde c_3 é o termo independente. Considerando o valor de $\Delta = c_2^2 - 4c_1c_3$ e uma vez que $c_1 > 0$, a determinação das raízes de (3.27) é dada por:

$$\Delta = c_2^2 - 4|c_1|c_3. \quad (3.28)$$

Segundo (3.28), o valor de Δ está relacionado ao sinal e ao módulo da superfície de deslizamento $s(n)$. A partir disso, pode-se proceder a análise conforme segue:

- Se $s(n) > 0$:

  (a) $c_1\eta^2 + c_2\eta - s(n) + CX1(n) < 0$

  (1) $|s(n)| > CX1(n) \Rightarrow c_3 < 0$.
  Raízes: $\Delta = c_2^2 + 4|c_1||c_3| \Rightarrow \Delta > c_2^2$. Considerando $\Delta = c_2^2\xi_1^2$, sendo $\xi_1 > 1$, as raízes podem ser escritas na forma:

$$\eta = -\frac{c_2}{2c_1} \pm \left|\frac{c_2\xi_1}{2c_1}\right| \quad (3.29)$$

  (2) $|s(n)| < CX1(n) \Rightarrow c_3 > 0$

Raízes: $\Delta = c_2^2 - 4|c_1||c_3| \Rightarrow \Delta < c_2^2$. Existem duas variações possíveis para Δ:
$1^a)$ $0 < \Delta < c_2^2$: Considerando $\Delta = \frac{c_2^2}{\xi_1^2}$, as raízes podem ser escritas na forma:

$$\eta = -\frac{c_2}{2c_1} \pm \left| \frac{c_2}{2c_1\xi_1} \right| \tag{3.30}$$

$2^a)$ $\Delta \leq 0$: Esta condição não é considerada pois não atende a restrição (3.15).

(b) $c_1\eta^2 + c_2\eta + s(n) + CX1(n) > 0$

Raízes: $\Delta = c_2^2 - 4|c_1||c_3| \Rightarrow \Delta < c_2^2$. Existem duas variações possíveis para Δ:

$1^a)$ $0 < \Delta < c_2^2$: Considerando $\Delta = \frac{c_2^2}{\xi_2^2}$, sendo $\xi_2 > \xi_1$, as raízes podem ser escritas na forma:

$$\eta = -\frac{c_2}{2c_1} \pm \left| \frac{c_2}{2c_1\xi_2} \right| \tag{3.31}$$

$2^a)$ $\Delta \leq 0$: Esta condição não é considerada pois não atende a restrição (3.15).

A partir de (3.29), (3.30) e (3.31) pode-se estabelecer a seguinte relação:

$$\left| \frac{c_2}{2c_1\xi_2} \right| < \left| \frac{c_2}{2c_1\xi_1} \right| < \left| \frac{c_2\xi_1}{2c_1} \right|. \tag{3.32}$$

Considerando $(-\frac{c_2}{2c_1})$ como ponto central dos intervalos de convergência e observando (3.32), pode-se traçar um diagrama identificando, em negrito, os intervalos de convergência para $s(n) > 0$ conforme indicado na Figura 3.1.

- Se $s(n) < 0$:

(a) $c_1\eta^2 + c_2\eta - s(n) + CX1(n) > 0 \Rightarrow c_1\eta^2 + c_2\eta + s(n) + CX1(n) > 0$

Raízes: $\Delta = c_2^2 - 4|c_1||c_3| \Rightarrow \Delta < c_2^2$. Existem duas variações possíveis para Δ:

$1^a)$ $0 < \Delta < c_2^2$: Considerando $\Delta = \frac{c_2^2}{\xi_2^2}$, as raízes podem ser escritas na forma:

$$\eta = -\frac{c_2}{2c_1} \pm \left| \frac{c_2}{2c_1\xi_2} \right| \tag{3.33}$$

$2^a)$ $\Delta \leq 0$: Esta condição não é considerada pois não atende a restrição (3.15).

(b) $c_1\eta^2 + c_2\eta + s(n) + CX1(n) < 0 \Rightarrow c_1\eta^2 + c_2\eta - s(n) + CX1(n) < 0$

(1) $|s(n)| > CX1(n) \Rightarrow c_3 < 0$.

Raízes: $\Delta = c_2^2 + 4|c_1||c_3| \Rightarrow \Delta > c_2^2$. Considerando $\Delta = c_2^2\xi_1^2$, as raízes podem ser escritas na forma:

$$\eta = -\frac{c_2}{2c_1} \pm \left| \frac{c_2\xi_1}{2c_1} \right| \tag{3.34}$$

(2) $|s(n)| < CX1(n) \Rightarrow c_3 > 0$

Raízes: $\Delta = c_2^2 - 4|c_1||c_3| \Rightarrow \Delta < c_2^2$. Existem duas variações possíveis para Δ:

$1^a)$ $0 < \Delta < c_2^2$: Considerando $\Delta = \frac{c_2^2}{\xi_1^2}$, as raízes podem ser escritas na forma:

$$\eta = -\frac{c_2}{2c_1} \pm \left| \frac{c_2}{2c_1\xi_1} \right| \tag{3.35}$$

$2^a)$ $\Delta \leq 0$: Esta condição não é considerada pois não atende a restrição (3.15).

A partir de (3.33), (3.34) e (3.35), pode-se estabelecer a mesma relação definida em (3.32) e traçar o mesmo diagrama identificando, em negrito, os intervalos de convergência para $s(n) < 0$, conforme indicado na Figura 3.1. □

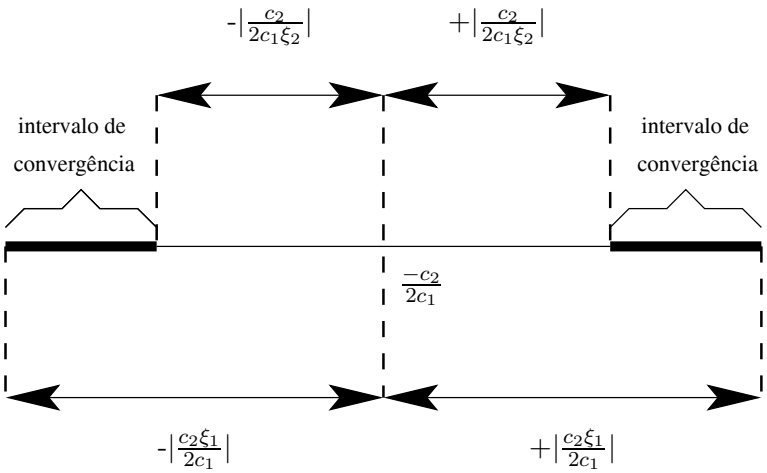


Figura 3.1: Intervalos de convergência para o algoritmo da primeira proposta.

Nota: O Teorema 3.2 garante a existência de intervalos reais para o ganho η que satisfazem as condições de convergência. Entretanto, o Teorema 3.2 não garante, *diretamente*, a existência de um *intervalo positivo* para o ganho η. Tanto para $s(n) > 0$ quanto para $s(n) < 0$, é garantida a existência de pelo menos uma raiz real positiva, o que reforça a existência de um intervalo positivo para η. Em (3.30), (3.31), (3.33) e (3.35), a existência de raízes reais positivas está condicionada a $-\frac{c_2}{2c_1} > 0$. Como $c_1 > 0$, a condição fica: $-c_2 > 0 \Rightarrow c_2 < 0$, a qual pode ser facilmente verificada a partir da aplicação da metodologia desenvolvida em um rede MLP de duas camadas.

Uma vez que $s(n)$ está relacionada com a topologia da rede utilizada, para se verificar a existência de um intervalo positivo para o ganho η, é necessário analisar o comportamento das condições de convergência para o perceptron linear, para o perceptron não-linear, e para a rede MLP de duas camadas com saída linear. A escolha desta topologia de rede MLP foi feita visando tornar mais simples, porém ainda efetivos, os cálculos envolvidos na determinação da resposta da rede a um estímulo.

3.1.1 Determinação de η para o Perceptron Linear

Seja a saída, no instante n, de um neurônio do tipo perceptron com função de ativação linear:

$$y(n) = \sum_{j=1}^{m_0} w_j(n)x_j(n), \tag{3.36}$$

onde m_0 é o número de entradas do neurônio. A análise para a determinação dos intervalos do ganho η é realizada para cada padrão de entrada do neurônio.

A saída do neurônio no instante $n+1$ é da forma:

$$y(n+1) = y(n) + \Delta y(n) = y(n) + \sum_{j=1}^{m_0} \Delta w_j(n)x_j(n). \tag{3.37}$$

Para que (3.37) possa ser calculada, é necessário a determinação de $\Delta w_j(n)$, que representa o ajuste dos pesos do perceptron no instante n. Uma expressão imediata pode ser obtida a partir de (2.43), conhecida como regra Delta, e que dá origem ao algoritmo LMS ou algoritmo de aprendizado do gradiente descendente. Assim, tem-se:

$$\Delta w_j(n) = -\eta \frac{\partial E(n)}{\partial w_j(n)} = -\eta\ 2\ \frac{1}{2}\ (d(n) - y(n))(-1)\frac{\partial y(n)}{\partial w_j(n)} = \eta e(n)x_j(n). \tag{3.38}$$

Uma vez definido $\Delta w_j(n)$, pode-se então calcular $y(n+1)$ como segue:

$$y(n+1) = y(n) + e(n)\sum_{j=1}^{m_0} x_j^2(n)\eta = y(n) + c\eta. \tag{3.39}$$

Portanto, usando (3.39) e considerando $c = e(n)\sum_{j=1}^{m_0} x_j^2(n)$, obtém-se as expressões para os coeficientes $c_1,\ c_2$ *e* c_3 de (3.26):

$$\begin{aligned}
c_1 &= \frac{1}{2}\left(C + \frac{1}{T}\right)c^2 = \frac{1}{2}\left(C + \frac{1}{T}\right)e^2(n)\left(\sum_{j=1}^{m_0} x_j^2(n)\right)^2 \\
c_2 &= -\left(C + \frac{1}{T}\right)ce(n) = -\left(C + \frac{1}{T}\right)e^2(n)\sum_{j=1}^{m_0} x_j^2(n) \\
c_3 &= \begin{cases} -s(n) + CX1(n), & \text{(na condição (3.11))} \\ s(n) + CX1(n), & \text{(na condição (3.12)).} \end{cases}
\end{aligned} \tag{3.40}$$

A partir da determinação dos coeficientes $c_1,\ c_2$ *e* c_3, pode-se aplicar o Teorema 3.2 para a determinação dos intervalos de convergência para o ganho η.

3.1.2 Determinação de η para o Perceptron Não-Linear

A saída característica deste tipo de neurônio é dada por:

$$y(n) = \varphi\left(\sum_{j=1}^{m_0} w_j(n)x_j(n)\right), \tag{3.41}$$

onde $\varphi(\cdot)$ corresponde a função de ativação do neurônio, contínua e diferenciável.

A abordagem adotada para a determinação da saída do neurônio consiste numa aproximação da função de ativação através de sua decomposição em série de Taylor, ao invés da propagação do sinal de saída do neurônio para a saída linear do mesmo utilizando a inversa da função de ativação. A abordagem foi escolhida pelo fato de que o uso dos primeiros termos da série de Taylor proporciona uma significativa simplificação e redução do custo matemático para a definição dos intervalos de convergência, apesar de limitar a capacidade de aproximação da função para regiões próximas ao ponto de interesse. Maiores detalhes a respeito da série de Taylor podem ser vistos no Apêndice B.

Seja a saída, no instante n, de um neurônio do tipo perceptron com função de ativação não-linear dada por (3.41). A saída no instante $n+1$ pode ser escrita como:

$$y(n+1) = y(n) + \Delta y(n) = y(n) + \varphi\left(\sum_{j=1}^{m_0} \Delta w_j(n)x_j(n)\right). \tag{3.42}$$

Aplicando-se em (3.42) a decomposição de primeira ordem da série de Taylor, obtém-se:

$$y(n+1) = y(n) + \dot{y}(n)\sum_{j=1}^{m_0} \Delta w_j(n)x_j(n), \tag{3.43}$$

onde $\left|\sum_{j=1}^{m_0} \Delta w_j(n)x_j(n)\right| \leq \xi$. Usando-se (3.38) para a variação dos pesos no instante n, é possível definir um intervalo para o ganho η em relação à expansão em série de Taylor:

$$\eta \leq \frac{\xi}{\left|e(n)\sum_{j=1}^{m_0} x_j^2(n)\right|}. \tag{3.44}$$

Pode-se verificar que (3.44) restringe o intervalo do ganho η de acordo com a precisão desejada (ξ) para a aproximação da função de ativação do neurônio. Decompondo-se (3.43) tem-se:

$$y(n+1) = y(n) + \dot{y}(n)e(n)\sum_{j=1}^{m_0} x_j^2(n)\eta = y(n) + c\eta. \tag{3.45}$$

Portanto, usando (3.45) e considerando $c = \dot{y}(n)e(n)\sum_{j=1}^{m_0} x_j^2(n)$, obtém-se as expressões

para os coeficientes $c_1,\ c_2\ e\ c_3$ de (3.26):

$$\begin{aligned} c_1 &= \frac{1}{2}\left(C+\frac{1}{T}\right)c^2 = \frac{1}{2}\left(C+\frac{1}{T}\right)\dot{y}^2(n)e^2(n)\left(\sum_{j=1}^{m_0}x_j^2(n)\right)^2 \\ c_2 &= -\left(C+\frac{1}{T}\right)ce(n) = -\left(C+\frac{1}{T}\right)\dot{y}(n)e^2(n)\sum_{j=1}^{m_0}x_j^2(n) \\ c_3 &= \begin{cases} -s(n)+CX1(n), & \text{(na condição (3.11))} \\ s(n)+CX1(n), & \text{(na condição (3.12)).} \end{cases} \end{aligned} \tag{3.46}$$

A partir da determinação dos coeficientes $c_1,\ c_2\ e\ c_3$, observando o limite imposto pela decomposição em série de Taylor, pode-se aplicar o Teorema 3.2 para a determinação dos intervalos de convergência para o ganho η.

3.1.3 Determinação de η para uma Rede MLP de Duas Camadas

Seja a saída linear do k-ésimo neurônio de uma rede MLP de duas camadas em relação a um vetor de entrada $\mathbf{x}(n)$:

$$y2_k(n) = \sum_{j=1}^{m_1+1} w2_{kj}(n)y1_j(n) = \sum_{j=1}^{m_1+1} w2_{kj}(n)\varphi\left(\sum_{i=1}^{m_0} w1_{ji}(n)x_i(n)\right).$$

Devido a existência de duas camadas, deve-se fazer o estudo do intervalo de convergência para a camada de saída e escondida, separadamente. Assim, tem-se:

- Camada de saída: Considerando somente os pesos da camada de saída como sendo os parâmetros de interesse, a saída k no instante n de uma rede MLP com saída linear é dada por:

$$y2_k(n) = \sum_{j=1}^{m_1+1} w2_{kj}(n)y1_j(n). \tag{3.47}$$

  Supondo que o ajuste dos pesos seja realizado, inicialmente, somente nos pesos da camada de saída, (3.47) pode ser comparada à (3.36) para o perceptron linear. Neste caso, as entradas do neurônio k correspondem ao vetor de saída dos neurônios da camada escondida (acrescidos do termo de bias) após a função de ativação, $\mathbf{y1}(n)$, e os pesos, ao vetor $\mathbf{w2}_k(n)$. Os coeficientes $c_1,\ c_2\ e\ c_3$ são obtidos a partir do uso das equações relativas ao neurônio linear *aplicando-se a análise para a rede com múltiplas saídas*. Assim, os coeficientes da equação de segundo grau associada às condições de convergência são definidos como:

$$c_1 = \frac{1}{2}\left(C+\frac{1}{T}\right)\sum_{k=1}^{m_2}\left[e_k^2(n)\left(\sum_{j=1}^{m_1+1} y1_j^2(n)\right)^2\right]$$

$$
\begin{aligned}
c_2 &= -\left(C+\frac{1}{T}\right)\sum_{k=1}^{m_2}\left(e_k^2(n)\sum_{j=1}^{m_1+1} y1_j^2(n)\right) \\
c_3 &= \begin{cases} -s(n)+CX1(n), & \text{(na condição (3.11))} \\ s(n)+CX1(n), & \text{(na condição (3.12)).} \end{cases}
\end{aligned} \tag{3.48}
$$

- Camada escondida: Considera-se agora o ajuste dos pesos da camada escondida, $\mathbf{W1}(n)$. Para isso, os pesos da camada de saída serão mantidos constantes. Logo, o k-ésimo neurônio da rede MLP de duas camadas com saída linear é dado por:

$$
y2_k(n) = \sum_{j=1}^{m_1+1} w2_{kj}(n)\varphi\left(\sum_{i=1}^{m_0} w1_{ji}(n)x_i(n)\right). \tag{3.49}
$$

A saída no instante $n+1$ é dada por:

$$
y2_k(n+1) = y2_k(n)+\Delta y2_k(n) = y2_k(n)+\sum_{j=1}^{m_1+1} w2_{kj}(n)\varphi\left(\sum_{i=1}^{m_0}\Delta w1_{ji}(n)x_i(n)\right). \tag{3.50}
$$

Aplicando-se em (3.50) a decomposição de primeira ordem da série de Taylor, obtém-se:

$$
y2_k(n+1) = y2_k(n) + \dot{y2}_k(n)\sum_{j=1}^{m_1+1} w2_{kj}(n)\sum_{i=1}^{m_0}\Delta w1_{ji}(n)x_i(n), \tag{3.51}
$$

onde $\left|\sum_{i=1}^{m_0}\Delta w1_{ji}(n)x_i(n)\right| \leq \xi$. Pode-se usar (3.38) para a variação dos pesos no instante n. Porém, para a camada escondida, não existe uma resposta desejada especificada para os neurônios desta camada. Consequentemente, um sinal de erro para um neurônio oculto deve ser determinado recursivamente, em termos dos sinais de erro de todos os neurônios aos quais o neurônio oculto está diretamente conectado, ou seja, $\Delta w1_{ji}(n) = \eta\sum_{k=1}^{m_2} e_k(n)w2_{kj}(n)x_i(n)$. A partir da expressão de $\Delta w1_{ji}(n)$ é possível definir um intervalo para o ganho η em relação à expansão em série de Taylor:

$$
\eta \leq \frac{\xi}{\left|\sum_{k=1}^{m_2} e_k(n)w2_{kj}(n)\sum_{i=1}^{m_0} x_i^2(n)\right|}. \tag{3.52}
$$

Apesar de (3.52) ser atribuída a um único neurônio, o limite para o ganho η deve ser definido em função de toda a rede, escolhendo-se o menor limite associado a um dos neurônios da rede. Decompondo-se (3.51) tem-se:

$$
y2_k(n+1) = y2_k(n) + \dot{y2}_k(n)\sum_{j=1}^{m_1+1}\sum_{k=1}^{m_2} e_k(n)w2_{kj}^2(n)\sum_{i=1}^{m_0} x_i^2(n)\eta, \tag{3.53}
$$

Portanto, usando (3.53) e considerando $c = \dot{y2}_k(n) \sum_{j=1}^{m_1+1} \sum_{k=1}^{m_2} e_k(n) w2_{kj}^2(n) \sum_{i=1}^{m_0} x_i^2(n)$, obtém-se os coeficientes c_1, c_2 e c_3 como segue:

$$\begin{aligned}
c_1 &= \frac{1}{2}\left(C + \frac{1}{T}\right) \sum_{k=1}^{m_2} \left[\dot{y2}_k^2(n) \sum_{j=1}^{m_1+1} \sum_{k=1}^{m_2} e_k^2(n) \left(w2_{kj}^2(n)\right)^2 \left(\sum_{i=1}^{m_0} x_i^2(n) \right)^2 \right] \\
c_2 &= -\left(C + \frac{1}{T}\right) \sum_{k=1}^{m_2} \left(\dot{y2}_k^2(n) \sum_{j=1}^{m_1+1} \sum_{k=1}^{m_2} e_k^2(n) w2_{kj}^2(n) \sum_{i=1}^{m_0} x_i^2(n) \right) \\
c_3 &= \begin{cases} -s(n) + CX1(n), & \text{(na condição (3.11))} \\ s(n) + CX1(n), & \text{(na condição (3.12)).} \end{cases}
\end{aligned} \tag{3.54}$$

Assim, a partir dos coeficientes obtidos em (3.48) e (3.54), pode-se aplicar o Teorema 3.2, sendo o intervalo final para o ganho η determinado pela intersecção dos intervalos definidos pelas equações de convergência obtidas para a camada escondida e de saída, observando o limite imposto pela decomposição em série de Taylor. Deve-se salientar também que, em (3.48) e (3.54), os coeficientes c_1, c_2 e c_3 são dependentes de C e T. Isto implica que, na determinação de C, o período de amostragem deve ser levado em conta.

3.2 Algoritmo com Ganho Adaptativo para Redes com uma Saída Escalar

Nesta seção é apresentado o algoritmo com ganho adaptativo para treinamento em tempo real de redes MLP com uma saída escalar que opera em modos quase-deslizantes. Para este algoritmo são adotadas as Definições 3.1 e 3.2.

Duas diferenças devem ser notadas entre o algoritmo desta seção e aquele apresentado na Seção 3.1. A primeira está relacionada à definição da superfície de deslizamento: neste caso, é usado o valor instantâneo da energia total do erro do único neurônio da camada de saída de uma rede MLP. A outra diferença está relacionada com o modo como os pesos da rede são atualizados. O algoritmo apresentado na Seção 3.1 atualiza os pesos da rede usando o gradiente da função erro em relação aos pesos (algoritmo BP), conforme mostrado em (3.38). Esta lei de correção de pesos, apesar de ser bastante usada para treinamento de redes MLP, apresenta algumas deficiências. Conforme mostrado em (Yu et al., 2002), somente a estabilidade (não a estabilidade assintótica) para um conjunto de pesos que corresponde ao mínimo global do algoritmo BP, de acordo com a teoria de estabilidade de Lyapunov, pode ser garantida. Além disso, o processo de aprendizado em tempo real usando o algoritmo BP apresenta oscilações (Zhao, 1996).

Visando melhorar as deficiências citadas anteriormente, propõe-se o uso da abordagem apresentada por (Topalov et al., 2003) para a correção dos pesos da rede. O uso desta abordagem implica na limitação para apenas uma saída escalar para rede MLP de duas camadas.

Segundo Topalov et al. (2003), a limitação de apenas uma saída escalar para a rede MLP não deve ser considerada tão restritiva em relação a aplicabilidade da proposta, uma vez que é possível se ter duas ou mais estruturas de redes MLP compartilhando as mesmas entradas.

Teorema 3.3 *Seja* $s(n) : \Re \rightarrow \Re$*, a superfície de deslizamento definida por* $s(n) = E(n)$*, onde* $E(n) \in \Re^+$*. Se* $E(n) = \frac{1}{2}e^2(n)$ *é definido como o valor instantâneo da energia do erro do neurônio da camada de saída de uma rede MLP, onde* $e(n) = d(n) - y(n)$ *é o sinal de erro entre o valor desejado e o valor atual na saída do neurônio de saída da rede na iteração* n*, então, para que o estado atual de* $s(n)$ *convirja para uma vizinhança* ε *de* $s(n) = 0$*, é necessário e suficiente que a rede satisfaça a seguinte condição:*

$$sign(s(n))[E(n+1) - E(n)] < 0 \tag{3.55}$$

sendo $sign(s(n)) = \begin{cases} +1, & s(n) \geq 0 \\ -1, & s(n) < 0 \end{cases}$ *a função sinal de* $s(n)$. $\diamond$

Prova: A partir da definição do valor absoluto da superfície de deslizamento como dado em (3.8), tem-se

$$|s(n+1)| < |s(n)| \Rightarrow sign(s(n+1))s(n+1) < sign(s(n))s(n).$$

Como $sign(s(n))sign(s(n)) = 1$, obtém-se

$$sign(s(n))[sign(s(n))sign(s(n+1))s(n+1) - s(n)] < 0.$$

Como a superfície de deslizamento não é atravessada durante a convergência em consequência da definição usada para $E(n)$, tem-se $sign(s(n+1)) = sign(s(n))$, então

$$sign(s(n)[s(n+1) - s(n)] < 0.$$

Substituindo-se a definição de $s(n)$ como dada no Teorema 3.3 e considerando $s(n+1) = E(n+1)$, tem-se

$$sign(s(n))[E(n+1) - E(n)] < 0.$$

Para a prova de que a condição do Teorema 3.3 é suficiente, deve-se considerar a definição usada para $E(n)$, ou seja, a superfície de deslizamento não é atravessada durante a convergência. Nesta situação tem-se

$$sign(s(n+1)) = sign(s(n)).$$

Considerando que $s(n) = E(n)$ e $s(n+1) = E(n+1)$, pode-se escrever (3.55) como

$$sign(s(n))[s(n+1) - s(n)] < 0 \Rightarrow sign(s(n+1))s(n+1) < sign(s(n))s(n).$$

Usando o valor absoluto da superfície de deslizamento como definido em (3.8) obtém-se

$$|s(n+1)| < |s(n)|.$$

□

A partir do Teorema 3.3, verifica-se que (3.55) é responsável pela convergência e existência de um regime quase-deslizante em torno de $s(n) = 0$. Uma vez que $sign(s(n)) > 0$, para o estudo da convergência da superfície de deslizamento $s(n) = E(n)$ é necessária a decomposição de $E(n+1) - E(n)$ em relação a um ganho η, de modo a se obter um conjunto de equações para estas variáveis e, a partir da condição definida pelo Teorema 3.3, determinar um intervalo em $\Re$ em função do ganho η, capaz de garantir a convergência do método proposto.

Teorema 3.4 *Seja $s(n) : \Re \rightarrow \Re$, a superfície de deslizamento definida por $s(n) = E(n)$, onde $E(n) \in \Re^+$ e é definido como no Teorema 3.3, então, para que o estado atual de $s(n)$ convirja para uma vizinhança ε de $s(n) = 0$, é necessário e suficiente que a rede satisfaça a seguinte condição:*

$$sign(s(n))[c_1\eta^2 + c_2\eta] < 0. \tag{3.56}$$

Se as restrições $c_1 > 0$ e $c_2 < 0$ são respeitadas, então, a existência de um intervalo para o ganho η que satisfaça a condição de convergência é garantida. ◇

Prova: Para que o Teorema 3.4 possa ser aplicado, é necessário que os coeficientes c_1 e c_2 da equação de segundo grau relacionada à condição de convergência, respeitadas as restrições impostas, sejam determinados. Para tanto, $E(n)$ deve ser decomposto considerando o valor desejado e o valor atual de saída do neurônio da camada de saída da rede. Assim, tem-se:

$$E(n) = \frac{1}{2}e^2(n) = \frac{1}{2}(d(n) - y(n))^2 = \frac{1}{2}\left(d^2(n) - 2d(n)y(n) + y^2(n)\right). \tag{3.57}$$

A expressão de $E(n)$ no próximo passo será:

$$E(n+1) = \frac{1}{2}\left(d^2(n+1) - 2d(n+1)y(n+1) + y^2(n+1)\right). \tag{3.58}$$

A partir de (3.57) e (3.58), é possível expandir os termos definidos em $E(n+1) - E(n)$ se a saída desejada no próximo passo for considerada igual à saída atual, i.e., $d(n+1) = d(n) = d$. Portanto:

$$E(n+1) - E(n) = \frac{1}{2}\left[2d(y(n) - y(n+1)) + y^2(n+1) - y^2(n)\right]. \tag{3.59}$$

Observando (3.59), pode-se identificar o termo $y(n+1)$ como sendo a variável de interesse

através da qual se deseja obter o ganho η. Então, fazendo

$$y(n+1) = y(n) + c\eta \tag{3.60}$$

$$y^2(n+1) = y^2(n) + 2y(n)c\eta + (c\eta)^2, \tag{3.61}$$

substituindo (3.60) e (3.61) em (3.59), e considerando $e(n) = d - y(n)$, obtém-se

$$E(n+1) - E(n) = \frac{1}{2}\left[c^2\eta^2 - 2ce(n)\eta\right]. \tag{3.62}$$

Finalmente, considerando (3.62) e (3.55), pode-se escrever (3.56) com os respectivos coeficientes dados por:

$$\begin{aligned} c_1 &= \frac{1}{2}c^2 \\ c_2 &= -ce(n) \\ c_3 &= 0. \end{aligned} \tag{3.63}$$

Para analisar o intervalo de convergência limitado pela condição (3.56) é necessário determinar os limites deste intervalo. Verifica-se, facilmente, que o intervalo de convergência é obtido a partir de uma parábola, sendo a concavidade desta parábola determinada pelo valor de c_1 (neste caso, concavidade positiva, pois $c_1 > 0$).

A forma geral para a equação de segundo grau relacionada à condição de convergência é dada por (3.27). Considerando o valor de $\Delta = c_2^2 - 4c_1c_3$, sendo $c_1 > 0$, $c_3 = 0$, e uma vez que $s(n) > 0$, a determinação das raízes de $c_1\eta^2 + c_2\eta < 0$ é obtida como segue:

$$\Delta = c_2^2 \Rightarrow \eta = \frac{-c_2 \pm c_2}{2c_1} = -\frac{c_2}{2c_1} \pm \left|\frac{c_2}{2c_1}\right| \tag{3.64}$$

Considerando $(-\frac{c_2}{2c_1})$ como ponto central do intervalo de convergência, pode-se traçar um diagrama identificando, em negrito, o intervalo de convergência conforme indicado na Figura 3.2. □

Nota: O Teorema 3.4 garante a existência de um intervalo positivo para o ganho η que satisfaz a condição de convergência. A existência de raízes reais positivas está condicionada à existência da condição $-\frac{c_2}{2c_1} > 0$. Como $c_1 > 0$, a condição fica: $-c_2 > 0 \Rightarrow c_2 < 0$, a qual pode ser facilmente verificada a partir da aplicação da metodologia desenvolvida em uma rede MLP de duas camadas.

3.2.1 Determinação da Expressão para a Correção dos Pesos da Rede

Conforme mencionado no início da Seção 3.2, será usada a abordagem apresentada por (Topalov et al., 2003) para a correção dos pesos da rede. Porém, como o algoritmo proposto é

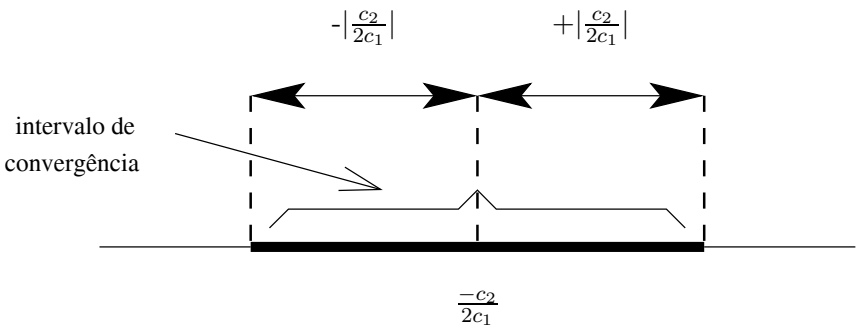


Figura 3.2: Intervalo de convergência para o algoritmo da segunda proposta.

desenvolvido considerando o tempo discreto, diferente do que acontece nos sistemas de tempo contínuo, a variação de pesos é feita a cada instante de amostragem nT, onde T é o período de amostragem. Portanto, em sistemas de tempo discreto, os pesos se mantém constantes entre os intervalos de amostragem, i.e., $nT \leq t < (n+1)T$, permitindo assim, o uso de expressões desenvolvidas em tempo contínuo para a abordagem discreta. Ainda, a Definição 2.1 e o Teorema 2.1 serão reescritos tendo em vista a definição de erro usada nesta tese.

Definição 3.3 *Um movimento de deslizamento irá ocorrer em uma superfície de deslizamento* $s(e(t)) = e(t) = d(t) - y(t) = 0$*, após o tempo de alcance* t_h*, se a condição* $s(t)\dot{s}(t) = e(t)\dot{e}(t) < 0$ *é verdadeira para todo* t *em um subintervalo não-trivial semi-aberto de tempo da forma* $[t, t_h) \subset (-\infty, t_h)$*.*

O algoritmo de aprendizado para os pesos da rede neural $\mathbf{W}1(t)$ e $\mathbf{w}2(t)$ deve ser obtido de tal forma que a condição de modos deslizantes da Definição 3.3 seja imposta. Denotando como $sign(e(t))$ a função sinal do erro como definida em (2.66), para possibilitar que $s = 0$ seja alcançado, usa-se o seguinte teorema:

Teorema 3.5 *Se o algoritmo de aprendizado para os pesos* $\mathbf{W}1(t)$ *e* $\mathbf{w}2(t)$ *é escolhido, respectivamente, como*

$$\dot{w}1_{ji}(t) = \left(\frac{w2_j(t)x_i(t)}{\mathbf{x}^T(t)\mathbf{x}(t)}\right)\eta\ sign(e(t)) \tag{3.65}$$

$$\dot{w}2_j(t) = \left(\frac{y1_j(t)}{\mathbf{y}1^T(t)\mathbf{y}1(t)}\right)\eta\ sign(e(t)) \tag{3.66}$$

com η *sendo uma constante positiva que satisfaça a seguinte inequação*

$$\eta > B_d + m_1 B_A B_{w1} B_{\dot{x}} B_{w2} \tag{3.67}$$

então, para qualquer condição inicial arbitrária $e(0)$*, o erro de aprendizado* $e(t)$ *irá convergir*

para zero durante um tempo finito t_h que pode ser estimado como

$$t_h \leq \frac{|e(0)|}{\eta - B_{\dot{d}} - m_1 B_A B_{w1} B_{\dot{x}} B_{w2}} \tag{3.68}$$

e um movimento de deslizamento irá ser mantido em $e = 0$ para todo $t > t_h$. $\diamond$

Prova: Considere a seguinte função candidata de Lyapunov [1]:

$$V = \frac{1}{2}\, e^2 \tag{3.69}$$

Então, usando a hipótese (3.67), diferenciando V obtém-se

$$\begin{aligned}
\dot{V} &= e(\dot{d} - \dot{y}) = \\
& e\left\{ \dot{d} - \left[\sum_{j=1}^{m_1+1} w2_j\, \varphi\left(\sum_{i=1}^{m_0} w1_{ji} x_i \right) \right]' \right\} = \\
& e\left\{ \dot{d} - \left[\sum_{j=1}^{m_1+1} \dot{w}2_j\, \varphi\left(\sum_{i=1}^{m_0} w1_{ji} x_i \right) + \right.\right. \\
& \qquad \left.\left. \sum_{j=1}^{m_1+1} w2_j\, \dot{\varphi}\left(\sum_{i=1}^{m_0} w1_{ji} x_i \right) \sum_{i=1}^{m_0} (\dot{w}1_{ji} x_i + w1_{ji} \dot{x}_i) \right] \right\} = \\
& e\left\{ \dot{d} - \left[\sum_{j=1}^{m_1+1} \dot{w}2_j y1_j + \sum_{j=1}^{m_1+1} w2_j A_j \sum_{i=1}^{m_0} (\dot{w}1_{ji} x_i + w1_{ji} \dot{x}_i) \right] \right\} = \\
& e\left\{ \dot{d} - \left[\sum_{j=1}^{m_1+1} \left(\frac{y1_j}{\mathbf{y1}^T \mathbf{y1}} \right) \eta sign(e) y1_j + \right.\right. \\
& \qquad \left.\left. \sum_{j=1}^{m_1+1} A_j \sum_{i=1}^{m_0} \left(\left(\frac{w2_j x_i}{\mathbf{x}^T \mathbf{x}} \right) \eta sign(e) x_i w2_j + w1_{ji} \dot{x}_i w2_j \right) \right] \right\} = \\
& e\left(\dot{d} - \eta sign(e) - \sum_{j=1}^{m_1+1} A_j \eta w2_j^2 sign(e) - \sum_{j=1}^{m_1+1} A_j w2_j \sum_{i=1}^{m_0} w1_{ji} \dot{x}_i \right) = \\
& e\dot{d} - \eta \mid e \mid - \eta \mid e \mid \sum_{j=1}^{m_1+1} A_j w2_j^2 - e \sum_{j=1}^{m_1+1} A_j w2_j \sum_{i=1}^{m_0} w1_{ji} \dot{x}_i = \\
& - \left(\eta + \eta \sum_{j=1}^{m_1+1} A_j w2_j^2 \right) \mid e \mid + \left(\dot{d} - \sum_{j=1}^{m_1+1} A_j w2_j \sum_{i=1}^{m_0} w1_{ji} \dot{x}_i \right) e \leq \\
& - \eta \mid e \mid + \left(\dot{d} - \sum_{j=1}^{m_1+1} A_j w2_j \sum_{i=1}^{m_0} w1_{ji} \dot{x}_i \right) e \leq
\end{aligned}$$

[1] Com o objetivo de tornar mais legível a prova do Teorema 3.5, a dependência temporal das variáveis foi omitida.

$$
\begin{aligned}
&-\eta \mid e \mid + \left(B_{\dot{d}} + m_1 B_A B_{w2} B_{w1} B_{\dot{x}}\right) \mid e \mid = \\
&\mid e \mid \left(-\eta + B_{\dot{d}} + m_1 B_A B_{w2} B_{w1} B_{\dot{x}}\right) < 0 \;\; \forall e \neq 0
\end{aligned} \tag{3.70}
$$

A desigualdade (3.70) significa que as trajetórias controladas do erro de aprendizagem $e(t)$ convergem para $s = 0$ de uma maneira estável. Pode-se mostrar que esta convergência se dá em um tempo finito. A equação diferencial que é satisfeita por meio do erro controlado $e(t)$ é a seguinte:

$$
\begin{aligned}
\dot{e} &= \dot{d} - \eta sign(e) - \left(\sum_{j=1}^{m_1+1} A_j w2_j^2\right) \eta sign(e) - \sum_{j=1}^{m_1+1} A_j w2_j \sum_{i=1}^{m_0} w1_{ji}\dot{x}_i \\
&= \dot{d} - \left(1 + \sum_{j=1}^{m_1+1} A_j w2_j^2\right) \eta sign(e) - \sum_{j=1}^{m_1+1} A_j w2_j \sum_{i=1}^{m_0} w1_{ji}\dot{x}_i.
\end{aligned} \tag{3.71}
$$

Para qualquer $t \leq t_h$, a solução $e(t)$ para esta equação, com $e(0)$ em $t = 0$, satisfaz

$$
\begin{aligned}
e(t) - e(0) &= \int_0^t \dot{e}(\tau) d\tau \\
&= \int_0^t \left[\dot{d}(\tau) - \left(1 + \sum_{j=1}^{m_1+1} A_j(\tau) w2_j^2(\tau)\right) \eta sign(e(\tau)) - \right. \\
&\qquad \left. \sum_{j=1}^{m_1+1} A_j(\tau) w2_j(\tau) \sum_{i=1}^{m_0} w1_{ji}(\tau)\dot{x}_i(\tau)\right] d(\tau).
\end{aligned} \tag{3.72}
$$

Em $t = t_h$, a solução tem valor zero e, portanto,

$$
\begin{aligned}
-e(0) &= \int_0^{t_h} \left[\dot{d}(t) - \left(1 + \sum_{j=1}^{m_1+1} A_j(t) w2_j^2(t)\right) \eta sign(e(0)) - \right. \\
&\qquad \left. \sum_{j=1}^{m_1+1} A_j(t) w2_j(t) \sum_{i=1}^{m_0} w1_{ji}(t)\dot{x}_i(t)\right] d(t) = \\
&\quad -\eta sign(e(0)) \left[t_h + \int_0^{t_h} \left(\sum_{j=1}^{m_1+1} A_j(t) w2_j^2(t)\right) dt\right] + \\
&\quad \int_0^{t_h} \left(\dot{d}(t) - \sum_{j=1}^{m_1+1} A_j(t) w2_j(t) \sum_{i=1}^{m_0} w1_{ji}(t)\dot{x}_i(t)\right) d(t).
\end{aligned} \tag{3.73}
$$

Multiplicando ambos os lados de (3.73) por $-sign(e(0))$, a estimativa de t_h em (3.68) pode ser encontrada usando a seguinte desigualdade

$$\begin{aligned} \mid e(0) \mid &= \eta t_h + \eta \int_0^{t_h} \left(\sum_{j=1}^{m_1+1} A_j(t) w2_j^2(t) \right) dt - \\ & sign(e(0)) \int_0^{t_h} \left(\dot{d}(t) - \sum_{j=1}^{m_1+1} A_j(t) w2_j(t) \sum_{i=1}^{m_0} w1_{ji}(t) \dot{x}_i(t) \right) d(t) \geq \\ & \eta \left[t_h + \int_0^{t_h} \left(\sum_{j=1}^{m_1+1} A_j(t) w2_j^2(t) \right) dt \right] - \left(B_{\dot{d}} + m_1 B_A B_{w2} B_{w1} B_{\dot{x}} \right) t_h \geq \\ & \left[\eta - \left(B_{\dot{d}} + m_1 B_A B_{w2} B_{w1} B_{\dot{x}} \right) \right] t_h. \end{aligned} \tag{3.74}$$

□

Obviamente, para todo $t < t_h$, levando em conta o ganho η escolhido em (3.67) para o controlador de modos deslizantes, segue, a partir de (3.71), que

$$\begin{aligned} e(t)\dot{e}(t) &= -\eta \mid e(t) \mid \left(1 + \sum_{j=1}^{m_1+1} A_j(t) w2_j^2(t) \right) + \\ & \left(\dot{d}(t) - \sum_{j=1}^{m_1+1} A_j(t) w2_j(t) \sum_{i=1}^{m_0} w1_{ji}(t) \dot{x}_i(t) \right) e(t) \leq \\ & \left(-\eta + B_{\dot{d}} + m_1 B_A B_{w2} B_{w1} B_{\dot{x}} \right) \mid e(t) \mid < 0 \end{aligned} \tag{3.75}$$

e um movimento de deslizamento existe em $e(t) = 0$ para $t > t_h$.

Assim, pode-se utilizar (3.65) e (3.66) para a correção dos pesos e, uma vez que $s(n)$ está relacionada com a topologia da rede utilizada, para se verificar a existência de um intervalo positivo para o ganho η, é necessário analisar o comportamento da condição de convergência para a rede MLP de duas camadas com uma saída linear.

3.2.2 Determinação de η para uma Rede MLP de Duas Camadas

Seja a saída linear do único neurônio de uma rede MLP de duas camadas em relação a um vetor de entrada $\mathbf{x}(n)$:

$$y2(n) = \sum_{j=1}^{m_1+1} w2_j(n) y1_j(n) = \sum_{j=1}^{m_1+1} w2_j(n) \varphi \left(\sum_{i=1}^{m_0} w1_{ji}(n) x_i(n) \right).$$

Devido a existência de duas camadas, deve-se fazer o estudo do intervalo de convergência para a camada de saída e escondida, separadamente. Assim, tem-se:

- Camada de saída: Considerando somente os pesos da camada de saída como sendo os

parâmetros de interesse, a saída no instante n de uma rede MLP com uma saída linear é dada por:

$$y2(n) = \sum_{j=1}^{m_1+1} w2_j(n) y1_j(n). \tag{3.76}$$

Supondo que o ajuste dos pesos seja realizado, inicialmente, somente nos pesos da camada de saída, (3.76) pode ser comparada à (3.36) para o perceptron linear. Neste caso, as entradas do único neurônio de saída correspondem ao vetor de saída dos neurônios da camada escondida (acrescidos do termo de bias) após a função de ativação, $\mathbf{y1}(n)$, e os pesos, ao vetor $\mathbf{w2}(n)$. Os coeficientes $c_1,\ c_2$ *e* c_3 são obtidos usando (3.63) e considerando (3.66) como a expressão para a correção dos pesos $\mathbf{w}2(n)$. A partir de (3.66), pode-se obter o coeficiente c como sendo $c = \frac{1}{\mathbf{y}1^T(n)\mathbf{y}1(n)} sign(e(n)) \sum_{j=1}^{m_1+1} y1_j^2(n)$. Assim, os coeficientes da equação de segundo grau associada à condição de convergência são definidos como:

$$\begin{aligned} c_1 &= \frac{1}{2}\left(\frac{1}{\mathbf{y}1^T(n)\mathbf{y}1(n)}\right)^2 \left(\sum_{j=1}^{m_1+1} y1_j^2(n)\right)^2 \\ c_2 &= -\frac{|e|}{\mathbf{y}1^T(n)\mathbf{y}1(n)} \sum_{j=1}^{m_1+1} y1_j^2(n) \\ c_3 &= 0. \end{aligned} \tag{3.77}$$

- Camada escondida: Considera-se agora o ajuste dos pesos da camada escondida, $\mathbf{W1}(n)$. Para isso, os pesos da camada de saída serão mantidos constantes. Logo, a saída do único neurônio da rede MLP de duas camadas com saída linear é dada por:

$$y2(n) = \sum_{j=1}^{m_1+1} w2_j(n)\varphi\left(\sum_{i=1}^{m_0} w1_{ji}(n)x_i(n)\right). \tag{3.78}$$

A saída no instante $n+1$ é dada por:

$$y2(n+1) = y2(n) + \Delta y2(n) = y2(n) + \sum_{j=1}^{m_1+1} w2_j(n)\varphi\left(\sum_{i=1}^{m_0} \Delta w1_{ji}(n)x_i(n)\right). \tag{3.79}$$

Aplicando-se em (3.79) a decomposição de primeira ordem da série de Taylor, obtém-se:

$$y2(n+1) = y2(n) + \dot{y2}(n) \sum_{j=1}^{m_1+1} w2_j(n) \sum_{i=1}^{m_0} \Delta w1_{ji}(n)x_i(n), \tag{3.80}$$

onde $|\sum_{i=1}^{m_0} \Delta w1_{ji}(n)x_i(n)| \leq \xi$. Neste caso, usa-se (3.65) para a variação dos pesos no instante n. Porém, para a camada escondida, não existe uma resposta desejada

especificada para os neurônios desta camada. Consequentemente, um sinal de erro para um neurônio oculto deve ser determinado considerando o sinal de saída do neurônio de saída e o peso que liga este neurônio ao neurônio oculto, i.e., usando (3.65) obtém-se $\Delta w1_{ji}(n) = \eta \left(\frac{1}{\mathbf{x}^T(n)\mathbf{x}(n)}\right) w2_j(n) x_i(n) sign(e(n))$. A partir da expressão de $\Delta w1_{ji}(n)$ é possível definir um intervalo para o ganho η em relação à expansão em série de Taylor:

$$\eta \leq \frac{\xi}{\left|\left(\frac{1}{\mathbf{x}^T(n)\mathbf{x}(n)}\right) w2_j(n) \sum_{i=1}^{m_0} x_i^2(n)\right|}. \tag{3.81}$$

Apesar de (3.81) ser atribuída a um único neurônio, o limite para o ganho η deve ser definido em função de toda a rede, escolhendo-se o menor limite associado a um dos neurônios da rede. Decompondo-se (3.80) tem-se:

$$y2(n+1) = y2(n) + \left(\frac{\dot{y2}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}\right) sign(e(n)) \sum_{j=1}^{m_1+1} w2_j^2(n) \sum_{i=1}^{m_0} x_i^2(n)\eta. \tag{3.82}$$

Finalmente, usando (3.63) obtém-se os coeficientes c_1, c_2 e c_3 considerando-se que $c = \left(\frac{\dot{y2}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}\right) sign(e(n)) \sum_{j=1}^{m_1+1} w2_j^2(n) \sum_{i=1}^{m_0} x_i^2(n)$:

$$\begin{aligned} c_1 &= \frac{1}{2}\left(\frac{\dot{y2}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}\right)^2 \left(\sum_{j=1}^{m_1+1} w2_j^2(n) \sum_{i=1}^{m_0} x_i^2(n)\right)^2 \\ c_2 &= -\left(\frac{\dot{y2}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}\right) |e(n)| \sum_{j=1}^{m_1+1} w2_j^2(n) \sum_{i=1}^{m_0} x_i^2(n) \\ c_3 &= 0. \end{aligned} \tag{3.83}$$

Uma vez obtidos os coeficientes das equações de convergência do erro, pode-se aplicar o Teorema 3.4, sendo o intervalo final para o ganho η determinado pela intersecção dos intervalos definidos pelas equações de convergência do erro para as camadas escondida e de saída, observando o limite imposto pela decomposição em série de Taylor.

3.3 Conclusão

Neste capítulo foram apresentados dois algoritmos para treinamento em tempo real de redes MLP de duas camadas com a camada de saída linear, os quais possibilitam a determinação de um ganho adaptativo, determinado iterativamente, a cada passo de atualização dos pesos da rede. Os algoritmos propostos seguem a mesma metodologia para obtenção do ganho adaptativo, diferindo em dois pontos principais: na definição de superfície de deslizamento e na expressão usada para atualização dos pesos da rede. Como consequência destas diferenças, a primeira proposta apresentada é mais generalista, possibilitando que haja mais de um

neurônio na camada de saída da rede, enquanto a segunda proposta é limitada a apenas uma saída escalar. Em contrapartida, a segunda proposta atualiza os pesos da rede usando uma lei que permite a estabilidade assintótica no senso de Lyapunov, para um conjunto de pesos que corresponde ao mínimo global.

Através do uso dos algoritmos propostos é possível a determinação de um intervalo resultante para o ganho η da rede, o qual é obtido através da intersecção dos intervalos definidos para a camada escondida e de saída, observando o limite imposto pela decomposição em série de Taylor. Os algoritmos propostos, no entanto, não definem o valor ótimo para o ganho η. Em princípio, qualquer valor dentro de um intervalo resultante positivo poderia ser usado. Questões de otimização não são abordadas pelos algoritmos apresentados. Porém, para fins de obtenção de resultados usando os algoritmos propostos, será adotada uma solução conservadora, utilizando-se o valor obtido a partir da série de Taylor.

No próximo capítulo, serão apresentados os resultados obtidos na simulação dos algoritmos propostos nas seguintes aplicações: aproximação de uma função periódica e no acionamento elétrico de um MI.

Capítulo 4

Avaliação dos Algoritmos Propostos

Neste capítulo são apresentados os resultados obtidos a partir de simulações dos algoritmos propostos no Capítulo 3. As simulações são realizadas considerando duas aplicações distintas para os algoritmos propostos. Na Seção 4.1 os algoritmos são utilizados na aproximação de uma função senoidal. Em seguida, estes mesmos algoritmos são utilizados no acionamento elétrico de um MI. Na Seção 4.2, o algoritmo da segunda proposta é usado no controle do MI, enquanto na Seção 4.3, o algoritmo da primeira proposta é utilizado para a observação do fluxo de estator do MI. A escolha da segunda proposta para o controle do MI e da primeira proposta para a observação do fluxo do MI foi feita por dois motivos: pela particularidade da aplicação, i.e., no caso do controle, necessita-se uma RNA com apenas uma saída, enquanto para o caso da observação do fluxo, a rede deve ter duas saídas; e, pela premissa de se ter o menor custo computacional possível, que não seria obtido usando-se a segunda proposta como observador do fluxo do MI pela necessidade de se ter duas estruturas de redes MLP compartilhando as mesmas entradas.

Por fim, na Seção 4.4 é feita uma descrição do desenvolvimento de uma bancada experimental que possibilita a implementação e avaliação de estratégias de acionamento elétrico de MI. Na Seção 4.5 são apresentadas as conclusões do capítulo.

4.1 Aproximação de Função

Nesta seção são apresentados os resultados de simulação da aplicação dos algoritmos propostos no aprendizado em tempo real da função $f(t) = e^{(-\frac{1}{3})}sin(3t)$. Foram considerados os seguintes parâmetros para as simulações: passo de integração = 10μs; tempo de simulação = 2s; período de amostragem = 250μs. As mesmas simulações foram também realizadas considerando o algoritmo BP padrão (Rumelhart et al., 1986), o algoritmo proposto por (Topalov et al., 2003) e os dois algoritmos para treinamento em tempo real apresentados por (Parma, 2000). Para estes algoritmos, os ganhos de treinamento (taxas de aprendizado) foram escolhidos de forma a se obter o melhor resultado, usando-se as mesmas condições iniciais, para cada um dos algoritmos simulados.

A topologia de rede usada na simulação dos algoritmos foi a seguinte: uma entrada, 5 neurônios na camada escondida e um neurônio na camada de saída. A dimensão da camada escondida da rede MLP foi definida em função da melhor resposta possível com o menor número de neurônios. A função tangente hiperbólica foi usada como função de ativação para os neurônios da camada escondida. Esta mesma função foi também usada como função de ativação para o neurônio da camada de saída nos algoritmos BP padrão e nas duas propostas de (Parma, 2000). Para os algoritmos apresentados nesta tese e aquele proposto por (Topalov et al., 2003), foi usada a saída linear para o neurônio da camada de saída.

Os resultados de simulação da primeira e segunda propostas são apresentados nas Figuras 4.1 e 4.2. Para as duas propostas foi usado $\xi = 1.5$ (ver Tabela B.1) para o intervalo de confiança para a aproximação da função tangente hiperbólica utilizando a expansão de primeira ordem da série de Taylor.

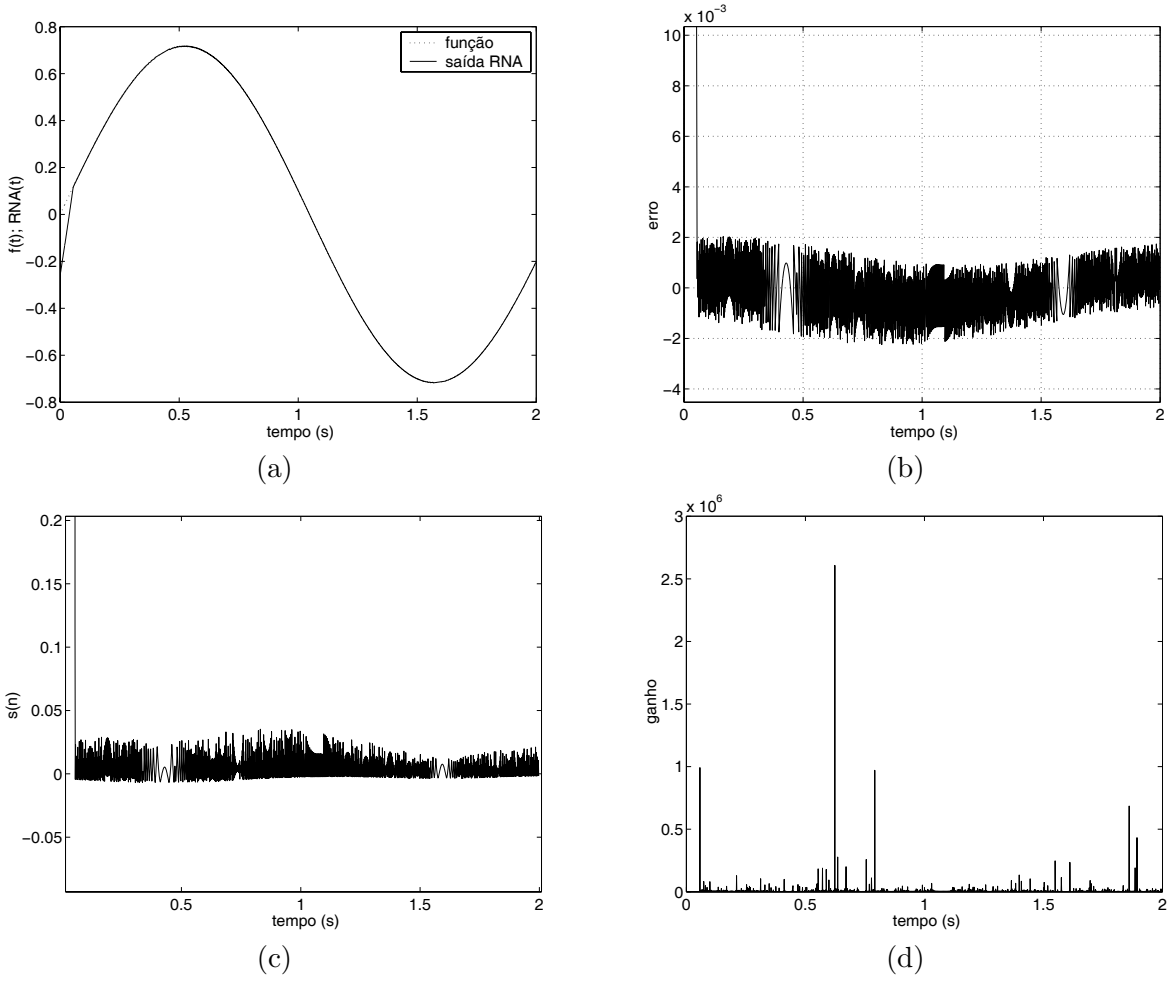


Figura 4.1: Resultados de simulação da aproximação de $f(t)$ usando a *primeira* proposta: (a) saída $f(t)$ x RNA(t); (b) erro entre saída $f(t)$ e saída da RNA; (c) comportamento de $s(n)$; (d) ganho adaptativo.

Na primeira proposta, foi adotado para o parâmetro C o valor de 10000. A função $f(t)$ é mostrada tracejada enquanto a saída da RNA é mostrada em traço contínuo. São também

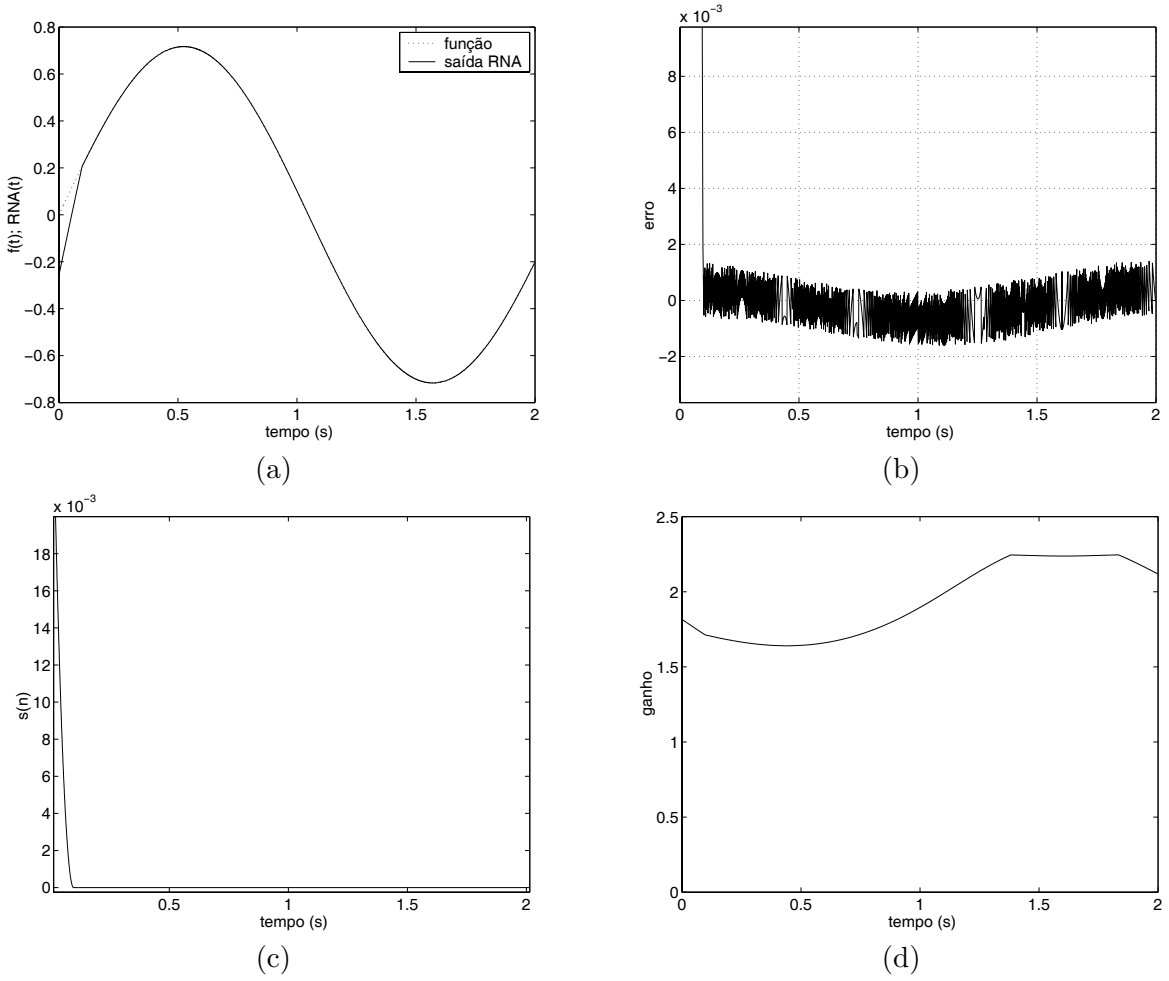


Figura 4.2: Resultados de simulação da aproximação de $f(t)$ usando a *segunda* proposta: (a) saída $f(t)$ x RNA(t); (b) erro entre saída $f(t)$ e saída da RNA; (c) comportamento de $s(n)$; (d) ganho adaptativo.

mostrados o gráfico do erro de aproximação para a função senoidal considerada, o comportamento da superfície de deslizamento $s(n)$ e os ganhos de treinamento determinados pelos algoritmos durante o tempo de simulação.

Ambos algoritmos apresentam desempenho semelhante, com erro de aproximação na mesma ordem de grandeza, a despeito da segunda proposta apresentar limites para o erro ligeiramente menores do que aqueles verificados para a primeira proposta. Em contrapartida, a primeira proposta leva menos tempo para alcançar a função senoidal, o que já era esperado devido a definição da superfície de deslizamento envolver não somente o erro de saída da rede mas também a derivada do erro. A diferença mais marcante está relacionada com os valores determinados para o ganho adaptativo. A primeira proposta apresenta ganhos extremamente altos enquanto a segunda proposta fica limitada a ganhos menores do que 2,5. Isto pode ser explicado pela maneira como os pesos são atualizados em cada proposta. O fato da primeira proposta utilizar o gradiente da função erro em relação aos pesos faz com que ocorram oscilações no processo de aprendizado implicando na necessidade da ganhos elevados para o

treinamento da rede. Estas oscilações também são sentidas no comportamento da superfície de deslizamento, conforme pode ser verificado no gráfico (c) da Figura 4.1.

Na Figura 4.3 são apresentados os resultados de simulação dos algoritmos propostos por (Parma, 2000) e por (Topalov et al., 2003).

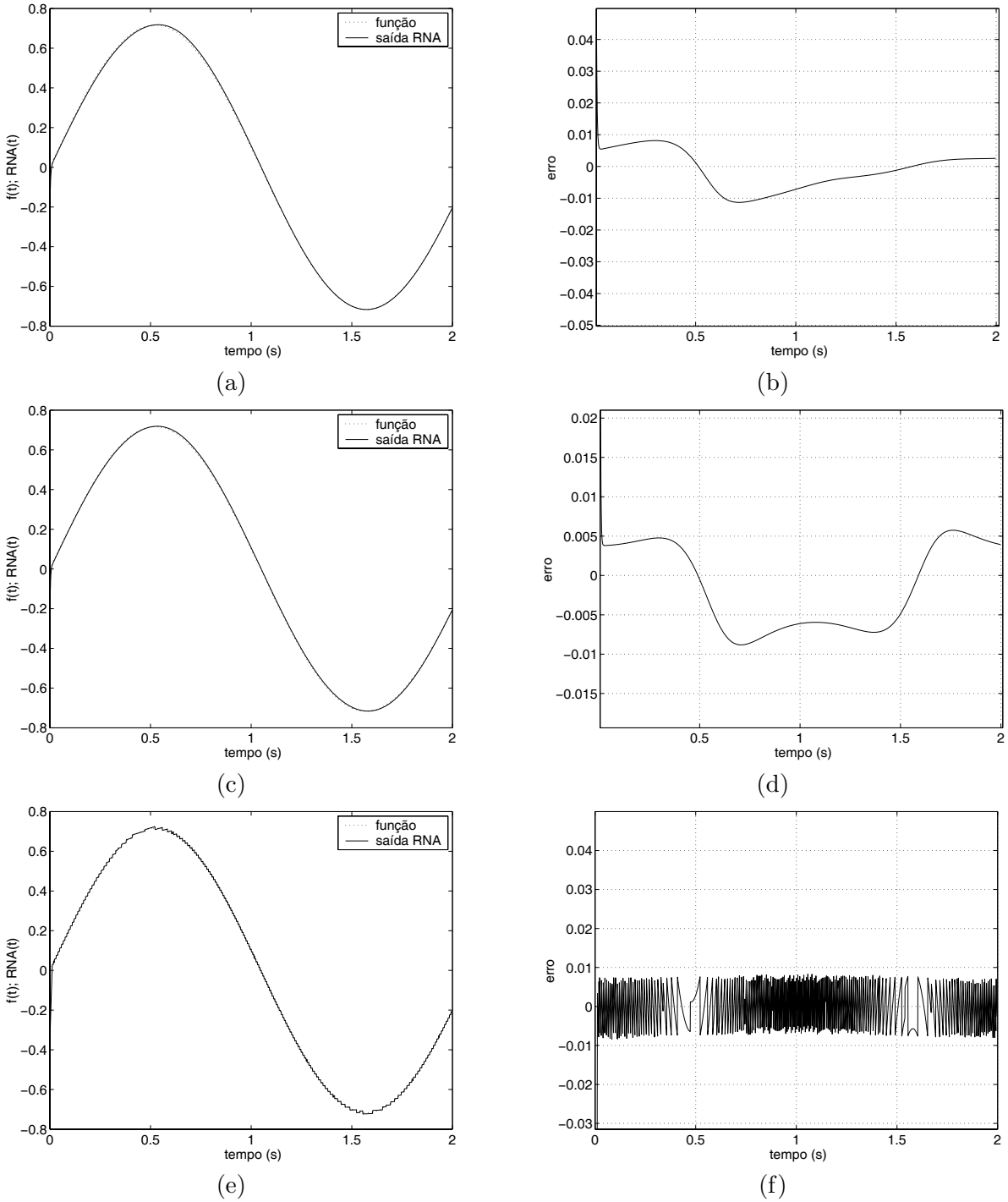
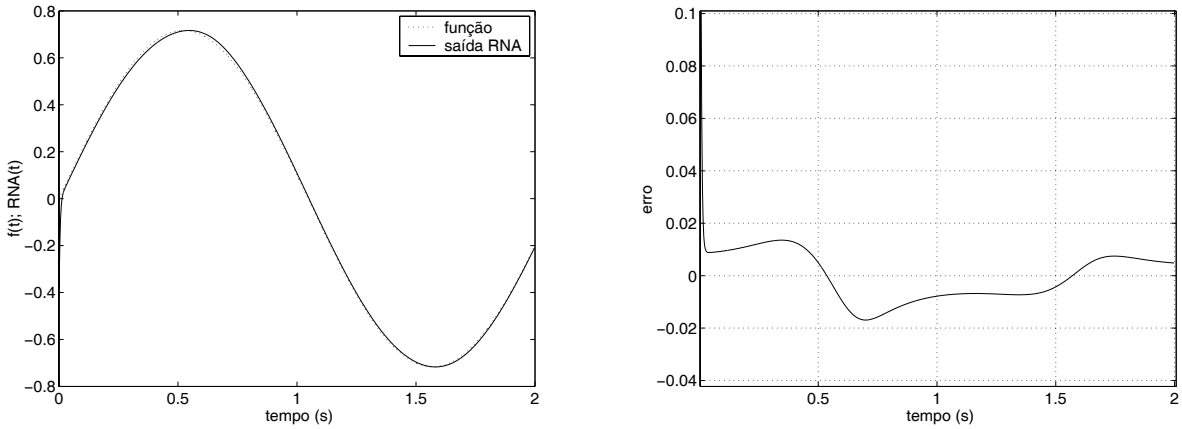


Figura 4.3: Resultados de simulação da aproximação de $f(t)$ usando as propostas de Parma e Topalov: gráficos (a) e (b) - 1a. proposta Parma; gráficos (c) e (d) - 2a. proposta Parma; gráficos (e) e (f) - proposta Topalov.

Os coeficientes e os ganhos dos algoritmos foram ajustados obtendo-se os seguintes valores: 1a. proposta Parma - $C1$=$C2$=10000, $\eta 1$=3000, $\eta 2$=10; 2a. proposta Parma - $C1$=$C2$=10000, $\eta 1$=200, $\eta 2$=100; proposta Topalov - η=10. Estes três algoritmos apresentaram resultados semelhantes, especialmente se for considerado o tempo necessário para alcançar a função senoidal, o qual é bem menor se comparado com as propostas apresentadas nesta tese. O fato dos algoritmos propostos nesta tese usarem um ganho adaptativo penaliza o tempo de alcance da função $f(t)$. Em contrapartida, se forem comparados os erros na aproximação da função, os algoritmos propostos têm um melhor desempenho.

Finalmente, na Figura 4.4 são mostrados os resultados obtidos usando o algoritmo BP padrão. Os valores ajustados de ganho para as camadas escondida e de saída foram, respectivamente, $\eta 1$=102 e $\eta 2$=12.



(a) (b)

Figura 4.4: Resultados de simulação da aproximação de $f(t)$ usando o algoritmo BP padrão: (a) saída $f(t)$ x RNA(t); (b) erro entre saída $f(t)$ e saída da RNA.

Como pode ser verificado facilmente, o algortimo BP padrão apresentou o maior erro na aproximação da função considerada. Este desempenho já era esperado pelas várias razões apontadas anteriormente. Os resultados deste algoritmo foram apresentados como uma forma de referência, visto que este algoritmo é o mais antigo entre os algoritmos simulados.

4.2 Controle do Motor de Indução

As técnicas de controle vetorial para o MI baseiam-se na representação das variáveis elétricas do modelo do motor em componentes ortogonais de forma semelhante a vetores. A partir desta representação é possível o controle desacoplado (ou quase-desacoplado) entre o fluxo e o conjugado, resultando em precisão, rapidez de resposta e eficiência do acionamento onde o motor está inserido (Leonhard, 1985).

A orientação pelo campo é uma técnica de controle vetorial que busca o controle desacoplado de fluxo magnético e conjugado através da decomposição das variáveis atuantes (vetor tensão ou vetor corrente) em componentes ortogonais (Leonhard, 1985). A decomposição das

variáveis ao longo de eixos ortogonais é feita na direção de um dos vetores enlace de fluxo: de rotor, de estator ou mútuo, dando origem às respectivas técnicas. A componente de eixo direto é conhecida como componente de produção de fluxo, enquanto que a componente de eixo em quadratura é conhecida como componente de produção de conjugado.

As técnicas de controle por orientação pelo campo foram generalizadas por (Dedoncker e Novotny, 1988) através do controlador universal orientado pelo campo (UFO - *universal field oriented controller*). Este controlador pode operar com orientação segundo os vetores fluxo de rotor, de estator ou mútuo, além de poder ser usado nos modos direto e indireto. O controlador proposto por Dedoncker e Novotny (1988) foi desenvolvido para alimentação em corrente, mas a alimentação também pode ser feita em tensão com algum método de controle de corrente, implicando na necessidade da existência de circuitos desacopladores.

O desenvolvimento de um controlador universal de orientação pelo campo para alimentação de tensão *sem* o uso de controladores de corrente foi generalizada em (Silva, 1995) na forma de um controlador universal denominado UFOV (*universal field oriented voltage*), análogo ao controlador UFO, podendo operar com orientação na direção dos vetores fluxo de rotor, de estator e mútuo. Entre os controladores UFOV, o controlador universal por tensão orientado pelo campo de estator (UFOVS - *universal field oriented voltage stator*) constitui o método ideal nesta classe de técnicas vetoriais, pois não apresenta acoplamento entre grandezas de eixo direto e em quadratura. Além disso, controladores UFOVS apresentam maior robustez à variação paramétrica do que os controladores UFO com orientação pelo campo de estator, pois não necessitam de circuitos desacopladores, sendo bastante apropriados para técnicas que não medem velocidade (*sensorless*). A principal desvantagem dos métodos de alimentação em tensão são os altos picos de corrente durante os transitórios. Este inconveniente pode ser reduzido através de um controle de malha fechada de fluxo e conjugado.

Nesta seção são apresentados os resultados de simulação da aplicação do algoritmo da *segunda* proposta no controle vetorial do MI. Este algoritmo foi usado como controlador na estrutura do controle direto orientado segundo fluxo de estator para alimentação em tensão sem uso de controlador de corrente. Esta estrutura apresenta três malhas de controle: controladores de fluxo, velocidade e conjugado, conforme pode ser visto na Figura 4.5 (Silva, 1995). Cada malha de controle pode usar um controlador convencional PI ou então, um controlador não convencional como, por exemplo, um neurocontrolador.

A obtenção do módulo e da posição do fluxo magnético pode ser feita através da inclusão de bobinas ou sensores no motor, ou utilizando-se estimadores e observadores de fluxo. A inclusão de bobinas ou sensores no motor é de difícil realização e via de regra implica em perda de robustez. Já os estimadores e observadores utilizam as equações do modelo da máquina de indução para obter o fluxo desejado sendo, por isso, de mais fácil implementação. Porém, estas estruturas se mostram dependentes do modelo da máquina adotado e dos respectivos parâmetros que são usados para implementar o observador.

Na Figura 4.6 (Parma, 2000) (Justino, 2004) é mostrada a estrutura de controle do MI usando controladores neurais. Esta estrutura é composta por duas redes independentes: a rede

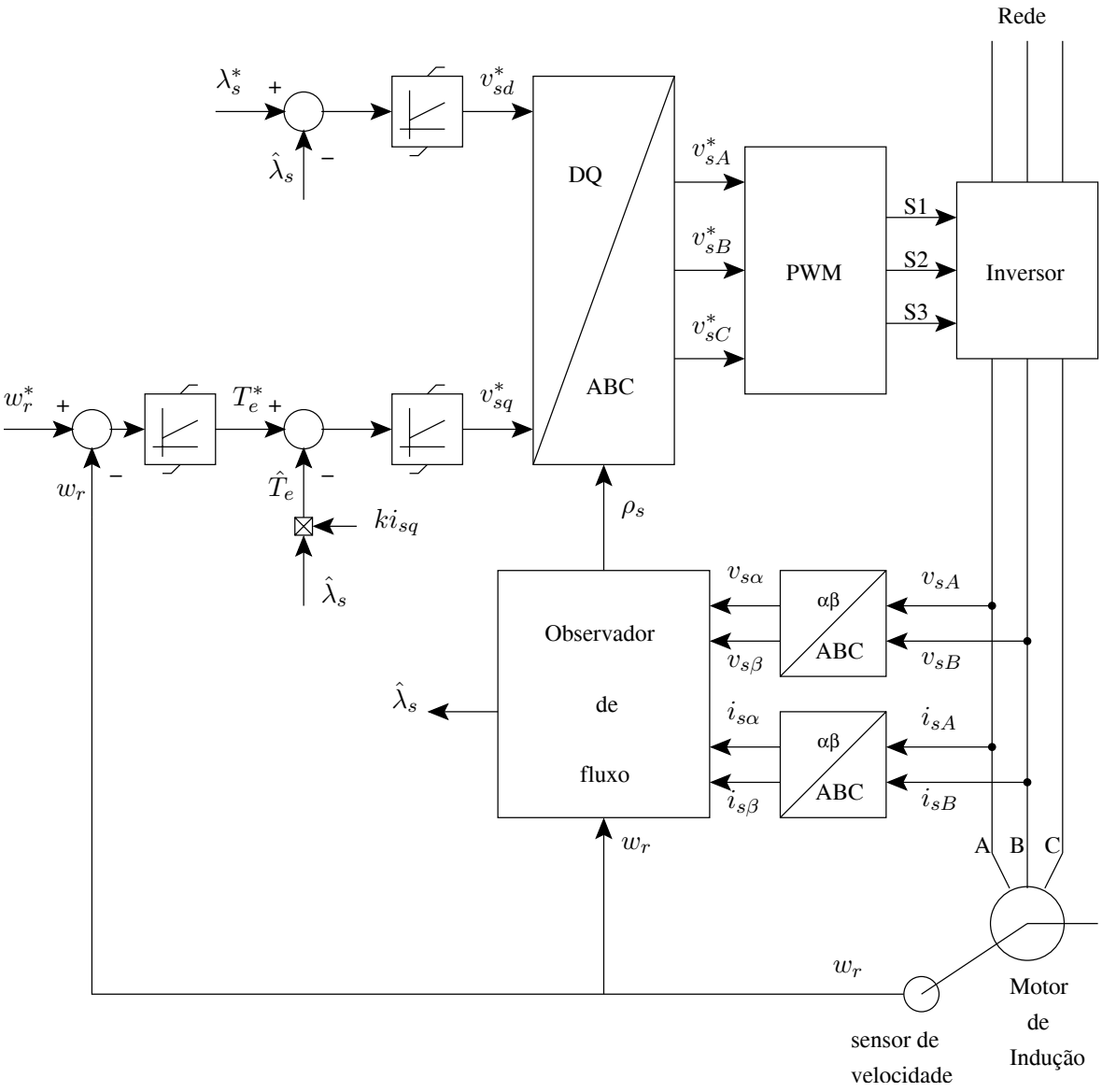


Figura 4.5: Estrutura do controle direto orientado segundo fluxo de estator usando PI's.

neural de controle de fluxo e a rede neural de controle de velocidade. No esquema adotado, a malha interna de conjugado não foi incluída. Neste caso, os picos de corrente durante os transitórios são limitados indiretamente através da limitação do conjugado.

Um programa escrito em linguagem C foi desenvolvido com o objetivo de simular as estruturas de controle mostradas nas Figuras 4.5 e 4.6. A integração numérica do modelo de equações diferenciais do MI é realizada usando o método de Runge-Kutta de quarta ordem. O MI foi alimentado por um inversor de frequência com uma frequência de PWM de 8 kHz. Os parâmetros usados na simulação e os parâmetros do MI são mostrados na Tabela C.1 e na Tabela C.2, respectivamente (ver Anexo C).

Foi usado o observador de fluxo de Gopinath A-3 (Hori et al., 1987) discretizado. Trata-se de um observador de ordem reduzida, que utiliza um modelo de estimador de corrente, com

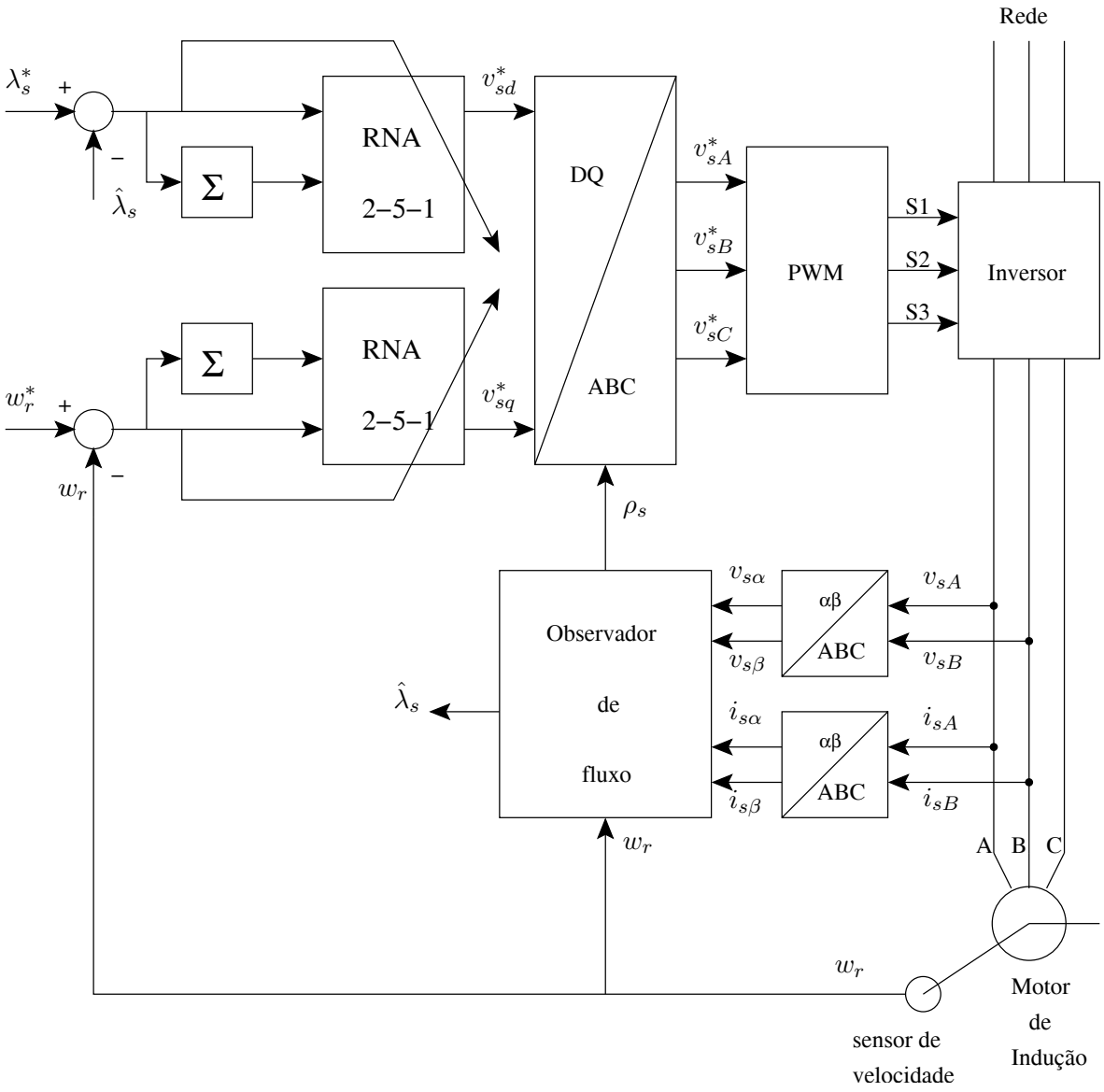


Figura 4.6: Estrutura do controle direto orientado segundo fluxo de estator usando neurocontroladores.

termo de correção do erro de predição baseado na derivada da corrente de estator.

O MI foi submetido aos transitórios de *partida e reversão de velocidade* (PRV) sem carga, e *aplicação e retirada de carga* (ARC) (constante) a uma velocidade constante. Estes transitórios permitem uma real excitação de todas as dinâmicas e não-linearidades do sistema de acionamento em estudo (Silva, 1995).

As Figuras 4.7 e 4.8 apresentam os resultados obtidos para o MI submetido aos transitórios de PRV e ARC usando as estruturas de controladores PI e neurais, respectivamente. Em ambos os transitórios, a velocidade de referência para o MI foi de 150 rad.ele/s em rampa. No transitório de ARC foi aplicado um conjugado de carga constante de 4 Nm.

Os pesos das RNA usadas nos controladores neurais são inicializados através de uma amos-

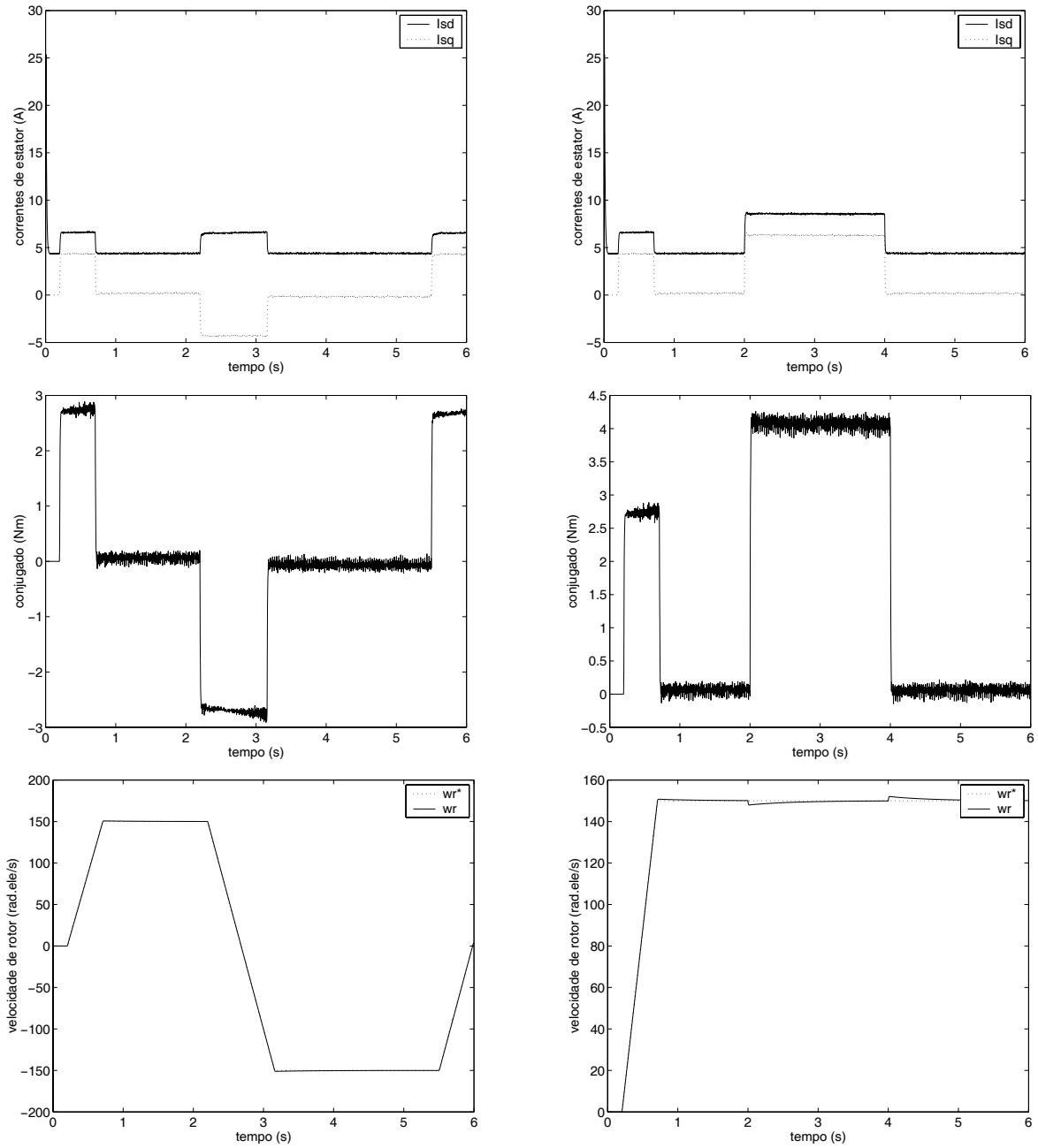


Figura 4.7: Resultados de simulação dos controladores PI: (a) partida (t=0,2s) e reversão de velocidade (t=2,2s) sem carga; (b) aplicação (t=2s) e retirada (t=4s) de carga (constante de 4 Nm) na velocidade de 150 rad.ele/s.

tragem de uma distribuição normal com média zero quando da primeira vez que o algoritmo de controle é iniciado. Os resultados usando neurocontroladores foram obtidos após dois passos de simulação para os transitórios de PRV e ARC.

Algumas simulações foram realizadas para escolher o número de neurônios da camada escondida, com o objetivo de se obter a configuração mais simples para aliviar o custo computacional e ao mesmo tempo, permitir que a rede ainda tivesse um bom desempenho. Assim,

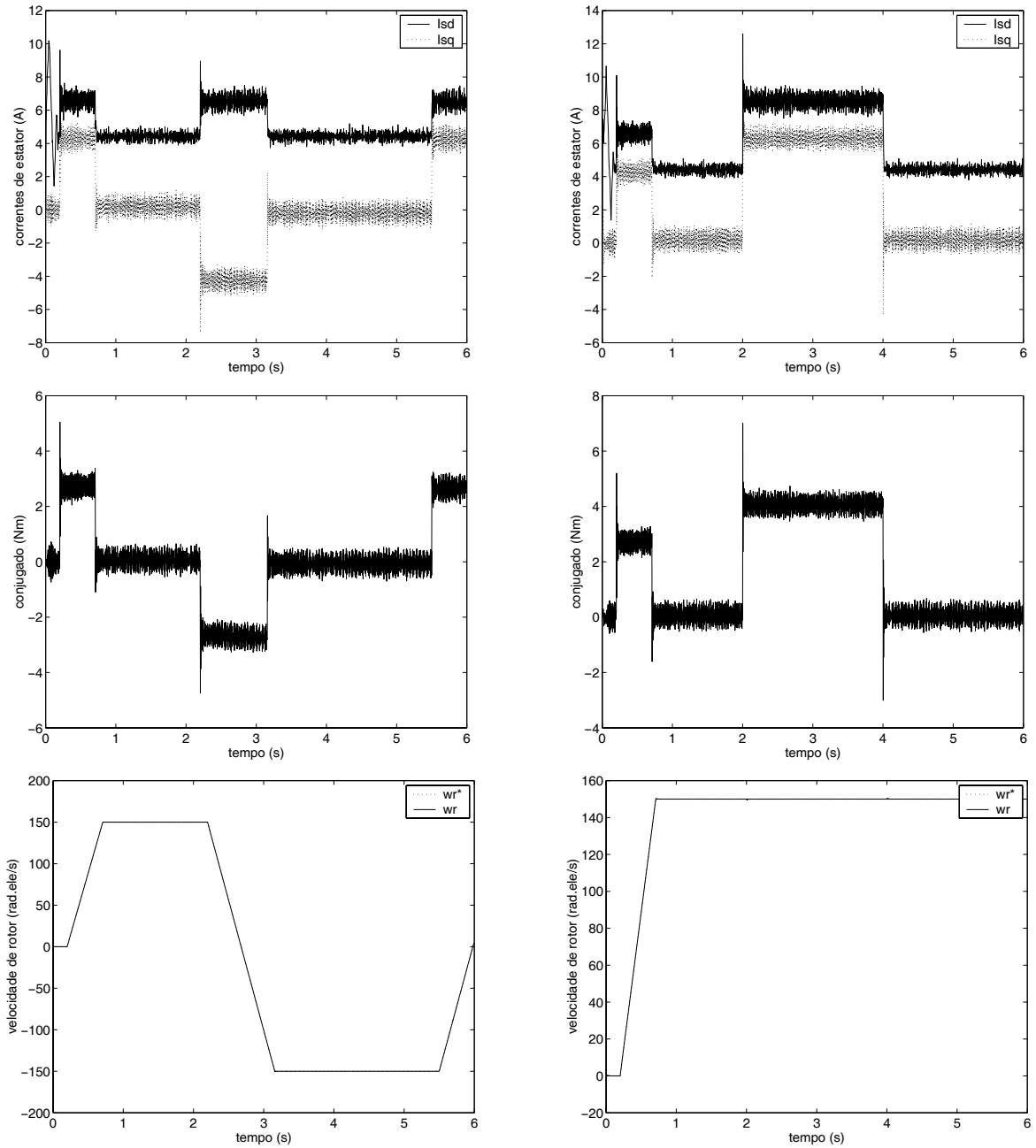


Figura 4.8: Resultados de simulação dos neurocontroladores: (a) partida (t=0,2s) e reversão (t=2,2s) de velocidade sem carga; (b) aplicação (t=2s) e retirada (t=4s) de carga (constante de 4 Nm) na velocidade de 150 rad.ele/s.

foram definidos cinco neurônios na camada escondida. Os outros parâmetros das RNA foram selecionados conforme a aplicação, i.e., a camada de saída para cada rede utilizada tinha que ter apenas um neurônio por causa dos sinais de tensão de referência usados na estrutura do controlador UFOVS empregado. Por sua vez, os sinais de entrada das redes foram selecionados para corresponder aos valores proporcional e integral usados nos controladores PIs.

Para ambos os controladores, após um intervalo de aproximadamente 0,2 s, o fluxo nominal

do MI é estabelecido, e a velocidade do rotor segue a referência. Durante este intervalo de tempo, a adaptação inicial dos parâmetros das redes neurais dos neurocontroladores é muito intensa, sendo notada especialmente nos gráficos da corrente de estator.

O sinal da corrente de estator se manteve dentro da faixa de valores nominais definidos para o motor simulado nos transitórios de PRV e ARC. Naturalmente, o comportamente da corrente se refletiu no conjugado eletromagnético do motor que, para os neurocontroladores, apresentou um sobresinal nos transitórios aos quais o motor foi submetido. O fato da estrutura dos neurocontroladores não incluir uma malha de conjugado influenciou o aparecimento destes sinais não desejáveis. O sinal de velocidade do rotor para ambos os controladores seguiu a referência, rejeitando a perturbação de carga. Pode-se notar, entretanto, uma melhor rejeição a perturbação apresentada pelo neurocontrolador se comparado ao controlador PI.

4.3 Observação do Fluxo de Estator do Motor de Indução

Considerando o acionamento do motor de indução, a correta estimação do fluxo, seja de estator, rotor ou mútuo, é a chave para o sucesso da implementação de qualquer estratégia de controle vetorial (Holtz e Quan, 2003).

A observação, por sua vez, é uma estimação em malha fechada que emprega, além dos sinais de entrada, um sinal de realimentação, obtido a partir dos sinais de saída do sistema e do modelo do processo.

Um requisito importante para uso de RNA na observação do fluxo do MI é que o treinamento seja em tempo real. A utilização deste tipo de treinamento possibilita um contínuo ajuste dos pesos da rede às exigências do sistema no qual a rede está inserida, neste caso, o MI. A Figura 4.9 apresenta os resultados de simulação da aplicação da primeira proposta apresentada no Capítulo 3 para treinamento de uma rede neural usada como observador do fluxo de estator do MI. As figuras 4.10 e 4.11 mostram os resultados experimentais obtidos para o observador neural utilizando a 2a. proposta on-line de (Parma, 2000). Foram consideradas as seguintes variáveis: módulo do fluxo de estator (do modelo do MI *versus* observador neural), torque eletromagnético e velocidade do motor. O MI foi submetido aos transitórios de partida e reversão de velocidade (PRV), e aplicação e retirada de carga (ARC).

O fluxo do MI pode ser estimado diretamente a partir da equação de tensão dada por (Novotny e Lipo, 1996)[1]:

$$\mathbf{v}_s = R_s \mathbf{i}_s + \frac{d\lambda_s}{dt} \Rightarrow \tag{4.1}$$

$$\lambda_s = \int (\mathbf{v}_s - R_s \mathbf{i}_s) dt. \tag{4.2}$$

A principal razão para o uso de (4.2) é a simplicidade. O estimador de fluxo de estator não depende da medida de velocidade se o referencial estacionário for adotado para os eixos d-q (Kovács e Rácz, 1984). Este fato torna a abordagem atrativa para uso no controle do MI sem

[1] As equações do modelo básico do MI são mostradas no Apêndice C

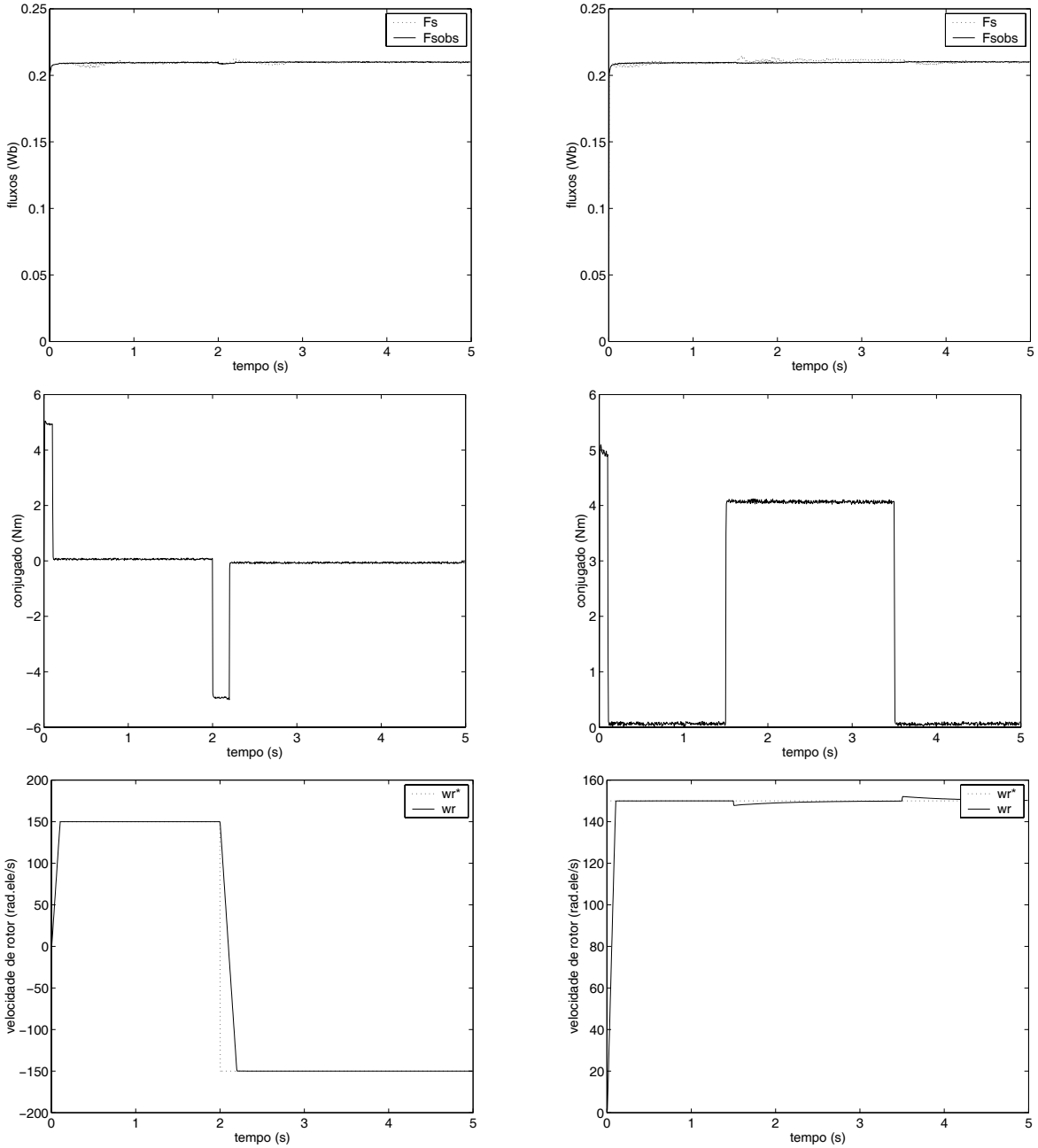
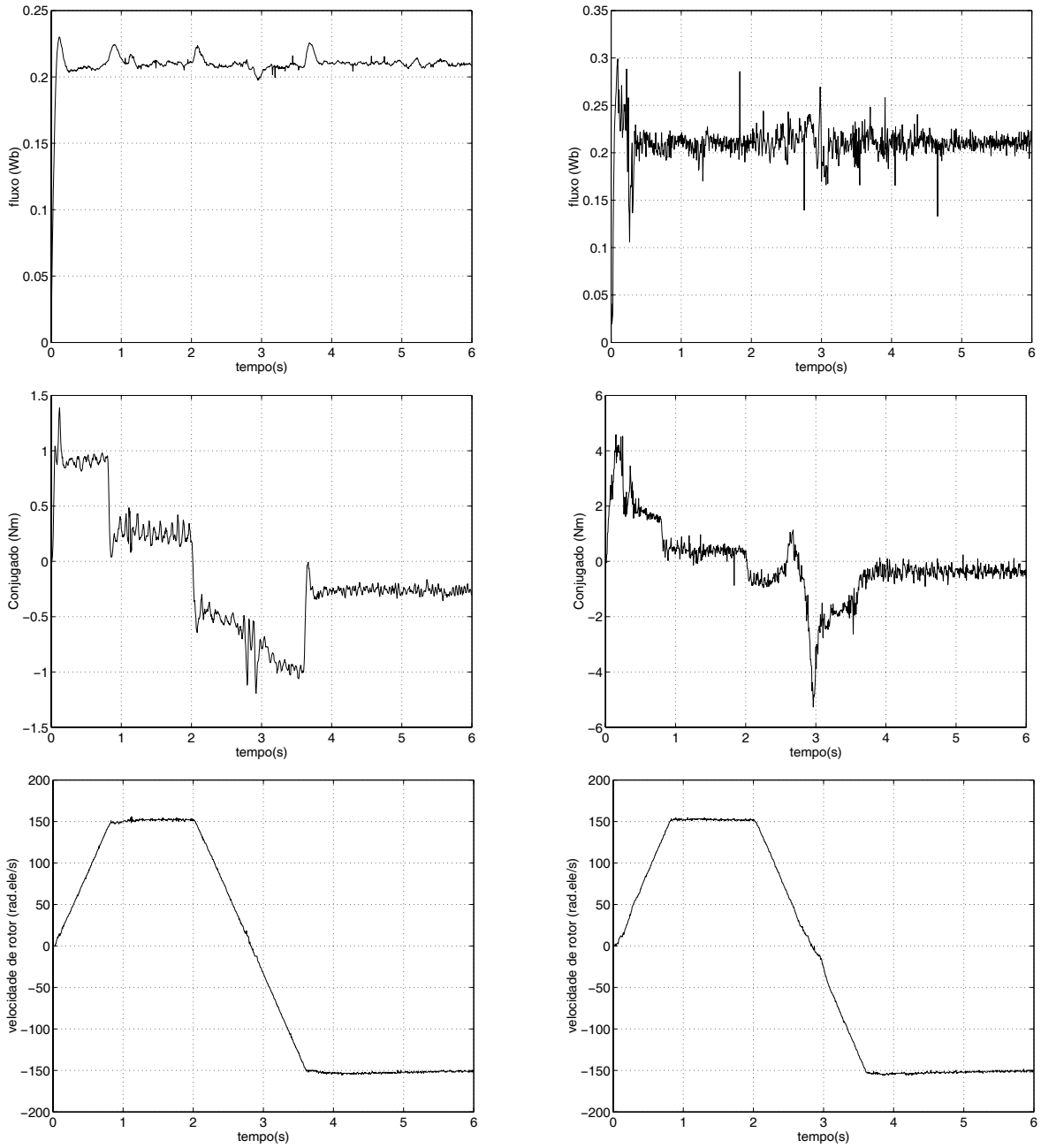


Figura 4.9: Resultados de simulação do observador neural: (a) partida e reversão (t=2s) de velocidade sem carga; (b) aplicação (t=1,5s) e retirada (t=3,5s) de carga (constante de 4 Nm) na velocidade de 150 rad.ele/s.

medição de velocidade. Além disso, pode-se verificar que a única dependência paramétrica é a resistência de estator, a qual pode ser obtida com razoável precisão (Novotny e Lipo, 1996). Soluções eficientes para a correção de *off-set* nas integrais de tensão e corrente podem ser verificadas em (Holtz e Quan, 2003) e (Gouvêa et al., 2004).



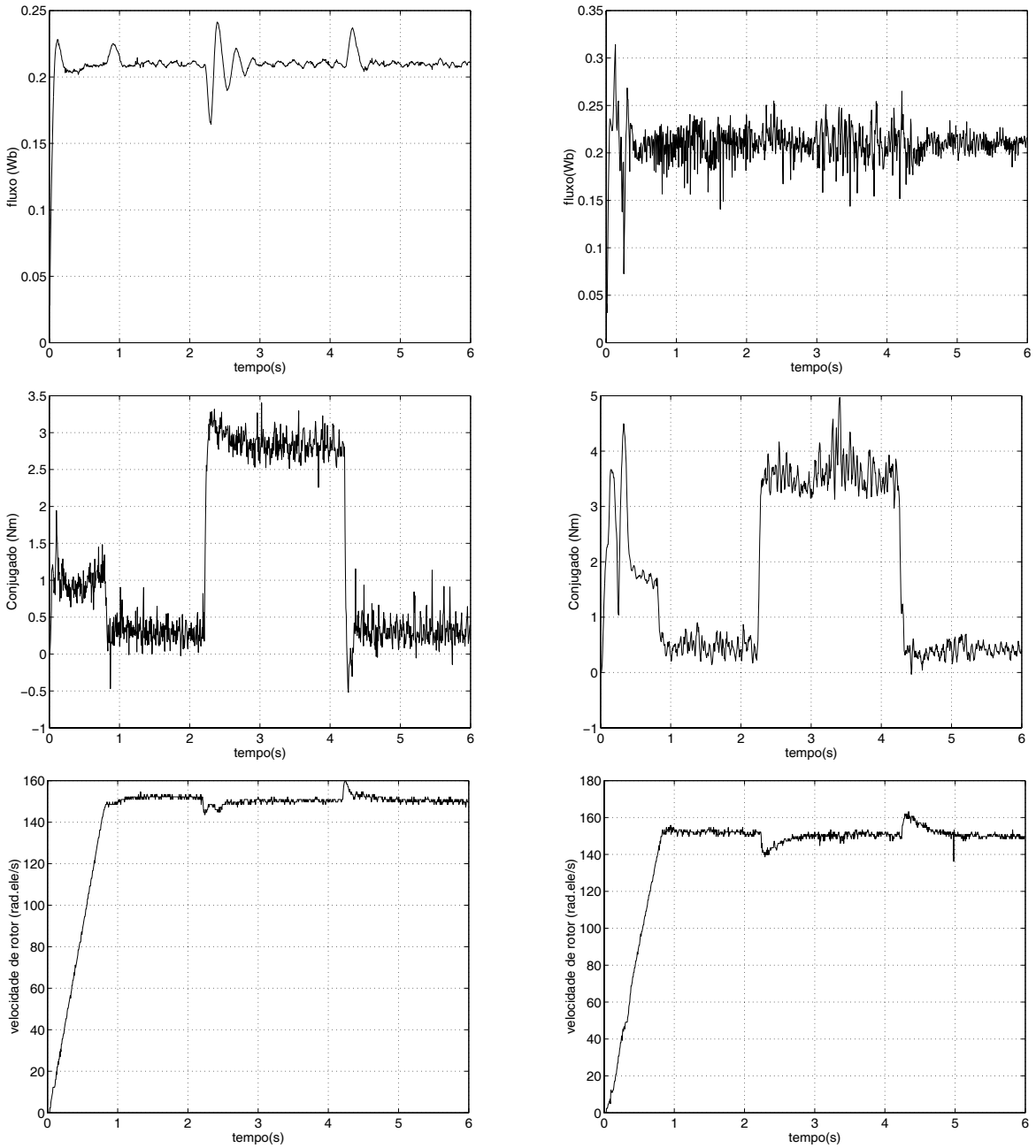
(a) (b)

Figura 4.10: Resultados experimentais: (a) partida e reversão (t=2s) de velocidade sem carga usando o observador de Gopinath; (b) partida e reversão (t=2s) de velocidade sem carga usando o observador neural.

Reescrevendo (4.2) em relação aos eixos d-q tem-se:

$$\mathbf{v}_{sd} = R_s \mathbf{i}_{sd} + \frac{d\lambda_{sd}}{dt} \tag{4.3}$$

$$\mathbf{v}_{sq} = R_s \mathbf{i}_{sq} + \frac{d\lambda_{sq}}{dt}, \tag{4.4}$$



(a) (b)

Figura 4.11: Resultados experimentais: (a) aplicação (t ≈ 2,2s) e retirada (t ≈ 4,2s) de carga (constante) na velocidade de 150 rad.ele/s usando o observador de Gopinath; (b) aplicação (t ≈ 2,2s) e retirada (t ≈ 4,2s) de carga (constante) na velocidade de 150 rad.ele/s usando o observador neural.

onde R_s é a resistência de estator; v_{sd} e v_{sq} são as tensões de estator, i_{sd} e i_{sq} são as correntes de estator, λ_{sd} e λ_{sq} são os fluxos de estator, todos referidos aos eixos d-q em referencial estacionário.

Assim, as componentes d-q da corrente de estator são usadas como entrada da RNA, sendo as componentes d-q de fluxo de estator as saídas da rede. Esta estrutura para o observador

neural de fluxo de estator do MI foi primeiramente proposta por (Parma et al., 1998b). A RNA utilizada é do tipo MLP 2-5-2. O número de neurônios na camada escondida foi determinado através da análise dos resultados de simulação, visando reduzir o custo computacional sem comprometer os resultados gerados pela rede. O observador neural apresentado em (Parma et al., 1998b) utiliza, para treinamento da rede neural, o algoritmo correspondente à 2a. proposta apresentada por (Parma, 2000). Outros trabalhos utilizando o mesmo observador podem ser vistos em (Nied et al., 2003a), (Nied et al., 2003b), (Nied et al., 2004).

Em (Nied et al., 2004) é feito um estudo comparativo entre um estimador e dois observadores de fluxo de estator do MI. Um dos observadores usados no estudo é o observador de Gopinath, enquanto o outro é o observador neural proposto por (Parma et al., 1998b). Os resultados experimentais obtidos permitiram avaliar o desempenho do estimador e dos observadores considerando transientes de velocidade e carga. O desempenho apresentado pelo observador neural confirmou os resultados obtidos anteriormente em simulação e demonstrou a viabilidade do uso de RNA para estimação de fluxo do MI.

Neste trabalho, o mesmo observador neural proposto por (Parma et al., 1998b) foi simulado porém, utilizando para treinamento da rede neural o algoritmo da primeira proposta mostrada no Capítulo 3. Foi adotado o mesmo programa usado para simular o controlador universal por tensão orientado pelo campo de estator porém, neste caso, a observação do fluxo é feita pelo observador neural ao invés do observador de Gopinath. O MI foi submetido aos transitórios de PRV sem carga (gráficos da Figura 4.9 (a)) e ARC (constante) a uma velocidade constante (gráficos da Figura 4.9 (b)). Porém, neste caso, pelo fato de se estar interessado no desempenho do observador ao invés do controlador, foi feita apenas uma reversão de velocidade e o tempo de simulação foi de 5 s. Foi adotada a velocidade de referência do rotor de 150 rad.ele/s em degrau. O comportamento das malhas de fluxo, velocidade e conjugado atestam o bom desempenho do observador neural.

4.4 Descrição de uma Bancada Experimental

Esta seção apresenta o projeto de desenvolvimento e implementação de uma plataforma para ensaios de sistemas de controle digital de tempo real usando um processador digital de sinais (DSP - *digital signal processor*). O desenvolvimento desta plataforma de ensaios está inserido no contexto do Programa Nacional de Cooperação Acadêmica (PROCAD), no qual a Universidade Federal de Minas Gerais (UFMG), a Universidade do Estado de Santa Catarina (UDESC) e a Universidade Federal de Pernambuco (UFPE) tiveram o projeto "Controle de Fluxo e Torque em Motores de Indução" aprovado junto a CAPES, entidade gestora do PROCAD, para o período de 2001-2005.

Este PROCAD teve como objetivo promover a integração de três equipes com afinidades de pesquisa evidentes, em torno do problema de identificação e controle robusto de motores de indução, favorecendo a consolidação das equipes de pesquisa emergentes como a UDESC e a UFPE.

O desenvolvimento desta plataforma experimental teve dois pré-requisitos: custo reduzido, dando-se preferência ao uso de hardware comercial, aumentando assim a confiabilidade da montagem final; e, alta flexibilidade. O atendimento aos pré-requisitos possibilita a implementação de várias estratégias de controle de sistemas, no caso particular, voltadas para o acionamento do MI, objetivo principal do desenvolvimento da plataforma.

A primeira versão implementada da plataforma usava dois DSPs na configuração mestre-escravo (Souza et al., 2002), (Nied et al., 2002). O módulo mestre, de alto desempenho, era dedicado ao gerenciamento das operações e execução de algoritmos mais complexos, o TMS320C6201, e o módulo escravo era dedicado ao controle e aquisição de sinais do sistema, o TMS320F2407, ambos da Texas Instruments Inc. A plataforma usava uma interface difundida no ambiente acadêmico e industrial, o Simulink^{TM}, e a programação em linguagem C era feita através do software Code Composer Studio^{TM}, da Texas Instruments Inc.

Porém, a limitação da taxa de amostragem em 1kHz motivou a busca por outra alternativa que permitisse o uso de frequências mais altas. Esta nova alternativa somente foi viabilizada após o lançamento do DSP TMS320F2812 (Texas, 2004) e da placa UPCC2812 da HPE (HPE, 2005b). A Figura 4.12 mostra o esquema adotado para a plataforma experimental.

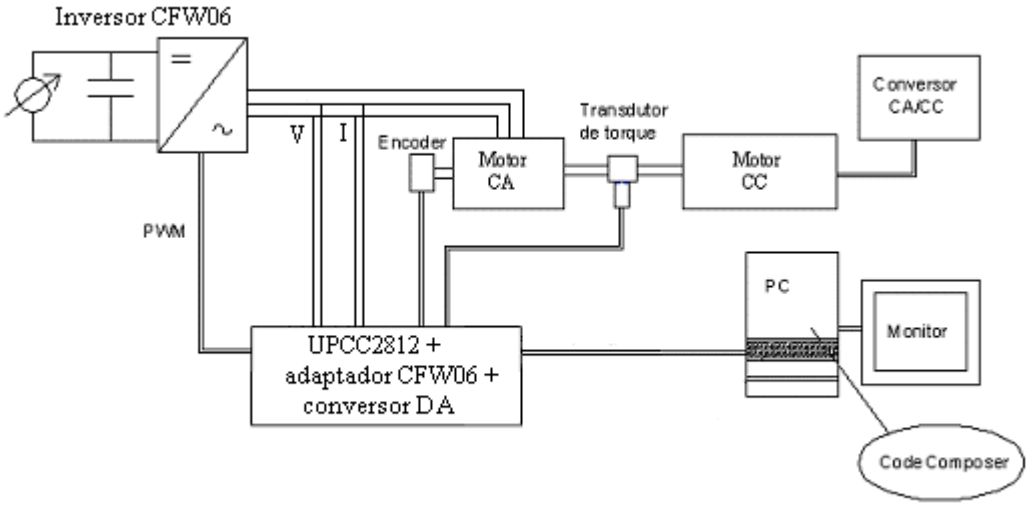


Figura 4.12: Plataforma experimental para acionamento de MI.

Além da placa UPCC2812, a plataforma é composta por um MI de 3 CV - 4 pólos, alimentado pelo inversor CFW06 da WEG, cujo controle é realizado pela UPCC2812 e usando o adaptador CFW06 da HPE (HPE, 2006) para comunicação com o inversor; junto com a placa UPCC2812 é usado um conversor DA (HPE, 2005a) de 12 bits, disponibilizando um total de 8 saídas analógicas; o ajuste da carga mecânica é feito através de um motor CC de 2,2 KW, acionado por um conversor estático. Além disso, a plataforma possui um encoder (1024 ppr) e um transdutor de torque (50 Nm). A implementação da programação do DSP e dos algoritmos de controle é feita usando a linguagem de programação C++ através do ambiente de programação Code Composer Studio^{TM}.

A seguir, são apresentadas as funcionalidades e características básicas das placas desenvolvidas pela HPE e utilizadas na plataforma. A Figura 4.13 mostra uma vista superior da UPCC2812, enquanto a Figura 4.14 apresentada o diagrama de blocos da placa (HPE, 2005b).

Figura 4.13: Controle universal de conversor de potência - UPCC2812.

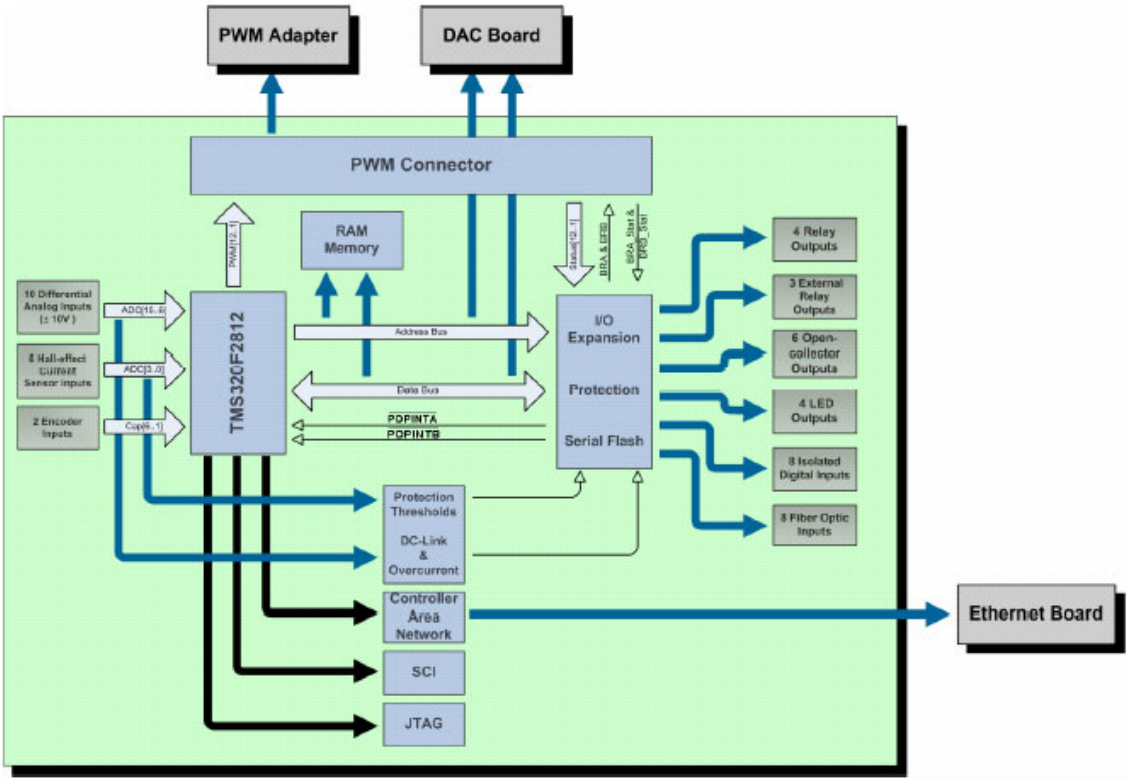


Figura 4.14: Diagrama de blocos da UPCC2812.

A UPCC2812 (*Universal Power Converter Control*) é uma placa de controle *ready-to-use* baseada no DSP TMS320F2812 da Texas Instruments Inc. O condicionamento de sinal e periféricos existentes na placa permitem ao usuário desenvolver, corrigir erros, analisar e aplicar a UPCC2812 no controle de conversores de potência. Com uma arquitetura modular, esta placa pode ser configurada para atender plenamente os requisitos específicos de aplicação, reduzindo os custos e o tempo de desenvolvimento de projeto.

A UPCC2812 permite a imediata utilização da maioria das funcionalidades do TMS320F2812, ampliando a disponibilidade de dispositivos de entrada/saída. A seguir, são relacionadas algumas das funcionalidades desta placa (HPE, 2005b):

- TMS320F2812 operando a 150 MHz;
- palavras de 64k na RAM da placa;
- memória flash serial de 2 Mbits (opcional);
- 6 canais analógicos com condicionamento de sinal para medida de corrente através de sensores de efeito Hall, incluindo a geração de sinal de proteção de *hardware*;
- 10 canais analógicos com condicionamento de sinal para medidas diferenciais ($\pm 10V$);
- 4 saídas a relé na placa (1NO + 1NC, 10A, 250V);
- 4 LEDs de saída na placa;
- 9 saídas de coletor aberto;
- 8 saídas analógicas (placa do conversor DA - opcional);
- 8 entradas digitais isoladas;
- 8 receptores de fibra ótica;
- conectores RS-232, SPI e CAN;
- 2 interfaces para encoder;
- proteção de sobretensão no link-DC;
- 12 saídas PWM + 2 choppers;
- 14 status de entradas de gate-drivers;
- interface para gate-drivers (opcional, de acordo com a disponibilidade);
- emulador compatível com XDS 510$^{\mathrm{TM}}$;
- conectores de expansão.

O adaptador CFW06 da UPCC2812 é uma placa de 17,8 x 4,1 cm conforme mostrado na Figura 4.15. Esta placa é composta por 2 conectores de interface, 3 *jumpers* de configuração e 2 conectores para interfaceamento direto com o módulo de potência CFW06 da WEG.

O módulo DAC da UPCC2812 é uma placa de 7,0 x 7,0 cm conforme mostrado na Figura 4.16. Esta placa é composta por 1 conector de entrada/saída, 9 pinos teste, 2 *jumpers* configuráveis, e 1 conector de interface.

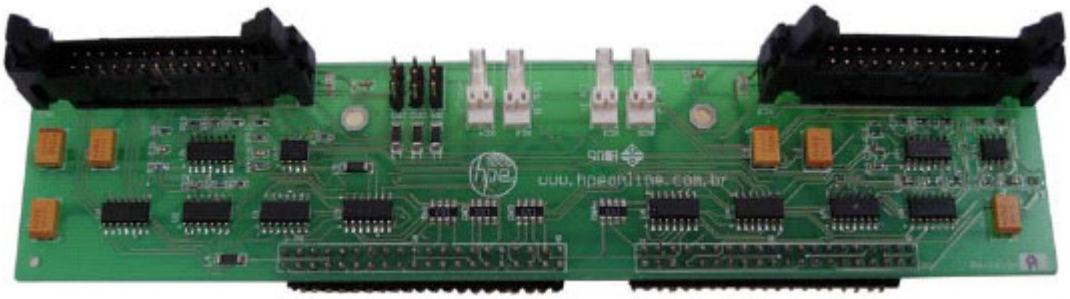


Figura 4.15: Adaptador da UPCC2812 para o CFW06.

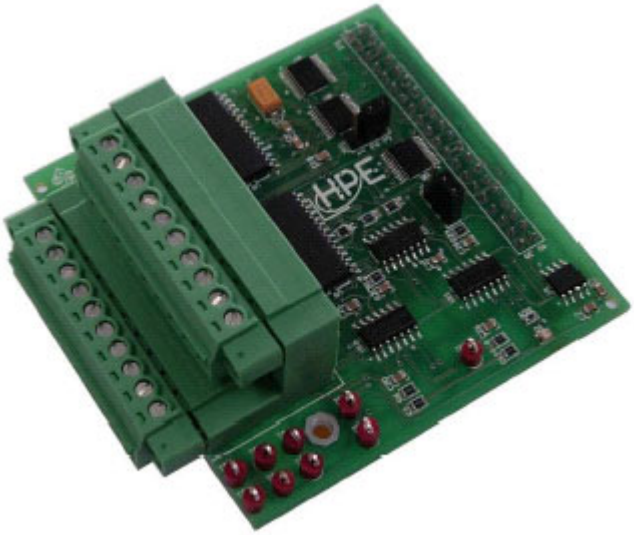


Figura 4.16: Conversor DA da UPCC2812.

A integração da placa UPCC2812, do adaptador da UPCC2812 para o CFW06, do conversor DA da UPCC2812 e do inversor CFW06 da WEG pode ser vista na Figura 4.17. Na Figura 4.18 é mostrada uma visão mais detalhada da integração das placas com o inversor da WEG CFW06 que comanda o MI.

Pode ser facilmente verificado que o adaptador da UPCC2812 para o CFW06 permanece com um conector de interface livre, permitindo a conexão do mesmo com mais um inversor. Em relação ao conversor DA, estão sendo usados 3 dos 9 pinos disponíveis, sendo dois ligados aos canais A e B do osciloscópio e o terceiro é o pino de referência.

Figura 4.17: Visão geral da plataforma experimental desenvolvida.

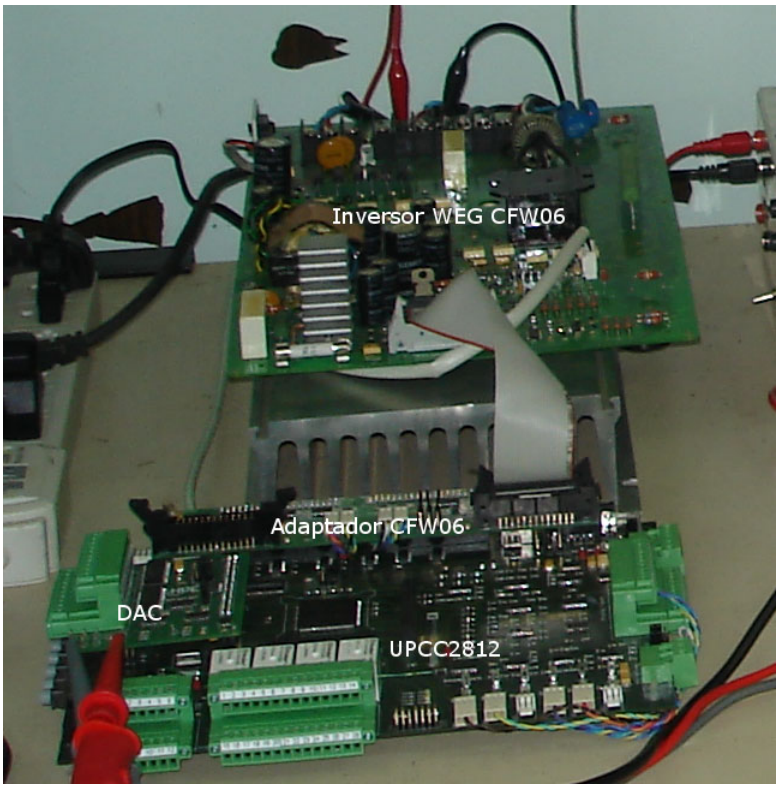


Figura 4.18: Detalhe da integração das placas com o inversor da plataforma experimental.

O estágio atual da operacionalização da plataforma experimental exige ainda a implementação e avaliação dos programas relacionados ao controle vetorial do MI e também, a interligação de alguns equipamentos de *hardware* com a placa UPCC2812 tais como, a carga mecânica através do motor CC e o transdutor de torque.

4.5 Conclusão

Neste capítulo foram apresentados os resultados de simulação dos algoritmos propostos no Capítulo 3 e os resultados experimentais obtidos para o observador neural utilizando a 2a. proposta para treinamento em tempo real de (Parma, 2000). Foi apresentado também uma descrição do desenvolvimento e implementação de uma plataforma experimental para avaliação de estratégias de acionamento elétrico do MI, possibilitando a avaliação de controladores e observadores de estado.

Para avaliação dos algoritmos propostos foram selecionadas duas aplicações quem têm em comum a necessidade de que o treinamento da RNA seja feito em tempo real, permitindo um contínuo ajuste dos pesos da rede às exigências do sistema no qual a rede está inserida. No caso da aproximação de uma função senoidal, os algoritmos propostos apresentaram o menor erro de treinamento. Nas aplicações onde a RNA foi usada como neurocontrolador e como observador neural, os algoritmos propostos apresentaram um bom desempenho.

Pode-se dizer, então, que os resultados obtidos indicam que os algoritmos propostos e implementados como neurocontrolador e como observador neural apresentam características interessantes, tais como: facilidade de uso, sem a necessidade da escolha de um ganho ou taxa de aprendizado para o treinamento da RNA e, comportamento adaptativo, sem a necessidade de qualquer informação do modelo matemático no qual a rede neural está inserida.

Com relação aos resultados experimentais, espera-se que, com a operacionalização da plataforma experimental descrita na Seção 4.4, seja possível implementar, avaliar e validar as diversas propostas de controladores e observadores para o MI, incluindo aquelas apresentadas nesta tese.

No próximo capítulo, serão apresentadas as conclusões deste trabalho e as propostas de continuidade.

Capítulo 5

Conclusões

A utilização da teoria de controle por modos deslizantes no problema de treinamento de redes MLP permite a análise da rede como um sistema a ser controlado, onde as variáveis de controle são os pesos e a saída da rede deve acompanhar a variável de referência. A partir disso, foi usada uma metodologia que permite a obtenção de um ganho adaptativo, determinado iterativamente, a cada passo de atualização dos pesos, dispensando a necessidade do uso de métodos heurísticos na determinação do ganho da rede. Esta metodologia foi usada para treinamento em tempo real de redes MLP com função de ativação linear na camada de saída.

O treinamento de RNA em tempo real pressupõe um processo de aprendizagem realizado enquanto o processamento de sinal está sendo executado pelo sistema, implicando na contínua adaptação dos parâmetros livres da rede neural às variações do sinal incidente em tempo real.

A partir da metodologia empregada, foram desenvolvidos dois algoritmos para treinamento em tempo real de redes MLP de duas camadas com a camada de saída linear. Os algoritmos propostos seguem a mesma metodologia para obtenção do ganho adaptativo, deferindo em dois pontos principais: na definição da superfície de deslizamento e na expressão usada para atualização dos pesos da rede.

Como consequência destas diferenças, a primeira proposta apresentada é generalista, possibilitando que haja um ou mais neurônios na camada de saída da rede, enquanto a segunda proposta é limitada a apenas um neurônio na saída da rede. Para esta segunda proposta, a existência de apenas um neurônio na camada de saída não restringe, em teoria, o escopo de aplicação da proposta, uma vez que é possível se ter duas ou mais estruturas de redes MLP compartilhando as mesmas entradas. Porém, para o treinamento de redes MLP em tempo real, é necessário que o algoritmo utilizado seja simples e com baixo custo computacional, implicando assim, no uso da proposta para aplicações com apenas um sinal de saída.

Em relação a atualização dos pesos da rede, a primeira proposta atualiza os pesos usando o gradiente da função erro em relação aos pesos (algoritmo BP). Esta lei de correção de pesos, apesar de ser bastante usada para treinamento de redes MLP, apresenta deficiências, como por exemplo, o fato de poder ser garantida somente a estabilidade (não a estabilidade assintótica)

para um conjunto de pesos que corresponde ao mínimo global do algoritmo BP, de acordo com a teoria de estabilidade de Lyapunov. Por sua vez, a segunda proposta atualiza os pesos da rede usando uma lei que permite a estabilidade assintótica conforme a teoria de estabilidade de Lyapunov, para um conjunto de pesos que corresponde ao mínimo global.

Através do uso dos algoritmos propostos é possível a determinação de um intervalo resultante para o ganho η da rede, o qual é obtido através da intersecção dos intervalos definidos para a camada escondida e de saída, observando o limite imposto pela decomposição em série de Taylor. Entretanto, os algoritmos propostos não definem o valor final para o ganho η. Assim, é possível, em princípio, ser usado qualquer valor dentro de um intervalo resultante positivo. Questões de otimização não são abordadas pelos algoritmos apresentados. Porém, tendo em vista a necessidade da obtenção de resultados práticos provenientes da aplicação dos algoritmos propostos, foi adotada uma solução conservadora utilizando-se o valor do ganho η obtido para o limite imposto pela decomposição em série de Taylor.

Em virtude das características dos algoritmos propostos, para a avaliação destes algoritmos foram selecionadas aplicações que demandavam adaptação dos parâmetros livres da rede neural em tempo real.

Numa primeira aplicação, as propostas apresentadas nesta tese foram usadas na aproximação de uma função senoidal. O erro de aproximação que os algoritmos propostos apresentaram foi o menor comparado com os valores do erro de aproximação apresentados pelos outros quatro algoritmos simulados.

Em seguida, a segunda proposta foi usada como controlador (neural) em uma estrutura de controle vetorial direto com alimentação em tensão orientado pelo fluxo de estator do MI. A motivação para o uso de redes neurais no controle do MI vem do fato de que, em algumas aplicações, quando incertezas e distúrbios são apreciáveis, técnicas de controle tradicionais não são capazes de garantir desempenho ótimo, ou podem requerer um tempo considerável no estágio de projeto devido à dependência da planta. Isto tem motivado a pesquisa do uso de RNA no controle de MI, com o objetivo de explorar a capacidade das redes neurais para mapeamentos não-lineares complexos.

Os resultados apresentados pelo neurocontrolador do MI usando o algoritmo da segunda proposta mostram um bom desempenho para as malhas de fluxo e velocidade. A falta da malha de conjugado na estrutura do neurocontrolador contribuiu para que houvesse um sobresinal no conjugado eletromagnético nos transitórios aos quais o MI foi submetido. Porém, a inclusão desta malha visando a diminuição ou eliminação do sobresinal no conjugado eletromagnético implica no aumento do custo computacional, exigindo um estudo de viabilidade da implementação desta malha considerando uma aplicação em tempo real.

A outra aplicação estava relacionada com o uso do algoritmo da primeira proposta como observador neural do fluxo de estator do MI. A mesma estrutura de controle do MI foi usada, substituindo-se o observador de Gopinath usado inicialmente pelo observador neural. A análise dos resultados obtidos mostra que o observador neural contribuiu para um bom desempenho das malhas de fluxo, velocidade e conjugado.

A partir dos resultados obtidos na simulação dos algoritmos propostos, pode-se identificar pelo menos duas características destes algoritmos: a facilidade de uso, uma vez que não há a necessidade da determinação do ganho (ou taxa de aprendizado) do algoritmo, o qual é obtido iterativamente pelo próprio algoritmo; dispensa a necessidade de qualquer informação a respeito do modelo matemático do sistema ao qual a rede está inserida.

Finalmente, pode-se dizer, a partir dos resultados obtidos, que a implementação dos algoritmos propostos como neurocontroladores e observador neural numa bancada experimental tem uma boa chance de sucesso. Com esta finalidade está em fase final de desenvolvimento uma plataforma experimental que conta com recursos de *hardware* e *software* modernos e que vão permitir implementar e validar as diversas estratégias de acionamento do MI envolvendo controladores e observadores de estado.

A seguir, são apresentadas as propostas de continuidade desta tese.

5.1 Propostas de Continuidade

As propostas de continuidade da tese podem ser sumarizados como:

- Implementação em bancada experimental dos algoritmos propostos como neurocontroladores e como observador neural de fluxo de estator do MI.
- Determinação do ganho η por algum algoritmo de otimização.
- Desenvolvimento de uma estrutura de acionamento do MI, sem sensor de velocidade, utilizando uma rede neural como estimador de velocidade.
- Aplicação dos algoritmos propostos em outros plantas que necessitem de adaptação paramétrica em tempo real.
- Aplicação da metodologia utilizada para obtenção de algoritmos de treinamento de outras topologias de RNA, diferentes das redes MLP.

Referências Bibliográficas

Bartoszewicz, A. (1998a). Discrete-time quasi-sliding-mode control strategies, *IEEE Tran. on Industrial Electronics* **45**(4): 633–637.

Bartoszewicz, A. (1998b). On the robustness of variable structure systems in the presence of measurement noise, *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, Vol. 3, IEEE Press, pp. 1733–1736.

Behera, L., Kumar, S. e Patnaik, A. (2006). On adaptive learning rate that guarantees convergence in feedforward networks, *IEEE Trans. on Neural Networks* **17**(5): 1116–1125.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*, Vol. I e II, Athenas Scientific, Belmont, MA.

Bishop, C. M. (1995). *Neural networks for pattern recognition*, Claredon Press, Oxford.

Braga, A. P., Carvalho, A. e Ludermir, T. (2000). *Redes Neurais Artificiais: teoria e aplicações*, LTC - Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, RJ.

Brent, R. P. (1991). Fast training algorithms for multilayer neural nets, *IEEE Trans. on Neural Networks* **2**(3): 346–354.

Butkov, E. (1978). *Física Matemática*, Editora Guanabara Dois S. A.

Cascella, G. L., Cupertino, F., Topalov, A. V., Kaynak, O. e Giordano, V. (2005). Adaptive control of electric drives using sliding-mode learning neural networks, *IEEE ISIE 2005*, IEEE Press, Dubrovnik, Croatia, pp. 125–130.

Costa, M. A. (2002). *Controle por Modos Deslizantes da Generalização em Aprendizado de Redes Neurais Artificiais*, Tese de doutorado, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, MG.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems* **2**(4): 304–314.

Dedoncker, R. W. e Novotny, D. W. (1988). The universal field oriented controller, *Proceedings of IEEE-IAS*, Pittsburg, USA, pp. 450–456.

Efe, M. O. e Kaynak, O. (2000). Stabilizing and robustifying the error backpropagation method in neurocontrol applications, *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, IEEE Press, San Francisco, CA, pp. 1882–1887.

Efe, M. O. e Kaynak, O. (2001). Variable structure systems theory based training strategies for computationally intelligent systems, *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society*, IEEE Press, pp. 1563–1576.

Efe, M. O., Kaynak, O. e Wilamowski, B. M. (2000). Stable training of computationally intelligent systems by using variable structure systems technique, *IEEE Trans. on Industrial Electronics* **47**(2): 487–496.

Emelyanov, S. V. (1959). Control of first order delay systems by means of an astatic controller and nonlinear correction, *Autom. Remote Control* (8): 983–991.

Fahlman, S. E. (1988). Faster-learning variations on backpropagation: an empirical study, *in* D. Touretzky, G. Hinton e T. Sejnowsky (eds), *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, Sao Mateo, CA, pp. 38–51.

Gao, W., Wang, Y. e Homaifa, A. (1995). Discrete-time variable structure control systems, *IEEE Trans. on Industrial Electronics* **42**(2): 117–122.

German, S., Bienenstock, E. e Dournsat, R. (1992). Neural networks and the bias/variance dilemma, *Neural Computation* **4**(1): 1–58.

Giordano, V., Topalov, A. V., Kaynak, O. e Turchiano, B. (2004). Sliding-mode approach for on-line neural identification of robotic manipulators, *2004 5th Asian Control Conference*, pp. 2060–2065.

Gouvêa, M. R., Figueiredo, E. S., Menezes, B. R., Parma, G. G., Caminhas, W. M. e Baccarini, L. M. R. (2004). Stator flux estimation with dc offset compensation, *Anais do XV Congresso Brasileiro de Automática*, Sociedade Brasileira de Automática, Gramado, RS.

Hagan, M. T. e Menhaj, M. B. (1994). Training feedforword networks with the marquardt algorithm, *IEEE Trans. on Neural Networks* **5**(6): 989–993.

Haykin, S. (1996). *Adaptive Filter Theory*, 3rd edn, Prentice-Hall, Englewood Cliffs, NJ.

Haykin, S. (2001). *Redes Neurais: princípios e prática*, 2 ed., Bookman, Porto Alegre, RS.

Hebb, D. O. (1949). *The Organization of Behaviour: A Neuropsychological Theory*, Wiley, New York.

Holtz, J. e Quan, J. (2003). Drift- and parameter-compensated flux estimator for persistent zero-stator-frequency operation of sensorless-controlled induction motors, *IEEE Trans. on Industrial Applications* **39**(4): 1052–1060.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Science of the USA*, Vol. 79, pp. 2554–2558.

Hori, Y., Cotter, V. e Kaya, Y. (1987). A novel induction machine flux observer and its application to a high performance ac drive system, *Proceedings of 10th World Congress on Automatic Control - IFAC*, Vol. 3, pp. 355–360.

HPE (2005a). *UPCC2812 DAC Module - Technical Reference*, High Power Engineering Ltda, Belo Horizonte, MG, Brasil. Rev. A.

HPE (2005b). *UPCC2812 Universal Power Converter Control - Technical Reference*, High Power Engineering Ltda, Belo Horizonte, MG, Brasil. Rev. B.

HPE (2006). *UPCC2812 CFW06 Adapter - Technical Reference*, High Power Engineering Ltda, Belo Horizonte, MG, Brasil. Rev. A.

Hung, J. Y., Gao, W. e Hung, J. C. (1993). Variable structure control: A survey, *IEEE Trans. on Industrial Electronics* **40**(1): 2–22.

Iiguni, Y., Sakai, H. e Tokumaru, H. (1992). A real-time learning algorithm for a multilayered neural network based on extended kalman filter, *IEEE Trans. Signal Process.* **40**(4): 959–966.

Itkis, U. (1976). *Control systems of variable structure*, John Wiley and Sons Inc., New York.

Justino, J. C. G. (2004). *Redes neurais artificiais com treinamento on-line aplicadas ao controle do motor de indução*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, MG.

Justino, J. C. G., Nied, A., Menezes, B. R. e Parma, G. G. (2004b). Neuro-controladores on-line baseados em sistemas de estrutura variável (sev) e controle por modos deslizantes (cmd): aplicação no acionamento do motor de indução, *Anais do Simpósio Brasileiro de Redes Neurais*, São Luis, MA.

Justino, J. C. G., Nied, A., Menezes, B. R., Parma, G. G. e Braga, A. P. (2003). Treinamento de redes neurais utilizando controle por modos deslizantes com ganho adaptativo, *Anais do VI Simpósio Brasileiro de Automação Inteligente*, Bauru, SP, pp. 206–211.

Justino, J. C. G., Nied, A., Parma, G. G. e Menezes, B. R. (2004a). Uso de neuro-controladores aplicados a máquinas de indução utilizando algoritmo baseado em sistemas de estrutura variável (sev) e controle por modos deslizantes (cmd), *Anais do XV Congresso Brasileiro de Automática*, Gramado, RS.

Justino, J. C. G., Parma, G. G., Nied, A. e Menezes, B. R. (2004c). Neurocontrollers and pi controllers applied in motor induction drives: expectations, advantages and disadvantages., *Anais da VI Conferência Internacional de Aplicações Industriais*, Joinville, SC.

Kaynak, O., Erbatur, K. e Ertugrul, M. (2001). The fusion of computationally intelligent methodologies and sliding-mode control - a survey, *IEEE Trans. on Industrial Electronics* **48**(1): 4–17.

Kovács, P. K. e Rácz, E. (1984). *Transient Phenomena in Electrical Machines*, Elsevier, Amsterdam, The Netherlands.

Kreyszig, E. (1993). *Advanced Engineering Mathematics*, 7th edn, John Wiley and Sons Inc.

Kuan, C. M. e Hornik, K. (1991). Convergence of learning algorithms with constant learning rates, *IEEE Trans. on Neural Networks* **2**(5): 484–489.

Leonhard, W. (1985). *Control of Electrical Drives*, Springer-Verlag, Berlin, Germany.

Ljung, L. (1987). *System Identification: Theory for the User*, Prentice-Hall, Englewood Cliffs, NJ.

Marquardt, D. W. (1963). An algorithm for least squares estimation of nonlinear parameters, *Journal of the Society for Industrial and Applied Mathematics* **11**(2): 431–441.

Mason, S. J. (1953). Feedback theory - some properties of signal-flow graphs, *Proceedings of the Institute of Radio Engineers* **41**: 1144–1156.

McCulloch, W. e Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics* **5**: 115–133.

Miloslavjevic, C. (1985). General conditions for the existence of a quasisliding mode on the switching hyperplane in discrete variable structure systems, *Automation and Remote Control* **46**: 307–314.

Minsky, M. L. e Papert, S. A. (1969). *Perceptrons*, MIT Press, Cambridge, MA.

Narendra, K. S. e Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks, *IEEE Trans. on Neural Networks* **1**(1): 4–27.

Nied, A., Gouvêa, M. R., Caminhas, W. M., Menezes, B. R., Junior, S. I. S. e Parma, G. G. (2004). Performance evaluation of flux estimators and observers for induction motors, *Anais da VI Conferência Internacional de Aplicações Industriais*, Joinville, SC.

Nied, A., Junior, A. P., Souza, A. H. e Menezes, B. R. (2002). Plataforma para ensaios de sistemas de controle digital de tempo real usando dsps, *Anais do XXX Congresso Brasileiro de Ensino de Engenharia*, Piracicaba, SP.

Nied, A., Junior, S. I. S., Menezes, B. R., Parma, G. G. e Justino, J. C. G. (2003a). Comparative study on flux observers for induction motor drives, *Proceedings of the 7th Brazilien Power Electronics Conference*, Fortaleza, CE.

Nied, A., Junior, S. I. S., Parma, G. G. e Menezes, B. R. (2003b). On-line training algorithms for an induction motor stator flux neural observer, *Proceedings of 29th annual conference of the IEEE Industrial Electronics Society - IECON2003*, IEEE Press, Roanoke, VA, pp. 129–134.

Nied, A., Junior, S. I. S., Parma, G. G. e Menezes, B. R. (2005a). On-line adaptive neural training algorithm for an induction motor flux observer, *Power Electronic Specialists Conference 2005 - PESC2005*, IEEE Press, Recife, PE.

Nied, A., Menezes, B. R. e Parma, G. G. (2005b). On-line neural training algorithm with sliding mode control and adaptive learning rate, *Seminário do Progama de Pós-Graduação em Engenharia Elétrica da UFMG*, Belo Horizonte, MG.

Nied, A., Seleme Junior, S. I., Parma, G. G. e Menezes, B. R. (2007). On-line neural training algorithm with sliding mode control and adaptive learning rate, *Neurocomputing* . in press.

Novotny, D. W. e Lipo, T. A. (1996). *Vector control and dynamics of AC drives*, 1st edn, Cleredon Press.

Oh, S. H. e Lee, S. Y. (1999). A new error function at hidden layers for fast training of multilayer perceptrons, *IEEE Trans. on Neural Networks* **10**(4): 960–964.

Parisi, R., Di Claudio, E. D., Orlandi, G. e Rao, B. D. (1996). A generalized learning paradigm exploiting the structure of feedforward neural networks, *IEEE Trans. on Neural Networks* **7**(6): 1450–1460.

Parma, G. G. (2000). *Treinamento de Redes Neurais Artificiais Baseado em Sistemas de Estrutura Variável com Aplicações em Acionamentos Elétricos*, Tese de doutorado, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, MG.

Parma, G. G., Menezes, B. R. e Braga, A. P. (1998a). Sliding mode algorithm for training multilayer neural network, *IEE Electronics Letters* **38**(1): 97–98.

Parma, G. G., Menezes, B. R. e Braga, A. P. (1999a). Neural networks learning with sliding mode control: the sliding mode backpropagation algorithm, *International Journal of Neural Systems* **9**(3): 187–193.

Parma, G. G., Menezes, B. R. e Braga, A. P. (1999b). Sliding mode backpropagation: control theory applied to neural networks learning, *Proceedings of the International Joint Conference on Neural Networks*, IEEE Computer Society Press, Washington-DC, pp. 1774–1778.

Parma, G. G., Menezes, B. R., Braga, A. P., Oliveira, J. C. R. e Aguirre, L. A. (1998b). Observador neural de fluxo estatórico com treinamento on-line, *Anais do XII Congresso Brasileiro de Automática*, Vol. 4, Sociedade Brasileira de Automática, Uberlândia, MG, pp. 1301–1306.

Proakis, J. G. (1989). *Digital Communications*, 2nd edn, McGraw-Hill, New York.

Reed, R. (1993). Pruning algorithms - a survey, *IEEE Trans. on Neural Networks* **4**(5): 740–746.

Riedmiller, M. e Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm, *Proceedings of the Int. Conf. on Neural Networks*, San Francisco, CA, pp. 586–591.

Rosenblatt, F. (1958). The preceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review* **65**: 386–408.

Rumelhart, D. E., Hinton, G. E. e Williams, R. J. (1986). *Learning internal representation by error propagation*, Vol. 1 of *Parallel distribuited processing: explorations in the microstructure of cognition*, MIT Press, Cambridge, MA, pp. 318–362.

Sarpturk, S., Istefanopulos, Y. e Kaynak, O. (1987). On the stability of discrete-time sliding mode control systems, *IEEE Trans. on Automatic Control* **32**(10): 930–932.

Silva, F. M. e Almeida, L. B. (1990). Speeding up backpropagation, *in* R. Eckmiller (ed.), *Advanced Neural Computers*, Amsterdam: Elsevier North Holland, pp. 151–158.

Silva, S. R. (1995). *Sistemas elétricos de alto desempenho a velocidade variável: estratégias de controle e aplicações*, Tese para concurso de professor titular, Escola de Engenharia, Belo Horizonte, MG.

Sira-Ramirez, H. e Colina-Morles, E. (1995). A sliding mode strategy for adaptive learning in adalines, *IEEE Trans. on Circuits and Systems - I: Fundamental Theory and Applications* **42**(12): 1001–1012.

Sira-Ramirez, H. J. e Zak, S. H. (1991). The adaptation of perceptrons with applications to inverse dynamics identification of unknown dynamic system, *IEEE Trans. on Systems, Man and Cybernetics* **21**(3): 634–643.

Souza, A. H., Zipf, A. J., Junior, A. P. e Nied, A. (2002). Plataforma para acionamento e controle de motores de indução usando dsp, *Anais do XIV Congresso Brasileiro de Automática*, Sociedade Brasileira de Automática, Natal, RN, pp. 298–303.

Teixeira, R. A. (2001). *Treinamento de Redes Neurais Artificiais Através de Otimização Multi-objetivo*, Tese de doutorado, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, MG.

Texas (2004). *TMS320F2812 Data Manual*, Texas Instruments Inc., Dallas, Texas, USA. Literature number SPRS 174K.

Tollenaere, T. (1990). Supersab: Fast adaptive back propagation with good scaling properties, *Neural Networks* **3**(5): 561–573.

Topalov, A. V. e Kaynak, O. (2003). A sliding mode strategy for adaptive learning in multilayer feedforward neural networks with a scalar output, *IEEE International Conference on Systems, Man and Cybernetics, 2003*, Vol. 2, IEEE Press, pp. 1636–1641.

Topalov, A. V., Kaynak, O. e Shakev, N. G. (2003). Variable structure systems approach for on-line learning in multilayer artificial neural networks, *Proc. IEEE 29th Annual Conference of the Industrial Electronics Society (2003)*, IEEE Press, Roanoke, VA, pp. 2989–2994.

Utkin, V. I. (1978). *Sliding modes and their application in Variable Structure Systems*, MIR, Moskow.

Utkin, V. I. (1992). *Sliding Modes in Control Optimization*, Springer-Verlag, Berlin, Germany.

Wan, E. A. (1990a). Temporal backpropagation: An efficient algorithm for finite impulse response neural networks, *in* M. Kaufmann (ed.), *Proc. of the 1990 Connectionist Models Summer School*, pp. 131–140.

Wan, E. A. (1990b). Temporal backpropagation for fir neural networks, *Proc. IEEE Int. Joint Conf. on Neural Networks*, Vol. 1, San Diego, CA, pp. 575–580.

Werbos, P. (1990). Backpropagation trough time: What it does and to do it, *Proceedings of The IEEE* **78**(10): 1550–1560.

Widrow, B. e Hoff, M. E. (1960). Adaptive switching circuits, *IRE WESCON Convention Record*, Vol. 4, IRE, New York, pp. 96–104.

Widrow, B. e Lehr, M. A. (1990). 30 years of adaptive neural networks: Perceptrons, madaline, and backpropagation, *Proceedings of The IEEE* **78**(9): 1415–1442. Special Issue on Neural Networks.

Widrow, B. e Steams, S. D. (1985). *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Williams, R. J. e Peng, J. (1989). Reinforcement learning algorithms as function optimizers, *International Joint Conference on Neural Networks*, Vol. 2, New York, pp. 89–95.

Yam, J. Y. F. e Chow, W. S. (1997). Extended least squares based algorithm for training feedforward networks, *IEEE Trans. on Neural Networks* **8**(3): 806–810.

Young, K. D., Utkin, V. I. e Ozguner, U. (1999). A control engineer's guide to sliding mode control, *IEEE Trans. Contr. Syst. Technol.* **7**(3): 328–342.

Yu, X., Efe, M. O. e Kaynak, O. (2002). A general backpropagation algorithm for feedforward neural networks learning, *IEEE Trans. on Neural Networks* **13**(1): 251–254.

Yu, X., Zhihong, M. e Rahman, S. M. M. (1998). Adaptive sliding mode approach for learning in a feedforward neural networks, *Neural Computing and Applications* **7**(4): 289–294.

Zhao, Y. (1996). On-line neural network learning algorithm with exponential convergence rate, *IEE Electronics Letters* **32**(15): 1381–1382.

Zhou, G. e Si, J. (1998). Advanced neural network training algorithm with reduced complexity based on jacobian deficiency, *IEEE Trans. on Neural Networks* **9**(3): 448–453.

Apêndice A

Modelo de um Neurônio

Um neurônio é uma unidade de informação que é fundamental para o operação de uma rede neural. A Figura A.1 mostra o modelo de um neurônio, que forma a base para o projeto de redes neurais artificiais (Haykin, 2001).

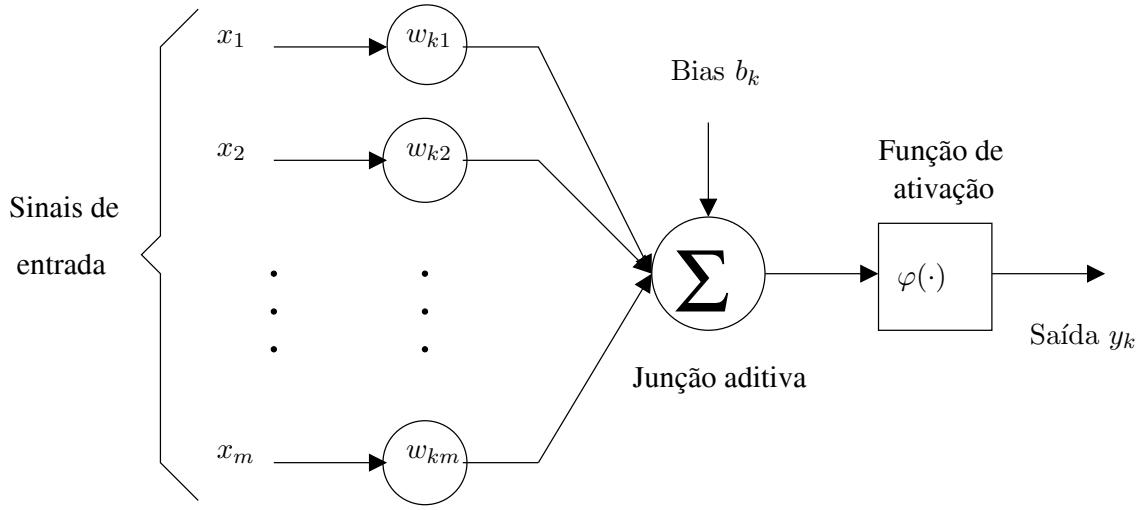


Figura A.1: Modelo não-linear de um neurônio.

Pode-se identificar três elementos básicos neste modelo:

1. Um conjunto de *sinapses* (ou *elos de conexão*), cada uma caracterizada por um *peso* ou *força* própria. Por exemplo, um sinal x_j na entrada da sinapse j conectada ao neurônio k é multiplicado pelo peso sináptico w_{kj}. Deve-se notar a maneira como são escritos os índices do peso sináptico w_{kj}. O primeiro índice se refere ao neurônio em questão e o segundo se refere ao terminal de entrada da sinapse à qual o peso se refere. Ao contrário de uma sinapse do cérebro, o peso sináptico de um neurônio artificial pode estar em um intervalo que inclui tanto valores positivos quanto negativos.

2. Um *somador* para somar os sinais de entrada, ponderados pelas respectivas sinapses do neurônio; as operações resultantes constituem um *combinador linear*.

3. Uma *função de ativação* para restringir a amplitude da saída do neurônio. A função de ativação também pode ser referida como *função restritiva* uma vez que restringe (limita) o intervalo de amplitude do sinal de saída a um valor finito. Tipicamente, o intervalo normalizado da amplitude da saída de um neurônio é escrito como o intervalo unitário fechado $[0, 1]$ ou, alternativamente, $[1, -1]$.

O modelo neuronal da Figura A.1 inclui também um *bias* aplicado externamente, representado por b_k. O bias b_k tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente, conforme pode ser visto na Figura A.2.

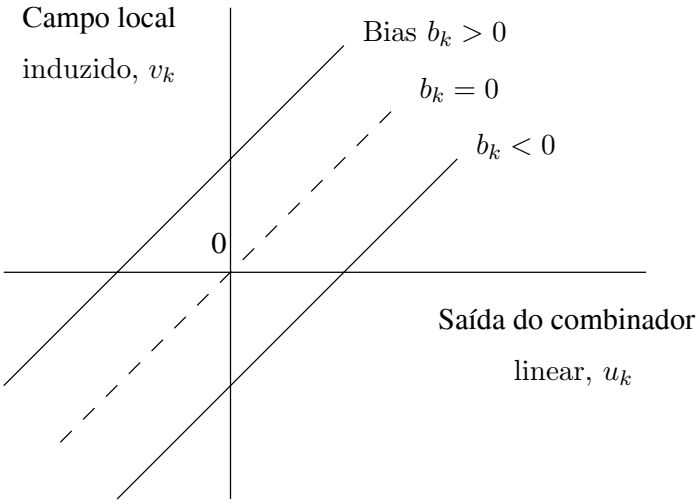


Figura A.2: Transformação afim produzida pela presença de um bias.

Em termos matemáticos, pode-se descrever um neurônio k a partir do seguinte par de equações:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{A.1}$$

e

$$y_k = \varphi(u_k + b_k) \tag{A.2}$$

onde $x_1, x_2, ..., x_m$ são os sinais de entrada; $w_{k1}, w_{k2}, ..., w_{km}$ são os pesos sinápticos do neurônio k; u_k é a saída do *combinador linear* devido aos sinais de entrada; b_k é o bias; $\varphi(\cdot)$ é a *função de ativação*; e y_k é o sinal de saída do neurônio. O uso do bias b_k tem o efeito de uma *transformação afim* à saída u_k do combinador linear no modelo da Figura A.2, como dado por

$$v_k = u_k + b_k \tag{A.3}$$

Em particular, dependendo se o bias b_k é positivo ou negativo, a relação entre o *campo local induzido* ou *potencial de ativação* v_k do neurônio k e a saída do combinador linear u_k é modificada como mostrado em (A.3).

O bias b_k é um parâmetro externo do neurônio artificial k. Assim, pode-se reescrever (A.1) a (A.3) da seguinte maneira:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{A.4}$$

$$y_k = \varphi(v_k) \tag{A.5}$$

Em (A.5) foi adicionada uma nova sinapse, cuja entrada e peso são $x_0 = +1$ e $w_{k0} = b_k$, respectivamente. Pode-se, portanto, reformular o modelo do neurônio k como mostrado na Figura A.3. Nesta figura, verifica-se que o efeito do bias é levado em conta de duas maneiras: (1) adicionando-se um novo sinal de entrada fixo em +1, e (2) adicionando-se um novo peso sináptico igual ao bias b_k. Apesar dos modelos das Figuras A.1 e A.3 serem diferentes na aparência, eles são matematicamente equivalentes.

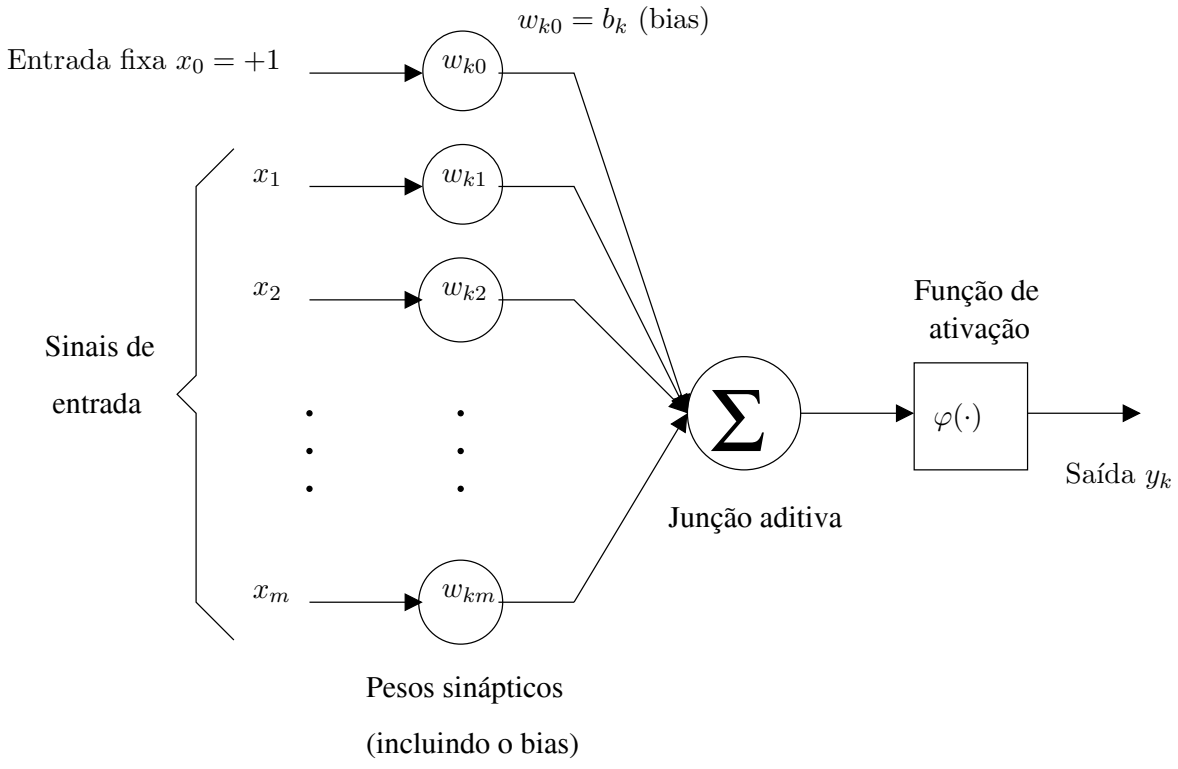


Figura A.3: Outro modelo não-linear de um neurônio.

Tipos de Função de Ativação

A função de ativação $\varphi(v)$ define a saída de um neurônio em termos do campo local induzido v. A seguir, são apresentados três tipos básicos de funções de ativação utilizadas nas RNA:

1. *Função de Limiar*. Para este tipo de função de ativação tem-se

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0. \end{cases} \tag{A.6}$$

Assim, a saída do neurônio k que emprega esta função de limiar é dada como

$$y_k = \begin{cases} 1, & \text{se } v_k \geq 0 \\ 0, & \text{se } v_k < 0, \end{cases} \tag{A.7}$$

onde v_k é o campo local induzido do neurônio. Tal neurônio é conhecido na literatura como *modelo de McCulloch-Pitts*, como reconhecimento ao trabalho pioneiro realizado por (McCulloch e Pitts, 1943). Neste modelo, a saída de um neurônio assume o valor 1 se o campo local induzido daquele neurônio é não-negativo, e 0 caso contrário. Esta definição descreve a *propriedade tudo-ou-nada* do modelo de McCulloch-Pitts.

2. *Função Linear por Partes*. Para este tipo de função de ativação tem-se

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq +\frac{1}{2} \\ v, & \text{se } +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & \text{se } v \leq -\frac{1}{2}, \end{cases} \tag{A.8}$$

onde assume-se que o fator de amplificação dentro da região linear de operação é a unidade. Esta forma de função de ativação pode ser vista como uma *aproximação* de um amplificador não-linear. Ainda, se a região linear de operação é mantida sem entrar em saturação, resulta num combinador linear e, a função linear por partes se reduz à função de limiar se o fator de amplificação da região linear é feito infinitamente grande.

3. *Função Sigmóide*. Esta função, cujo gráfico tem um formato de um s, é a forma mais comum de função de ativação utilizada na construção de RNA. Ela é definida como uma função *estritamente crescente* que exibe um balanceamento adequado entre comportamento linear e não-linear. Uma função sigmóide muito usada é a *função logística*, definida como

$$\varphi(v) = \frac{1}{1 + \exp(-av)}, \tag{A.9}$$

onde a é o parâmetro da função sigmóide. Variando-se o parâmetro a, obtém-se funções sigmóides com diferentes inclinações. Enquanto a função de limiar assume o valor de 0 ou 1, uma função sigmóide assume um intervalo contínuo de valores entre 0 e 1. Observe também que a função sigmóide é *diferenciável*, enquanto a função de limiar não possue esta propriedade. No limite (quando a se aproxima do infinito), a função sigmóide se torna uma função de limiar.

As funções de ativação definidas em (A.6), (A.8) e (A.9) se estendem de 0 a +1. Algumas

vezes é desejável que a função de ativação se estenda de -1 a +1, assumindo assim uma forma anti-simétrica em relação à origem. Desta forma, a função de limiar é agora definida como

$$\varphi(v) = \begin{cases} 1, & \text{se } v > 0 \\ 0, & \text{se } v = 0 \\ -1, & \text{se } v < 0, \end{cases} \tag{A.10}$$

a qual é normalmente definida como *função sinal*. Para a forma correspondente de uma função sigmóide, pode-se usar a função *tangente hiperbólica*, definida por

$$\varphi(v) = \tanh(v). \tag{A.11}$$

O fato de uma função de ativação do tipo sigmóide poder assumir valores negativos permite que o aprendizado do neurônio possa se dar mais rapidamente (em termos do número de iterações de treinamento) do que se ela assumisse apenas valores positivos.

A.1 Redes Neurais Vistas como Grafos Orientados

O diagrama em blocos das Figuras A.1 e A.3 fornece uma descrição funcional dos vários elementos que constituem o modelo de um neurônio artificial. Pode-se simplificar a aparência do modelo utilizando a idéia de grafos de fluxo de sinal sem sacrifício de quaisquer detalhes do modelo (Haykin, 2001). Os grafos de fluxo de sinal e suas regras de funcionamento foram desenvolvidos por (Mason, 1953) para redes lineares. A presença de não-linearidade no modelo de um neurônio limita o escopo da aplicação de grafos de fluxo de sinal em RNA. Apesar disso, os grafos de fluxo de sinal fornecem um método elegante para retratar o fluxo dos sinais em uma rede neural.

Com base nas regras definidas por (Mason, 1953), pode-se construir, por exemplo, o grafo do fluxo de sinal mostrado na Figura A.4 como o modelo de um neurônio, correspondente ao diagrama de blocos da Figura A.3:

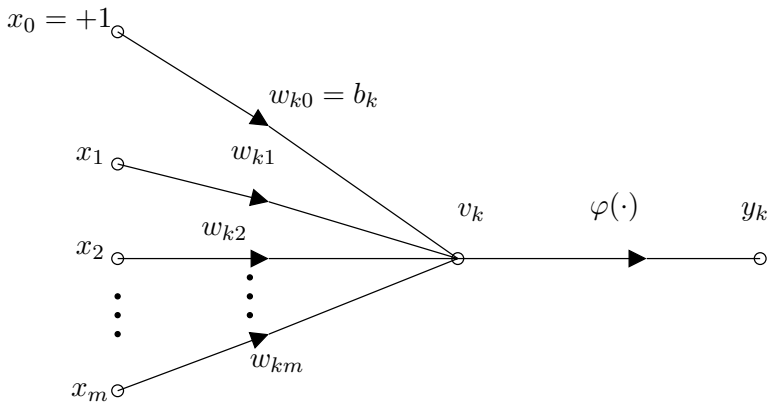


Figura A.4: Grafo de fluxo de sinal de um neurônio.

Com base no grafo de fluxo de sinal mostrado na Figura A.4 como modelo de um neurônio, pode-se apresentar a seguinte definição matemática de uma rede neural.

Definição A.1 *Uma rede neural é um grafo orientado constituído de nós com elos de interligação sinápticos e de ativação e é caracterizada por quatro propriedades:*

1. *Cada neurônio é representado por um conjunto de elos sinápticos lineares, um bias aplicado externamente e um elo de ativação possivelmente não-linear. O bias é representado por um elo sináptico conectado a uma entrada fixa em +1.*
2. *Os elos sinápticos de um neurônio ponderam os seus respectivos sinais de entrada.*
3. *A soma ponderada dos sinais de entrada define o campo local induzido do neurônio em questão.*
4. *O elo de ativação limita o campo local induzido do neurônio para produzir uma saída.*

O estado do neurônio pode ser definido em termos do seu campo local induzido ou de seu sinal de saída (se o neurônio possuir função de ativação linear, o sinal de saída coincide com seu campo local induzido). Um grafo orientado obtido a partir da Definição A.1 é *completo*, descrevendo não somente o fluxo de sinal de neurônio para neurônio, mas também o fluxo de sinal dentro de cada neurônio.

Apêndice B

Equações Matemáticas

Neste apêndice são apresentadas algumas equações matemáticas que serviram de base para a dedução dos algoritmos propostos apresentados no Capítulo 3.

B.1 Determinação das raízes de um polinômio de 2^o grau

A equação de um polinômio de ordem 2 em relação a variável x pode ser descrita como:

$$ax^2 + bx + c, \tag{B.1}$$

onde $a,\ b,\ c$ são coeficientes reais. Para o cálculo das raízes, deve-se determinar o valor de Δ:

$$\Delta = b^2 - 4ac. \tag{B.2}$$

As raízes são definidas de acordo com o valor de Δ, ou seja:

- $\Delta \geq 0$, as raízes são calculadas utilizando a seguinte expressão:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}. \tag{B.3}$$

- $\Delta < 0$, não existem raízes reais distintas.

- $\Delta = 0$, existe uma raiz real dupla definida como:

$$x = \frac{-b}{2a}. \tag{B.4}$$

O comportamento da concavidade da parábola está associado ao sinal do coeficiente a, como pode ser visto na Figura B.1.

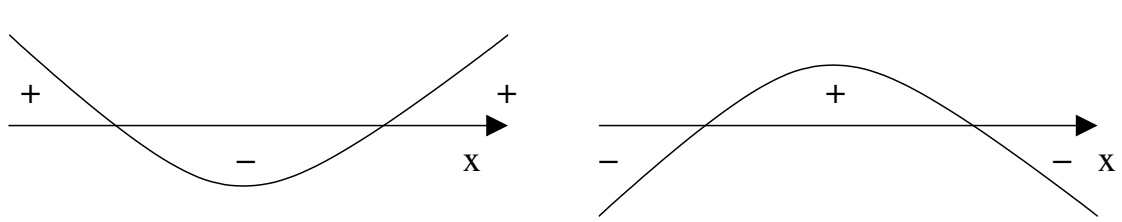


Figura B.1: (a) Coeficiente $a > 0$; (b) Coeficiente $a < 0$.

Assim, polinômios de ordem 2 com coeficiente a positivo tem concavidade positiva enquanto aqueles com coeficiente a negativo tem concavidade negativa.

B.2 Decomposição da Função de Ativação em Série de Taylor

Seja a expansão em série de Taylor de uma função $f(x)$ no ponto $x_0 + \Delta x$:

$$f(x_0 + \Delta x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(\Delta x)^n, \tag{B.5}$$

onde $\Delta x = x - x_0$.

É possível estimar uma aproximação da função desejada utilizando apenas alguns termos da série, como uma aproximação de primeira ordem:

$$f(x_0 + \Delta x) = f(x_0) + \dot{f}(x_0)\Delta x. \tag{B.6}$$

Para a aproximação expressa por (B.6) existe um erro associado que pode ser minimizado com a redução do valor de Δx. Este erro depende da complexidade da função a ser aproximada. De forma sucinta, a aproximação de uma função através da decomposição de primeira ordem é dada por (B.6) para $|\Delta x| \leq \xi$, onde ξ é o valor positivo que representa o intervalo no qual a aproximação é válida. O valor de ξ varia dependendo do tipo de função a ser aproximada e do erro de aproximação desejado, conforme pode ser verificado na Tabela B.1.

ξ	0.001	0.01	0.05	0.1	0.2	0.5	1.0	1.5
Erro médio	1.33e-14	1.33e-10	8.33e-8	1.33e-6	2.12e-5	8.01e-4	1.15e-2	4.92e-2

Tabela B.1: Erros médios e intervalos de confiança para a aproximação da função tangente hiperbólica utilizando a expansão de primeira ordem em série de Taylor.

A partir da Tabela B.1 é possível identificar os valores dos erros médios de aproximação da função tangente hiperbólica bem como os valores de ξ associados.

Apêndice C

Parâmetros e Equações do Motor de Indução

Neste apêndice são mostrados os parâmetros (Tabela C.1) usados na simulação do MI a partir do programa desenvolvido em linguagem C. São mostrados também os parâmetros (Tabela C.2) e as principais equações do modelo do MI.

Parâmetros	**Valores**
Passo de integração (μs)	1
Tempo de simulação (s)	5
Frequência de amostragem (kHz)	4
Tensão do elo DC (V)	300
Carga usada no ensaio de aplicação e retirada de carga (Nm)	4
Velocidade de referência do MI (rad.ele/s)	150
Fluxo de referência (Wb)	0,21

Tabela C.1: Parâmetros da simulação.

Parâmetros	**Valores**
Potência (CV)	2
Tensão de alimentação (V) - Δ/Y	92,4/160
Corrente nominal (A) - Δ/Y	14,4/8,3
Velocidade nominal (rpm)	1715
Resistência de estator (Ω)	0,995
Resistência de rotor (Ω)	0,696
Indutância de dispersão de estator (mH)	2,362
Indutância de dispersão de rotor (mH)	3,525
Indutância de magnetização (mH)	45,601
Coeficiente de perdas rotacionais (Ws^2/rad^2)	0,0008718
Momento de inércia (Nms^2)	0,006547

Tabela C.2: Parâmetros do MI.

O modelo básico do motor de indução em referencial girante à velocidade arbitrária ω_e é dado por (Novotny e Lipo, 1996):

- Equações de tensão:

$$\mathbf{v}_s = R_s\mathbf{i}_s + \frac{d\lambda_s}{dt} + j\omega_e\lambda_s \tag{C.1}$$

$$\mathbf{v}_r = R_r\mathbf{i}_r + \frac{d\lambda_r}{dt} + j(\omega_e - \omega_r)\lambda_r \tag{C.2}$$

- Equações de enlace de fluxo:

$$\lambda_s = L_s\mathbf{i}_s + M\mathbf{i}_r \tag{C.3}$$

$$\lambda_r = L_r\mathbf{i}_r + M\mathbf{i}_s \tag{C.4}$$

- Equações de conjugado eletromagnético:

$$\frac{2}{p}J\frac{d\omega_r}{dt} = T_e - T_c \tag{C.5}$$

$$T_e = \frac{3}{2}\frac{p}{2}\mathrm{Im}\{\lambda_s^* i_s\}. \tag{C.6}$$